Markt&Technik

Markus Breuer

# AMIGA 500-BUCH

Hardware ★ Workbench 1.3 und Extras-Diskette
★ CLI-Einführung und Überblick
★ Amiga-Grafik ★ Sound-Erzeugung
★ Amiga-Basic

2., überarbeitete und erweiterte Auflage

Für Workbench 1.3

Amiga-500-Buch

Markus Breuer

# AMIGA 500-Buch

Hardware ★ Workbench 1.3 und Extras-Diskette
★ CLI-Einführung und Überblick ★
Amiga-Grafik ★ Sound-Erzeugung
★ Amiga-Basic

2., überarbeitete und erweiterte Auflage

Markt&Technik Verlag AG

CIP-Kurztitelaufnahme der Deutschen Bibliothek

**Breuer, Markus**
Amiga-500-Buch : Hardware, Workbench 1.3 und Extras-Diskette, CLI–Einführung und Überblick, Amiga-Grafik, Sound-Erzeugung, Amiga-Basic / Markus Breuer. –
2., überarb. u. erw. Aufl. – Haar bei München : Markt-u.-Technik-Verl., 1989.
ISBN 3-89090-300-2

Die Informationen im vorliegenden Produkt werden ohne Rücksicht auf einen eventuellen Patentschutz veröffentlicht.
Warennamen werden ohne Gewährleistung der freien Verwendbarkeit benutzt.
Bei der Zusammenstellung von Texten und Abbildungen wurde mit größter Sorgfalt vorgegangen.
Trotzdem können Fehler nicht vollständig ausgeschlossen werden.
Verlag, Herausgeber und Autoren können für fehlerhafte Angaben und deren Folgen weder eine juristische Verantwortung noch irgendeine Haftung übernehmen.
Für Verbesserungsvorschläge und Hinweise auf Fehler sind Verlag und Herausgeber dankbar.

Alle Rechte vorbehalten, auch die der fotomechanischen Wiedergabe und der Speicherung in elektronischen Medien.
Die gewerbliche Nutzung der in diesem Produkt gezeigten Modelle und Arbeiten ist nicht zulässig.

Amiga 500 ist eine Produktbezeichnung der Commodore-Amiga Inc., USA
Graphicraft, Textcraft, Musicraft sind Produktbezeichnungen der Commodore-Amiga Inc., USA.
Amiga-BASIC ist eine Produktbezeichnung der Microsoft Inc., USA
Deluxe Paint ist eine Produktbezeichnung von Electronic Arts, USA
Deluxe Print ist eine Produktbezeichnung von Electronic Arts, USA
Deluxe Video ist eine Produktbezeichnung von Electronic Arts, USA

15 14 13 12 11 10 9 8 7 6 5 4 3
92 91 90

ISBN 3-89090-300-2

© 1989 by Markt & Technik Verlag Aktiengesellschaft,
Hans-Pinsel-Straße 2, D-8013 Haar bei München/West-Germany
Alle Rechte vorbehalten
Dieses Produkt wurde mit Desktop-Publishing-Programmen erstellt
und auf der Linotronic 300 belichtet
Einbandgestaltung: Grafikdesign Heinz Rauner
Druck: Kösel, Kempten
Printed in Germany

# Inhaltsverzeichnis


| | | | |
|---|---|---|---|
| | | **Vorwort** | **19** |
| **Kapitel** | **1** | **Vorhang auf: der Amiga 500!** | **23** |
| | 1.1 | Äußerlichkeiten | 24 |
| | 1.2 | Innere Werte | 26 |
| | 1.3 | Verbindungen zur Außenwelt | 27 |
| | 1.4 | Zubehör | 28 |
| | 1.4.1 | Zusätzliche Diskettenlaufwerke | 28 |
| | 1.4.2 | Zusätzlicher RAM-Speicher | 29 |
| | 1.4.3 | Bildschirme | 29 |
| | 1.4.4 | Drucker | 30 |
| | 1.4.5 | Sonstiges Zubehör | 31 |
| | 1.5 | Was Sie außer der Hardware noch bekommen | 32 |
| | 1.6 | Die Fähigkeiten des Amiga 500 | 33 |
| | 1.6.1 | Grafik-Fähigkeiten | 33 |
| | 1.6.2 | Sound-Fähigkeiten | 35 |
| | 1.6.3 | Sprach-Fähigkeiten | 36 |
| | 1.6.4 | Sound- und Grafik-Hardware | 36 |
| | 1.7 | Die Software des Amiga 500 | 37 |
| | 1.7.1 | Die Workbench | 37 |
| | 1.7.2 | Das CLI | 38 |
| | 1.7.3 | Multitasking | 38 |

| | | | |
|---|---|---|---|
| | 1.8 | Etwas Historie | 39 |
| **Kapitel** | **2** | **Erste Handgriffe auf der Workbench** | **43** |
| | 2.1 | Los geht's: Starten des Amiga 500 | 43 |
| | 2.2 | Wie Sie einen Neustart erzwingen | 47 |
| | 2.3 | Wozu die Workbench-Diskette gut ist | 47 |
| | 2.4 | Ein besonderes Nagetier: die Maus | 48 |
| | 2.4.1 | Funktionsweise der Maus | 49 |
| | 2.4.2 | Erste Maus-Spiele | 50 |
| | 2.4.3 | Die Tasten der Maus | 51 |
| | 2.5 | Was darf es sein: die Menüs des Amiga | 53 |
| | 2.5.1 | Wie Sie die Befehls-Menüs sichtbar machen | 53 |
| | 2.5.2 | Wie Sie einen Menübefehl erteilen | 54 |
| | 2.6 | Schöne Aussichten: Fenster | 56 |
| | 2.6.1 | Wie Sie Fenster verschieben | 56 |
| | 2.6.2 | Überlappende Fenster | 57 |
| | 2.6.3 | Fenster-Gadgets | 158 |
| | 2.7 | Sicher ist sicher: Kopien | 59 |
| | 2.7.1 | Vorbereitungen | 60 |
| | 2.7.2 | Kopieren mit einem Diskettenlaufwerk | 61 |
| | 2.7.3 | Kopieren mit zwei Diskettenlaufwerken | 64 |
| | 2.7.4 | Abschluß des Kopiervorgangs | 65 |
| | 2.7.5 | Kopieren Sie alle Originaldisketten! | 65 |
| **Kapitel** | **3** | **Objekte der Workbench** | **67** |
| | 3.1 | Allgemeine Eigenschaften von Piktogrammen | 67 |
| | 3.1.1 | Ein Piktogramm auswählen | 68 |
| | 3.1.2 | Ein Piktogramm verschieben | 68 |
| | 3.1.3 | Ein Piktogramm umbenennen | 68 |
| | 3.1.4 | Wie Fehlermeldungen aussehen | 70 |
| | 3.1.5 | Informationen zu einem Piktogramm erfragen | 72 |
| | 3.2 | Ordnung auf der Werkbank: Schubladen | 72 |
| | 3.2.1 | Welche Aufgaben Schubladen und Disketten haben | 72 |
| | 3.2.2 | Eine Schublade öffnen | 73 |
| | 3.2.3 | Eine Schublade schließen | 75 |
| | 3.2.4 | Der Aufbau eines Schubladenfensters | 76 |
| | 3.2.5 | Rollbalken | 76 |

3.2.6 Eine Schublade bewegen 79
3.2.7 Eine Schublade duplizieren 80
3.2.8 Das Info-Fenster einer Schublade 81
3.2.9 Eine neue Schublade erzeugen 82
3.2.10 Eine besondere Schublade: der Mülleimer 82

3.3 Disketten und Festplatten 84
3.3.1 Der Aufbau eines Diskettenfensters 84
3.3.2 Das Info-Fenster einer Diskette 85
3.3.3 Eine Diskette kopieren 86
3.3.4 Eine Diskette löschen oder initialisieren 87
3.3.5 Festplatten 88

3.4 Dateien, Werkzeuge und Projekte 89
3.4.1 Werkzeuge und Projekte 89
3.4.2 Operationen für Werkzeuge und Projekte 90

3.5 Weitere wichtige Workbench-Befehle 91
3.5.1 Löschen mit dem Discard-Befehl 91
3.5.2 Eine Schublade aufräumen 93
3.5.3 Einen Schnappschuß machen 93
3.5.4 Mehr als ein Piktogramm mit einem Befehl bearbeiten 93
3.5.5 Kopieren mit der RAM-Disk 94
3.5.6 Die »Tastatur-Maus« 95
3.5.7 Ein völlig verdecktes Fenster nach vorne holen 96

3.6 Die Menübefehle der Workbench 97

**Kapitel 4 Zusammenhänge: Werkzeuge und Projekte 99**

4.1 Werkzeuge ohne Projekte: Uhr und Rechner 99
4.1.1 Wie Sie ein Werkzeug starten 99
4.1.2 Das Fenster eines Werkzeugs 101
4.1.3 Die Menüleiste eines Werkzeugs 102
4.1.4 Der Rechner 104

4.2 Ein Werkzeug für Projekte: Notepad 106
4.2.1 Wie Sie den Notizblock öffnen 106
4.2.2 Wie Sie Text in den Notizblock eintragen 106
4.2.3 Wie Sie kleine Fehler korrigieren 108
4.2.4 Wie Sie im Text blättern 108
4.2.5 Wie Sie Textpassagen selektieren 109
4.2.6 Wie Sie Texte umstellen 111
4.2.7 Wie Sie Texte suchen und ersetzen 112
4.2.8 Wie Sie Texte gestalten 114
4.2.9 Wie Sie den Notizblock nach Ihrem Geschmack gestalten 117

| | | | |
|---|---|---|---|
| | 4.3 | Wichtige Eigenschaften von Projekten | 118 |
| | 4.3.1 | Das Project-Menü | 119 |
| | 4.3.2 | Das Infofenster eines Werkzeugs | 122 |
| | 4.3.4 | Ein Piktogramm vor versehentlichem Löschen schützen | 123 |
| | 4.3.3 | Ein Piktogramm mit einem Kommentar versehen | 123 |
| | 4.3.5 | Das Feld TOOL TYPES im Infofenster | 124 |
| | 4.3.5 | Das Infofenster schließen | 124 |
| | 4.3.6 | Die Parameter von Notepad | 125 |
| | 4.4 | Zusammenhänge zwischen Werkzeugen und Projekten | 126 |
| | 4.4.1 | Öffnen eines Projektes | 126 |
| | 4.4.2 | Das Infofenster eines Projektes | 127 |
| | 4.4.3 | Parameter eines Projektes | 128 |
| | 4.5 | Wie man sich zerreißt: Multitaskings | 128 |
| | 4.5.1 | Vorteile des Multitasking | 129 |
| | 4.5.2 | Das Problem Speicherplatz | 131 |
| | 4.5.3 | Das Problem Bildschirm | 132 |
| | 4.6 | Screens, die Superfenster | 132 |
| | 4.6.1 | Wie Sie Screens bewegen | 133 |
| | 4.6.2 | Wie Sie den »Screen-Stapel« umschichten | 134 |
| **Kapitel** | **5** | **Werkzeuge der Workbench** | **137** |
| | 5.1 | Preferences – das wichtigste Werkzeug überhaupt | 137 |
| | 5.1.1 | Preferences starten | 137 |
| | 5.1.2 | Wichtige Tasten im Preferences-Hauptfenster | 138 |
| | 5.1.3 | Ganz nach Geschmack: die Workbench-Farben | 139 |
| | 5.1.4 | Wem die Stunde schlägt: Einstellen von Uhrzeit und Datum | 140 |
| | 5.1.5 | Das Kleingedruckte: Einstellen von Schriftgröße und Interlace | 141 |
| | 5.1.6 | Nichts für Schwindelige: Zentrieren des Bildes | 142 |
| | 5.1.7 | Einstellen der automatischen Tastenwiederholung | 143 |
| | 5.1.8 | Einstellen der »Mausübersetzung« | 143 |
| | 5.1.9 | Ändern des Mauszeigers | 145 |
| | 5.1.10 | Einstellen der seriellen Schnittstelle | 146 |
| | 5.1.11 | Die Druckeranpassung | 148 |
| | 5.1.12 | Einstellungen für Grafikdruck 1 | 150 |
| | 5.1.13 | Einstellungen für Grafikdruck 2 | 152 |
| | 5.2 | Werkzeuge in der Utilities-Schublade | 157 |
| | 5.2.1 | Alte Bekannte: Clock, Notepad Calculator | 157 |
| | 5.2.2 | Einen Drucker installieren: InstallPrinter | 157 |
| | 5.2.3 | Den Amiga reden lassen: das Programm Say | 159 |
| | 5.2.4 | ClockPtr | 161 |
| | 5.2.5 | Graphic Dump | 162 |
| | 5.2.6 | More | 162 |

| | | | |
|---|---|---|---|
| | 5.2.7 | PrintFiles | 165 |
| | 5.2.8 | Cmd | 166 |
| | 5.3 | Werkzeuge in der System-Schublade | 166 |
| | 5.3.1 | Format und DiskCopy | 167 |
| | 5.3.2 | NoFastMem | 167 |
| | 5.3.3 | FastMemFirst | 168 |
| | 5.3.4 | MergeMem | 168 |
| | 5.3.5 | InitPrinter | 169 |
| | 5.3.6 | SetMap | 169 |
| | 5.3.7 | FixFonts | 171 |
| | 5.3.8 | CLI | 171 |
| | 5.3.9 | Shell | 172 |
| | 5.4 | Werkzeuge auf der Extras-Diskette | 172 |
| | 5.4.1 | Basic | 173 |
| | 5.4.2 | Für Verspielte: das Werkzeug IconEd | 174 |
| | 5.4.3 | IconMerge | 182 |
| | 5.4.4 | MEmacs | 184 |
| | 5.4.5 | KeyToy2000 | 184 |
| | 5.4.6 | Palette | 186 |
| | 5.4.7 | FreeMap | 187 |
| | 5.4.8 | PerfMon | 187 |
| | 5.4.9 | Fed | 188 |
| | 5.4.10 | Terminal | 188 |
| **Kapitel** | **6** | **Einführung in CLI und Shell** | **191** |
| | 6.1 | CLI, Shell und Amiga-DOS | 192 |
| | 6.1.1 | Was ist Amiga-DOS? | 192 |
| | 6.1.2 | Welche Aufgaben haben CLI und Shell? | 192 |
| | 6.1.3 | Unterschiede zwischen CLI und Shell | 193 |
| | 6.2 | Wie Sie die Shell starten | 194 |
| | 6.2.1 | Vorarbeiten für die Arbeit mit der Shell | 194 |
| | 6.2.2 | Erste Schritte in der Shell | 195 |
| | 6.2.3 | Wie Sie die Shell wieder verlassen | 197 |
| | 6.3 | Befehle eingeben und korrigieren | 198 |
| | 6.3.1 | Eingeben eines Befehls | 198 |
| | 6.3.2 | Editieren einer Befehlszeile | 199 |
| | 6.3.3 | Anhalten und Fortsetzen der Ausgabe | 200 |
| | 6.3.4 | Abbrechen eines Befehls | 200 |
| | 6.4 | Dateien und Directories | 201 |
| | 6.4.1 | Was eine Datei ist | 201 |
| | 6.4.2 | Was ein Directory ist | 202 |

| | | | |
|---|---|---|---|
| | 6.4.3 | Der Dateienbaum | 202 |
| | 6.4.4 | Pfadnamen – Wege zu Dateien | 203 |
| | 6.4.5 | Das aktuelle Directory | 204 |
| | 6.4.6 | Die aktuelle Diskette | 205 |
| | 6.4.7 | Laufwerksnamen | 205 |
| | 6.5 | Arbeiten mit der Shell | 206 |
| | 6.5.1 | Directories betrachten | 206 |
| | 6.5.2 | Programme starten | 208 |
| | 6.5.3 | Disketten kopieren | 208 |
| | 6.5.4 | Eine Datei kopieren | 210 |
| | 6.5.5 | Eine Datei umbenennen | 211 |
| | 6.5.6 | Eine Datei löschen | 211 |
| | 6.5.7 | Den Inhalt einer Datei auf dem Bildschirm betrachten | 212 |
| | 6.5.8 | Den DIR-Befehl »interaktiv« verwenden | 213 |
| | 6.5.9 | Parameter zu Befehlen erfragen | 214 |
| | 6.5.10 | Mehrere Dateien mit einem Befehl bearbeiten – Muster | 215 |
| | 6.6 | Directories und Geräte | 219 |
| | 6.6.1 | Das aktuelle Dateiverzeichnis wechseln | 219 |
| | 6.6.2 | Auf und ab im Dateienbaum | 220 |
| | 6.6.3 | Neue Directories erzeugen und löschen | 222 |
| | 6.6.4 | Geräte – nicht nur Wurzeln der Dateienbäume | 223 |
| | 6.7 | Ein-/Ausgabe-Umleitung | 224 |
| | 6.7.1 | Ein- und Ausgaben eines Shell-Kommandos umleiten | 224 |
| | 6.7.2 | Ein-/Ausgabe-Umleitung auf Geräte und Dateien | 225 |
| | 6.8 | Alternative Wege zu Programmen und Dateien | 226 |
| | 6.8.2 | Der Suchpfad für Programme | 227 |
| | 6.8.1 | »Logische« Geräte | 229 |
| | 6.9 | Wie Sie parallel mit mehreren Shells arbeiten | 231 |
| | 6.9.1 | Mehrere Shell-Fenster | 232 |
| | 6.9.2 | Hintergrund-Befehle | 234 |
| | 6.10 | Zusammenhänge zwischen Shell und Workbench | 235 |
| **Kapitel** | **7** | **Die Amiga-Shell – Nachschlageteil** | **237** |
| | 7.1 | Überblick über die Shell-Befehle | 237 |
| | 7.2 | Format eines Shell-Befehls | 238 |
| | 7.3 | Die wichtigsten Shell-Befehle: | 240 |
| | | ADDBUFFERS | 243 |
| | | ALIAS | 243 |
| | | ASK | 244 |
| | | ASSIGN | 245 |

AVAIL 246
BINDDRIVERS 247
BREAK 248
CD 248
CHANGETASKPRI 249
CLOCK 250
CLOCKPTR 251
CMD 252
COPY 253
DATE 255
DELETE 256
DIR 257
DISKCHANGE 259
DISKCOPY 259
DISKDOCTOR 261
ECHO 262
ED 263
EDIT 263
ELSE 264
ENDCLI 264
ENDIF 265
ENDSHELL 265
ENDSKIP 265
EVAL 266
EXECUTE 267
FAILAT 268
FASTMEMFIRST 268
FAULT 269
FF 269
FILENOTE 270
FORMAT 271
GETENV 272
GRAPHICDUMP 273
ICONX 274
IF 275
INFO 277
INITPRINTER 278
INSTALL 278
JOIN 279
LAB 280
LIST 280
LOADWB 282
LOCK 283
MAKEDIR 283
MEMACS 284
MERGEMEM 285
MORE 285
MOUNT 286

NEWCLI 287
NEWSHELL 287
NOFASTMEM 288
PATH 289
PREFERENCES 290
PRINTFILES 290
PROMPT 291
PROTECT 291
QUIT 293
RELABEL 293
REMRAD 294
RENAME 294
RESIDENT 295
RUN 296
SAY 298
SEARCH 299
SETCLOCK 300
SETDATE 301
SETENV 301
SETPATCH 302
SKIP 303
SORT 303
STACK 304
STATUS 305
TYPE 306
WAIT 306
WHICH 307
WHY 308

7.4 Geräte 309
7.5 Logische Laufwerke 313
7.6 Überblick 315

**Kapitel 8 Shell für Fortgeschrittene 319**

8.1 Arbeiten im Shell-Fenster 319
8.1.1 Das Gedächtnis der Shell 320
8.1.2 Korrigieren einer Befehlszeile 321

8.2 Textbearbeitung in der Shell 322
8.2.1 Überblick 322
8.2.2 Wie Sie das Werkzeug Notepad als Editor »mißbrauchen« 323

8.3 Der mausgesteuerte Editor MEMACS 325
8.3.1 Die Geschichte von MEMACS 325
8.3.2 MEMACS aufrufen und verlassen 326
8.3.3 Der Bildschirmaufbau 327

| | | |
|---|---|---|
| 8.3.4 | Texte abspeichern und einlesen | 328 |
| 8.3.5 | Im Text bewegen | 329 |
| 8.3.6 | Text eingeben und bearbeiten | 330 |
| 8.3.7 | Mit Textblöcken arbeiten | 331 |
| 8.3.8 | Texte suchen und ersetzen | 331 |
| 8.3.9 | Mit mehreren Puffern und Fenstern arbeiten | 332 |
| 8.3.10 | Befehlsüberblick | 334 |
| 8.4 | Kommandofolgen | 344 |
| 8.4.1 | Wozu Kommandofolgen? | 344 |
| 8.4.2 | Eine einfache Kommandofolge | 345 |
| 8.4.3 | Kommandofolgen ohne EXECUTE aufrufen | 347 |
| 8.4.4 | Kommandofolgen mit Parametern | 347 |
| 8.4.5 | Spezielle Punktbefehle | 351 |
| 8.4.6 | Die IF-ELSE-ENDIF-Anweisung | 352 |
| 8.4.7 | Mögliche Bedingungen für den IF-Befehl | 354 |
| 8.4.8 | Die Sprunganweisungen SKIP und QUIT | 356 |
| 8.4.9 | Wie Sie Fehler in Kommandofolgen abfangen | 357 |
| 8.4.10 | Schleifen | 358 |
| 8.4.11 | Umgebungsvariablen | 359 |
| 8.4.12 | Kommandofolgen automatisch erstellen lassen | 361 |
| 8.4.13 | Ein paar besondere Kommandofolgen | 363 |
| 8.4.14 | Ändern der Start-Kommandofolgen | 367 |
| 8.5 | Ein ganz spezielles Gerät: die RAM-Disk | 369 |
| 8.5.1 | Was ist eine RAM-Disk? | 370 |
| 8.5.2 | Die RAM-Disk RAM: | 370 |
| 8.5.3 | Die RAM-Disk RAD: | 371 |
| 8.5.4 | Die RAM-Disk RAD: benutzen | 371 |
| 8.5.5 | Die Größe von RAD: ändern | 373 |
| 8.5.6 | Eine RAM-Disk richtig benutzen | 373 |
| 8.5.7 | Programme und Dateien in der RAM-Disk | 375 |
| 8.5.8 | Kombination von RAM-Disk und Startup-Sequence | 376 |
| 8.6 | Tips für Shell-Anwender | 377 |
| 8.6.1 | Wie Sie auf der Workbench-Diskette Platz schaffen | 377 |
| 8.6.2 | Die Mini-Workbench für Sparsame | 384 |
| 8.6.3 | Die RAM-Workbench | 386 |
| 8.6.4 | Schriften löschen und zusätzliche Schriftarten installieren | 388 |
| 8.6.5 | Piktogramme vom Shell-Fenster aus gesehen | 389 |
| 8.6.6 | Kommandofolgen von der Workbench aus starten | 391 |
| **Kapitel 9** | **Ausbaumöglichkeiten** | **393** |
| 9.1 | Ausbau des Amiga 500 – ein Überblick | 393 |
| 9.1.1 | Jeder Computer ist erweiterbar | 393 |
| 9.1.2 | Externe Erweiterungen | 394 |

| | | | |
|---|---|---|---|
| | 9.1.3 | Interne Erweiterungen | 394 |
| | 9.1.4 | Steckkarten | 394 |
| | 9.1.5 | Die Erweiterungsschnittstellen des Amiga 500 | 395 |
| | 9.2 | Bildschirme für den Amiga 500 | 396 |
| | 9.2.1 | Wie das Bild aus dem Amiga kommt | 396 |
| | 9.2.2 | Der Standard: Modell 1084 S | 396 |
| | 9.2.3 | Für Sparsame: Anschluß eines Fernsehers | 397 |
| | 9.2.4 | Andere RGB-Monitore | 398 |
| | 9.2.5 | Eine Klasse für sich: MultiSync-Monitore | 398 |
| | 9.3 | Schwarz – oder bunt – auf weiß: Drucker | 399 |
| | 9.3.1 | Die verschiedenen Druckprinzipien | 399 |
| | 9.3.2 | Text und Bild | 401 |
| | 9.3.3 | Schwarzweiß oder bunt? | 401 |
| | 9.4 | Diskettenlaufwerke und Festplatten | 402 |
| | 9.4.1 | Welche Alternativen zur Wahl stehen | 402 |
| | 9.4.2 | Externe Diskettenlaufwerke | 403 |
| | 9.4.3 | Festplatten | 403 |
| | 9.5 | Speichererweiterungen | 404 |
| | 9.5.1 | Die einfachste Speichererweiterung A 501 | 404 |
| | 9.5.2 | Externe Speicherkarten | 405 |
| | 9.6 | Erweiterungen für Video-Fans | 406 |
| | 9.6.1 | Der Mediencomputer und das Fernsehen | 406 |
| | 9.6.2 | Die Verbindung zum Videorekorder | 407 |
| | 9.6.3 | Video-Digitizer | 408 |
| | 9.6.4 | GenLock – Bilder im Gleichschritt | 409 |
| | 9.7 | Erweiterungen für Musiker | 410 |
| | 9.7.1 | Der Amiga als Kompositionshilfe | 410 |
| | 9.7.2 | Der Amiga als Instrument | 410 |
| | 9.7.3 | Der MIDI-Standard | 411 |
| | 9.7.4 | Sound-Digitizer | 412 |
| | 9.8 | Beschleuniger | 413 |
| **Kapitel** | **10** | **Festplatten für den Amiga 500** | **415** |
| | 10.1 | Grundsätzliches zum Thema Festplatte | 415 |
| | 10.1.1 | Wozu Festplatten gut sind | 417 |
| | 10.1.2 | Aus welchen Teilen eine Festplatte besteht | 418 |
| | 10.1.3 | Festplatten und Kontroller | 419 |
| | 10.1.4 | Autoboot oder nicht | 420 |
| | 10.1.5 | Das FastFileSystem | 421 |

10.1.6 Festplatten-Partitionen 422
10.1.7 Welche Festplatte für welchen Zweck? 423
10.1.8 Was Sie bei der Arbeit mit einer Festplatte beachten müssen 423

10.2 Installation und Inbetriebname einer Amiga-Festplatte 425
10.2.1 Installation von Controller und Festplatte 425
10.2.2 Vorbereiten der Festplatte 426
10.2.3 Festplatten-Partitionen 428
10.2.4 Die Mountlist 428
10.2.5 Die Festplatte formatieren 431
10.2.6 Die Workbench auf die Festplatte bringen 432
10.2.7 Grundsätzliches zum Start mit der Festplatte 433
10.2.8 Start mit nicht autobootfähige Festplatten 434
10.2.9 Start mit autobootfähigen Festplatten 436
10.2.10 Festplattenstart mit Hilfe der RAM-Disk RAD: 437

10.3 Arbeiten mit einer Festplatte 439
10.3.1 Eine Alternative zur Installation der Workbench 440
10.3.2 Schubladenanordnung auf einer Festplatte 441
10.3.3 Sicherungskopien anlegen 442
10.3.4 Spezielle Software und Hardware zum Sichern von Festplatten 442

10.4 Programme auf die Festplatte bringen 444
10.4.1 Mögliche Probleme 444
10.4.2 Vorsichtsmaßnahmen 444
10.4.3 Die Verbindung zwischen Programmen und Projekten erhalten 445
10.4.4 Ein Traum: Selbstinstallierende Programme 446

**Kapitel 11 Chips und Tricks 447**

11.1 Grundlagen der Amiga-Grafik 447
11.1.1 Das digitale Prinzip 448
11.1.2 Pointilismus und digitale Farben 448
11.1.3 Das Prinzip der »bitmapped« Grafik 449
11.1.4 Farbe ins Bild 450
11.1.5 Wie Farben auf den Bildschirm kommen 451
11.1.6 Malen nach Zahlen 453

11.2 Die Grafikbetriebsarten des Amiga 454
11.2.1 Bildauflösung 454
11.2.2 Die Farbpalette 456
11.2.3 Nachträgliches Ändern der Palette 458
11.2.4 Der Interlace-Modus 459

11.3 Ungewöhnliche grafische Fähigkeiten des Amiga 500 461
11.3.1 HAM: Mehr Farben aus dem Computer 461
11.3.2 Grafiken, die größer als der Bildschirm sind 462

11.3.3 Zwei Bilder übereinander: der Dual-Playfield-Modus 463
11.3.4 Sprites & BOBs 464
11.3.5 AnimObjects 466
11.3.6 Kollisionserkennung 466

11.4 Klangerzeugung 467
11.4.1 Töne, wie sie der Amiga sieht 467
11.4.2 »Vollsynthetische« Klänge 469
11.4.3 Die Hüllkurve eines Tons 470
11.4.4 Klangerzeugung mit dem Amiga 472
11.4.5 Spezialeffekte 473
11.4.6 Sprachsynthese 474
11.4.7 MIDI 475

11.5 Eine Handvoll Chips 476
11.5.1 Die CPU – Herr im Hause Amiga 476
11.5.2 Teamwork 477
11.5.3 Paula die musikalische 479
11.5.4 Denise das Grafik-Wunder 480
11.5.5 Agnus der Rechenkünstler 481
11.5.6 Chip-RAM und FastMem 484
11.5.7 Begriffsver- und -entwirrungen 485

**Kapitel 12 Die Programmierung des Amiga 500 487**

12.1 Architektur der Amiga-Software 487
12.1.1 Grundriß der Amiga-Systemsoftware 488
12.1.2 Exec 489
12.1.3 Bibliotheken 491
12.1.4 Geräte 491
12.1.5 Die Grafik-Bibliothek 491
12.1.6 Intuition 492
12.1.7 Amiga-DOS 493

12.2 Grundlagen der Programmierung 493
12.2.1 Sprachen 494
12.2.2 Interpreter 494
12.2.3 Compiler 495
12.2.4 Vergleich von Compiler und Interpreter 495

12.3 Programmiersprachen für den Amiga 500 495
12.3.1 Assembler und Maschinencode 496
12.3.2 Basic 496
12.3.3 C 496
12.3.4 Pascal und Modula 497

**Kapitel 13 Amiga-Basic 499**

13.1 Basic 500

13.2 Der Basic-Dialekt »Amiga-Basic« 501
13.2.1 Microsoft-Basic 501
13.2.2 Moderne Features von Microsoft-Basic 502

13.3 Amiga-Basic und der Amiga 500 503
13.3.1 Die Bedienung von Amiga-Basic 503
13.3.2 Unterstützung der Amiga-Fähigkeiten 504

13.4 Umgang mit Amiga-Basic 507
13.4.1 Amiga-Basic starten 507
13.4.2 Programme eingeben und editieren 508
13.4.3 Programme starten und wieder anhalten 509
13.4.4 Fehler in Programmen suchen 509
13.4.5 Programme abspeichern 510
13.4.6 Programme einlesen 511

13.5 Grafik mit Amiga-Basic 512
13.5.1 Pixel 512
13.5.2 Linien 513
13.5.3 Mondrian 513
13.5.4 Eigene Fenster und Screens 514

13.6 Ereignisreiche Programme 516
13.6.1 Das einfachste Malprogramm der Welt 516
13.6.2 Die ON MOUSE GOSUB-Anweisung 516
13.6.3 Eigene Menüs in Basic-Programmen verwenden 518

13.7 Klangerzeugung mit Amiga-Basic 520
13.7.1 Ein paar schräge Töne 520
13.7.2 Sprachsynthese 522

13.8 Libraries: Amiga-Basic ohne Grenzen 523

**Anhang A Fehlercodes und ihre Bedeutung 525**
**Anhang B ANSI-Escape-Sequenzen 531**

**Stichwortverzeichnis 533**

**Hinweis auf weitere Markt&Technik-Produkte 543**

# Vorwort

Der Commodore Amiga 500 ist in vieler Hinsicht ein ganz besonderer Computer. Er beeindruckt zunächst einmal vor allem durch zwei Fähigkeiten: eine qualitativ hochwertige, ungewöhnlich vielseitige und schnelle Farbgrafik und die Möglichkeit, mit mehreren Programmen gleichzeitig zu arbeiten (Multitasking). Beides erscheint auf den ersten Blick vielleicht als mehr oder weniger überflüssige Spielerei. Es wird aber zu einer nahezu unentbehrlichen und nur noch ungern wieder aufgegebenen Annehmlichkeit, wenn man erst einmal eine Weile damit gearbeitet hat. Lassen Sie sich nicht von anders lautenden Meinungen beirren! Grafik und Multitasking sind keine grundsätzlich notwendigen Bestandteile der Computer-Bedienung – schließlich sind die Computer Jahrzehnte ohne sie ausgekommen. Beides trägt aber ungemein zu einer einfach zu erlernenden, schnellen und angenehmen Bedienung eines Computers bei!

Im Amiga 500 werden auf allen Gebieten – sowohl bei der Hardware wie auch bei der Software – die allerneuesten Technologien eingesetzt. Für Sie als (potentiellen) Käufer dieses Computers bedeutet dies, daß Sie ungemein leistungsfähige Hardware zu einem vergleichsweise geringen Preis bekommen. (In der Computerbranche dürfte das allerdings nicht mehr allzu beeindruckend wirken – schon nächstes Jahr wird sicherlich ein noch leistungsfähigerer Computer für noch weniger Geld auf den Markt kommen.)

Die Hauptaufgabe dieses Buches ist es aber nicht, Sie mit den phantastischen Fähigkeiten des Amiga zu beeindrucken – selbst wenn sich das bisher vielleicht so angehört haben sollte. Vielmehr werden der Amiga und die Programme, die Sie brauchen, um ihn nutzbringend einzusetzen, exakt beschrieben. Das soll Ihnen im täglichen Umgang mit dem Gerät und diesen Programmen helfen und Ihnen zugleich eine Entscheidungshilfe und Vergleichsmöglichkeiten mit anderen Computern geben, wenn Sie erst planen, sich (noch) einen anzuschaffen. (Direkte Vergleiche mit anderen Computern werden aber vermieden. Sie würden zu unfair ausfallen...)

## Überblick über das Buch

Dieses Buch läßt sich grob in drei Teile gliedern, die auch im Inhalt immer wieder angesprochen werden. In jedem dieser Teile geht es ein wenig »tiefer« in die Materie hinein. Es

ist deshalb nicht unbedingt sinnvoll oder gar nötig, sämtliche Kapitel in einem Zug durchzuarbeiten. Sie können auch immer erst dann mit einem neuen Kapitel beginnen, wenn Sie bei Ihrer Arbeit auf einen neuen Aspekt gestoßen sind, über den Sie mehr erfahren möchten. Das ist aber Geschmackssache! Wie Sie dieses Buch benutzen, bleibt letztlich Ihnen überlassen – hoffentlich nicht nur als Stütze für einen wackligen Tisch.

Im ersten Teil dieses Buches werden die wesentlichen Grundzüge der Bedienung des Amiga beschrieben. Sie sollten in diesem Teil mit der Philosophie des Amiga vertraut werden – seiner stark grafisch orientierten Bedienung, der Anwendung der Maus und aller wichtigen Handgriffe, die Sie später tagtäglich benötigen werden. Hierzu wird das Programm, das den Mittler zwischen Ihnen und der Hardware des Amiga spielt, die sogenannte Workbench (zu deutsch: Werkbank), ausführlich beschrieben.

Der zweite Buch-Teil beschäftigt sich mit einer Art, den Amiga 500 zu bedienen, die nicht ganz so bildlich und intuitiv zu begreifen ist wie die Workbench: dem sogenannten CLI. Diese zweite »Benutzerschnittstelle« des Amiga hat trotz gewisser Nachteile und Schwierigkeiten, die sie dem Anfänger bereitet, auch einige Vorteile. Diese kommen vor allem dem intensiven Anwender zugute und denjenigen unter den Lesern, die mit dem kleinen Nagetier namens »Maus« auf Kriegsfuß stehen.

Der dritte Teil dieses Buches schließlich enthält vor allem Informationen für diejenigen Leser, die es immer ganz genau wissen wollen. Die Hardware des Amiga (und was man damit machen kann) wird beschrieben und ein erster Eindruck davon vermittelt, welche Möglichkeiten der Amiga dem Programmierer bietet. Auch die »Kontakte zur Außenwelt«, Anschlüsse für Drucker und andere Zusatzgeräte, werden hier beschrieben. Die Programmierer unter den Lesern – und solche, die es werden wollen – lernen schließlich anhand von ein paar kleinen Beispielprogrammen in Amiga-Basic, was man alles schon mit dieser einfachen und leicht erlernbaren Sprache anstellen kann, die jedem Amiga beim Kauf beiliegt.

## Was dieses Buch sein will – und was nicht

Ein einführendes Buch über einen Computer stellt zwangsläufig immer einen Kompromiß dar. Es muß ganz verschiedene Leserkreise, mit und ohne Computer-Vorwissen, mit und ohne Interesse für die Hardware und Programmierung, und mit und ohne den Computer, um den es geht, ansprechen. Ich habe versucht, für jeden dieser potentiellen Leserkreise etwas einzubringen, mußte dafür aber zwangsläufig in Kauf nehmen, diejenigen, die etwas mehr zu einem bestimmten Aspekt wissen wollen, zu enttäuschen. Leser, die bislang überhaupt nichts über Computer wissen, werden deshalb in einigen Kapiteln im dritten Teil des Buches manchmal Schwierigkeiten haben, dem Text zu folgen.

Für diese beiden enttäuschten Gruppen ist aber Rettung nahe. Es gibt sowohl über die Computertechnik »an sich« einige sehr gute einführende Werke. Aber auch der an bestimmten Details interessierte Leser wird in anderen Büchern zu den Computern der Amiga-Familie, die speziell auf diese Details – wie Programmierung, Hardware und Grafik – eingehen, finden können, was er sucht. Diese Bücher sind bereits erhältlich und es erscheinen auch laufend neue. Der Markt&Technik Verlag bietet zum Beispiel schon eine breite Palette von Büchern und Zeitschriften zum Amiga an.

In diesem Buch werden Sie aber zunächst einmal einen »Rundumschlag« finden, der alle Aspekte des Amiga 500 zumindest streift. Ich hoffe, damit nicht nur dem frischgebackenen Amiga-500-Besitzer helfen zu können, sondern vielleicht auch dem einen oder anderen noch Unentschlossenen dabei zu helfen, eine Kaufentscheidung für oder gegen den Amiga 500 zu treffen.

Manche Kapitel beschäftigen sich aber trotz dieses überblickartigen Charakters recht detailliert mit einigen besonders wichtigen Aspekten des Amiga und sind für den täglichen Einsatz bei der Arbeit am Gerät gedacht. So werden Sie in den folgenden Kapiteln unter anderem auch vollständige »Handbücher« für den Umgang mit der Workbench und dem CLI finden, die kaum Fragen offenlassen dürften. Ich habe dabei versucht, vor allem die Themenkreise etwas mehr auszuarbeiten, die im Original-Handbuch, das zusammen mit dem Amiga 500 ausgeliefert wird, etwas stiefmütterlich behandelt werden.

## Zur Vorgeschichte dieses Buches

Zum Abschluß dieses Vorwortes muß ich fairerweise aber auch noch erwähnen, daß das vorliegende Buch nicht hundertprozentig neu ist. Wer schon *Das Amiga-Handbuch* (M&T 90223) besitzt oder kennt, dem werden viele Passagen bekannt vorkommen. Die einzelnen Kapitel dieses Buches stellen fast alle Überarbeitungen der Kapitel von *Das Amiga-Handbuch* dar oder bauen auf diesem Buch auf. *Das Amiga-Handbuch* beschäftigt sich ausschließlich mit dem Amiga 1000 und erschien zu einem Zeitpunkt (im Frühjahr '86), als es den Amiga in Deutschland noch nicht einmal offiziell zu kaufen gab.

Seitdem hat sich einiges an der Amiga-Software geändert (unter anderem wurden die ersten Betriebssystemversionen 1.0 und 1.1 durch die weiterentwickelte Workbench 1.2 ersetzt). Diese Änderungen wurden natürlich berücksichtigt. Darüber hinaus ist der Amiga 500 natürlich auch in manch anderer Hinsicht ein anderer Computer als der Amiga 1000, wodurch sich weitere Änderungen ergaben. Und als ich gerade dabei war, die sich aus diesen beiden Gründen ergebenden Änderungen auszuführen, habe ich gleich einige Kapitel völlig neu geschrieben und an anderen Stellen die Reihenfolge der Darstellung geändert, wo ich mit meinem früheren »Schaffen« unzufrieden war.

Die vorläufig letzte Überarbeitung dieses Buches wurde im Frühjahr 1989 nötig, als Commodore eine weitere neue Version der Betriebssystemsoftware (Workbench 1.3) auf den Markt brachte. Diese bringt soviele Vorteile (und vor allem Neuerungen) gegenüber den früheren Software-Versionen, daß eine Überarbeitung des Manuskripts angebracht erschien. Auch bei dieser Überarbeitung sind einige Kapitel umgeschrieben worden, um vor allem dem »Computer-Neuling« den Zugang zu diesem phantastischen Computer so leicht wie möglich zu machen.

## Danksagungen

Welches Vorwort wäre komplett ohne Danksagungen? Dieses jedenfalls nicht. Ich möchte an dieser Stelle einer ganzen Reihe von Menschen danken, ohne die dieses Buch nicht zustande gekommen wäre – oder aber nicht in dieser Form.

Für einen nie versiegenden Strom an Informationen und vielfältige Unterstützung beim Kampf mit den Herstellern sowie die Versorgung mit Hard- und Software möchte ich Frau Baumann, Frau Scholz, Herrn Peters und Frau Jurk vom Markt&Technik Buchverlag danken. Speziell Frau Baumann möchte ich dafür danken, daß sie mich damals (im Spätsommer 1985) mit dem ersten Buchprojekt zum Amiga betraute. Ihre regelmäßigen Telefonanrufe hielten mich damals, trotz widriger Umstände, bei der Stange. Und auch im Laufe der folgenden Jahre bewies sie viel Geduld mit mir, wenn wieder und wieder der Abgabetermin für ein Manuskript nahte und der Herr Breuer einmal mehr in Verzug geriet.

Und last but not least auch ein Wort des Dankes an meine Freundin Maria. Sie hat seit dem Entstehen der ersten Fassung dieses Buches (im Herbst 1985 über Amiga 1000) meine Arbeit mit viel Geduld und Nachsicht begleitet und dürfte nun wissen, was das Wort »Computer-Witwe« bedeutet!

Es ist schwer, die Begeisterung in Worte zu fassen, die man erlebt, wenn man die ersten Stunden und Tage mit einem Amiga arbeitet. Ich hoffe, daß trotzdem ein wenig dieser Begeisterung in den folgenden Kapiteln wiederzufinden ist.

Markus Breuer

# 1 Vorhang auf: Der Amiga 500!

Der Commodore Amiga ist ein Vertreter einer neuen Computer-Generation. Diese neue Generation legt – im Vergleich zu ihren Vorgängern – wesentlich mehr Wert auf die einfache Bedienung des Computers als auf die »nackten Daten«. Man könnte einen Vergleich mit Autos ziehen und sagen, daß die Polsterung der Sitze, die Qualität der Stereo-Anlage und auch die Bremsen an erster Stelle kommen und irgendwann viel später die PS- (alias kW-) Leistung des Motors. Fairerweise muß man allerdings zugeben, daß eine solche Umorientierung bei den Computern erst ab dem Moment möglich ist, in dem nach Erledigung der notwendigen Rechen- und Steuerungsaufgaben noch genügend Computerleistung für solche Komfort-Aspekte »übrigbleibt«. Beim Amiga ist das der Fall. Er ist leistungsfähig genug, Komfort **und** Geschwindigkeit bieten zu können.

Deshalb darf man aber die »nackten Daten« eines solchen Computers nicht vergessen, denn erst sie machen den Komfort möglich. In diesem Kapitel erhalten Sie deshalb zunächst einen kleinen Überblick über die Leistungsdaten des Amiga 500 und eher »äußerlichen Aspekte« seiner Erscheinung. Dies beginnt bei der Hardware und wird dann mit der Software fortgesetzt, die Sie erhalten, wenn Sie einen Amiga 500 kaufen. Dieser Überblick führt Sie praktisch im Laufschritt durch die Welt des Amiga 500. Er wird Ihnen deshalb zwangsläufig etwas oberflächlich vorkommen. Aber keine Angst: Fast alle angeschnittenen Themen werden in den folgenden Kapiteln noch im Detail erläutert.

Das Wort »Amiga« ist spanisch und bedeutet in etwa »Freundin«. Genau das sollte dieser Computer im Idealfall auch für Sie werden – natürlich nur im platonischen Sinn. (Wie auch im wirklichen Leben, kann Ihnen diese neue Freundin übrigens auch viel Ärger bereiten.)

Betrachten wir aber zunächst einmal ganz unvoreingenommen ihr äußeres Erscheinungsbild. Man soll zwar nicht zuviel auf Äußerlichkeiten geben; vielleicht ist es aber trotzdem ganz interessant für Sie. Wenn Sie bereits längere Zeit einen Amiga besitzen, können Sie dabei wahrscheinlich wenig Neues lernen. Sie sollten die folgenden Abschnitte trotzdem überfliegen, weil einige Begriffe eingeführt werden, die auch im restlichen Buch immer wieder gebraucht werden.

# 1.1 Äußerlichkeiten

Äußerlich macht der Amiga 500 einen recht attraktiven Eindruck. Das »Basispaket«, das natürlich ausgebaut werden kann (siehe unten), besteht aus drei Komponenten:

- Einer Systemeinheit mit eingebautem Diskettenlaufwerk
- Einer Stromversorgung (Netzteil)
- Einer Maus

Optional kann (und sollte!) man auch noch ein zusätzliches Diskettenlaufwerk erwerben, das außen an die Haupteinheit angeschlossen wird. Unbedingt notwendig zur Arbeit mit dem Amiga 500 ist ferner ein Monitor oder Fernsehgerät. Das folgende Bild zeigt eine Arbeitskonfiguration aus Amiga 500, Maus und Monitor.




***Bild 1.1:*** *Der Amiga 500*

Die einzelnen Komponenten dieses Computersystems sind über Kabel miteinander verbunden. Die Maus wird zum Beispiel über ein langes Kabel rechts an der Rückseite der Systemeinheit angeschlossen. Der Bildschirm steht normalerweise hinter der Haupteinheit (sie ist nicht groß genug, als daß er darauf Platz hätte) und ist über ein Kabel mit ihr verbunden. Besonders wichtig ist auch noch das Netzteil, das Sie am besten hinter dem Bildschirm oder auf dem Fußboden plazieren. Falls Sie sich für den Fußboden entscheiden – wo es am wenigsten stört – müssen Sie sich allerdings bei jedem Einschalten des Amiga

bücken. Ein genialer Commodore-Ingenieur hat nämlich den Netzschalter in das Netzteil verlegt. Am Amiga 500 selbst gibt es keine Möglichkeit, ihn aus- oder einzuschalten.

Die Länge der verschiedenen Kabel läßt eine fast beliebige Positionierung der Komponenten zu. Ob Sie die Maus zum Beispiel lieber rechts oder links neben der Tastatur liegen haben wollen, bleibt Ihnen überlassen. Dicht neben der Tastatur sollte sie aber auf alle Fälle liegen (mehr dazu weiter unten).

Die Tastatur des Amiga ist – anders als Sie es vielleicht schon bei anderen Computern gesehen haben – direkt in die Systemeinheit eingebaut, aber trotzdem angenehm flach. Sie enthält neben dem normalen Satz Schreibmaschinentasten einen separaten Zehnerblock für die schnelle und komfortable Zahleneingabe, einen umgekehrt T-förmigen Block mit Tasten zur Steuerung einer Schreibmarke (Cursor) und eine Reihe von zehn Funktionstasten am oberen Rand. Mit diesen Funktionstasten können Sie innerhalb von Programmen oft bestimmte wichtige Funktionen mit nur einem Tastendruck auslösen.

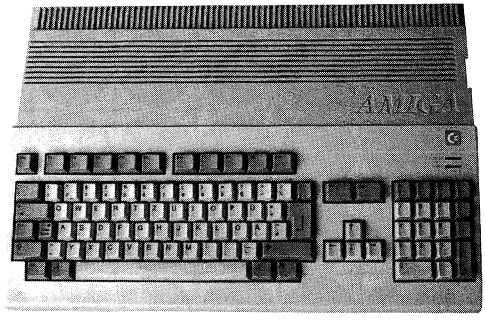


***Bild 1.2:*** *Die Tastatur des Amiga*

Links und rechts neben der Leertaste liegen zwei besondere Tasten. Sie tragen das Amiga-Markenzeichen; links als fetten Buchstaben [*A*] und rechts in Konturschrift [*A*]. Es sind (in Kombination mit anderen Tasten) spezielle Funktionstasten, wie sie heute auch bei vielen anderen Rechnern zu finden sind (zum Beispiel Apple II, Apple Macintosh und Atari ST). Sie werden in den folgenden Kapiteln noch mehr über diese Tasten erfahren.

Direkt neben der Amiga- und Commodore-Taste liegt auf beiden Seiten noch eine sog. »Alt-Taste«. Hält man die Alt-Taste fest und drückt dann einen »normalen« Buchstaben, so erhält man besondere Zeichen, wie zum Beispiel á, œ, Σ und ¿. Viele Buchstabentasten sind mit einem solchen alternativen Symbol versehen, weshalb man den Amiga wohl zu Recht als einen internationalen Computer bezeichnen kann. Fast jeder Buchstabe, den man sich vorstellen kann, steht mit der Alt-Taste zu Verfügung.

Der Anschlag der Tasten ist leicht und recht angenehm, sowohl für diejenigen, die nach dem Adler-System (Ziel suchen und dann hinabstoßen) die Buchstaben finden, als auch für die

Zehn-Finger-Tipper. Der Tastenhub ist weder extrem lang noch besonders gering. Ein Buchstabe gilt allerdings schon als getippt, bevor man beim Anschlag ankommt.

Wichtigstes Eingabemedium des Amiga ist neben der Tastatur die sogenannte »Maus«. Die mechanische Maus des Amiga (es gibt auch optische Mäuse) besitzt eine elegante »Stromlinienform« und liegt gut in der Hand. An der Oberseite der Maus befinden sich zwei Knöpfe. Ein Drücken dieser Knöpfe wirkt in bestimmten Situationen als Befehl an den Computer. An der Unterseite der Maus ist eine kleine Öffnung zu sehen, aus der eine Gummikugel etwas herausschaut. Diese Gummikugel dreht sich, wenn Sie die Maus über eine glatte Oberfläche schieben. Diese Drehung wird über das angeschlossene Kabel dem Computer mitgeteilt und kann für Steuerungszwecke am Bildschirm verwendet werden. (Mehr darüber im nächsten Kapitel.)

Die Kugel (und damit die Maus) rollt im Allgemeinen sehr flüssig, und falls Sie das einmal nicht mehr tut, ist sie mit Sicherheit verschmutzt und sollte gereinigt werden. Sie können die kleine Kugel dazu aus dem Gehäuse nehmen und sie mit einem weichen Tuch abwischen. Wenn Sie die Kugel herausnehmen, werden Sie drei kleine Rädchen im Innern des Maus-Gehäuses entdecken. Auch diese Rädchen müssen von Zeit zu Zeit gereinigt werden. Verwenden Sie dazu am besten ein Wattestäbchen, das Sie in etwas medizinischen Alkohol getaucht haben.

Das eingebaute Diskettenlaufwerk des Amiga 500 hat ein Fassungsvermögen (eine Kapazität) von 880 Kbyte, was ziemlich viel für ein Diskettenlaufwerk ist. 880 Kbyte sind mehr als 900 000 Buchstaben. Falls Sie nur Texte auf einer solchen Diskette speichern, so passen etwa 400 eineinhalbzeilig getippte Manuskriptseiten darauf. Die verwendeten Disketten mit 3,5 Zoll Durchmesser und Plastikschale sind handlich, robust und zuverlässig. Sie setzen sich jetzt allmählich als Standard bei neueren Mikrocomputern durch. Selbst IBM, der große »blaue Riese« des Computergeschäfts, setzt diese Disketten bei seiner neuesten Computerserie, den PS/2-Systemen, ein.

## 1.2 Innere Werte

Jetzt haben Sie schon die äußere Erscheinung des Amiga 500 etwas näher kennengelernt. Er besitzt aber auch innere Werte: Die Systemeinheit enthält unter anderem die folgenden Komponenten:

- Die CPU Motorola MC 68000
- 512 Kbyte RAM-Speicher für Programme und Daten
- 256 Kbyte ROM-Speicher für die wichtigsten Teile des Betriebssystems
- Ein 3,5-Zoll-Diskettenlaufwerk mit 880 Kbyte Kapazität
- Drei spezielle Chips für Sound und Grafik
- Diverse Zusatz-Chips für Hilfsaufgaben

Für Fachleute: Bei der CPU handelt es sich um den inzwischen recht weit verbreiteten Motorola-Prozessor 68000, der auch im Atari ST, dem Apple Macintosh und anderen leistungsfähigen PCs zu finden ist. Sie arbeitet im Amiga 500 mit einer Taktfrequenz (was

man in etwa mit »Geschwindigkeit« gleichsetzen kann) von etwas über 7 MHz. Der Speicher und einige andere Chips werden zwar mit einer Frequenz von etwas unter 14,5 MHz getaktet. Die CPU selbst bekommt davon aber nur jeden zweiten Takt ab, so daß Sie effektiv nur mit der Hälfte dieser Taktfrequenz (ziemlich exakt 7,19 MHz) läuft. Die anderen Takte sind für die drei Spezial-Chips reserviert.

Mit den 256 Kbyte ROM-Speicher in der Haupteinheit des Amiga hat es eine besondere Bewandtnis. Wie die meisten 68000er-Computer wurde der Amiga von vornherein mit einem großen ROM (Festwertspeicher) entworfen. Dieses ROM sollte alle wesentlichen Teile des Betriebssystems sowie die wichtigsten Hilfsroutinen enthalten, damit alle Programmierer sofort diese Routinen nutzen können, und sie nicht noch einmal selbst entwickeln müssen. Wie scheinbar allgemein üblich, gelang es den Software-Ingenieuren aber nicht, rechtzeitig zur Fertigstellung der Hardware auch die Software fertigzustellen. Man hat deshalb (beim Amiga 1000) zunächst die ROMs aus dem Design herausgenommen und zusätzlichen RAM-Speicher an dessen Stelle eingebaut. Beim Einschalten des Amiga mußte dieser RAM-Speicher von der sogenannten »Kickstart-Diskette« mit der vorläufigen Version der Software gefüllt werden, die eigentlich in das ROM gehen sollte.

Bei den Nachfolgern des Amiga 1000, dem Amiga 2000 und dem Amiga 500, war die Software dann fertig und wurde wirklich in Form von ROM-Chips in die Computer eingebaut. Der Name »Kickstart« geistert auch heute noch durch die Amiga-Welt, und wenn Sie ihn einmal hören sollten, wissen Sie nun, wo er seinen Ursprung hat. Das ROM des Amiga 500 wird zum Beispiel meist »Kickstart-ROM« genannt, weil es dieselben Daten enthält, die früher auf der Kickstart-Diskette des Amiga 1000 zu finden war.

## 1.3 Verbindungen zur Außenwelt

Der Amiga ist für die Kommunikation mit der Außenwelt – nicht nur mit dem Benutzer – gut gerüstet. An seiner Haupteinheit befinden sich reichlich Anschlüsse (sogenannte »Schnittstellen«), die fürs erste den meisten Anwendern völlig genügen dürften.

Hinten an der Haupteinheit befinden sich (von links nach rechts):

- Zwei Anschlüsse für je eine Maus oder einen Steuerknüppel (Joystick)
- Zwei Anschlüsse für Stereo-Tonausgabe
- Ein Anschluß für ein zusätzliches Diskettenlaufwerk
- Eine serielle Schnittstelle (nach RS232-Standard)
- Eine parallele Schnittstelle (nach Centronics-Standard)
- Der Stromanschluß (für das Netzteil)
- Ein Anschluß für einen RGB-Monitor
- Ein Anschluß für Video-Monitore oder Fernsehbildschirme

Die interessantesten dieser Anschlüsse dürften für Sie wahrscheinlich die parallele Schnittstelle sein, an die Sie eine Vielzahl handelsüblicher Drucker anschließen können, sowie die serielle Schnittstelle, an die Sie ebenfalls Drucker oder zum Beispiel ein Modem (für die

Kommunikation mit anderen Computern über Telefonleitungen) anschließen können. Einer der Monitor-Ausgänge, der erste Mausanschluß und vielleicht auch der zusätzliche Diskettenlaufwerksanschluß, ist bei Ihnen ja wahrscheinlich schon belegt.

An der linken Seite des Amiga-500-Gehäuses befindet sich, versteckt hinter einer abnehmbaren Plastikleiste, die »Erweiterungsschnittstelle«. Hier werden praktisch alle Signale, die intern im Amiga 500 erzeugt werden (für Fachleute: der komplette »Systembus«), nach außen geführt. Festplattenlaufwerke und zusätzlicher RAM-Speicher können hier zum Beispiel mit dem Amiga 500 verbunden werden. Hersteller von Zusatzgeräten, aber auch Commodore selbst, bieten bereits eine Reihe solcher Zusatzprodukte für die Erweiterungsschnittstelle an oder planen solche Erweiterungen.

An der Unterseite des Amiga 500 schließlich befindet sich, ebenfalls unter einer Plastikabdeckung verborgen, ein leeres Fach und eine Steckleiste, die speziell für Speichererweiterungen gedacht ist. Mehr dazu aber im folgenden Abschnitt.

## 1.4 Zubehör

Ähnlich wie bei einigen Automobilmarken der Luxusklasse, ist es meist mit dem Kauf des »Grundmodells« – und damit im übertragenen Sinn des Amiga 500 – allein nicht getan. Damit sind Sie zwar »fahrtüchtig«; viel Freude werden Sie an dieser Art zu fahren aber nicht haben. Deshalb ist es ganz sinnvoll, möglichst früh einen Blick in die Zubehörliste zu werfen, damit Sie wissen, was in dieser Hinsicht auf Sie zukommt. (Sie selbst sollten vielleicht auch einen Blick in die Zubehör-Preisliste werfen, damit Sie auch wissen, was auf Ihr Portemonnaie noch alles zukommt.)

### 1.4.1 Zusätzliche Diskettenlaufwerke

Der Amiga 500 enthält ein Diskettenlaufwerk. Dieses ist mit einer Kapazität von 880 Kbyte für den Anfang mehr als ausreichend. Ein einziges Diskettenlaufwerk kann beim späteren intensivem Gebrauch des Amiga 500 andererseits aber auch rasch »eng« werden. Insbesondere die Anfertigung von Diskettenkopien ist mit nur einem Laufwerk eine recht lästige Angelegenheit. Deshalb bietet der Amiga 500 die Möglichkeit, an dem Anschluß, der mit »Disk Drive« beschriftet ist, ein zusätzliches Diskettenlaufwerk anzuschließen. Commodore bietet ein solches zusätzliches Diskettenlaufwerk für 3,5-Zoll-Disketten unter der Bezeichnung Amiga 1010 an. Dieses Laufwerk wird in einem separaten Gehäuse ausgeliefert, das vom Design und Farbton her zum Amiga 500 paßt. Es besitzt seinerseits auch wieder einen Anschluß für ein weiteres Diskettenlaufwerk. Auf diese Weise kann man mehrere zusätzliche Diskettenlaufwerke an den Amiga 500 anschließen, obwohl nur ein Anschluß für diesen Zweck vorhanden ist.

Neben einem (oder mehreren) zusätzlichen 3,5-Zoll-Laufwerk(en) können aber auch Laufwerke für 5,25-Zoll-Disketten angeschlossen werden, die bis heute von vielen anderen Computerherstellern verwendet werden. Ein solches Laufwerk wird von Commodore nicht mehr angeboten (früher gab es eines unter der Bezeichnung Amiga 1020). Bei diversen Zubehörfirmen sind 5,25-Zoll-Laufwerke aber nach wie vor erhältlich. Sie können mit einem solchen Laufwerk dann auch auf den preiswerteren 5,25-Zoll-Disketten Daten und

Programme speichern, müssen dafür aber auch mit einer geringeren Kapazität vorliebnehmen. (Auf eine 5,25-Zoll-Diskette passen nur etwas weniger als halb so viele Daten wie auf eine Amiga-3,5-Zoll-Diskette.) Ein 5,25-Zoll-Laufwerk versetzt den Amiga 500 aber auch – mit der entsprechenden Software – in die Lage, Daten von Disketten zu lesen, die von anderen Computern beschrieben wurden, die 5,25-Zoll-Laufwerke verwenden (zum Beispiel von einem IBM PC).

### 1.4.2 Zusätzlicher RAM-Speicher

In einem vorangehenden Abschnitt wurde ja bereits erwähnt, daß an der Unterseite des Amiga 500 schon Platz für eine Speichererweiterung vorgesehen ist. Commodore bietet unter der Bezeichnung A 501 eine solche Speichererweiterung an, die den RAM-Speicher des Amiga 500 von 512 Kbyte auf 1024 Kbyte (= 1 Mbyte) aufstockt. Sie werden dadurch in die Lage versetzt, größere und leistungsfähigere Programme zu benutzen, größere Datenmengen zu bearbeiten oder auch mit mehreren Programmen gleichzeitig zu arbeiten.

Mit der Speichererweiterung A 501 bekommen Sie zugleich auch eine Uhr mit Batterieversorgung spendiert. Der Amiga 500 ist dann (nachdem Sie die Uhr einmal richtig gestellt haben) immer über die aktuelle Zeit im Bilde. Sie können ihn dann auch als Uhr, Wecker und Terminkalender »mißbrauchen«. Zugleich sorgt diese Uhr auch dafür, daß bei jeder Änderung, die Sie an einer Datei vornehmen, das Datum der Änderung mit abgespeichert wird. Dies ist eine unschätzbare Hilfe, wie Sie bei der täglichen Arbeit mit dem Amiga 500 bald merken werden.

Beides – Speichererweiterung und Uhr – sind außerordentlich sinnvolle Ergänzungen des Amiga 500. Man sollte die Speichererweiterung A 501 deshalb am besten gleich beim Kauf des Grundgeräts miterwerben (vielleicht macht Ihnen Ihr Händler dann ja auch ein günstiges Paketangebot).

### 1.4.3 Bildschirme

Ein wirklich unverzichtbares Zubehör des Amiga 500 ist ein Bildschirm. Hier stehen Ihnen sehr viele Möglichkeiten zur Wahl. Welche davon für Sie die beste ist, hängt von den Programmen, die Sie benutzen, Ihren Ansprüchen hinsichtlich der Bildqualität, und nicht zuletzt von Ihrem Geldbeutel ab.

Einen fast idealen Bildschirm für Ihren Amiga 500 stellt zweifellos der speziell für die verschiedenen Amiga-Modelle entwickelte RGB-Monitor dar, wie ihn die Firma Commodore zum Beispiel unter den Bezeichnungen A 1084 S (früher auch A 1081 und A 1082) anbietet. Ein solcher »RGB-Monitor« – oder genauer Rot-Grün-Blau-Monitor – erhält die Signale für die drei Grundfarben, aus denen sich die Farbe eines Bildschirmpunktes zusammensetzt, über drei getrennte Kabel. Die Bildqualität, besonders die Farbtrennung und Schärfe, ist deshalb typischerweise wesentlich besser als bei einem sogenannten Composite-Monitor oder gar einem einfachen Farbfernseher. Ein (analoger) RGB-Monitor kann deshalb die volle Palette von 4096 Farben, zu der der Amiga fähig ist, darstellen. Neben diesen grafischen Fähigkeiten kann er aber auch 80spaltigen Text in akzeptabler Qualität zeigen. Das Bild ist stabil, flimmerfrei und die Farbtrennung ist bis in die Ecken hinein hervorragend (keine Regenbogen).

Die billigste – und leider auch von der Bildqualität her minderwertigste – Lösung des Bildschirmproblems ist es, einen Fernseher oder Video-Monitor zu verwenden. Hierzu benötigen Sie allerdings einen Fernseher mit Video-Eingang. (Der Amiga 500 besitzt keinen HF-Ausgang, den Sie mit dem Antenneneingang eines Fernsehers verbinden können.) Mit einem Fernseher erhalten Sie aber nur Schwarzweiß-Bilder (selbst wenn es ein Farbfernseher ist). Commodore hat bei früheren Amiga-Modellen eine Farbausgabe auf Farbfernsehern versucht, war aber mit der Qualität so unzufrieden, daß diese Möglichkeit im Amiga 500 nicht mehr vorgesehen wurde.

Wer einen guten Video-Monitor besitzt oder aber unbedingt Wert darauf legt, sich die Augen zu verderben, kann einen zusätzlichen Adapter oder »Modulator« namens A 520 erwerben, der auf den RGB-Ausgang des Amiga 500 gesteckt wird. Dieser Modulator ermöglicht den Anschluß eines Farbfernsehers an den Amiga 500, der dann auch ein farbiges Bild liefert und sich nicht auf Schwarz und Weiß beschränkt. Auch für den Anschluß des Amiga 500 an einen Videorecorder – eine sehr interessante Verbindung übrigens – ist der A-520-Adapter notwendig. Damit ist allerdings nicht ganz die Bildqualität eines RGB-Monitors zu erreichen, wenngleich schon eine bessere Qualität als bei der Verwendung eines Fernsehers. Diese Alternative dürfte vor allem dann ideal sein, wenn man bereits einen guten Video-Monitor besitzt – zum Beispiel als Bestandteil einer Heim-Videoanlage. Wesentlich preiswerter als ein RGB-Monitor ist ein guter Video-Monitor nämlich nicht.

Wer bereits einen anderen IBM-kompatiblen Computer mit RGB-Monitor besitzt, von denen es eine sehr große Auswahl gibt, kann auch diesen Monitor am Amiga 500 verwenden. Dann muß er sich allerdings meist mit 16 statt mit 4096 Farben begnügen (mehr schafft die herkömmliche Standard-IBM-Grafik nämlich nicht). Sehr gute Bildschärfe und alle 4096 Farben, zu denen der Amiga 500 fähig ist, kann man hingegen auch mit den sogenannten MultiSyn- oder MultiScan-Monitoren erzielen, die normalerweise ebenfalls für IBM-kompatible Computer verkauft werden. (Sie werden meist im Zusammenhang mit den sogenannten »EGA-« oder »VGA-Karten« verwendet.) Solche Monitore sind aber meist deutlich teurer als der A 1084 (sogar teurer als der Amiga 500 selbst). Ihre Anschaffung lohnt deshalb nur im Zusammenhang mit einem weiteren Computer oder bei besonders hohen Qualitätsansprüchen.

### 1.4.4 Drucker

Wer die Werke, die er mit seinem Computer zustande gebracht hat, nicht nur auf dem Bildschirm betrachten will, sondern sie auch schwarz (oder bunt) auf weiß in der Hand halten möchte, wird nicht um die Anschaffung eines Druckers herumkommen. Hier ist die Auswahl ähnlich groß wie bei den Monitoren, das Problem andererseits aber nicht ganz so drängend, da man meist auch sehr gut – zumindest eine Zeitlang – ohne Drucker auskommen kann.

Commodore selbst bietet, speziell für die verschiedenen Amiga-Modelle, eine Reihe von Druckern für verschiedene Bedürfnisse und Geldbeutel an. Bei diesen Geräten handelt es sich um grafikfähige Nadeldrucker der mittleren Preisklasse. Es gibt sie sowohl als reine Schwarzweißdrucker wie auch als Mehrfarbdrucker, was besonders für den wichtig ist, der die fantastischen Bilder, die man auf einem Amiga erstellen kann, auch zu Papier bringen möchte. Ein schwarzweißer Ausdruck eines vielfarbigen Bildes ist meist nur ein recht müder Abklatsch, der kaum noch erahnen läßt, wie das Original am Bildschirm aussah.

Daneben können Sie am Amiga 500 aber auch nahezu alle Computerdrucker verwenden, die im Handel angeboten werden. Die Software des Amiga unterstützt eine ungewöhnlich breite Palette von Druckern. Zwei wichtige Gesichtspunkte sind bei der Auswahl des Druckers aber zu beachten. Der erste davon ist die Grafikfähigkeit. Die Software des Amiga 500 ist stark grafikorientiert. Auf dem Bildschirm erscheinen nicht nur Zahlen und Buchstaben, sondern oft auch Symbole, Piktogramme, Skizzen und Bilder. Wenn Sie diese auch auf Papier bekommen wollen, brauchen Sie einen grafikfähigen Drucker. Fast alle heutzutage angebotenen Nadel- oder Matrixdrucker sind grafikfähig. Typenraddrucker, Kugelkopfdrucker oder zu Druckern umfunktionierte Schreibmaschinen sind jedoch nicht grafikfähig. Wenn Sie nicht absolut sicher sind, daß Sie immer nur Texte ausdrucken wollen, sollten Sie keinen solchen nicht-grafikfähigen Drucker kaufen.

Der zweite wichtige Gesichtspunkt ist die Farbfähigkeit des Druckers. Der Amiga 500 kann auf einem guten Monitor bis zu 4096 verschiedene Farben darstellen, während die meisten Drucker nur in einer Farbe drucken können. Wer die Bildschirmfarben auch zu Papier bringen will, muß also einen Mehrfarbdrucker kaufen. Diese Drucker haben jedoch zwei entscheidende Nachteile: sie sind entweder recht teuer oder bringen nur eine dürftige Qualität zu Papier und sind zudem noch recht langsam. Auch die laufenden Kosten – also die Kosten pro ausgedruckte Seite – sind bei Mehrfarbdruckern unverhältnismäßig viel höher als bei einfachen Matrixdruckern.

Falls Sie sich aber auch von diesen Tatsachen nicht abschrecken lassen, berücksichtigen Sie bitte auch noch, daß kein heute zu einem Preis unter 30 000 DM erhältlicher Drucker das zu Papier bringen kann, was Sie auf dem Bildschirm sehen. Die meisten Mehrfarbdrucker verfügen nur über etwa ein Dutzend unterschiedlicher Farben; ein Bruchteil dessen, was der Bildschirm zuwege bringt. Und den gedruckten Farben mangelt es meist auch an Leuchtkraft und Brillanz. Für den Privatbedarf kann ein Mehrfarbdrucker schon akzeptable Ergebnisse liefern. Für den betrieblichen Einsatz sind seine Erzeugnisse meist zu dürftig und entstammen gar zu deutlich dem Computer.

Ganz anders sieht die Situation bei der nächsten Druckerkategorie aus. Die besten Druckergebnisse kann man nämlich zweifellos mit einem Laserdrucker erzielen. Die Ausdrucke dieser etwa 5 000 DM bis 10 000 DM teuren Geräte stehen der Qualität einer Tageszeitung oder Illustrierten kaum nach. Sie sind zudem auch voll grafikfähig – bieten aber nur eine Druckfarbe: Schwarz. Die Anschaffung eines Druckers, der fünfmal soviel kostet wie der Computer selbst, dürfte jedoch nur in den wenigsten Anwendungsfällen gerechtfertigt sein.

### 1.4.5 Sonstiges Zubehör

Neben diesem »üblichen« Zubehör, wie es Drucker und Bildschirme darstellen, gibt es für ganz spezielle Anwendungen natürlich auch noch ganz spezielles Zubehör. So gibt es zum Beispiel Zusatzprodukte, mit denen man Bilder, die man mit einer Videokamera aufgenommen hat, »in den Amiga 500 hineinbekommen kann« und sie dann verändern und wieder anzeigen und ausdrucken kann. Ähnliches ist mit Tönen möglich.

Wem 1 Mbyte RAM-Speicher nicht genügt, der kann seinem Amiga 500 auch noch mehr Speicher spendieren. Oder man kann noch höhere Kapazitäten und höhere Geschwindigkeiten als mit Disketten erreichen, indem man eine »Festplatte« an den Amiga 500 anschließt. Die Erweiterungsmöglichkeiten sind fast unbegrenzt. Dies wird vor allem durch die

Erweiterungsschnittstelle ermöglicht, die sich hinter einer Plastikabdeckung links am Gehäuse des Amiga 500 befindet.

## 1.5 Was Sie außer der Hardware noch bekommen

Damit soll aber die Betrachtung der Amiga-Hardware erst einmal abgeschlossen werden. In den folgenden Abschnitten wird die »weiche Ware« (Software) des Amiga 500 etwas näher vorgestellt. Neben der reinen Hardware bekommt man schon beim Kauf des Basispakets auch etwas Software dazu. Diese ist auf drei Disketten enthalten:

- Der Workbench-Diskette
- Der Extras-Diskette
- Der Erste-Schritte-Diskette

Wenn Sie ein spezielles »Aktionspaket« des Amiga kaufen, zum Beispiel aus der Kiss-Me-Amiga-Aktion von Commodore, erhalten Sie unter Umständen noch weitere Software und/oder Hardware. Die eben genannten Disketten stellen lediglich die Software-Grundausstattung dar, die (fast) jedem Amiga 500 beim Kauf beiliegt. (Die Erste-Schritte-Diskette ist allerdings in manchen Amiga-500-Paketen auch nicht vorhanden.)

Die Workbench-Diskette ist die »Boot-Diskette«, mit der der Amiga 500 nach dem Start »hochgefahren« oder »gebootet« wird, wie die Computerleute sagen. Sie sorgt für das Erscheinen der grafischen Benutzerschnittstelle des Amiga 500, der »Workbench« (zu deutsch Werkbank), mit der der Benutzer wohl die meiste Zeit zu tun haben wird. Sie enthält ansonsten noch einige kleine Hilfs- und Demonstrations-Programme, mit denen man die Möglichkeiten des Amiga 500 etwas austesten kann.

Die Extras-Diskette oder ExtrasD-Diskette enthält vor allem Amiga-Basic. Amiga-Basic ist eine sehr komfortable Variante des bekannten und eigentlich auf fast allen Mikrocomputern zu findenden Microsoft-Basic, der am weitesten verbreiteten Computer-Software überhaupt. Mehr dazu aber in einem späteren Kapitel. Zusätzlich zu der Programmiersprache selbst befinden sich auch noch einige Beispielprogramme samt den dazugehörigen Hilfsdateien und einige weitere kleine Programme auf der Extras-Diskette.

Die Erste-Schritte-Diskette schließlich enthält das Tutorial »Erste-Schritte«. Ein solches »Tutorial«, manchmal sagt man auch »Guided Tour« dazu, ist ein Programm, das Sie Schritt für Schritt durch eine Folge von Lernschritten führt. In diesem Fall erlernen Sie dabei die Grundzüge der Bedienung des Amiga. Sie brauchen diese Diskette nur in das Diskettenlaufwerk Ihres Amiga zu schieben, ihn einzuschalten und schon geht es los. Folgen Sie dann einfach den Anweisungen, die am Bildschirm erscheinen!

Mit dieser Grundausstattung werden Sie sich als frischgebackener Amiga-Besitzer aber wohl höchstens ein oder zwei Tage beschäftigen können, falls Sie nicht Amiga-Basic benutzen wollen, um eigene Programme zu entwickeln. Für den Amiga-500-Anwender (Anwender hier im Gegensatz zum Programmierer, der sich seine eigenen Programme schreibt) gibt es bereits sehr viel zusätzliche Software. Schließlich ist der Amiga 500 beziehungsweise sein Vorgängermodell Amiga 1000, schon seit Jahren auf dem Markt - ausreichend Zeit für die Softwareentwickler, um viele Programme dafür zu schreiben. Nich

nur Commodore, sondern auch viele andere Softwarehäuser bieten inzwischen Programme für fast jeden Anwendungszweck an. Die Vielfalt und Leistungsfähigkeit der Programme für den Amiga steht dem Softwareangebot für andere Computer kaum nach.

Außer den eben genannten Disketten erhalten Sie beim Kauf natürlich auch noch Handbücher für die Bedienung Ihres neuen Computers, und zwar zwei Stück. Eines behandelt den Amiga 500 (das Gerät) selbst, die »Workbench« und das »CLI« (über beides werden Sie weiter unten noch mehr erfahren). Dieses Handbuch ist recht kurz und knapp gefaßt und erklärt nur die wichtigsten Grundlagen der Bedienung. Es enthält aber auch wichtige Informationen über die Hardware des Amiga 500 und die Anschlüsse für Zusatzgeräte. Das zweite Handbuch beschäftigt sich mit Amiga-Basic und den wichtigsten Grundzügen der Programmierung des Amiga 500. Neben einigen meist kleinen Beispielen enthält es vor allem ein sehr vollständiges Nachschlagewerk zu Amiga-Basic.

## 1.6 Die Fähigkeiten des Amiga 500

Nachdem Sie nun – zumindest ungefähr – wissen, wie der Amiga 500 aussieht und was Sie alles erhalten, wenn Sie einen kaufen, sollen Sie nun aber auch einen kleinen Eindruck davon erhalten, wozu er fähig ist, wenn er die richtige Software »zu fressen bekommt«. Diese Leistungsdaten sind sehr beeindruckend – besonders dann, wenn Sie bereits andere Computer kennen, die meist nicht so leistungsfähig sind. Sie sollten sich davon aber nicht blenden lassen. Das wichtigste bei jedem Computer ist, daß er Ihnen Arbeit abnimmt. Dem Amiga 500 mit seinem schnellen 16-Bit-Hauptprozessor fällt das natürlich nicht schwer. Mit seinen phantastischen Grafik- und Sound-Fähigkeiten hat er aber dazu noch das Potential, Ihnen diese Arbeitserleichterung so angenehm und unterhaltsam wie möglich zu gestalten.

### 1.6.1 Grafik-Fähigkeiten

Der eine Aspekt, der am Amiga von allen Fachleuten nahezu uneingeschränkt gelobt wird, sind seine Grafikfähigkeiten. Sie sind den Möglichkeiten vergleichbarer Computer in vieler Hinsicht weit voraus. Drei speziell für diesen Zweck entworfene und von Commodore gebaute Chips (sogenannte »Custom-VLSI-Chips«) sorgen für eine Vielfalt an Möglichkeiten und vor allem für eine Geschwindigkeit beim Aufbau der Grafiken, wie man sie bisher – zumindest bei Computern in dieser Preislage – für unmöglich hielt.

Die Grafik des Amiga 500 ist »bitmapped«. Dies bedeutet, daß die Farbe jedes Punktes auf dem Bildschirm durch ein oder mehrere Bits im Speicher bestimmt wird. Im simpelsten Fall gibt es eine 1:1-Zuordnung, bei der jeder Punkt auf dem Bildschirm genau einem Bit im Speicher entspricht. Da der Amiga 500 mehrere Farben gleichzeitig in einer Grafik verwenden kann, liegt der Fall natürlich etwas komplizierter – aber im wesentlichen stimmt die 1:1-Zuordnung, von der auch der Begriff »bitmapped «abgeleitet ist. Mehr zu diesem Thema können Sie in Kapitel 11 erfahren.

Andere Computer haben oft noch einen anderen Betriebsmodus, in dem sie nur Text (Buchstaben, Zahlen und Sonderzeichen) und keine Grafiken auf dem Bildschirm darstellen können. Beim Amiga 500 gibt es einen solchen Modus nicht. Auch Text muß, wie ein Kreis

oder ein Rechteck, als ein Punktmuster auf den Bildschirm gemalt werden. Dies ist natürlich schwieriger, aber dafür auch viel flexibler. Durch die Geschwindigkeit, die die speziellen Grafik-Chips dem Amiga 500 verschaffen, spielt der zusätzliche Aufwand zudem kaum eine Rolle.

Die Amiga-Grafik kennt allerdings auch mehrere (grafische) Betriebsarten. Die horizontale Auflösung (Anzahl der Punkte in einer Bildzeile) kann 320 oder 640 Punkte betragen. Und die vertikale Auflösung (Anzahl der Bildzeilen beziehungsweise Anzahl der Punkte in einer Bildspalte) kann entweder 256 oder 512 Zeilen betragen. Die Auflösung mit den 512 Zeilen wird allerdings auf eine Art und Weise erreicht, die zu erhöhtem Bildschirmflimmern führt. Diese Methode nennt man »Zeilensprungverfahren«. Sie wird auch bei Fernsehern angewandt, weshalb man nicht allzu lange nahe vor einem Fernseher sitzen kann, ohne Kopfschmerzen zu bekommen.

Die Bitmapped-Grafik des Amiga 500 ist mehrfarbig. Je nachdem, wieviel Speicher man opfern will, kann man für jeden Bildpunkt zwischen 2, 4, 8, 16 und 32 möglichen Farben wählen. Bei mehr als 2 Farben wird auch mehr als 1 Bit pro Punkt benötigt (siehe auch Kapitel 11). Deshalb benötigen Bilder um so mehr Speicherplatz, je mehr verschiedene Farben darin vorkommen. Diese jeweils zu Verfügung stehenden 2, 4, 8, 16 oder 32 Farben sind aber nicht fest »eingebaut« wie bei anderen Computern. Diese Farben können aus einer Menge von 4096 verschiedenen Farben ausgewählt werden. Man sucht sich also »normalerweise« zunächst aus dieser Menge von 4096 Farben eine »Palette« von maximal 32 Farben aus und kann mit diesen dann sein Bild malen.

Das Wort »normalerweise« im letzten Satz wurde mit Bedacht in Anführungszeichen gesetzt, weil es einen Betriebsmodus gibt, in dem – mit gewissen Einschränkungen – alle 4096 möglichen Farben gleichzeitig am Bildschirm erscheinen können. Dies ist der sogenannte »Hold-and-modify-« oder »HAM-Modus«, über den Sie in einem folgenden Kapitel noch mehr erfahren werden. Zusätzlich ist es möglich, den Bildschirm in horizontale Streifen zu unterteilen, in denen verschiedene Grafik-Betriebsarten genutzt werden. Zum Beispiel kann die obere Hälfte oder das obere Drittel des Bildschirms 32 andere Farben verwenden als der untere Teil. Wie Sie sehen, sind die Grafikmöglichkeiten des Amiga 500 sehr vielseitig.

Bisher haben wir uns nur mit den statischen Aspekten der Amiga-Grafik beschäftigt, also ruhenden, unveränderlichen Bildern. Die Grafik-Chips des Amiga 500 bieten aber auch die Möglichkeit, 8 kleine Bilder (die sogenannten »Sprites«) mit hoher Geschwindigkeit über den Bildschirm zu bewegen. Dies ist nicht nur für Videospiele wichtig, wie Sie vielleicht zunächst meinen werden. Sprites können auch für durchaus sinnvolle Aufgaben verwendet werden.

Die Software des Amiga 500 bietet darüber hinaus aber in Zusammenarbeit mit anderen Teilen der Grafik-Chips auch noch weitergehende Möglichkeiten für solche bewegte Grafiken (der Fachmann spricht von »animierten« Grafiken oder von »Animation«), die dem Programmierer sehr viel Arbeit abnehmen. Hierzu gehören die BOBs (BlitterObjects) und AnimObjects (animierte Objekte).

Ein AnimObject kann zum Beispiel Bilder der verschiedenen Bewegungsphasen einer gehenden Person enthalten. Je mehr Phasen man verwendet, desto flüssiger wirkt diese Bewegung. Gibt man dem AnimObject dann eine Geschwindigkeit, dann könnte diese Person zum Beispiel langsam oder schnell über den Bildschirm schreiten. All dies geschieht

– nach dem einmaligen »Einschalten« des Vorgangs – ohne weiteres Zutun des eigentlichen Programms. Dieses kann sich währenddessen um andere Dinge kümmern.

### 1.6.2 Sound-Fähigkeiten

Nicht ganz so große Publizität wie seine Grafikfähigkeiten haben die Möglichkeiten des Amiga 500 zur Klangerzeugung gefunden – unverdienterweise, denn sie sind mindestens ebenso beeindruckend. Der Amiga 500 besitzt insgesamt 4 voneinander unabhängige Tonkanäle, von denen jeweils 2 zu einem Stereo-Kanal zusammengefaßt werden können. Jeder dieser 4 Kanäle kann eine völlig beliebige Wellenform mit beliebiger »Hüllkurve« ausgeben. Auf dem einen Kanal könnte zum Beispiel ein Klavier spielen, während auf dem anderen eine Trompete erklingt. Dazu wird nur in sehr geringem Maße der Hauptprozessor des Amiga beziehungsweise das Programm benötigt, das den Ton irgendwann einmal »angestoßen« hat. Die Klangerzeugung wird nahezu vollautomatisch von den Spezial-Chips des Amiga erledigt.

Man darf von der Sound-Hardware des Amiga 500 aber natürlich auch nicht zuviel verlangen. Es handelt sich nicht um einen professionellen Synthesizer, und die Klangqualität kann Hi-Fi-Tests gewiß nicht standhalten. Für die Darstellung eines Klangs werden im Amiga 500 zum Beispiel »8 Bit große« Werte verwendet. Hi-Fi-Geräte in Digital-Technik verwenden üblicherweise 16 Bit, also die doppelte Datenmenge, was zu einer wesentlich besseren Klangqualität führt. Trotzdem können die vom Amiga 500 erzeugten Töne oft recht natürlich klingen.

Die vom Amiga 500 erzeugten Töne müssen aber nicht »natürlich« sein und klingen. Sie können auch vollständig »synthetisch« sein, also mehr oder weniger auf dem Papier beziehungsweise am Computer entworfen worden sein. Kreieren Sie sich selbst ein elektronisches Instrument! Sie können aber auch einen natürlichen Klang nehmen, seine Charakteristika analysieren (ihn »digitalisieren«), abspeichern und dann von der Amiga-Sound-Hardware wieder ausgeben lassen. Der Fachmann nennt das »Sampling«. Dabei kann man den »gesampelten« Klang in sehr weiten Grenzen ändern und mißbrauchen. Es wäre zum Beispiel durchaus möglich, Beethovens 5. Symphonie mit digitalisierten Kochtopfdeckeln zu spielen.

Das berühmte »Bouncing-Ball-Demo«, mit dem der Amiga im Jahr seiner Markteinführung alle Computerfans (aber auch Fachleute) verblüffte, nutzt zum Beispiel die Möglichkeit, einen Klang aufzunehmen und dann verändert wiederzugeben, in recht eindrucksvoller Weise. Der Ton, den man hört, wenn der sich drehende Ball an eine Wand oder auf den Boden schlägt, ist in Wirklichkeit das Geräusch eines Garagentors, vor das der damalige Amiga-Software-Chef Bob Pariseau mit einem Mikrophon schlägt.

Wie es inzwischen scheinbar schon guter Ton bei neuen Computern zu werden scheint, ist der Amiga 500 natürlich auch für die Kommunikation mit MIDI-Geräten (MIDI = Musical Instruments Digitital Interface; der wohl wichtigste Standard für die Verbindung digitaler Musikinstrumente, wie Synthesizer und Keyboards) vorbereitet. Hierzu muß ein entsprechendes Zusatzgerät an die serielle Schnittstelle des Amiga angeschlossen werden. Wem die Klangqualität des Amiga also auch dann noch nicht ausreicht, wenn er ihn an seine Stereoanlage angeschlossen hat, der sollte sich die Anschaffung eines solchen »MIDI-Interfaces« und eines MIDI-fähigen Keyboards überlegen.

### 1.6.3 Sprach-Fähigkeiten

Eine Sound-Fähigkeit des Amiga 500 darf man auf keinen Fall vergessen: die Sprachausgabe. Der Amiga 1000 war seinerzeit der erste Computer, bei dem Software für Sprachsynthese sofort mitgeliefert wurde. Daran hat sich natürlich auch beim Amiga 500 nichts geändert. Sprachsynthese, also vom Computer erzeugte Sprache, kann in jedem Programm eingesetzt werden und ist völlig problemlos zu programmieren. In der simpelsten Form übergibt man einfach einen Text an eine bestimmte Routine, die ihn daraufhin in akustischer Form ausgibt. Die Aussprache kann man zusätzlich in verschiedenster Form beeinflussen. Die Stimmlage kann geändert werden; der Sprecher kann entweder im tiefen Baß oder als helle Frauenstimme sprechen. Die Geschwindigkeit der Sprache kann innerhalb weiter Grenzen geändert werden (in weiteren Grenzen als sinnvoll ist). Und schließlich kann die Stimme entweder sehr ausdrucksvoll, mit betonten Worten und Satzzeichen, oder monoton, computerhaft, sprechen.

Wir können also schon bald Programme erwarten, die uns unsere Fehler nicht nur am Bildschirm melden oder durch einen nervösen Summton mitteilen, sondern auch rufen: »Heh du Dummkopf, was hast du da schon wieder angestellt?« Die Entscheidung, ob das einen echten Fortschritt darstellt, überlasse ich Ihnen.

### 1.6.4 Sound- und Grafik-Hardware

Nachdem Sie nun einen ersten Eindruck von den Grafik- und Sound-Fähigkeiten des Amiga 500 bekommen haben, möchte ich auch noch ganz kurz darauf eingehen, wie sie zustandekommen – die Details können Sie im letzten Teil dieses Buches nachlesen. Eine allzu genaue Kenntnis der Hardware-Zusammenhänge ist sowieso beim Amiga 500 nicht nötig (nicht einmal für die Programmierung), da einem die Amiga-Systemsoftware den größten Teil der bei anderen Computern nötigen »Bit-Fummelei« abnimmt. Falls Sie allerdings ein Videospiel schreiben wollen, wie es die Welt noch nicht gesehen hat, werden Sie kaum um eine direkte Manipulation der Grafik-Chips herumkommen. Dann sind allerdings auf dem Amiga wirklich Dinge möglich, die »die Welt noch nicht gesehen hat«.

Die Qualität und Geschwindigkeit der Amiga-Grafik hat viele Gründe. Beides liegt aber im wesentlichen in den drei »Custom-VLSI-Chips« (Spezial-Chips) begründet, die schon mehrfach erwähnt wurden. Zum Beispiel hat jeder der drei Spezial-Chips im Amiga 500 ungefähr die Komplexität der CPU MC 68000, des Hauptprozessors im Amiga 500, der einer der modernsten und leistungsfähigsten Chips ist, die man heute in PCs einsetzt. Der Amiga 500 verfügt damit also – jedenfalls für bestimmte Aufgaben – über etwa drei- bis viermal soviel Rechenleistung wie ein Computer, der »nur« einen 68000er-Prozessor enthält. Die drei Chips tragen sinnigerweise die Spitznamen »Agnus«, »Denise« und »Paula« (obwohl gelegentlich auch andere Namen in der Computer-Presse zu lesen waren).

In diesen Chips befinden sich unter anderem Zusatz-Prozessoren oder sogenannte »Koprozessoren«, die dem eigentlichen Mikroprozessor, dem MC 68000, viel Arbeit abnehmen. Sie arbeiten gleichzeitig »neben« dem Mikroprozessor, während dieser mit anderen Dingen beschäftigt ist. Einer dieser Koprozessoren ist zum Beispiel der sogenannte Blitter, der unter anderem sehr schnell Linien zeichnen, Flächen mit einer bestimmten Farbe füllen und beliebige Teilbilder von einem Ort am Bildschirm zu einer anderen Stelle bewegen kann.

Auch die Sound-Hardware holt sich ganz selbständig die Daten, die sie zur Tonerzeugung benötigt, aus dem Speicher des Amiga 500. Erst wenn diese Daten zu Ende sind, meldet sie dies dem Programm und bittet um neue Daten. Zwischendurch aber wird das eigentliche Programm nicht durch die Erzeugung von Musik beeinträchtigt oder gebremst.

Der Amiga 500 wird durch seine drei Zusatz-Chips ein echter Multi-Prozessor-Rechner, in dem zu jedem Zeitpunkt mehrere Mikroprozessoren gleichzeitig mit der Erfüllung der Wünsche des Benutzers beschäftigt sind. Er ist deshalb jedem Computer der gleichen Leistungsstufe, der alle Arbeit nur dem einen zentralen Mikroprozessor überläßt, haushoch überlegen!

## 1.7 Die Software des Amiga 500

Wie schon erwähnt wurde, kann die Hardware eines Computers noch so beeindruckend und technisch hochentwickelt sein – ohne Software, die diese Hardware entsprechend nutzt, bleibt sie unwichtig und nutzlos. Sie werden nun ein wenig über diese Software erfahren. Dabei geht es aber noch nicht um die echten Anwendungsprogramme, die Ihnen zum Beispiel die Textverarbeitung oder Buchhaltung erleichtern. Sie erhalten statt dessen einen kurzen Überblick über die sogenannte »Betriebssystem-« oder »System-Software«, mit der Sie immer zu tun haben, wenn Sie einen Computer bedienen – ohne daß es Ihnen die meiste Zeit bewußt wird.

Diese System-Software erlaubt unter anderem auch erst die Kommunikation der Computer-Hardware mit der Außenwelt. (Diese Außenwelt sind Sie selbst, aber auch Drucker und andere Geräte, die Sie an Ihren Amiga 500 anschließen.) Sie stellt zum Beispiel auch die Vorgänge im Innern des Computers oder den Inhalt des Speichers am Bildschirm dar.

### 1.7.1 Die Workbench

Nach dem Einschaltvorgang, der im folgenden Kapitel noch in aller Ausführlichkeit erläutert wird, erscheint normalerweise die »Workbench«. Die Workbench (zu deutsch Werkbank) ist die »Benutzeroberfläche« des Amiga. Auf dieser Werkbank können Sie Programme starten, Dateien und Disketten kopieren, löschen und umbenennen.

Disketten, Dateien und Programme werden, wenn Sie die Workbench benutzen, als Symbole auf dem Bildschirm dargestellt. Alle wichtigen Operationen können mühelos durchgeführt werden, indem Sie einfach mit einem kleinen Zeiger, den man mit der Maus bewegen kann, auf bestimmte Objekte am Bildschirm deuten und einen der beiden Knöpfe auf der Maus drücken. Bestimmte Bewegungen dieses Zeigers zu bestimmten Punkten können ebenfalls Befehle bedeuten.

Ihre Dateien und Programme können Sie in Schubladen ablegen und so etwas Ordnung in Ihre Disketten bringen. Diese Schubladen werden Ihnen am Bildschirm auch wirklich als kleine Schubladen gezeigt. Sie können diese öffnen und nachschauen, welche Dateien darin enthalten sind. Sie können sie aber auch schließen und auf diese Weise Platz auf dem Bildschirm schaffen, weil Sie nicht mehr alle Dateien sehen, sondern nur noch die Schublade.

Das soll aber zunächst einmal an Informationen über die Workbench ausreichen. Sie wird noch in großer Ausführlichkeit in den folgenden Kapiteln des ersten Buchteils beschrieben. Wie Sie aber recht bald bemerken werden, ist diese grafische Art der Computer-Bedienung sehr leicht zu erlernen und auch zu behalten. Sie brauchen sich nur sehr wenige Dinge zu merken. Die Möglichkeiten, die Ihnen in jedem Augenblick zur Verfügung stehen, werden Ihnen immer gezeigt. Kurz: Die Bedienung ist sehr intuitiv. (Das Paket von Programmroutinen in der Amiga-Systemsoftware, das für diese Dinge zuständig ist, heißt deshalb treffenderweise auch »Intuition«.)

### 1.7.2 Das CLI

Eine zweite, von der Workbench völlig verschiedene Methode, den Amiga 500 zu bedienen, stellt das CLI dar. »CLI« bedeutet »command line interface«, was man in etwa mit Kommandozeilen-Schnittstelle übersetzen könnte. Diese Bedienungsweise ist der herkömmlichen Bedienungsweise, wie sie auch auf anderen Computern üblich ist, sehr ähnlich. Sie müssen dabei einen Befehl, meist ein kurzes englisches Wort, und einige Parameter eintippen, die diesem Befehl sagen, was er genau tun soll. Die Ergebnisse des Befehls werden Ihnen dann üblicherweise auch in Textform am Bildschirm aufgelistet.

Für viele Anwender gibt es wenig Gründe, sich mit dem CLI zu beschäftigen. Befehle in Textform sind wesentlich langwieriger einzugeben, schwerer zu behalten, und bieten viele Möglichkeiten, Fehler zu machen. Die neue Form des CLI, die zusammen mit der Version 1.3 der Workbench-Software ausgeliefert wird, »Shell« genannt, ist zwar wesentlich komfortabler als das CLI, aber immer noch schwieriger als die Workbench. Die Workbench ist eine viel »freundlichere« Umgebung. Einige Arbeiten gehen aber mit einer textuellen Bedienungsweise, also dem CLI oder der Shell, wesentlich einfacher. Die Programmierung zum Beispiel ist ein solches Anwendungsgebiet, in dem man mit dem CLI viel zügiger arbeiten kann. CLI oder Shell als zusätzliche Möglichkeit, den Amiga zu bedienen, hat also durchaus seine Daseinsberechtigung. Im zweiten Teil dieses Buches werden Sie deshalb über die Shell und das CLI noch mehr erfahren.

### 1.7.3 Multitasking

Eine der Software-Eigenschaften, die es vor dem Amiga bei Personalcomputern so gut wie gar nicht gab, ist die Fähigkeit, mehrere Programme gleichzeitig laufen zu lassen. Programme nennt man in diesem Zusammenhang oft auch Tasks oder Prozesse, und die Eigenschaft, mehrere Programme gleichzeitig verarbeiten zu können, heißt dementsprechend Multitasking.

Wenn man von der Workbench aus (siehe oben) ein Programm startet, »übernimmt« es nicht die ganze Computer-Hardware, wie das bei anderen Computer-Modellen üblich ist. Die Programme, die zu diesem Zeitpunkt schon aktiv waren, zum Beispiel auch die Workbench selbst, laufen einfach weiter. Das hört sich im ersten Moment sehr beeindruckend an, insbesondere, weil dazu bis heute nur wesentlich größere und teurere Rechner fähig waren. Andererseits fragt man sich dann bald auch, wozu das Ganze? Schließlich kann man ja keine zwei oder drei Programme gleichzeitig bedienen, selbst wenn sie gleichzeitig ausgeführt werden.

Dieser Einwand stimmt, und man soll die Vorteile des Multitasking deshalb auch nicht überschätzen. Andererseits ist Multitasking bei langwierigen Vorgängen aber auch eine ganz praktische Sache. Die meisten Druckvorgänge laufen zum Beispiel auf dem Amiga standardmäßig als separates Programm »im Hintergrund«. Während der Drucker noch rattert, kann man sich schon einmal anderen Dingen zuwenden. Genauso kann während des Formatierens (Vorbereitung zur Datenaufnahme) einer Diskette bereits mit anderen Programmen gearbeitet werden. Auch langwierige Sortier- oder Rechenvorgänge können sehr gut ohne Ihren andauernden Eingriff ablaufen, und Sie können unterdessen schon einmal einen Brief schreiben. Und schließlich ist auch der Wechsel von einem Programm zum anderen, ein Vorgang, den Sie häufiger benötigen werden, als Sie jetzt vielleicht meinen, wesentlich schneller, wenn jedes Programm nicht immer neu gestartet werden muß.

Alles in allem sind die Multitasking-Möglichkeiten des Amiga sehr vielversprechend. Ich bin sicher, daß Sie diese Möglichkeit nicht mehr missen wollen, wenn Sie sie erst einmal eine Weile genutzt haben.

## 1.8 Etwas Historie

Zum Abschluß dieser Einleitung noch ein paar Informationen zur Geschichte des Amiga 500. Es ist ja immer ganz interessant, auch das Vorleben einer neuen Freundin (oder eines neuen Freundes) kennenzulernen. Und dieses Vorleben ist im Falle des Amiga recht bewegt:

Wie kein anderer Computer vor ihm, wurde der Amiga schon lange vor seinem wirklichen Erscheinen von den wildesten Gerüchten begleitet. Die ersten tauchten etwa im Winter '83 auf der COMDEX (eine große und wichtige Computermesse, die jedes Jahr einmal im Frühjahr und einmal im Herbst in den USA stattfindet) in den USA auf. »Amiga« war damals noch der Name einer Firma, die nur »Joysticks«, kleine Steuerknüppel für Videospiele, verkaufte – sehr gute allerdings. Einen Personalcomputer zu bauen und noch dazu einen, der alles bisher Dagewesene in den Schatten stellen sollte, traute dieser Firma eigentlich kaum jemand zu.

Als man jedoch den Namen eines der Männer hörte, die bei der Entwicklung des Amiga mitarbeiten sollten, wurden die Zweifel etwas geringer. Für das Design der drei Spezial-Chips und eines großen Teils der grundsätzlichen »Architektur« des Amiga ist Jay Miner verantwortlich, der auch schon die Grafik-Chips für die Atari 400/800er Serie entwarf, die auch heute noch im Atari 130 XE eingesetzt werden. Es sind im wesentlichen diese Chips, die für die relativ guten Grafikfähigkeiten der alten 8-Bit-Ataris (und für den Ruhm Miners) verantwortlich sind. Die Philosophien hinter dem Atari-Chip-Satz und den drei Custom-Chips des Amiga 500 sind sich sehr ähnlich, nur sind die Amiga-Chips ungleich leistungsfähiger, da sie dem neuesten Stand der Mikroprozessor-Technik entsprechen. Der Amiga kann deshalb mit Fug und Recht als ein Nachfolger der alten Atari-PCs bezeichnet werden – eher jedenfalls als die neue Atari-ST-Serie, die eine völlige Neuentwicklung ist. Die Chips von Jay Miner sollten diesem neuen Heimcomputer Fähigkeiten geben, wie sie sonst nur in speziellen Grafikcomputern für einige Zehntausend Mark zu finden sind.

So waren denn auch alle Journalisten, denen Prototypen des neuen Amiga (oder »Lorraine«, wie er damals noch hieß) gezeigt wurden, hellauf begeistert. Die Firma Amiga machte in der Folge allerdings eine vor allem auch finanziell schwierige Zeit durch. Die Entwicklung der

Grafik-Chips dauerte unerwartet lange und wurde dadurch natürlich auch unerwartet teuer. Es wurde dadurch mit der Zeit recht unsicher, ob der Amiga jemals das Licht der Computerwelt erblicken würde.

In der Zwischenzeit brach nämlich auch der Heimcomputer-Markt mehr oder weniger in sich zusammen. Mit neuen Heimcomputern glaubte niemand mehr Geld verdienen zu können. Der Amiga wurde (allerdings nicht allein deshalb) immer größer und teurer und mauserte sich langsam aber sicher zu einem vollständigen Personalcomputer, der nichts mehr mit den beschränkten Heimcomputern der damaligen Zeit zu tun hatte. Die Firma Amiga benötigte allerdings immer mehr Geld für die immer länger dauernde Entwicklung eines sich kontinuierlich ändernden Computer-Projekts. Amiga hatte auch Geld von der Firma Atari für die Entwicklung neuer Grafik-Chips bekommen. Als sich das Geschäft mit Atari zerschlagen hatte, mußten diese Summen zusammen mit hohen Vertragsstrafen zurückgezahlt werden. Kurz: Amiga war in großen Schwierigkeiten.

Glücklicherweise fand sich – wie wir alle inzwischen wissen – aber rechtzeitig ein finanzstarker Geldgeber: Commodore. Im Sommer '85 wurde der Amiga dann endlich im Rahmen einer millionenschweren Multimedia-Show in New York vorgestellt. (Seine deutsche Premiere hatte er auf der Hannovermesse/CeBIT im März '86.) Kaum einen der Anwesenden dürfte die Vorstellung kaltgelassen haben. Ganz abgesehen von reichlich Prominenz, mit der sich Amiga schmückte, so zum Beispiel dem inzwischen verstorbenen Pop-Künstler Andy Warhol und der Rocksängerin Debby Harrie (Blondie), wurden vor allem die grafischen und musikalischen Fähigkeiten des Amiga groß herausgestellt. Diese Fähigkeiten hatten aber ihren Preis. Aus dem preiswerten Heimcomputer für etwa 500 Dollar, den Amiga zunächst entwickeln wollte, war inzwischen ein leistungsfähiger PC geworden, der zwar auch in der Geschäftswelt Erfolg haben sollte aber auch circa das Dreifache kostete. (Mit dem Amiga 500 ist Commodore dann später wieder zu dem zunächst geplanten preiswerten Heimcomputer zurückgekehrt.

In diesen fast zwei Jahren zwischen den ersten Gerüchten über den Amiga und seiner öffentlichen Präsentation kursierten die wildesten Gerüchte über eine wahre Supermaschine durch die Presse und die Computernetzwerke. Zu einem Zeitpunkt wurden die Gerüchte sogar Amiga selbst zu unglaublich, worauf ein Papier veröffentlicht wurde, in dem eine ungefähre Beschreibung an die Presse gegeben wurde, was die Maschine kann, aber auch, was sie nicht kann.

Nach dem spektakulären Auftakt hatte der Amiga (genauer der Amiga 1000) allerdings einige Startschwierigkeiten. Das von Commodore angepeilte Publikum »Geschäftswelt« konnte aus verschiedenen Gründen heraus nicht so recht mit dem neuen Computer warmwerden. Einer der wichtigsten Gründe dürfte dabei die mangelnde Verfügbarkeit professioneller Software gewesen sein. Dieser Mangel dauerte über ein Jahr an und ist erst seit Anfang 1987 wirklich behoben.

Als Heimcomputer wäre der Amiga 1000 andererseits ganz hervorragend geeignet gewesen und hätte sicherlich auch großen Erfolg gehabt – wäre da nicht der hohe Preis gewesen. In Deutschland kostete der Amiga 1000 bei der Markteinführung zusammen mit einem hochwertigen RGB-Monitor zum Beispiel fast 6000 DM. Diese Entwicklung bedeutete einmal mehr stürmische Zeiten für den Amiga – und Commodore.

Commodore geriet in der Folgezeit – nicht allein wegen des Amiga – in große finanzielle Schwierigkeiten, die die Zukunft des Amiga wieder ungewiß erscheinen ließen. Eine

Zeitlang sah es fast so aus, als wäre das Ende des Amiga gekommen. Die Amiga-Entwicklungsabteilung wurde aufgelöst und sämtliche Weiterentwicklungen des Amiga-Konzepts gestoppt. Die ehemaligen Mitglieder der Firma Amiga, nun Angestellte von Commodore, wurden entlassen.

Doch auch diese Schwierigkeiten haben Commodore und der Amiga inzwischen überwunden. Man kann sogar sagen, daß die Schwierigkeiten relativ unbeschadet überstanden wurden und die Amiga-Familie heute besser dasteht als jemals zuvor. An zwei verschiedenen Stellen in der Welt – im Commodore-Stammwerk in West Chester, USA, und in Deutschland im Commodore-Werk Braunschweig – wird die Amiga-Entwicklung fortgeführt. Nachdem die Preise gehörig gesenkt wurden, hat der Amiga inzwischen sowohl in den USA als auch hier bei uns in Europa große Verkaufserfolge zu verzeichnen. Im Weihnachtsgeschäft '86 ging der Amiga über die Ladentheken wie die sprichwörtlichen »warmen Semmeln«. Sowohl »Aufsteiger«, die zuvor einen weniger leistungsfähigen Computer besessen hatten, als auch Leute, die zuvor noch nie mit einem Computer gearbeitet hatten, entschieden sich für den Kauf eines Amiga.

Auch die Weiterentwicklung des Amiga hat inzwischen Früchte getragen – zwei völlig verschiedene Früchte allerdings. In Braunschweig entstand der Amiga 2000, eine leistungsfähige und ungemein ausbaufähige Weiterentwicklung des Amiga 1000. In West Chester, dem Commodore-Stammwerk in den USA, wurde gleichzeitig der Amiga 500 entwickelt.

Der Amiga 2000 besitzt von vorneherein schon 1 Mbyte RAM-Speicher und im Gehäuse Platz für insgesamt 3 Diskettenlaufwerke und eine Festplatte. In seinem relativ großen Gehäuse sind Anschlüsse (Steckplätze) vorhanden, in die man Erweiterungen (mehr Speicher, schnellere Prozessoren, andere Video-Adapter und so weiter) einsetzen kann. Mit einer speziellen Einsteckkarte kann man den Amiga 2000 auch IBM-kompatibel machen und so eine Vielzahl der sehr professionellen Softwarepakete nutzen, die für IBM-PCs entwickelt wurden. Einige der Steckplätze des Amiga 2000 sind sogar fähig, Steckkarten, die für IBM-PCs entwickelt wurden, aufzunehmen. Für nicht einmal 5000 DM erhält man mit dem Amiga 2000 eigentlich zwei Computer auf dem Platz und zum Preis von einem.

Während der Schwerpunkt bei der Entwicklung des Amiga 2000 eindeutig bei der Steigerung der Leistungsfähigkeit und vor allem auch bei der Ausbaufähigkeit lag, wurde der Amiga 500 von vorneherein als kompakter, preiswerter Heimcomputer entwickelt. Er vereint Systemeinheit, Tastatur und ein Diskettenlaufwerk in einem kompakten, handlichen Gehäuse. Auch der interne Aufbau wurde vereinfacht und besser gegliedert, was vor allem durch neue, noch leistungsfähigere Chips möglich wurde, die Commodore in der Zwischenzeit entwickelt hat. Alles das hat zur Konsequenz, daß der Amiga 500 wesentlich kostengünstiger gebaut – und deshalb auch wesentlich preiswerter verkauft – werden kann. Nicht nur äußerlich erinnert der Amiga 500 etwas an den C 64 – den erfolgreichsten Computer aller Zeiten – oder dessen Nachfolgemodell C 128. Er ist »handlich«, für jedermann erschwinglich und damit ist in jeder Hinsicht der rechtmäßige Erbe des in die Jahre gekommenen C 64, der nicht mehr ganz dem Stand der Technik entspricht.

Man sollte sich aber von den Äußerlichkeiten dieses kleinen, fast unscheinbaren Computers nicht täuschen lassen. Der Amiga 500 ist ein echter Wolf im Schafspelz. In seinem kompakten Gehäuse enthält er alles (sogar etwas mehr), was in dem wesentlich größeren Amiga 1000 zu finden ist. Das Gehäuse ist das eines Heimcomputers, sein Inhalt und seine Leistungsfähigkeit entsprechen einem vollwertigen PC!

# 2 Erste Handgriffe auf der Workbench

Das Programm, mit dem Sie einen Großteil Ihrer Arbeit am Amiga machen werden, ist die Workbench (oder zu deutsch: Werkbank). Von dieser Werkbank aus werden Sie Ihre Disketten verwalten, Dokumente anlegen, kopieren, umbenennen, ablegen, löschen, Programme starten und so weiter. Prinzipiell finden fast alle Arbeiten, die nicht unmittelbar etwas mit einem sogenannten Anwendungsprogramm zu tun haben, auf der Workbench statt.

Die wichtigsten Eigenschaften der Workbench und der zusammen mit der Workbench ausgelieferten Programme werden Sie in diesem und den folgenden Kapiteln kennenlernen. Beachten Sie dabei bitte, daß die vorliegende Fassung dieses Buchs auf Grundlage der Workbench-Version 1.3 geschrieben wurde. Diese Version ist seit Frühjahr 1989 in Deutschland erhältlich und sollte eigentlich zusammen mit jedem neuen Amiga ausgeliefert werden. Sollten Sie jedoch noch eine ältere Version (zum Beispiel 1.2) der Workbench-Diskette erhalten haben – was Sie an der Diskettenbeschriftung erkennen können – sollten Sie sich die neue Version 1.3 bei Ihrem Händler besorgen. Selbst wenn Sie dafür ein paar DM zusätzlich anlegen müssen – es lohnt sich!

## 2.1 Los geht's: Starten des Amiga 500

Um auf die Workbench zu gelangen, müssen Sie natürlich zunächst einmal Ihren Amiga und den zugehörigen Monitor einschalten. Nachdem Sie dies getan haben, geschieht einen Moment lang scheinbar gar nichts. Der Amiga testet sich in dieser Zeit selbst. Er prüft währenddessen zum Beispiel seinen Speicher daraufhin, ob man aus diesem genau dieselben Zahlen lesen kann, die kurz zuvor hineingeschrieben wurden, schaut nach, ob die Speichererweiterung installiert ist und so weiter. Wenn alles zu seiner Zufriedenheit ausgefallen ist, leuchtet die rote Betriebslampe (Power) rechts oben auf dem Gehäuse auf und er bittet Sie, die Workbench-Diskette einzulegen. Dazu sehen Sie am Bildschirm – sofern dieser korrekt angeschlossen und eingeschaltet wurde – das Bild einer Hand, die eine Diskette hält.

Dieses Bild erscheint nur dann nicht, wenn Sie stolzer Besitzer einer Festplatte sind, diese »autoboot-fähig« ist **und** bereits mit der vom Amiga benötigten Workbench-Software

versehen wurde. In allen anderen Fällen müssen Sie die Workbench-Diskette nach dem Einschalten einlegen, um mit dem Amiga 500 arbeiten zu können. (Mehr zum Thema Festplatte erfahren Sie in den Kapiteln 9 und 10.)

Eine kleine Sicherheitsvorkehrung sollten Sie vor dem Einlegen der Workbench-Diskette beachten: Aktivieren Sie bitte den Schreibschutz dieser Diskette und aller anderen Disketten, die Sie beim Kauf erhalten haben! Dies geht, indem Sie das kleine Plastikplättchen, das in einer der Diskettenecken zu finden ist, so verschieben, daß eine Öffnung entsteht, durch die Sie hindurchsehen können. Sie verhindern auf diese Weise, daß eine der relativ empfindlichen Disketten bei einem kleinen Fehler Ihrerseits, den Sie bei den folgenden Experimenten machen könnten, beschädigt werden kann. (Falls Ihnen diese kurze Erklärung des Schreibschutzes nicht genügt, können Sie eine etwas detailliertere am Ende dieses Kapitels finden.)

Bild 2.1 zeigt Ihnen, wie die Aufforderung, die Workbench-Diskette einzulegen, im wesentlichen aussieht. Achten Sie auf den Text, der – verkehrt herum – auf der Diskette steht.

Selbst wenn Sie ein zweites (externes) Diskettenlaufwerk besitzen, hilft es Ihnen übrigens nichts, die Workbench-Diskette in dieses zu schieben. Der Amiga sucht die Workbench-Diskette nur im internen (eingebauten) Laufwerk an der rechten Seite des Amiga 500!

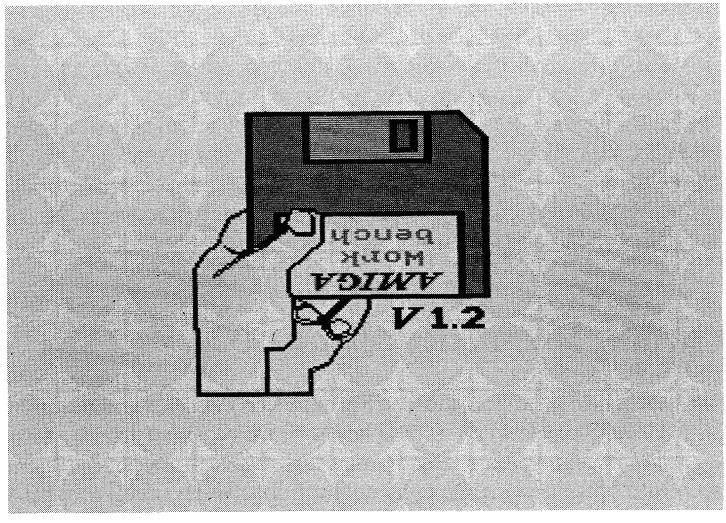



***Bild 2.1:*** *Die Aufforderung zum Einlegen der Workbench-Diskette*

Erscheint das Bild 2.1 erneut auf Ihrem Bildschirm, obwohl Sie die Workbench-Diskette eingelegt haben, so ist Ihre Workbench-Diskette wahrscheinlich defekt. Wenn Sie noch eine

Kopie davon besitzen, versuchen Sie es mit dieser Kopie noch einmal. Falls das auch nichts fruchtet, schalten Sie Ihren Amiga noch einmal ganz aus und nach etwa 10 Sekunden wieder ein. Klappt es auch jetzt nicht, bleibt Ihnen nichts anderes übrig, als sich an Ihren Amiga-Händler zu wenden.

Ist aber alles gut gegangen und der Amiga hat Ihre Workbench-Diskette »verdaut«, so verschwindet Bild 2.1 nach wenigen Sekunden, und das Diskettenlaufwerk beginnt hörbar zu arbeiten. Sie können übrigens immer an zwei Dingen feststellen, daß der Amiga auf die Disketten in einem der Diskettenlaufwerke »zugreift« (Daten davon liest oder darauf schreibt). Zum einen hören Sie dann ein leises, schabendes, schnarrendes oder surrendes Geräusch und zweitens leuchtet das kleine grüne Lämpchen oben rechts auf dem Amiga (neben dem das Wort *Drive* steht) auf. Während Sie eine dieser beiden Tatsachen registrieren, also die Diskettenlampe an ist oder Sie das Diskettengeräusch hören, sollten Sie niemals auf den Auswurfknopf unter dem Diskettenschlitz drücken! Diese Warnung werden Sie auf den folgenden Seiten noch mehrmals lesen. Sie ist aber so wichtig, daß sie einige Wiederholungen verträgt. Gerade die 3,5-Zoll-Disketten des Amiga sind zwar außerordentlich robust. Es bekommt aber keiner Diskette – und auch dem Laufwerk selbst nicht – wenn die Diskette aus dem Laufwerk entfernt wird, während der Computer gerade auf ihre Daten zugreift.

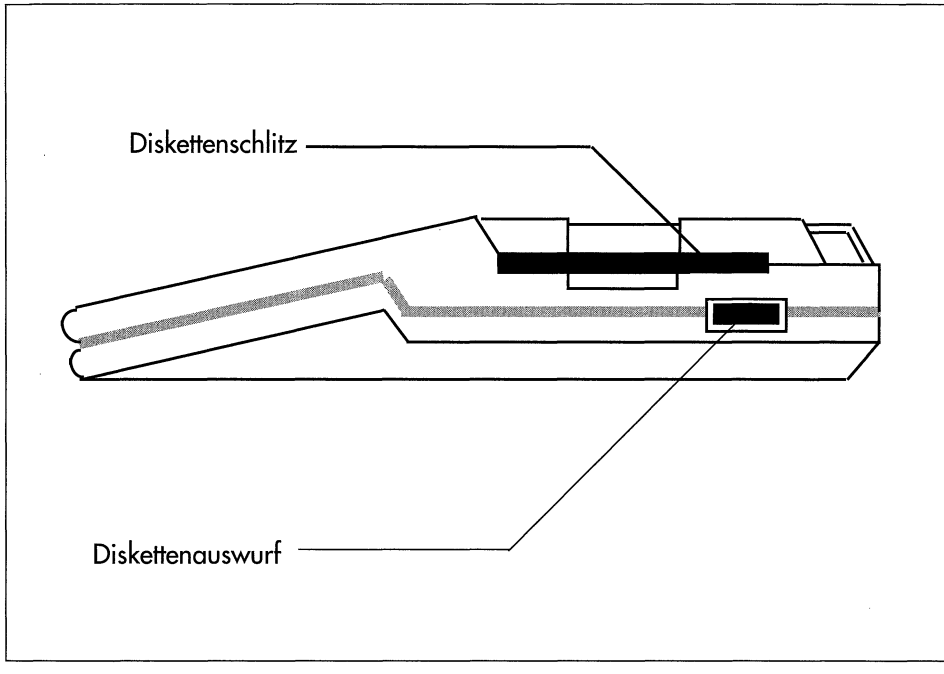



***Bild 2.2:*** *Das eingebaute Diskettenlaufwerk des Amiga 500*

Nach einem kurzen Moment, während dem der Amiga Informationen von der Workbench-Diskette einliest, belebt sich der Bildschirm des Amiga wieder, und es erscheinen eine

Copyright-Meldung und dann einige Meldungen, die Sie jetzt noch nicht verstehen werden. Nach einer Wartezeit von einer knappen Minute taucht dann schließlich die Workbench am Bildschirm auf. Der Bildschirm färbt sich blau und am oberen Bildschirmrand sehen Sie einen weißen Streifen, in dem am linken Rand steht: Workbench release 1.3: 368248 free memory. (Die Zahl vor free memory kann bei Ihnen allerdings eine andere sein.) Dieser Streifen heißt »Titelzeile« oder »Titelleiste« des Bildschirms.

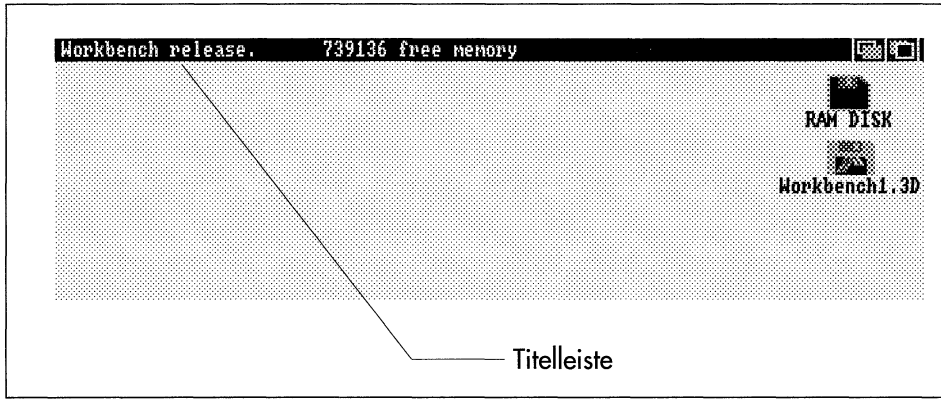



***Bild 2.3:*** *Die Titelleiste des Workbench-Bildschirms*

Am rechten Rand der Titelzeile stehen zwei kleine Symbole, über die Sie in einem späteren Kapitel noch mehr erfahren werden. Falls Sie ein zweites Diskettenlaufwerk besitzen und sich in diesem schon eine Diskette befand, als Sie den Amiga gestartet haben, so erscheinen am rechten oberen Rand der blauen Fläche zwei kleine Bilder, die Disketten zumindest nicht unähnlich sind. Dies sind Diskettenpiktogramme oder Disketten-Icons. (Die kleinen Bildchen, die beim Amiga die verschiedenen Objekte symbolisieren, mit denen wir es zu tun haben, nennen wir generell Piktogramme oder Icons.) Sie befinden sich nun auf der Workbench!

Wollen Sie die Workbench wieder verlassen, also den Amiga ausschalten, nehmen Sie bitte zuvor die Workbench-Diskette und/oder sämtliche anderen Disketten aus dem externen Diskettenlaufwerk. Ein Druck auf den Auswurfknopf an jedem Diskettenlaufwerk macht Ihren Disketten in diesem Fall das Leben leichter. Sie sollten niemals »radikale« Aktionen wie das Ein- oder Ausschalten des Amiga, durchführen, solange sich eine Diskette in einem Laufwerk befindet!

Bevor wir nun in die Tiefen der Workbench eindringen, noch ein Wort zur Terminologie. Der Vorgang, den der Amiga soeben durchlaufen hat, war der Einschaltvorgang. Er beinhaltet das Einschalten des Amiga und das Starten der Workbench mit Hilfe der Workbench-Diskette.

Das Starten der Workbench heißt auch Neustart des Amiga. Der Computerfachmann sagt dazu auch oft »Rebooten«. Selbst nach einem schwerwiegenden Programmfehler, der Ihren Amiga total »in Unordnung« gebracht hat, genügt immer ein Neustart, um wieder sauber von vorne anzufangen.

Der Begriff »booten« oder »rebooten« kommt übrigens von dem englischen Wort »bootstraps« (Schnürsenkel) und hat etwas mit der Vorstellung zu tun, daß man sich selbst an seinen Schnürsenkeln aus einem Sumpf zieht. Im deutschsprachigen Raum bevorzugt man allerdings das Ziehen an den eigenen Haaren – siehe Münchhausen. Wenn man versteht, was beim Start eines Computers mittels Software, die er erst von der Diskette lesen muß, vor sich geht, liegt dieser zunächst absurd erscheinende Vergleich gar nicht einmal so fern.

## 2.2 Wie Sie einen Neustart erzwingen

Sollte Ihnen einmal das schreckliche Ereignis eines solchen Programmabsturzes widerfahren oder sollte es Ihnen passieren, daß Ihr Amiga einmal auf keinen Ihrer Versuche, seine Aufmerksamkeit zu erregen, reagiert, dann können Sie einen Neustart erzwingen. Drücken Sie dazu die Taste mit dem [*A*]-Symbol links und die Taste mit dem [*A*]-Symbol rechts neben der Leer-Taste auf der Tastatur und dann gleichzeitig auf die [Ctrl]-Taste. Sie können die Tasten auch nacheinander drücken; es muß nur einen kurzen Augenblick geben, in dem alle drei zugleich niedergedrückt sind. Bevor Sie dies machen, vergewissern Sie sich, daß keines der Diskettenlaufwerke aktiv ist.

Nachdem Sie die drei Tasten losgelassen haben, blinkt wieder die Betriebsleuchte (die rote) auf dem Gehäuse des Amiga auf und nach kurzer Zeit erscheint die Aufforderung, die Workbench-Diskette einzulegen (Bild 2.1). Verfahren Sie dann wie oben beschrieben und warten Sie, bis erneut der Workbench-Bildschirm mit den zwei Diskettenpiktogrammen auftaucht.

Die Tastenkombination für den Neustart sollte nur in absoluten Notfällen angewandt werden! Alle Daten, die zu dem Zeitpunkt, zu dem Sie den Amiga neu starten, noch nicht auf Disketten gesichert waren, gehen dabei verloren. Aus Versehen kann diese Tastenkombination glücklicherweise aber kaum vorkommen, außer Sie besitzen 20 Zentimeter lange Finger oder lassen Gegenstände auf die Tastatur fallen und halten dabei gleichzeitig Tasten fest.

Probieren Sie den »gewaltsamen« Neustart doch sofort einfach einmal aus. Dabei kann im Augenblick nichts Schwerwiegendes passieren! Warten Sie dazu, bis sämtliche Disketten-Aktivitäten beendet und alle entsprechenden Lampen erloschen sind und nehmen Sie dann die Diskette(n) aus dem (den) Laufwerk(en). Dann drücken Sie die drei Tasten nieder und lassen sie wieder los.

## 2.3 Wozu die Workbench-Diskette gut ist

Bevor wir uns näher mit der Bedienung des Amiga beschäftigen, ein Wort zur Workbench-Diskette. Besonders, wenn Sie schon Kontakt mit anderen Computern, insbesondere Heimcomputern, hatten, werden Sie sich vielleicht wundern, wozu diese Diskette gut ist. Bei vielen anderen Computern – zum Beispiel beim Commodore C 64 – genügt es ja, den Computer einfach einzuschalten und man kann sofort mit der Arbeit beginnen. Beim Amiga ist das ein klein wenig anders.

Jeder Computer benötigt, damit man mit ihm arbeiten kann, ja die sogenannte Software, also Programme. Erst ein Textverarbeitungsprogramm »bringt dem Computer bei«, Ihre Eingaben auf dem Bildschirm darzustellen und gegebenenfalls abzuspeichern. Der Computer selbst, die Hardware, ist ohne Software unfähig. Einen umfangreichen und wichtigen Teil dieser Software werden Sie allerdings niemals bemerken. Diese sogenannte Systemsoftware ist unsichtbar im Hintergrund tätig und erleichtert den Anwendungsprogrammen, wie zum Beispiel der eben schon erwähnten Textverarbeitung, die Arbeit. Die Systemsoftware verwaltet zum Beispiel den Speicher des Computers und teilt ihn den verschiedenen Programmen, die Speicher benötigen, zu. Sie legt auf Wunsch der Anwendungsprogramme Daten auf der Diskette oder Festplatte ab und findet sie später auch wieder. Und schließlich sorgt sie zum Beispiel auch dafür, daß auf einen einfachen Programmbefehl hin, wie *Linie von (10,10) zu (100,100)*, tatsächlich eine Linie auf dem Bildschirm erscheint.

Bei vielen preiswerten Heimcomputern ist die Systemsoftware in Form sogenannter ROMs vollständig in den Computer eingebaut. ROMs sind spezielle Speicherchips, deren Inhalte nur gelesen und nie wieder geändert werden können (ROM = Read Only Memory; oder auf deutsch Nur-Lese-Speicher). Dies hat den Vorteil, daß diese Daten jederzeit – und vor allem auch sofort nach dem Einschalten – zur Verfügung stehen (änderbare Speicher-Chips enthalten nach dem Einschalten nichts). Ein Nachteil dieser Lösung ist allerdings, daß Software in ROMs sehr schlecht geändert werden kann, wenn nachträglich Fehler darin gefunden werden oder wenn Verbesserungen eingeführt werden sollen. Beim Amiga hat man sich deshalb für eine zweigeteilte Lösung entschieden: Ein Teil der Systemsoftware ist fest in ROMs eingebaut und ein anderer Teil befindet sich auf der Workbench-Diskette und wird beim Start eingelesen. Dieser zweite Teil kann sehr leicht geändert und verbessert werden. Trotzdem wurde aus »Bequemlichkeit« nicht alle Systemsoftware auf diese Diskette verlagert. Die Software in den ROMs (256 Kbyte) braucht nämlich beim Start nicht von der Diskette eingelesen zu werden, was den Startvorgang beschleunigt. Oder anders gesagt: Sie brauchen nach dem Einschalten nicht so lange zu warten, bis Sie mit der Arbeit beginnen können!

## 2.4 Ein besonderes Nagetier: Die Maus

Das wichtigste Bedienungselement Ihres Amiga ist die Maus. Sie kann für einen großen Teil aller anfallenden Arbeiten auf der Workbench und auch innerhalb der meisten Programme, die Sie für den Amiga kaufen können, eingesetzt werden. Speziell zusammen mit der Workbench ist die Tastatur nahezu überflüssig – ohne Maus sind Sie hingegen (beinahe) rettungslos verloren. Dies wird Sie wahrscheinlich verblüffen, wenn Sie schon andere Computer kennen. Bei vielen anderen Computern (mit wenigen Ausnahmen) ist die Tastatur nämlich elementar für die ganze Bedienung des Gerätes, und ohne Tastatur wären Sie völlig aufgeschmissen. Auf dem Amiga ist das anders – und deshalb einfacher, wie Sie bald merken werden. Die Maus – mit der zugehörigen Software – hat einige Vorteile gegenüber der üblichen Computer-Bedienung mit Befehlen, die über die Tastatur eingegeben werden müssen.

Dies soll nicht heißen, daß Sie auf dem Amiga Textverarbeitung ohne Tastatur betreiben können. Für viele Anwendungen ist die Tastatur nach wie vor unentbehrlich. Wenn man aber von den Vorgängen auf der Workbench zum normalen (nicht computerisierten) Leben ein-

Analogie zieht, dann kann man sagen, daß Sie immer dann ohne Tastatur auskommen werden, wenn Sie beim entsprechenden Vorgang im wirklichen Leben ohne Schreibmaschine oder Kugelschreiber auskommen würden. Die Tastatur wird beim Amiga wirklich nur für die Texteingabe benutzt, und sonst nicht.

### 2.4.1 Funktionsweise der Maus

Bild 2.4 zeigt Ihnen die Maus des Amiga von oben. Die Ähnlichkeit mit einer lebenden Maus ist nicht gerade frappierend, aber mit etwas Mühe zu erkennen. Ich habe allerdings schon Computer-Mäuse gesehen, die von ihrem Besitzer einen Pelz verpaßt bekommen haben und dann doch recht putzig aussahen. Wenn Sie also ihren Computer gern warm und flauschig bedienen wollen ... (geschickt gemacht, schadet eine solche Pelzhülle der korrekten Funktion überhaupt nicht).

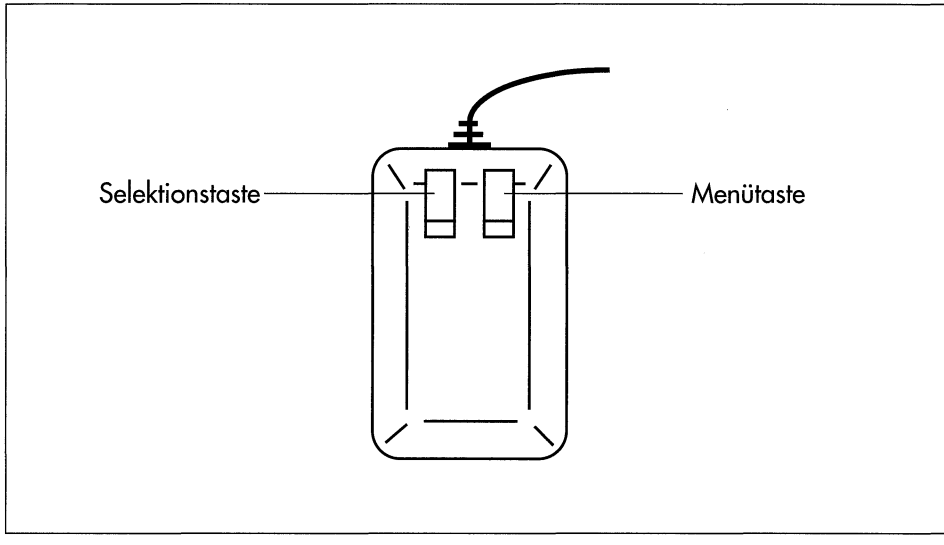



***Bild 2.4:*** *Die Amiga-Maus*

Die Oberseite der Maus ist glatt und sollte gut in die locker darauf gelegte Handfläche passen. Ob sie das auch tut, darüber kann man geteilter Meinung sein – Geschmackssache. Die Form der Amiga-Maus ist aber wohl unbestreitbar schnittig und modern. An ihrer Oberseite befinden sich zwei Tasten. Wird eine dieser Tasten gedrückt, so bewirkt dies bei den meisten Programmen eine bestimmte Aktion – abhängig vom gerade laufenden Programm können es allerdings verschiedene Aktionen sein.

Die linke der beiden Tasten heißt »Aktions-« oder »Selektionstaste«, die rechte »Menütaste«. (Keine Angst, Sie haben sich nicht in ein Kochbuch verirrt. Die Namen der Knöpfe werden weiter unten noch erläutert.)

Die Unterseite der Maus ist (zumindest für ihre Funktionsweise) noch etwas interessanter. Falls Sie dieses Buch nicht deshalb gekauft haben, weil Sie erst noch Besitzer eines Amiga

werden wollen, sondern es schon sind, nehmen Sie einmal die Maus zur Hand und drehen Sie sie auf ihre Rückseite. In der Mitte des Mausbodens sehen Sie ein kleines Loch und darin eine kleine Gummikugel. Immer wenn Sie die Maus – richtig herum – auf eine glatte Oberfläche legen und sie dann bewegen, dreht sich diese Kugel und überträgt diese Drehung auf zwei kleine Rädchen in ihrem Innern. Diese Rädchen senden elektrische Impulse zum Amiga, die besagen, in welche Richtung und wie weit Sie die Maus bewegt haben.

Wie diese Impulse genutzt werden, ist prinzipiell natürlich Sache des gerade laufenden Programms. Meist werden sie aber dazu verwendet, einen kleinen Zeiger auf dem Bildschirm synchron zur Maus zu bewegen.

### 2.4.2 Erste Maus-Spiele

Falls Sie noch nicht mit der Maus vertraut sind, sollten Sie jetzt versuchen, es zu werden. Begeben Sie sich dazu auf die Workbench (wie oben beschrieben) und nehmen Sie die Maus in die Hand. Legen Sie sie dann auf eine glatte Unterlage. Sie sollten dabei eine freie Fläche wenigstens von der Größe eines DIN-A4-Blattes zur Verfügung haben. Später werden Sie lernen, auch mit weniger Platz auszukommen; zunächst kann es aber gar nicht genug Platz für die Maus geben. In welche Hand Sie die Maus nehmen, ist im wesentlichen Geschmacksfrage. Zu Beginn ist es aber wahrscheinlich einfacher, wenn Sie sie als Linkshänder in die linke und als Rechtshänder in die rechte Hand nehmen. Mit ein wenig Übung wird diese Unterscheidung aber bald gegenstandslos werden und Sie werden die Maus mit der Hand bewegen, in deren Nähe der meiste Platz auf dem Schreibtisch ist.

Wenn Sie die Maus nun auf einer ebenen Oberfläche bewegen (und der Amiga eingeschaltet ist), können Sie sehen, wie ein kleiner roter Pfeil auf dem Bildschirm diesen Bewegungen folgt. Dies ist der Mauszeiger – das vielleicht wichtigste Objekt auf dem Bildschirm. Die folgende Abbildung zeigt Ihnen diesen Zeiger in vergrößerter Form – leider nur in Schwarzweiß. Auf dem Bildschirm erscheint er vor allem rot, besitzt in Wirklichkeit aber mehrere Farbtöne, damit er auch über den unterschiedlichsten Hintergrundfarben sichtbar bleibt.

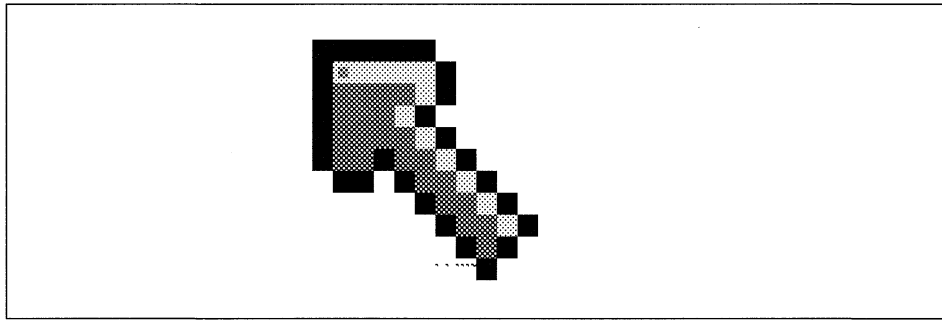


***Bild 2.5:*** *Der Mauszeiger des Amiga*

Versuchen Sie nun einmal, die vier Ecken des Bildschirms mit diesem Zeiger gezielt anzufahren. Dies ist unter Umständen gar nicht so einfach, wenn Sie die Maus nicht richtig

halten. Richtig gehalten wird die Maus immer so, daß ihr Schwanz (der Kabelanschluß) geradlinig auf den Bildschirm weist. Schieben Sie die Maus auf der Unterlage nach vorn, so geht der Zeiger auf dem Bildschirm nach oben. Ziehen Sie sie an Ihren Körper, geht der Zeiger nach unten. Bewegen Sie die Maus nach rechts, und der Zeiger geht nach rechts. Bewegen Sie sie nach links, und der Zeiger folgt nach links.

Drehen Sie die Maus testweise einmal so, daß der Schwanz nach links zeigt, und bewegen Sie sie dann nacheinander in die vier Bildschirmecken. Wie Sie sehen, ist dies viel schwieriger als eben mit dem Schwanz nach vorn, da die Entsprechung von Hand- beziehungsweise Mausbewegung zu Zeigerbewegung nicht mehr stimmt!

Versuchen Sie auch einmal, die Spitze des Mauszeigers genau auf die obere linke Ecke des Diskettensymbols in der rechten oberen Bildschirmecke zu bringen. Sie werden merken, daß es gar nicht so einfach ist, die Handbewegungen mit der Bildschirmdarstellung zu koordinieren und der Versuchung zu widerstehen, permanent auf die Maus zu schauen. Sie werden sich aber bald daran gewöhnen. Die Erfahrung zeigt, daß die Maus nach maximal etwa einer Stunde keinem Anwender mehr Probleme macht. Die Kopplung zwischen Maus und Mauszeiger wird auch Ihnen bald so natürlich erscheinen wie ein direktes Zeigen mit dem Finger. Sie ist sogar so selbstverständlich, daß in diesem Buch manchmal der Fehler gemacht wird, von der Maus zu sprechen, wenn eigentlich der Mauszeiger gemeint ist. Da Maus und Mauszeiger geradezu unzertrennlich sind, ist dieser Fehler hoffentlich verzeihlich.

### 2.4.3 Die Tasten der Maus

Wie die Maus bewegt wird und was diese Bewegung bewirkt, haben wir ja jetzt gesehen. Auf der Oberseite der Maus befinden sich aber auch noch zwei Tasten. Da beide Tasten häufig gebraucht werden, sollten Sie sich eine Handhaltung angewöhnen, mit der Sie beide unabhängig voneinander niederdrücken können. Zeige- und Mittelfinger eignen sich dazu gut. Sie können die Maus dann mit Daumen und Ringfinger sehr gut umfassen und Zeige- und Mittelfinger immer locker auf den Knöpfen liegen lassen. Ganz so locker wird Ihre Handhaltung am Anfang vielleicht nicht sein, aber keine Angst: das legt sich mit der Zeit (und etwas Übung).

Bewegen Sie den Mauszeiger nun einmal in die freie blaue Fläche in der Bildschirmmitte, drücken Sie dann kurz auf den linken Mausknopf und lassen ihn sofort wieder los. Es passiert nichts! Wenn Sie dabei aber auf den Bildschirm schauen, werden Sie feststellen, daß der Amiga Ihren Knopfdruck bemerkt hat. Die Titelzeile des Bildschirms flackert jedesmal leicht, wenn Sie den linken Knopf drücken. Dieser Vorgang des Drückens und Loslassens der Maustaste heißt »Mausklick« und das entsprechende Verb heißt »klicken« (mit der Maus). Merken Sie sich bitte diese beiden Begriffe; Sie werden sie immer wieder benötigen.

Gehen Sie nun mit dem Mauszeiger über das oberste Diskettenpiktogramm in der rechten oberen Bildschirmecke (unter diesem Piktogramm steht der Namen *RAM DISK*). Die Spitze des roten Pfeils sollte sich über der weißen Fläche des Symbols befinden. Drücken Sie nun wieder kurz die linke Maustaste und lassen sie sofort wieder los. Das Piktogramm wechselt daraufhin seine Farbe. Im folgenden heißt dieser Vorgang immer »Selektieren«, »Auswählen« oder auch »Markieren eines Objekts«. Die linke Taste heißt dementsprechend Selektions-Taste oder Aktions-Taste.

Bewegen Sie den Zeiger nun wieder auf die blaue Bildschirmfläche und klicken Sie erneut mit dem linken Knopf. Das Diskettensymbol erhält daraufhin wieder seine alten Farben. Sie haben das Symbol »deselektiert«. Versuchen Sie es zur Übung gleich noch einmal: Anklicken des Diskettenpiktogramms (Selektieren) und dann Anklicken der blauen Fläche (Deselektieren).

Sie haben damit schon eine wesentliche Operation zur Bedienung der Workbench und des Amiga erlernt: das Auswählen eines Objektes zur Bearbeitung. Wenn Sie später irgendeinen Befehl geben (zum Beispiel »kopiere«) müssen Sie dem Amiga zuvor immer sagen, mit welchem Objekt er diesen Befehl durchführen soll (*was* er kopieren soll). Das vorherige Anklicken eines oder mehrerer Objekte erfüllt genau diesen Zweck. Sie suchen sich zum Beispiel eine Diskette aus und sagen dem Amiga dann »kopiere (diese Diskette)«.

Mit dem linken Knopf können Sie aber nicht nur Objekte auswählen und wieder deselektieren, sondern unter anderem auch Objekte bewegen. Dazu müssen Sie die linke Maustaste drücken und dann festhalten. Probieren Sie es aus: Bewegen Sie den Mauszeiger über eines der beiden Diskettensymbole, drücken Sie die linke Taste und halten sie gedrückt. Sie sehen, wie sich die Farben des Diskettensymbols ändern, sobald die Taste niedergedrückt wird. Bewegen Sie nun diesen Zeiger in die Mitte des Bildschirms und halten dabei die Maustaste immer unten. Das Diskettensymbol folgt dieser Bewegung.

Dieser Vorgang heißt »Ziehen der Maus«. Beim Ziehen bleibt im Gegensatz zum einfachen Bewegen die Maustaste gedrückt, während die Maus beziehungsweise der Mauszeiger bewegt wird. Sobald Sie die Maustaste nun loslassen, bleibt das Diskettensymbol an der Stelle liegen, an der sich auch der Mauszeiger in diesem Moment befand. Sie haben das Diskettenpiktogramm auf diese Weise verlegt. Versuchen Sie, es nun wieder zurückzulegen. Dazu »ergreifen« Sie das Piktogramm wieder mit der Maus (drücken die Maustaste nieder, während der Zeiger sich über dem Piktogramm befindet) und bewegen den roten Mauszeiger in die rechte obere Bildschirmecke. Dort lassen Sie die Maustaste los, und die Diskette liegt wieder (fast) an ihrer alten Stelle.

Wir werden später noch eine Reihe anderer Piktogramme kennenlernen, die auf der Workbench des Amiga erscheinen können. Alle diese Piktogramme haben gemeinsam, daß man sie mit der linken Maustaste zur Bearbeitung auswählen (selektieren) und sie bei gedrückter linker Maustaste auch ergreifen und bewegen kann.

Die Maus besitzt aber noch eine zweite Taste. Diese rechte Taste, die sogenannte »Menütaste«, haben wir überhaupt noch nicht verwendet. Falls Sie sie schon aus Versehen gedrückt haben, werden Sie gesehen haben, daß sie nicht dieselbe Funktion wie die linke hat – die obigen Beispiele sind »mit rechts« alle nicht durchführbar. Vielleicht haben Sie aber schon eine Veränderung auf dem Bildschirm bemerkt, als Sie die Menütaste herunterdrückten. Falls nicht, tun Sie es jetzt noch einmal.

Sobald Sie mit der rechten Taste klicken (und solange Sie sie festhalten), ändert sich die Titelzeile des Bildschirms. Anstelle des Textes *Workbench release 1.3...* erscheinen darin die drei Worte *Workbench*, *Disk* und *Special*. Dies sind die Titel der drei »Menüs«, aus denen Sie die Befehle der Workbench auswählen können. Solange die Menütitel sichtbar sind, nennt man die Titelzeile des Bildschirms manchmal auch Menüzeile, um ihre spezielle Funktion noch etwas zu betonen. Menüs sind eine sehr bequeme Möglichkeit, einem Computer Befehle zu erteilen und zudem eine, die Eingabefehler fast unmöglich macht. In

den nächsten Abschnitten werden Sie noch mehr darüber erfahren. Fast alle Amiga-Programme verwenden solche Menüs.

## 2.5 Was darf es sein? – Die Menüs des Amiga

Mit diesen Menüs wollen wir uns nun etwas näher beschäftigen. Sie haben übrigens ihren Namen daher, daß sie dem Benutzer des Computers – also Ihnen – die zur Verfügung stehenden Programmbefehle anbieten, wie eine Speisekarte die Gänge eines Menüs zur Wahl anbietet. Guten Appetit also zu dieser Mahlzeit!

### 2.5.1 Wie Sie die Befehls-Menüs sichtbar machen

Um die Menüs des Amiga näher kennenzulernen, halten Sie zunächst einmal wieder die Menütaste (die rechte) der Maus nieder und bewegen Sie dann den Mauszeiger in die Titelzeile über das erste Wort *Workbench*. Das Bild, das erscheint, sobald der Zeiger sich über dem Wort *Workbench* befindet, dürfte ungefähr so aussehen wie Bild 2.6.




***Bild 2.6:*** *Das Workbench-Menü*

Ein weißes Rechteck »klappt« unter dem Menütitel hervor und zeigt Ihnen einige der Befehle, die die Workbench anbietet. Es kann allerdings durchaus sein, daß einige Befehls-Namen, die Sie sehen, etwas schlecht lesbar sind. In Bild 2.6 trifft das zum Beispiel für den Befehl Discard zu. (Die Gründe dafür werden Sie später noch erfahren. Zunächst soll uns das einmal nicht zu sehr kümmern.)

Bewegen Sie den Mauszeiger (bei gedrückter Menütaste) nun nach rechts über das nächste Wort in der Menüzeile: *Disk*. Belassen Sie den Mauszeiger unbedingt in der Titelzeile und halten Sie die Maustaste gedrückt! (Falls Ihr Zeigefinger erlahmen sollte, können Sie den

Mauszeiger über die freie blaue Fläche bewegen und dann die Menütaste loslassen. Beginnen Sie dann nach einer Erholungszeit wieder von vorne.) Sobald sich der Mauszeiger über dem Wort *Disk* befindet, verschwindet das erste Menü und ein neues taucht auf. Wenn Sie den Mauszeiger weiter nach rechts bewegen, wiederholt sich der Vorgang: das zweite Menü verschwindet und ein drittes erscheint unter dem Wort *Special*. Das ist eine gute Methode, ein Programm kennenzulernen. Bewegen Sie die Maus in der Titelzeile hin und her und Sie können alle Befehle, die Ihnen in einem Programm – in diesem Fall dem Workbench-Programm – zur Verfügung stehen, schnell überblicken.

Auch für die Zukunft ist es eine gute Angewohnheit, bei gedrückter Menütaste über die Titelzeile des Bildschirms zu fahren, wenn man ein neues Programm kennenlernen möchte oder nicht mehr weiter weiß. Dies ist eine unschätzbare Orientierungshilfe in jedem Programm. Das Schöne an den Menübefehlen des Amiga ist auch, daß man jederzeit sieht, welche Befehle gerade sinnvoll sind und welche nicht. Die in einem Moment nicht möglichen oder sinnlosen Befehle erscheinen stets gerastert oder »geisterhaft«, während die Befehle, die sinnvoll sind und die Sie wirklich auswählen können, fett und gut lesbar sind. Solange Sie keine Objekte (Piktogramme) in der Workbench ausgewählt haben, sind allerdings fast alle Einträge in den drei Menüs grau.

### 2.5.2 Wie Sie einen Menübefehl erteilen

Bis jetzt haben wir uns ja nur den Inhalt der Menüs angeschaut, jedoch noch nichts damit gemacht. Nun aber geht es »zur Sache«. Genauer gesagt, werden Sie jetzt einige Menübefehle ausprobieren. Rekapitulieren Sie dazu noch einmal, wie ein Menü aufgebaut ist: Jedes Menü hat einen Titel, der in der Titelzeile (oder »Menüzeile«) zu sehen ist, sobald die Menütaste der Maus (die rechte Taste) gedrückt wird. Bewegt man dann die Spitze des Mauszeigers bei gedrückter Menütaste über einen Menütitel, klappt das entsprechende Menü herunter. In jedem Menü erscheint eine Reihe von kurzen Worten. Dies sind die Namen der einzelnen Menübefehle, die die Workbench versteht. Manchmal nennt man diese Worte statt »Menübefehl« auch »Menüpunkt«.

Klappen Sie nun einmal das Menü *Special* herunter und belassen Sie den Mauszeiger über dem Namen *Special*. Wie Sie sehen können, stehen Ihnen in diesem Menü die Befehle die Befehle *Clean Up* (engl. für aufräumen), *Last Error* (engl. für letzter Fehler), *Redraw* (engl. für neu zeichnen) und *Version* (engl. für Versionsnummer zeigen) zur Verfügung. Einige dieser Befehle sind wahrscheinlich grau, können also im Augenblick nicht ausgewählt werden. Der Befehl *Version*, der Ihnen die Versionsnummer der Betriebssystem-Software zeigt, wird jedoch niemals grau, steht also immer zur Verfügung. Genau diesen Befehl werden Sie jetzt aufrufen.

Bewegen Sie dazu den Mauszeiger bei gedrückter Menütaste innerhalb des weißen Kastens, der die Menüpunkte enthält, nach unten. Jeder Menüpunkt, der dabei unter den Mauszeiger gerät und nicht grau gezeichnet war, wechselt dabei seine Farbe. Er erscheint orange auf schwarz statt blau auf weiß. Probieren Sie es einmal aus, indem Sie den Mauszeiger im Menü *Special* auf und ab bewegen. Lassen Sie dabei die Menütaste aber bitte nicht los! Bewegen Sie dann den Mauszeiger aus dem Rechteck des Menüs heraus. Sobald der Zeiger das Menü verläßt, werden alle Befehle wieder normal gezeichnet; das Menü bleibt aber sichtbar, solange die Menütaste gedrückt bleibt.

Bewegen Sie nun den Zeiger über das Wort *Version*, so daß es orange auf schwarz geschrieben wird und lassen Sie die Menütaste los. Und nun stellen Sie sich bitte einen kleinen Tusch vor, denn soeben haben Sie dem Amiga den ersten Befehl erteilt! Er führt ihn auch sofort aus. Der Bildschirm flackert kurz auf und zeigt Ihnen dann oben links in der Titelzeile die genaue Versionsnummer der Workbench (zum Beispiel *Kickstart Version 34.5. Workbench Version 34.20*). Zur Eingewöhnung versuchen Sie es am besten gleich noch einmal.

Jedesmal, wenn ein Menüpunkt orange auf schwarz hervorgehoben wird und Sie in diesem Augenblick die Menütaste loslassen, erteilen Sie dem Amiga einen Befehl. Sie brauchen diesen Befehl also nicht – wie bei anderen Computern – sorgfältig einzutippen, sondern wählen ihn einfach aus dem Menü aus; genauso, wie Sie auch ein Gericht aus der Speisekarte eines Restaurants auswählen würden.

Der *Version*-Befehl ist ein Menübefehl, der keine zusätzlichen Informationen über die Objekte, mit denen er arbeiten soll, benötigt. Dies stellt unter den Menübefehlen aber eher eine Ausnahme dar. Fast alle anderen Befehle benötigen die Angabe eines oder mehrerer Objekte, mit denen der Befehl etwas tun soll. (Solche Angaben nennt der Computerfachmann »Argumente« oder »Parameter« des Befehls.) Wir wollen uns nun einmal einen solchen Befehl mit Argumenten anschauen.

Selektieren Sie dazu das einzige Piktogramm *Workbench1.3D* oben rechts am Bildschirm (durch einen Klick mit der linken Taste auf dieses Piktogramm). Drücken Sie dann wieder die Menütaste und fahren Sie mit dem Mauszeiger über das Wort *Workbench* in der Menüzeile. Der erste Befehl in diesem Menü heißt *Open* (engl. für öffnen). Bewegen Sie den Mauszeiger über das Wort *Open* und lassen Sie die Menütaste los. Sie haben dem Amiga dadurch sinngemäß den Befehl »Öffne diese Diskette« gegeben.

Das erste, was geschieht, ist, daß ein weißer Rahmen auf dem Bildschirm erscheint. Dieser Rahmen trägt am oberen linken Rand den Titel *Workbench1.3D* und es tauchen nun langsam – unter hörbarem Arbeiten des Diskettenlaufwerks – nacheinander eine Reihe von Piktogrammen darin auf: einige Schubladen, eine Uhr, ein Mülleimer und ein Piktogramm, das dem Amiga selbst verdächtig ähnelt. Dieser weiße Rahmen und die Fläche in seinem Innern ist ein sogenanntes »Fenster«, in dem uns der Inhalt der Diskette namens Workbench gezeigt wird.

Diese Eigenschaft des *Open*-Befehls, die hier demonstriert wurde, sollten Sie sich unbedingt merken. Obwohl Sie den *Open*-Befehl auf die verschiedensten Objekte anwenden können, zeigt er Ihnen stets (meist in einem Fenster), was sich in diesem Objekt verbirgt. Das Pendant zum Befehl *Open* (öffnen) ist natürlich der Befehl *Close* (engl. für schließen). Dieser Befehl (*Close*) findet sich im Menü Workbench direkt unter *Open*. Wählen Sie ihn nun aus. Wie Sie sehen, verschwindet das Fenster daraufhin sofort und Sie blicken wieder auf die blaue Fläche der Workbench.

Wenn Sie den Befehl *Close* nicht auswählen können, weil er blaß erscheint, haben Sie, nachdem Sie Open gewählt haben, wahrscheinlich zwischendurch einmal auf die Selektionstaste gedrückt. Das macht aber gar nichts. Bewegen Sie dann den Mauszeiger über das Diskettenpiktogramm und drücken Sie einmal auf die Selektionstaste (Sie haben die Diskette dadurch natürlich selektiert). Nun sollten Sie *Close* auswählen können.

Klicken Sie nun wieder das Diskettenpiktogramm der Workbench-Diskette an und wählen Sie erneut *Open* aus. Beachten Sie diesmal auch, was dabei mit dem Mauszeiger geschieht. Sie werden bemerken, daß er seine Form ändert, während sich das Fenster, das aus dem Diskettenpiktogramm hervorgegangen ist, langsam mit Objekten füllt. Er wird zu einer Art Sprechblase (wie in einem Comic-Heftchen), in der zwei »Z« stehen. Der Amiga sagt Ihnen auf diese Weise, daß er beschäftigt ist (beziehungsweise ein kleines Nickerchen macht) und nicht mehr sofort auf alle Ihre Eingaben reagieren kann. Immer, wenn Sie diese kleine Sprechblase sehen, ist ein Programm mit einer etwas länger andauernden Tätigkeit beschäftigt. Falls Sie zur Ungeduld neigen, wird Sie diese Sprechblase bestimmt bald etwas nerven. Manche Dinge dauern nun aber einmal etwas länger und leider können Sie auch recht wenig dagegen tun. Der Kauf einer – bedauerlicherweise meist recht teuren – Festplatte lindert solche Beschwerden allerdings erheblich.

Hier wollen wir nun die Besprechung der Menüs erst einmal abbrechen und uns den Fenstern des Amiga zuwenden – neben der Maus und den Menüs wohl dem dritten wichtigen Konzept der Amiga-Software.

## 2.6 Schöne Aussichten: Fenster

Die Fenster des Amiga besitzen einige interessante Eigenschaften, die die Arbeit mit diesem Computer wesentlich erleichtern. Unter anderem ermöglichen sie es, »mehr Informationen gleichzeitig auf dem Bildschirm erscheinen zu lassen als eigentlich Platz haben«. Sollten Sie – berechtigte – Zweifel an dieser Aussage haben, legen Sie dieses Buch bitte trotzdem nicht beiseite. Die Behauptung steht ja in Anführungsstrichen und ist also mit Vorsicht zu genießen. Ein Quentchen Wahrheit steckt aber trotzdem darin!

### 2.6.1 Wie Sie Fenster verschieben

Das einfachste, was wir mit einem Fenster machen können, das auf den *Open*-Befehl hin erschien, ist, es zu bewegen. Die Bewegung eines Fensters geht ganz ähnlich vonstatten wie das Bewegen des Diskettenpiktogramms in einem der vorangegangenen Abschnitte. Während man das Disketten-Symbol aber einfach an beliebiger Stelle irgendwo mit der Maus greifen konnte, sind Fenster etwas wählerischer. Öffnen Sie jetzt am besten gleich ein Fenster, um sich davon zu überzeugen. Klicken Sie dazu auf das Diskettenpiktogramm xx und wählen Sie dann den Befehl *Open* aus dem Menü *Workbench*. Das gewünschte Fenster erscheint dann am Bildschirm.

Wichtig für viele Operationen mit einem Fenster ist die »Titelleiste«. Dies ist der breite Streifen am oberen Fensterrand, der unter anderem den Namen des Fensters am linken Rand enthält. Diese Titelzeile samt Fensternamen ist zunächst vielleicht etwas undeutlich und gerastert – ähnlich wie ein Menübefehl, der nicht ausgewählt werden kann. Klicken Sie, wenn das bei Ihnen der Fall ist, auf die blaue Fläche im Innern des Fensters (nicht auf eines der Symbole darin). Das Fenster wird dadurch »zur weiteren Bearbeitung« selektiert. Die vorher schlecht leserliche Schrift des Fenster-Titels wird klar und die kleinen blauen Pünktchen neben dem Namen werden zu zwei blauen Linien. Dies bedeutet, daß das Fenster nun selektiert ist.

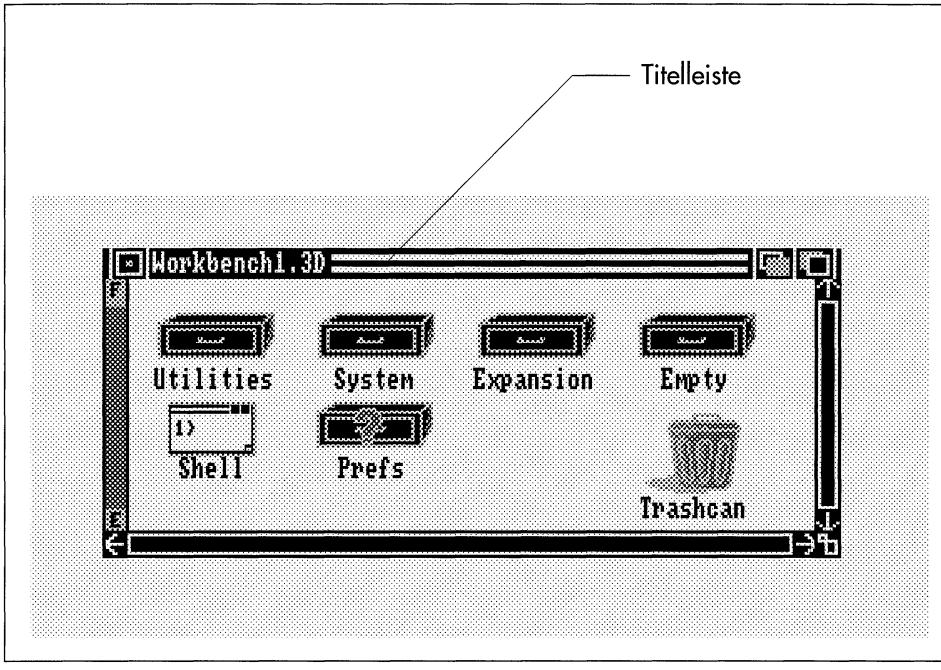



***Bild 2.7:*** *Ein Fenster*

Wenn Sie jetzt mit der **linken** Maustaste auf den Namen oder die vier blauen Linien in der Titelzeile des Fensters klicken, ergreifen Sie das Fenster – genauso, wie Sie ein Diskettenpiktogramm ergreifen können, indem Sie darauf klicken. Halten Sie die Maustaste fest und bewegen Sie die Maus. Den Mausbewegungen folgt jetzt ein orangefarbenes Rechteck, welches den Rahmen des Fensters symbolisiert. Lassen Sie die Maustaste in irgendeinem Augenblick los, so springt auch der Rest des Fensters samt Inhalt an die Stelle, an der sich der orangefarbene Rahmen in diesem Augenblick befand.

Experimentieren Sie etwas mit dem Bewegen von Fenstern. Es ist eine Sache, die in vielen Programmen häufig gebraucht wird. Prägen Sie sich alles, was Ihnen dabei auffällt, gut ein. Achten Sie zum Beispiel darauf, daß die Umrißlinie des Fensters immer komplett auf dem Bildschirm bleibt. Sie stößt bei der Bewegung am Bildschirmrand an und auch der Mauszeiger kann dann in dieser Richtung nicht weiter bewegt werden. Fenster müssen immer komplett auf dem Bildschirm sichtbar sein!

### 2.6.2 Überlappende Fenster

Eine der wichtigsten Eigenschaften der Fenster ist die Fähigkeit, sich zu überlappen. Ein Fenster kann ein anderes ganz oder teilweise verdecken, wie ein Blatt Papier auf dem Schreibtisch ein anderes Blatt ganz oder teilweise verdecken kann. Wendet sich das Interesse wieder einem (teilweise) verdeckten Fenster zu, so zieht man es auf dem Bildschirm einfach zur Seite, damit wieder mehr oder alles davon sichtbar wird.

Um das ausprobieren zu können, brauchen wir natürlich ein zweites Fenster. Dazu klicken wir im Fenster *Workbench* das Schubladenpiktogramm an, unter dem der Name *System* steht, und wählen dann wieder den Befehl *Open* aus dem Menü Workbench aus. Daraufhin erscheint ein neues kleines Fenster am Bildschirm. Schubladen enthalten – wie es sich gehört – andere Dinge, die sichtbar werden, wenn man sie öffnet. Was das Fenster enthält, welches aus der Schublade System hervorgegangen ist, soll uns hier zunächst nicht weiter interessieren. Wir benötigen nur das Fenster selbst. Um noch ein Fenster zu erhalten, öffnen Sie jetzt bitte auch die Schublade Utilities.

Nehmen Sie nun abwechselnd die beiden Fenster und schieben Sie sie auf dem Bildschirm hin und her. Sie sehen dabei sehr deutlich, daß jedes dieser Fenster ein »Blatt für sich« ist. Die Symbole des Fensterinhalts, die verdeckt wurden, erscheinen sofort wieder, sobald der entsprechende Teil des Fensters wieder freigelegt wird. Spinnen wir den Vergleich von Fenstern mit Blättern Papier auf einem Schreibtisch etwas weiter, so müßte es auch bei Fenstern eine Möglichkeit geben, eines ganz oben auf »den Stapel« oder »nach hinten« zu legen. Bis jetzt können Sie Fenster ja nur gegeneinander verschieben. Aber auch die Reihenfolge der Fenster »im Stapel« kann auf dem Amiga geändert werden.

### 2.6.3 Fenster-Gadgets

Zum Ändern der Reihenfolge der Fenster »im Stapel« dienen die beiden kleinen Symbole am rechten Rand der Titelleiste. Es sind sogenannte *Gadgets* (ausgesprochen *Gädschets*; zu deutsch etwa Geräte oder Instrumente), die, ähnlich wie eine Menüauswahl, einen Befehl an ein Programm schicken, wenn sie angeklickt werden. Das linke der beiden Symbole heißt Back-Gadget und legt ein Fenster »ganz nach hinten«, das rechte Symbol heißt Front-Gadget und kann ein Fenster »ganz nach oben« holen.

Verteilen Sie die drei Fenster nun so auf dem Bildschirm, daß sie sich gegenseitig überdecken, aber von jedem wenigstens ein »Zipfelchen« der Titelleiste zu sehen ist. Nun können Sie die Gadgets in Ruhe ausprobieren. Klicken Sie in das Back-Gadget und das Fenster fällt nach hinten. Alle Teile anderer Fenster, die von diesem verdeckt wurden, werden wieder sichtbar. Ist das Front-Gadget eines teilweise verdeckten Fensters sichtbar, so klicken Sie darauf und beobachten Sie, wie es ganz nach oben kommt und sein kompletter Inhalt sichtbar wird.

Interessiert Sie der Inhalt eines Fensters überhaupt nicht mehr, so können Sie das Fenster ganz schließen. Hierzu können Sie natürlich den Befehl *Close* verwenden, der jedes Fenster wieder schließen kann, das mit Open geöffnet wurde. Es gibt allerdings auch eine »Abkürzung«, die Ihnen den ach so langen Weg in die Menüzeile spart. Klicken Sie in das kleine quadratische Gadget am linken Rand der Titelzeile eines Fensters, so wirkt dies, als hätten Sie das Fenster selektiert und dann den Befehl *Close* ausgewählt. Dieses Gadget heißt Schließ-Gadget (oder englisch *close gadget*).

Allen Gadgets, die Sie bis jetzt kennengelernt haben, ist eine Eigenschaft gemeinsam, die Sie sofort austesten können. Diese Gadgets bewirken nämlich nur dann etwas, wenn Sie die linke Maustaste niederdrücken, während sich der Mauszeiger über dem Gadget befindet, und die Taste dann aber auch wieder loslassen, während sich die Spitze des Mauszeigers immer noch innerhalb des Gadgets befindet. Probieren Sie es einfach aus. Wenn Sie zum Beispiel in ein Front-Gadget klicken und die Aktionstaste festhalten, können sie sehen, wie

sich die Färbung des Gadgets ändert. Bewegen Sie den Mauszeiger bei gedrückter Taste vom Gadget fort, erhält es die ursprüngliche Färbung wieder. Wenn Sie jetzt die Taste loslassen, geschieht gar nichts. Bewegen Sie aber vor dem Loslassen der Taste die Maus wieder über das Gadget, so schlägt dessen Farbe um. Wenn Sie jetzt die Taste loslassen, »wirkt das Gadget« und das Fenster kommt nach oben.

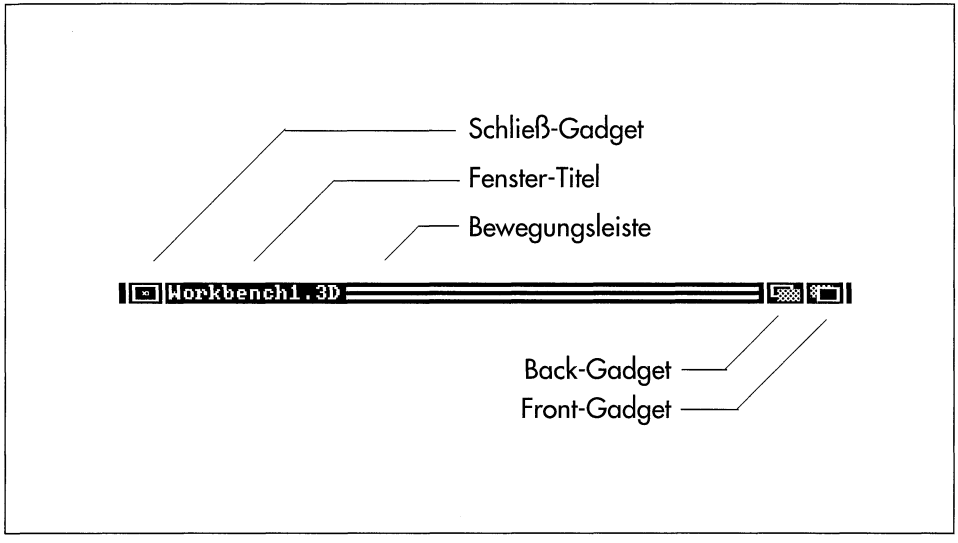



***Bild 2.8:*** *Die Titelleiste eines Fensters*

Somit haben Sie jetzt mehrere übereinanderliegende Blätter auf dem Bildschirm, von denen jedes fast so groß werden kann wie ein ganzer Bildschirm und von denen Sie beliebig eines nach oben holen oder auch ganz beiseite legen (schließen) können. Die Behauptung, mit Fenstern könne man mehr Informationen auf dem Bildschirm zeigen, als eigentlich Platz haben, klingt damit also weniger absurd.

Fenster können noch eine ganze Menge anderer Eigenschaften haben. Für dieses Kapitel haben wir uns aber lange genug mit Fenstern beschäftigt. Zum Abschluß des Kapitels sollen Sie jetzt noch eine ganz elementare Operation kennenlernen, die ein Computerbesitzer gar nicht oft genug durchführen kann.

## 2.7 Sicher ist sicher: Kopien

Eine Fähigkeit, die jeder Anwender eines Computers möglichst zu Anfang seines Kontakts mit einem neuen Computer-Modell erlernen sollte, ist das Anfertigen von Sicherungskopien. Die 3,5-Zoll-Disketten des Amiga sind zwar ein relativ robustes Speichermedium – aber bekanntlich hält nichts ewig. Bereits eine kleine Unachtsamkeit Ihrerseits oder ein kleiner Materialfehler können ihr schaden. Unter den Begriff Unachtsamkeit würde zum Beispiel fallen, wenn Sie eine Diskette auf einen Lautsprecher, ein Gerät mit einem starken

Elektromotor, unter ein Telefon oder auf die Heizung legen. Am gefährlichsten sind alle Ablagen, bei denen die Diskette einem starken magnetischen Feld ausgesetzt wird (wie es Elektromotoren und Telefonklingeln zum Beispiel erzeugen). Aber auch von den modischen Halogenlampen mit eingebautem Transformator, die auf mehr und mehr Schreibtischen zu finden sind, sollten Sie die Disketten fernhalten.

Die möglichen Schäden, die dabei auftreten, können unter Umständen in kleinem Rahmen bleiben. Sie könnten zum Beispiel nur ein Dokument betreffen, das Sie auf einer Diskette gespeichert haben, es könnte aber auch eine ganze Diskette unbrauchbar werden, so daß alle Daten darauf verlorengehen. Wenn Sie dann keine Sicherungskopie dieser Diskette haben, ist möglicherweise die Arbeit von Wochen verloren.

Deshalb gilt die Grundregel Nummer 1: Man kann gar nicht oft genug Kopien seiner wichtigen Disketten anlegen!

Dies gilt in dieser Form natürlich nur für Disketten, bei denen sich die darauf gespeicherten Daten andauernd ändern. Eine Diskette wie die Workbench-Diskette, die Sie nie oder nur sehr selten ändern, muß nur einmal beziehungsweise einmal nach jeder größeren Änderung kopiert werden. Üblicherweise verfährt man dabei so, daß man nach dem Kopieren das Original an einer sicheren Stelle unterbringt (am besten so, daß auch neugierige Kinder nicht daran kommen können) und mit der Kopie weiterarbeitet. Man spricht in diesem Zusammenhang oft auch von einer »Arbeitskopie«.

Von besonders wichtigen oder schwer wieder zu beschaffenden Disketten sollten Sie sogar mindestens 2 Kopien anlegen.

### 2.7.1 Vorbereitungen

Bevor Sie mit dem eigentlichen Kopieren anfangen, müssen Sie zunächst einige Vorbereitungen treffen. Legen Sie die Original-Diskette, also die Diskette, die Sie kopieren wollen, bereit und sehen Sie nach, ob sie eindeutig beschriftet ist. »Eindeutig« heißt in diesem Fall, daß die Beschriftung den Inhalt deutlich beschreibt und sie nicht mit einer anderen Diskette verwechselt werden kann. Falls dies nicht der Fall ist, holen Sie die eindeutige Beschriftung nach – zum Beispiel mit einem permanenten (!) Faserschreiber.

Aktivieren Sie nun den Schreibschutz der Original-Diskette, falls Sie das noch nicht getan haben! Schieben Sie dazu den kleinen Plastik-Schieber, den Sie in einer Ecke der Diskette finden, zum Rand der Diskette, so daß dadurch ein Loch entsteht, durch das Sie hindurchschauen können. (Der Schreibschutz-Schieber befindet sich in der rechten oberen Ecke der Diskette, wenn Sie auf die Diskettenoberseite schauen und der Metallschieber nach unten zeigt.) Dieser Schreibschutz macht es physikalisch unmöglich, daß bei einem Fehler während des Kopiervorgangs irgend etwas am Original zerstört wird.

Legen Sie dann die Diskette bereit, die die Kopie aufnehmen soll. Es kann eine neue Diskette frisch aus der Packung sein oder eine schon gebrauchte, deren Inhalt nicht mehr wichtig ist. Versehen Sie sie mit exakt derselben Beschriftung wie die Original-Diskette und setzen Sie noch groß KOPIE dahinter (oder BACKUP, wenn Sie professionell wirken wollen). Achten Sie darauf, daß der Schreibschutz der Kopie nicht aktiviert ist! Der Schreibschutz-Schieber muß also in einer Position stehen, die das kleine Loch in der entsprechenden Diskettenecke verdeckt, so daß Sie nicht mehr hindurchschauen können.

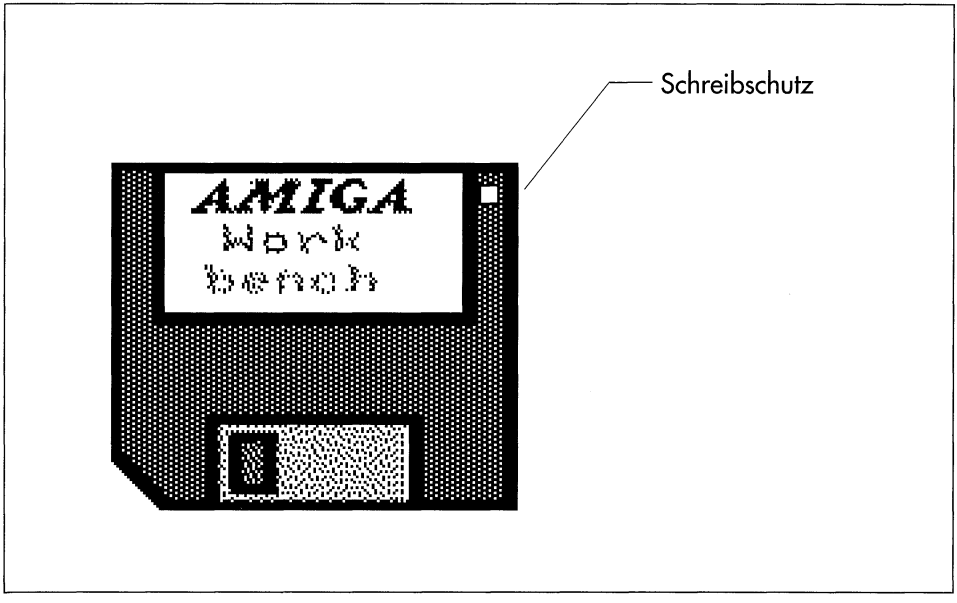



***Bild 2.9:*** *Eine Diskette und der Schreibschutz-Schieber*

Nun sind Sie bestens gerüstet, für »die Dinge, die da kommen mögen«. Sind Sie den Anweisungen exakt gefolgt, so können Sie keinen Fehler machen, der Ihre wertvolle Original-Diskette beschädigt.

Die Anfertigung einer Sicherungskopie läuft auf der Workbench ab. Starten Sie deshalb nun, falls nötig – den Amiga, legen Sie die Workbench-Diskette ein und warten Sie, bis die Workbench erscheint. Falls Sie sich bereits auf der Workbench befinden, schließen Sie alle Fenster und beenden Sie alle Programme, sofern Sie welche gestartet haben.

Der genaue Ablauf des Kopiervorgangs hängt nun stark davon ab, ob Sie ein oder zwei Diskettenlaufwerke besitzen. Naturgemäß ist es mit einem zweiten Laufwerk einfacher. Vielleicht ist das ein Anreiz für Sie, sich ein solches anzuschaffen.

### 2.7.2 Kopieren mit einem Diskettenlaufwerk

Besitzen Sie nur ein Diskettenlaufwerk, so steht Ihnen eine kleine Tortur bevor. Das Kopieren mit einem Diskettenlaufwerk ist kein reines Vergnügen.

Falls Sie nicht die Workbench-Diskette selbst kopieren wollen, werfen Sie diese zunächst erst einmal aus. Halten Sie sie aber bereit; sie wird gleich wieder gebraucht. Schieben Sie die Diskette ein, die Sie kopieren wollen und warten Sie, bis das entsprechende Symbol am Bildschirm erscheint.

Selektieren Sie nun die eben eingelegte Original-Diskette (mit der linken Maustaste) und drücken Sie dann die Menütaste der Maus. Gehen Sie mit dem Mauszeiger über das Wort *Workbench* in der Menüleiste und fahren Sie dann – bei immer noch gedrückter Taste – über

den Befehl *Duplicate*. (Falls dieser grau ist und nicht aktiviert werden kann, ist die Diskette nicht mehr selektiert. Klicken Sie erneut auf das Piktogramm der zu kopierenden Diskette.) Lassen Sie die Maustaste los, wenn das Wort *Duplicate* orange auf schwarz erscheint.

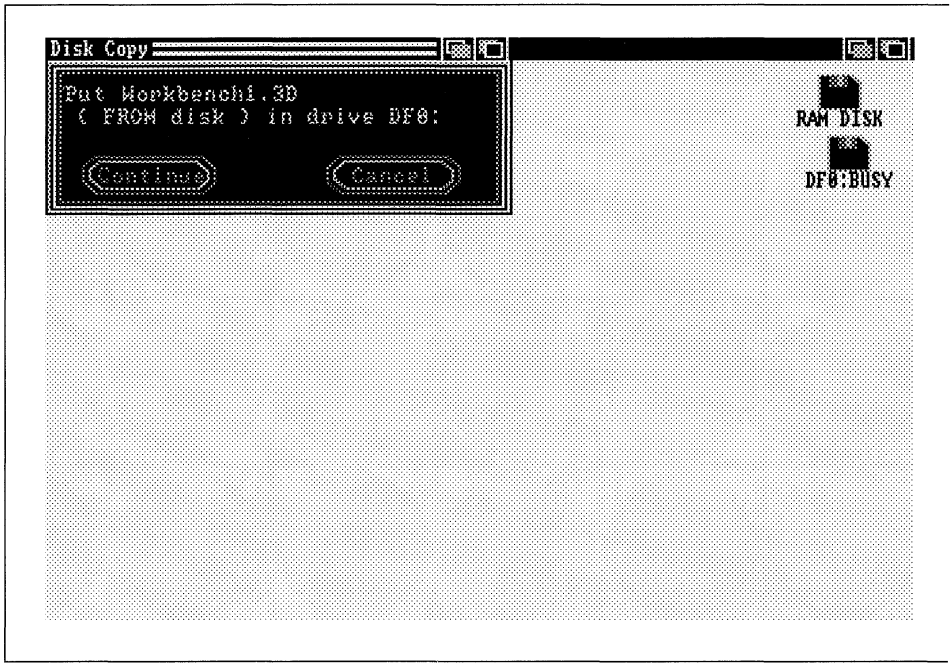



***Bild 2.10:*** *Aufforderung zum Einlegen der FROM-Diskette*

Es erscheint nun ein neues Fenster namens *Disk Copy* mit der Aufforderung, die *FROM disk* ins Diskettenlaufwerk einzulegen. Mit *FROM disk* ist immer die Originaldiskette, also die zu kopierende Diskette, gemeint. Am unteren Rand des Fensters *Disk Copy* befinden sich zwei abgerundete Rechtecke – sogenannte Gadgets – in Form von schwarz und orange umrahmten Worten. Diese Gadgets verhalten sich genauso wie Tasten an einem Radio oder Kassettenrecorder. Das eine trägt den Namen *Continue* (engl. für Fortfahren), das andere *Cancel* (engl. für Abbrechen). Wenn Sie sicher sind, daß die richtige Diskette im Laufwerk liegt, klicken Sie auf *Continue*. Falls nicht, so klicken Sie bitte auf *Cancel*, wodurch der gesamte Kopiervorgang abgebrochen wird.

Haben Sie *Continue* geklickt, so startet der Kopiervorgang. Im Fenster *Disk Copy* wird der Fortschritt des Vorgangs angezeigt. Die Meldung *reading 23, 56 to go* bedeutet zum Beispiel, daß bereits 24 von insgesamt 80 »Häppchen« der Diskette gelesen wurden. Der Amiga beginnt in vielen Fällen bei Null und nicht bei Eins zu zählen. Deshalb erscheint im Meldungsfenster eine 23, obwohl bereits das 24ste Häppchen gelesen wurde. (Für den Fachmann: Bei diesen Häppchen handelt es sich um die 80 Spuren der Diskette.) Der Amiga liest nun nämlich vom Inhalt der Original-Diskette soviel wie möglich in seinen RAM-Speicher. Ist dieser voll, so werden Sie gebeten, die *DESTINATION disk*, also die Diskette, die die

Kopie aufnehmen soll, einzulegen. Warten Sie einen Moment und achten Sie darauf, daß das Diskettenlämpchen erloschen ist. Entfernen Sie dann das Original aus dem Laufwerk und schieben die Diskette ein, die die Kopie aufnehmen soll. Wenn Sie damit fertig sind, klicken Sie wieder auf *Continue*!

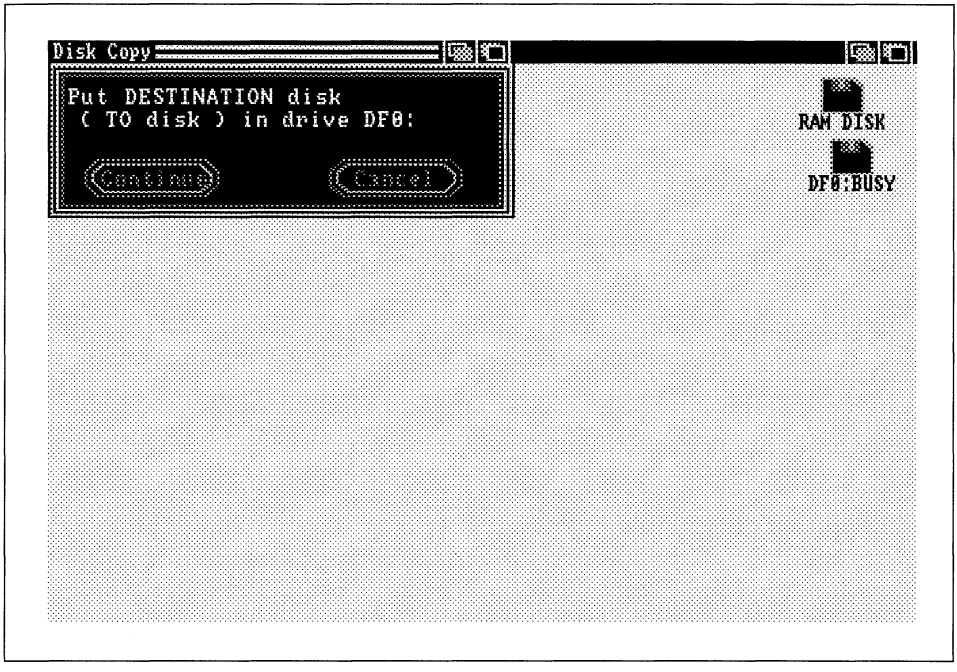



***Bild 2.11:*** *Aufforderung zum Einlegen der DESTINATION-Diskette*

Der Amiga schreibt nun die Daten von der Original-Diskette, die er in seinen RAM-Chips gespeichert hat, auf die Kopie. Auch dieser Vorgang wird wieder protokolliert. Die Meldung *writing 23, 56 to go* bedeutet zum Beispiel, daß bereits 24 von insgesamt 80 »Häppchen« auf die zweite Diskette geschrieben wurden. Nach kurzer Zeit fordert Sie der Amiga wieder auf, die Original-Diskette (*FROM disk*) einzulegen, damit er die nächsten Häppchen davon lesen kann. Folgen Sie dieser Aufforderung und klicken dann wieder auf *Continue*!

Dieses Wechseln zwischen Originaldiskette (FROM disk) und Kopie (DESTINATION disk) wird noch einige Male nötig werden. Beim Amiga 500 ohne Speichererweiterung sind etwa 10 bis 12 Diskettenwechsel nötig; beim Amiga mit Speichererweiterung sind es nur etwa 5. Achten Sie dabei immer auf den Text im Fenster. Wird die *FROM disk* verlangt, müssen Sie das Original einlegen. Wird die *DESTINATION disk* verlangt, müssen Sie die Kopie einlegen. Achten Sie vor dem Herausnehmen der Disketten immer darauf, daß das rote Diskettenlämpchen aus ist und das Laufwerk keine hörbaren Geräusche von sich gibt. Wenn Sie mit dem Diskettenwechsel fertig sind, müssen Sie jedesmal auf *Continue* klicken (oder auf *Cancel*, wenn Sie die Tortur abbrechen wollen).

### 2.7.3 Kopieren mit zwei Diskettenlaufwerken

Diskettenkopieren mit zwei Laufwerken ist wesentlich angenehmer. Legen Sie dazu die Original-Diskette ins interne Laufwerk und die Diskette, die die Kopie aufnehmen soll, ins externe Laufwerk. Ist die zweite Diskette neu und bislang unbenutzt, erscheint sie unter dem Namen *DF1:BAD*. Klicken Sie dann mit der linken Maustaste auf das Piktogramm der Original-Diskette und ziehen den Zeiger über das Piktogramm der Kopie. Nun lassen Sie die Maustaste los. Dieses Legen der einen Diskette auf die andere sagt der Workbench, daß Sie eine Kopie mit zwei Laufwerken machen wollen. Falls auf Ihrer Workbench noch ein anderes Piktogramm namens *RAM Disk* erscheint, achten Sie bitte darauf, das Piktogramm der zu kopierenden Diskette nicht auf *RAM Disk* zu legen. Mit dieser Diskette hat es eine besondere Bewandtnis. Nähere Erläuterungen zum Thema RAM-Disk finden Sie im zweiten Teil dieses Buches (ab Kapitel 6), der sich hauptsächlich mit dem CLI beschäftigt.

Falls Sie nicht die Workbench-Diskette selbst kopieren, erscheint jetzt ein Fenster mit der Aufforderung, die Workbench einzulegen. Werfen Sie dazu die Original-Diskette aus und legen Sie die Workbench-Diskette ein. Nach wenigen Augenblicken verschwindet diese Aufforderung und der Amiga verlangt wieder nach der Originaldiskette. Tauschen Sie daraufhin wieder die Workbenchdiskette gegen die Originaldiskette aus. Dann erscheint ein neues Fenster namens *Disk Copy*.

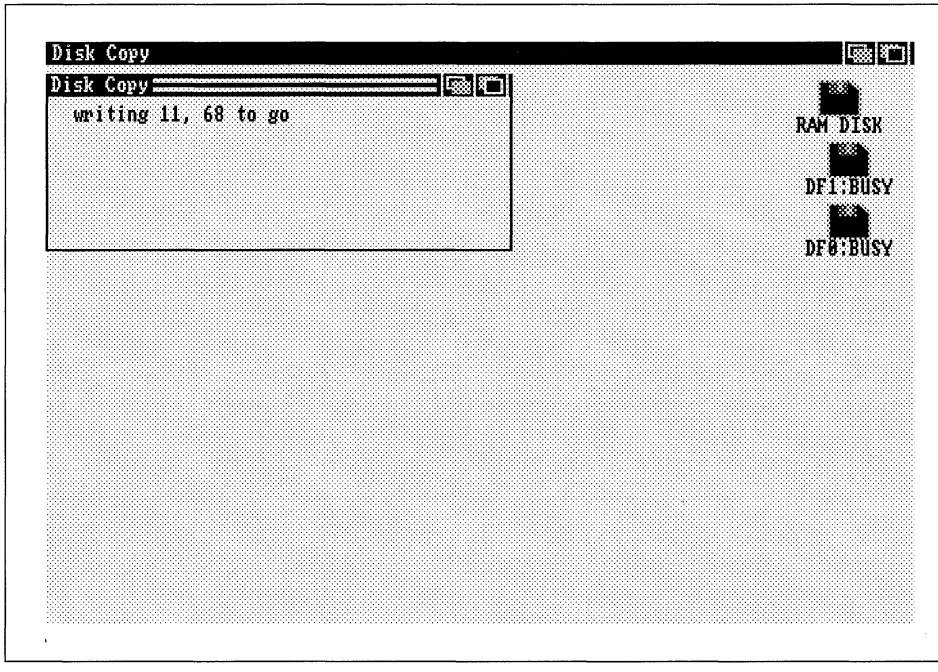



***Bild 2.12:*** *Das Disk-Copy-Fenster beim Kopieren mit zwei Laufwerken*

In diesem Fenster erscheint unter anderem die Aufforderung, das Original (*FROM disk*) ins interne Laufwerk (*drive DF0:*) und die Kopie (*DESTINATION disk*) ins externe Laufwerk (*drive DF1:*) einzulegen. Tun Sie das jetzt und vergewissern Sie sich auf alle Fälle noch einmal davon, daß in jedem Laufwerk die richtige Diskette liegt! Am unteren Rand des Fensters *Disk Copy* befinden sich zwei schwarz und orange umrahmte Worte, die Druckknöpfe – sogenannte Gadgets – darstellen, und die sich genauso verhalten wie eine Taste an einem Radio oder Kassettenrecorder. Das eine Wort lautet *Continue* (engl. für Fortfahren), das andere *Cancel* (engl. für Abbrechen). Wenn Sie sicher sind, daß in jedem Laufwerk die richtige Diskette liegt, klicken Sie auf *Continue*. Falls nicht, so klicken Sie bitte auf *Cancel*, was den gesamten Kopiervorgang abbricht.

Nachdem Sie auf *Continue* geklickt haben, beginnt der Kopiervorgang. Er dauert insgesamt etwa 100 Sekunden. Sie können den Fortschritt im Fenster *Disk Copy* beobachten, um ungefähr zu wissen, wie lange es noch dauert. Es erscheinen abwechselnd Meldungen der Form *reading 23, 56 to go* und *writing 23, 56 to go*. Dies bedeutet, daß der Amiga bereits 24 von 80 nötigen »Häppchen« (Spuren) der Originaldiskette gelesen beziehungsweise schon auf die Kopie geschrieben hat.

### 2.7.4 Abschluß des Kopiervorgangs

Am Ende des Ganzen steht dann eine komplette Kopie Ihres kostbaren Originals. Der alte Inhalt der Diskette, die die Kopie enthält, wurde dabei allerdings vollständig zerstört! Die Kopie erscheint übrigens auch unter dem Namen *copy of <Originalname>* auf der Workbench. Auf Wunsch können Sie diesen Namen auch ändern. Den dazu notwendigen Befehl lernen Sie im folgenden Kapitel kennen.

Falls es sich beim Original um die Kickstart-, Workbench- oder die Original-Diskette eines Programms handelt, legen Sie die Diskette beiseite und arbeiten Sie ab jetzt mit der Kopie weiter. Handelt es sich um eine Daten-Diskette, deren Inhalt häufig geändert wird, legen Sie die Kopie beiseite und arbeiten Sie mit dem Original weiter.

Es gibt allerdings einige kommerzielle Programme, die Sie auf diese Weise nicht kopieren können. Diese Disketten haben einen Kopierschutz, der sie vor unautorisiertem (und leider auch vor rechtmäßigem) Kopieren schützt. Es ist traurig, daß Software-Hersteller meinen, zu solchen Methoden greifen zu müssen, aber man kann vorläufig wenig dagegen tun – außer solche Programme zu boykottieren! Es gibt jedoch auch einige Spezial-Kopierprogramme auf dem Markt, denen solche Kopierschutzverfahren nichts ausmachen (und die Sie bei jedem guten Amiga-Händler erwerben können). Ein gutes Kopierprogramm dieser Art gehört zur Grundausstattung jeder Softwarebibliothek, denn auch nur vorübergehender erzwungener Verzicht auf ein wichtiges Programm kann unter Umständen große Umstände (und Kosten) bereiten.

### 2.7.5 Kopieren Sie alle Originaldisketten!

Das Kopieren einer (beziehungsweise mehrerer) Disketten sollten Sie am besten sofort einmal ausprobieren, falls Sie das nicht schon getan haben. Legen Sie sich dazu Kopien der originalen Workbench-Diskette und der Extras-Diskette an, die Ihrem Amiga 500 beim

Kauf beilagen. Dazu brauchen Sie lediglich diese beiden Disketten und zwei leere Disketten bereithalten und dann den Anweisungen in Abschnitt 2.7.2 (wenn Sie nur ein Diskettenlaufwerk besitzen) oder Abschnitt 2.7.3 (wenn Sie ein zweites Diskettenlaufwerk besitzen) zu folgen.

Bringen Sie danach die Original-Disketten in Sicherheit (aber merken Sie sich auch, wo Sie sie hingelegt haben) und arbeiten Sie in Zukunft nur noch mit den soeben angefertigten Kopien weiter!

Sofern Sie beim Kauf des Amiga noch weitere Disketten mit zusätzlichen Programmen erhalten haben, sollten Sie mit diesen genauso verfahren. Machen Sie sich auch von diesen Kopien und legen Sie die Originale an einen sicheren Ort. Kaufen Sie sich am besten eine Diskettenbox (im Zubehörhandel und in allen größeren Schreibwarengeschäften erhältlich), in denen Sie ausschließlich Originaldisketten aufheben. Sie werden im Laufe Ihrer Computer-Karriere sicherlich noch genug Programme kaufen und kopieren, um diese Diskettenbox nach und nach zu füllen.

# 3 Objekte der Workbench

Nachdem Sie im Kapitel 2 einen Überblick über die Funktionsweise der Amiga-Werkbank (Workbench) bekommen haben, wollen wir nun »zur Sache« kommen und uns die Objekte auf der Werkbank einmal näher betrachten. Dies sind die verschiedenen Symbole, die auf dem blauen Bildschirmhintergrund und in den Fenstern der Workbench zu sehen sind. Wir nennen sie Piktogramme, und jedes von ihnen repräsentiert ein Objekt, das von der Workbench aus auf irgendeine Weise manipuliert werden kann. Diese Piktogramme entsprechen den Disketten, Programmen und Dateien, mit denen Sie beim Umgang mit dem Amiga täglich zu tun haben werden. Besonders wichtig sind auch noch die Schubladen, die Ihnen helfen, Ordnung auf Ihren Disketten zu halten.

Die im folgenden beschriebenen Befehle und Operationen sollten Sie bei einem ersten Lesen dieses Buches am besten immer sofort ausprobieren. Verwenden Sie dazu sicherheitshalber eine Kopie und nicht die Original-Workbench-Diskette. Wie Sie eine solche Kopie anlegen können, haben Sie ja im letzten Kapitel erfahren.

## 3.1 Allgemeine Eigenschaften von Piktogrammen

Das erste Piktogramm, das Sie auf der Workbench kennengelernt haben, war das einer Diskette. Die folgende Abbildung zeigt es Ihnen noch einmal in etwas vergrößerter Form.

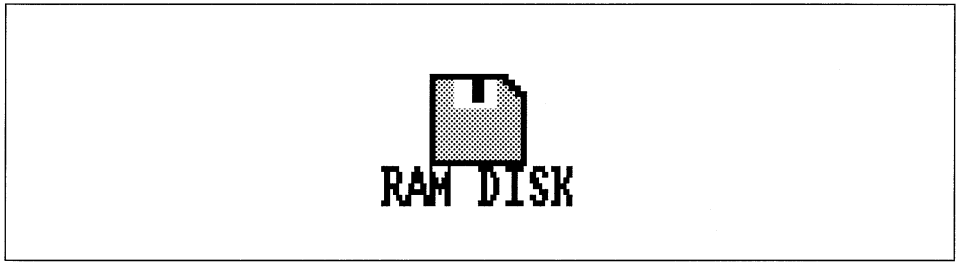



***Bild 3.1:*** *Ein Diskettenpiktogramm*

### 3.1.1 Ein Piktogramm auswählen

»Normalerweise« erscheint ein Diskettenpiktogramm weiß auf blauem Grund mit einem orangen Fleck am oberen Rand, der den Metallschieber einer 3,5-Zoll-Diskette andeuten soll. Klicken Sie es aber mit der linken Maustaste an, so wird es schwarz und der orangefarbene Klecks wird blau. Jedes Piktogramm reagiert so oder so ähnlich darauf, wenn es mit der Maus ausgewählt beziehungsweise selektiert wurde.

Bevor Sie einen Workbench-Befehl aus einem der Menüs auswählen können, ist oft die Selektion eines Piktogramms nötig. Diese Befehle bearbeiten immer nur das vorher selektierte Piktogramm. Um eine Diskette zu duplizieren (kopieren), müssen Sie zum Beispiel erst das entsprechende Diskettenpiktogramm auswählen und dann den Befehl *Duplicate* aus dem Menü *Workbench* aufrufen.

### 3.1.2 Ein Piktogramm verschieben

Beim Auswählen eines Piktogramms wird die linke Maustaste nur kurz niedergedrückt und dann sofort wieder losgelassen. Halten Sie die Maustaste jedoch fest, nachdem Sie das Piktogramm angeklickt haben, können Sie es verschieben. Dazu bewegen Sie einfach den Mauszeiger zu der gewünschten Stelle und lassen die Taste dann wieder los. Aber diese Handgriffe kennen Sie ja schon aus Kapitel 2. Die Bewegung eines Piktogramms mit der Maus nennen wir in Zukunft »Ergreifen und Verschieben eines Piktogramms« (an eine bestimmte Stelle). Die Mausbewegungen und Maustastenbetätigungen, die dazu nötig sind, werde ich nicht jedesmal ausführlich beschreiben.

Ein Piktogramm bleibt so lange an der Stelle liegen, an der Sie es zuletzt hingelegt haben, bis Sie die Workbench verlassen. Das Verlassen der Workbench ist normalerweise nur durch ein Abschalten oder Neustarten (siehe Kapitel 2) des Amiga zu erreichen. Nach dem Abschalten »vergißt« das Piktogramm seine neue Position. Beim nächsten Start der Workbench liegt es wieder an der alten Position. Möchten Sie, daß ein bestimmtes Piktogramm stets an einer anderen Position erscheint, so können Sie auch dies erreichen, indem Sie einen »Schnappschuß« der Piktogrammposition machen. Was Sie dazu tun müssen, wird weiter unten noch genauer beschrieben.

### 3.1.3 Ein Piktogramm umbenennen

Unter jedem Piktogramm, das Sie am Bildschirm sehen, steht ein Name. Dieser dient vor allem dazu, Piktogramme mit demselben Aussehen unterscheiden zu können. Den Namen, beziehungsweise den Text, der unter dem Piktogramm erscheint, können Sie jederzeit ändern. Probieren Sie es aus: Selektieren Sie dazu das Diskettenpiktogramm der Workbench-Diskette und wählen Sie dann den Befehl *Rename* (engl. *to rename* = umbenennen) aus dem Menü *Workbench*. Mitten auf dem Bildschirm erscheint daraufhin ein schmaler, weiß umrandeter Streifen, in dem der Name der Diskette steht (zum Beispiel »Workbench1.3D«). Klicken Sie mit der linken Maustaste in diesen Streifen und Sie können den Text ändern.

Ein Buchstabe des Namens erscheint schwarz auf orange, nachdem Sie den Rahmen angeklickt haben, während alle anderen Buchstaben weiß auf blau dargestellt werden. Das

orangefarbige Rechteck ist die aktuelle »Schreibmarke« (der Fachmann sagt auch »Cursor« dazu). Wenn Sie auf der Tastatur nun zu tippen beginnen, erscheint der neue Text immer vor dieser Schreibmarke und die dahinterstehenden Buchstaben werden automatisch nach rechts geschoben. Sie können den Buchstaben, auf dem die Schreibmarke steht, auch löschen. Drücken Sie hierzu die [Del]-Taste oben rechts auf der Tastatur. Der Buchstabe bei der Schreibmarke verschwindet und die Buchstaben rechts davon werden gleichzeitig nach links verschoben. Wenn Sie einen völlig neuen Namen eingeben wollen, tippen Sie einfach ein paarmal auf [Del], bis der alte Name völlig verschwunden ist und geben dann den neuen ein.

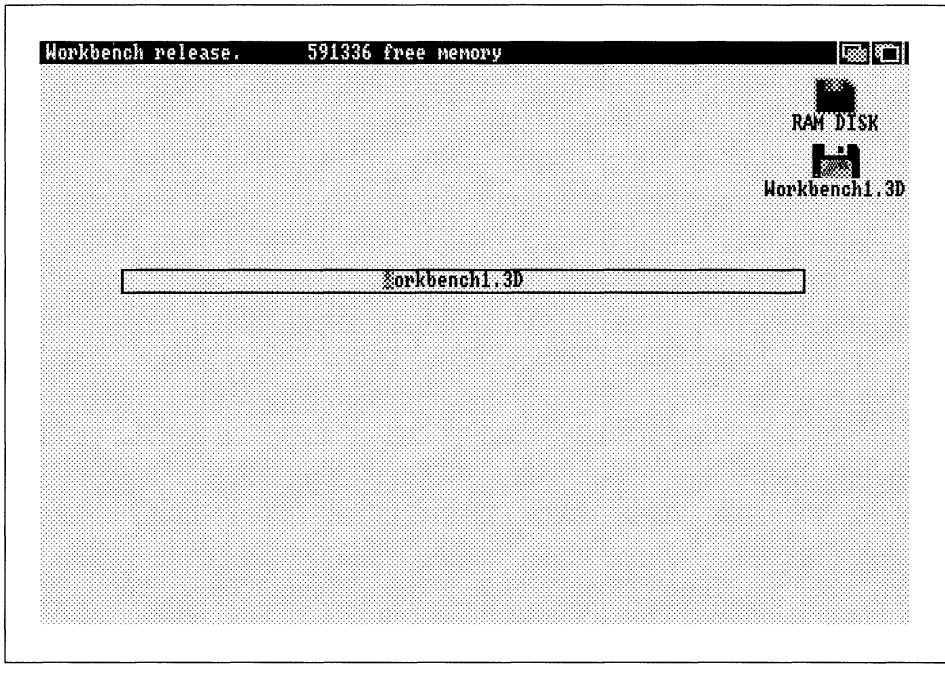



***Bild 3.2:*** *Ein Rahmen für die Texteingabe*

Wenn Sie einen Tippfehler machen, können Sie immer den letzten eingegebenen Buchstaben mit der [Backspace]-Taste (links neben der [Del]-Taste) löschen. Die [Backspace]-Taste löscht aber nicht nur den zuletzt eingetippten Buchstaben, sondern immer den Buchstaben links von der Schreibmarke, so daß Sie also auch mehrere Buchstaben auf diese Weise wieder löschen können.

Rechts unten auf der Tastatur befinden sich weiterhin vier Tasten, die je einen Pfeil tragen. Dies sind die Cursortasten, die Sie zum Verschieben der Schreibmarke verwenden können. Die beiden nach links und rechts weisenden Pfeile bewegen die Schreibmarke jeweils einen Buchstaben nach links beziehungsweise rechts, sofern dies möglich ist. (Die beiden Pfeile nach oben und unten haben beim Ändern eines Piktogrammnamens keine Wirkung.) Statt mit diesen beiden Tasten können Sie die Schreibmarke aber auch mit der Maus bewegen.

Klicken Sie dazu einfach den Buchstaben mit der Spitze des Mauszeigers an, zu dem Sie die Schreibmarke bewegen wollen.

Probieren Sie diese vielfältigen Möglichkeiten zur Änderung des Namens am besten alle einmal aus. Ändern Sie dabei – beziehungsweise am Ende Ihrer Experimente – den im Rahmen angezeigten Namen in »WB-Diskette« (Workbench-Diskette) um.

Wenn Sie mit dem Namen in dem weißen Rahmen zufrieden sind, müssen Sie die [Return]-Taste drücken. Das zuvor ausgewählte Piktogramm – in diesem Fall das Piktogramm der Workbench-Diskette – erhält dann den neuen Namen. (Falls die Diskette schreibgeschützt war, erhält das Piktogramm keinen neuen Namen, sondern es erscheint ein neues, kleines »Fenster« mit einer Fehlermeldung. Dazu mehr im folgenden Abschnitt.) Die [Return]-Taste schließt typischerweise die Eingabe von Text in einen solchen Rahmen ab!

Experimentieren Sie am besten noch etwas mit dem *Rename*-Befehl und prägen Sie sich die bei der Änderung des Namens zur Verfügung stehenden Möglichkeiten gut ein. Auch andere Befehle oder Programme verlangen manchmal Texteingaben, die immer ähnlich ablaufen, wie gerade beschrieben, und die oft auch mit der Betätigung der [Return]-Taste abgeschlossen werden.

### 3.1.4 Wie Fehlermeldungen aussehen

Der Befehl *Rename* bietet eine gute Gelegenheit, auch einmal die Fehlermeldungen des Amiga kennenzulernen. Fehlermeldungen zu erkennen und ihre Ursachen zu beheben, ist ein wichtiger Bestandteil der Arbeit mit dem Computer – denn Fehler macht schließlich jeder einmal.

Nehmen Sie dazu nun die Workbench-Diskette aus dem Laufwerk und aktivieren Sie den Schreibschutz (indem Sie den kleinen Plastikschieber zum Rand der Diskette schieben). Legen Sie die Diskette dann zurück in das Laufwerk, wählen Sie das Piktogramm mit der Maus aus und wählen Sie den Befehl *Rename* aus dem Menü *Workbench*. Geben Sie ein paar Buchstaben ein und drücken Sie auf die [Return]-Taste. Sofort erscheint ein neues Fenster auf dem Bildschirm, das Ihnen mitteilt, daß die Diskette schreibgeschützt (engl. *write protected*) ist.

Dieses Fenster ist ein sogenanntes »Requester« oder »Dialogfenster«. Es verlangt eine Reaktion von Ihnen (engl. *to request* = verlangen). Hierzu enthält es zwei Knöpfe, die Sie mit dem Mauszeiger betätigen können. Wenn Sie *Cancel* anklicken, wird der *Rename*-Befehl abgebrochen und die Diskette behält den alten Namen. Sie können aber auch die Disketten aus dem Laufwerk nehmen, den Schreibschutz entfernen und dann auf *Retry* (engl. für *versuche es noch einmal*) klicken. Dann erhält die Diskette den neuen Namen.

Wenn eine Fehlermeldung auf dem Bildschirm erscheint und Sie *Cancel* drücken, erscheint in der Titelzeile noch einmal eine kurze Meldung, die verschwindet, sobald Sie eine Maustaste drücken. Diese Meldung beginnt mit einer Zahl, die den aufgetretenen Fehler eindeutig identifiziert. Falls Sie einmal nicht wissen, was Sie falsch gemacht haben oder wie Sie den Fehler beheben, können Sie mit Hilfe der Fehlernummer im Anhang dieses Buches nachschlagen. Dort sind alle möglichen Fehlermeldungen mit Erläuterungen und Lösungsvorschlägen aufgelistet.




***Bild 3.3:*** *Die erste Fehlermeldung*

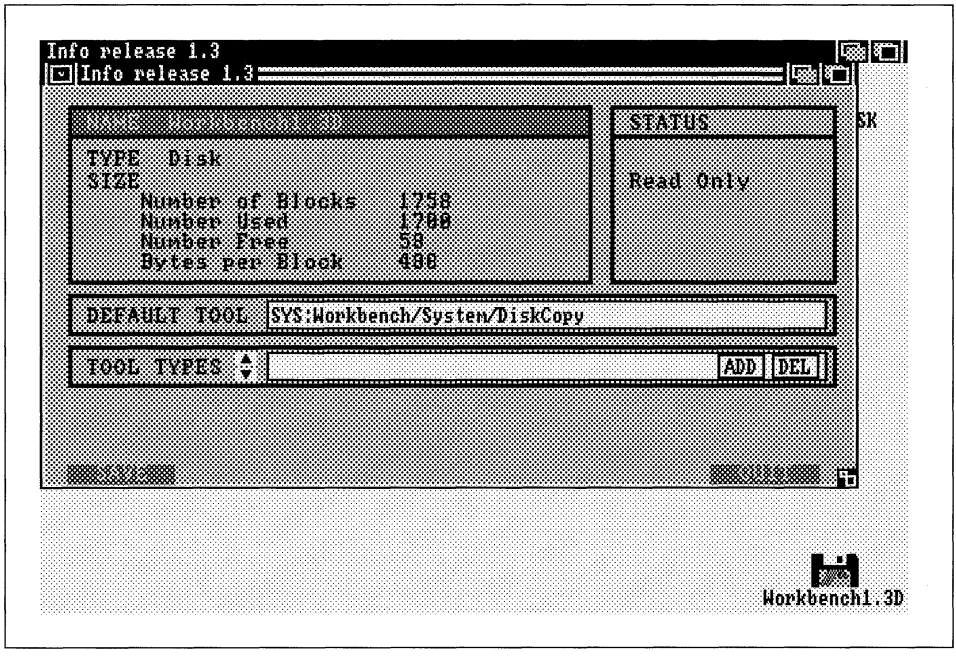



***Bild 3.4:*** *Das Info-Fenster einer Diskette*

### 3.1.5 Informationen zu einem Piktogramm erfragen

Wenn Sie ein Piktogramm auf der Workbench sehen, kennen Sie zunächst einmal nur seinen Namen und können aus seinem Aussehen vielleicht darauf schließen, um welche Art von Piktogramm es sich handelt. Diese Informationen werden Ihnen oft auch genügen. Wenn Sie aber detaillierte Informationen über ein Piktogramm wünschen, so können Sie diese mit dem Befehl *Info* erhalten.

Probieren Sie es aus: Selektieren Sie dazu das Piktogramm der Workbench-Diskette, und wählen Sie dann den Befehl *Info* aus dem Menü *Workbench*. Daraufhin erscheint ein neues Fenster am Bildschirm, das den Namen *Info Release 1.3* trägt.

In diesem Fenster wird unter anderem noch einmal der Name des Piktogramms aufgelistet und darunter, neben dem Wort TYPE, erfahren Sie, um welche Art Piktogramm es sich handelt (in diesem Fall um eine Diskette; engl. *disk*). Abhängig von diesem Typ tauchen darunter noch eine ganze Reihe weiterer Informationen auf. Bei einem Diskettenpiktogramm sind dies zum Beispiel ein paar Zahlen, die Aufschluß darüber geben, wieviele Daten sich schon auf der Diskette befinden und wieviel Platz noch auf ihr frei ist (mehr darüber in den folgenden Abschitten dieses Kapitels).

Das Info-Fenster können Sie wieder verschwinden lassen, indem Sie sein Schließ-Gadget in der oberen linken Ecke oder den Knopf *Quit* in der rechten unteren Ecke des Fensters mit der Maus anklicken. Den Knopf *Save* daneben sollten Sie nicht betätigen. Er ist nur sinnvoll, wenn Sie die Felder neben *Default Tool* oder *Tool Type* modifiziert haben (was Sie vorläufig auf keinen Fall tun sollten).

## 3.2 Ordnung auf der Werkbank: Schubladen

Nach diesen mehr allgemeinen Eigenschaften von Piktogrammen werden wir uns nun mit einer besonderen Sorte von Piktogrammen beschäftigen, den Schubladen. Man kann sagen, daß die Schubladen die wichtigsten Objekte sind, die sich auf der Workbench befinden. Wir haben die Schubladen ja schon kurz in Kapitel 2 kennengelernt, als wir Fenster zum Experimentieren benötigten. In Schubladen (und Diskettenpiktogrammen) verbergen sich immer andere Objekte. Die Schubladen sind an sich keine »richtigen« Objekte; sie enthalten nur andere Objekte. (Schubladen entsprechen auf einer nicht ganz so bildlichen Ebene den Directories oder Dateiverzeichnissen, die Sie in den Kapiteln über das CLI und die Shell noch kennenlernen werden. Wenn Sie schon mit MS-DOS- oder UNIX-Computern gearbeitet haben, wird Ihnen dieses Konzept sicherlich vertraut sein.)

### 3.2.1 Welche Aufgaben Schubladen und Disketten haben

Schubladen dienen der Gliederung der Informationen beziehungsweise der Objekte auf einer Diskette oder Festplatte. Sie sind nicht unbedingt notwendig, aber praktisch. Prinzipiell wäre es zum Beispiel möglich, alle Objekte, die eine Diskette enthält, im Diskettenfenster selbst unterzubringen. Solange es 10 oder 20 Objekte sind, mag das völlig ausreichen. Wenn aber einige hundert oder (bei Verwendung einer Festplatte) gar einige tausend Objekte auf einem Datenträger liegen, wird man rasch die Übersicht verlieren und nur sehr schwer ein bestimmtes Objekt finden, das man gerade sucht.

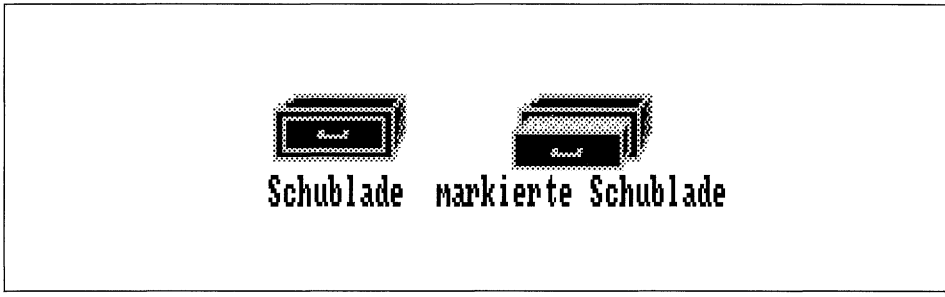



***Bild 3.5:*** *Schubladen auf der Workbench*

Auf einer richtigen Werkbank würden Sie ja auch nicht alle Gegenstände einfach herumliegen lassen, sondern sie in Kisten, Kästchen und Schubladen legen, die jeweils eine zusammengehörige Gruppe von Gegenständen enthält. Eine Kiste für die Bohrer, eine für Nägel, Schrauben, Dübel, Schraubenschlüssel, Schraubendreher und so weiter. Genauso funktionieren die Schubladen der Workbench. Wenn Sie alle persönlichen Briefe, die Sie mit einem Textverarbeitungssystem erstellt haben, in eine Schublade legen, alle Rechnungen in eine andere und die kleinen Hilfsprogramme, die Sie nicht täglich benötigen, in eine andere, so wird das Ihre Chancen, schnell einen bestimmten Brief oder ein bestimmtes Programm zu finden, wesentlich erhöhen.

Und genau so, wie Schubladen auf einer Werkbank ihrerseits wieder Kisten und Kästchen enthalten können, die vielleicht jeweils Nägel einer bestimmten Größe enthalten, können auch die Schubladen auf der elektronischen Workbench weitere Schubladen enthalten. Haben Sie zum Beispiel eine Schublade mit Briefen, so könnten Sie in dieser wiederum Schubladen einrichten, die ältere, neue und noch nicht abgesandte Briefe enthalten. Verwalten Sie sehr viele Briefe elektronisch auf einer Diskette, so dürfte dies wiederum Ihre Chancen, einen bestimmten zu finden, wesentlich verbessern – wie schon zuvor die Entscheidung, Briefe und Rechnungen in getrennte Schubladen zu legen.

Während Sie auf einer richtigen Werkbank aber nicht unendlich viele Schubladen ineinander verschachteln können, ist dies im Prinzip mit den »elektronischen Schubladen« auf der Workbench möglich. Sie können Schubladen in Schubladen in Schubladen in Schubladen in Schubladen ... haben. (Sie kennen ja diese russischen Püppchen, die man öffnen kann, und darin wieder ein Püppchen findet, das man öffnen kann und so weiter. Genauso ist es mit den Schubladen der Workbench.) Wie sinnvoll das in so extremer Form ist, darüber läßt sich streiten. Müssen Sie, um etwa einen Brief bearbeiten zu können, erst 20 Schubladen öffnen, ist die Arbeitserleichterung, die Schubladen verschaffen können, nicht mehr sehr groß. Man muß auch bei der Verwendung von Schubladen das richtige Mittelmaß finden. Eine maximale Schachtelungstiefe von fünf bis sechs Schubladen ist durchaus sinnvoll.

### 3.2.2 Eine Schublade öffnen

Durch den Befehl *Open* im Menü *Workbench* können Schubladen – genau wie übrigens auch fast alle anderen Piktogramme – geöffnet werden. Daraufhin öffnet sich ein sogenanntes »Fenster«, das Ihnen den Inhalt der Schublade zeigt. Den Umgang mit Fenstern haben Sie ja bereits in Kapitel 2 kennengelernt.

Probieren Sie es aus: Markieren Sie mit einem Mausklick das Piktogramm der Workbench-Diskette (mit dem Namen »WB-Disk«) und wählen Sie den Befehl *Open* aus dem Menü *Workbench*. Dann öffnet sich das Diskettenfenster der Workbench-Diskette. Markieren Sie jetzt die Schublade *Utilities* und wählen Sie noch einmal *Open* aus dem Menü *Workbench*.

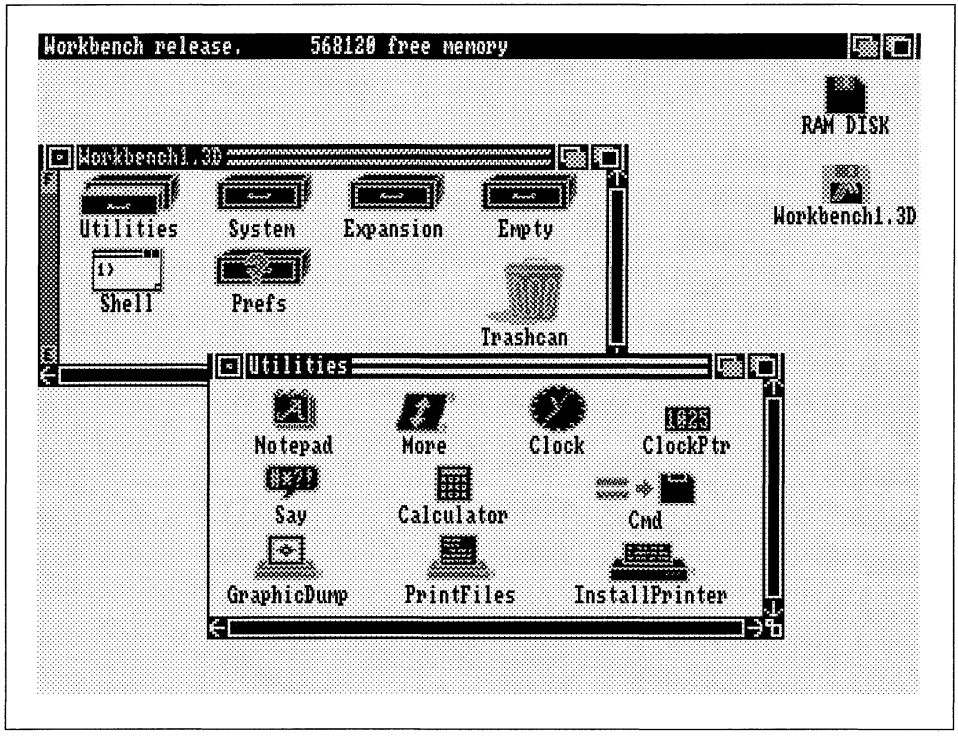



***Bild 3.6:*** *Das Fenster einer Schublade*

*Open* hat auch eine Abkürzung, die so praktisch ist, daß ich die Prophezeiung wage, daß Sie später fast nur noch diese Abkürzung wählen werden und nie mehr den Menü-Befehl *Open*. Wenn Sie zweimal schnell hintereinander mit der linken Maustaste auf ein Schubladen- oder ein Diskettenpiktogramm klicken, erzielen Sie dieselbe Wirkung wie beim Auswählen des Befehls *Open* aus dem *Workbench*-Menü. Diesen Vorgang werden wir im folgenden immer »Doppelklick« nennen.

Probieren Sie am besten sofort aus: Verschieben Sie dazu gegebenenfalls das Fenster »WB-Diskette« (das Fenster der Workbench-Diskette), bis Sie die Schublade *System* sehen können. Bewegen Sie dann die Spitze des Mauszeigers über diese Schublade und drücken Sie zweimal schnell hintereinander die linke Maustaste nieder.

Jedes Piktogramm kann entweder mit dem Menü-Befehl *Open* oder durch einen Doppelklick mit der linken Maustaste darauf geöffnet werden. Man könnte auch sagen, *Open* und Doppelklick sind äquivalent.

### 3.2.3 Eine Schublade schließen

Ein gerade aktives Schubladenfenster können Sie jederzeit schließen, indem Sie den Befehl *Close* aus dem Menü *Workbench* auswählen. Falls Sie sich den »Weg ins Menü« sparen wollen, können Sie auch einfach ins Schließ-Gadget, links oben in der Titelleiste des Fensters, klicken. Auch dadurch wird das Fenster wieder geschlossen.

Probieren Sie am besten beide Möglichkeiten sofort einmal aus: Das oberste Fenster auf dem Bildschirm dürfte im Augenblick noch den Namen *System* tragen (dieses Fenster haben Sie ja zuletzt geöffnet.) Wenn Sie jetzt den Befehl *Close* aus dem Menü *Workbench* auswählen, wird dieses Fenster geschlossen. Beachten Sie bitte, daß der *Close*-Befehl immer das »aktive« Fenster schließt. Das muß nicht unbedingt immer das oberste Fenster sein. Sie erkennen das aktive Fenster an seinem gut lesbaren Titel. Die Titelzeilen der anderen (nicht aktiven) Fenster werden hingegen in gerasterter (gepunkteter) Schrift dargestellt. Sie können ein beliebiges Fenster aktiv machen, indem Sie mit der Maus in das Fenster klicken oder das »zugehörige« Schubladenpiktogramm anklicken.

Diese Vorarbeiten brauchen Sie bei der Verwendung des Schließgadgets nicht zu machen. Klicken Sie einfach in das Schließ-Gadget eines beliebigen Fensters und schon verschwindet es vom Bildschirm. (Beachten Sie aber, daß nicht jedes Fenster zwangsläufig ein Schließgadget besitzen muß. Requester verfügen zum Beispiel meist nicht über diese praktische »Abkürzung«.)

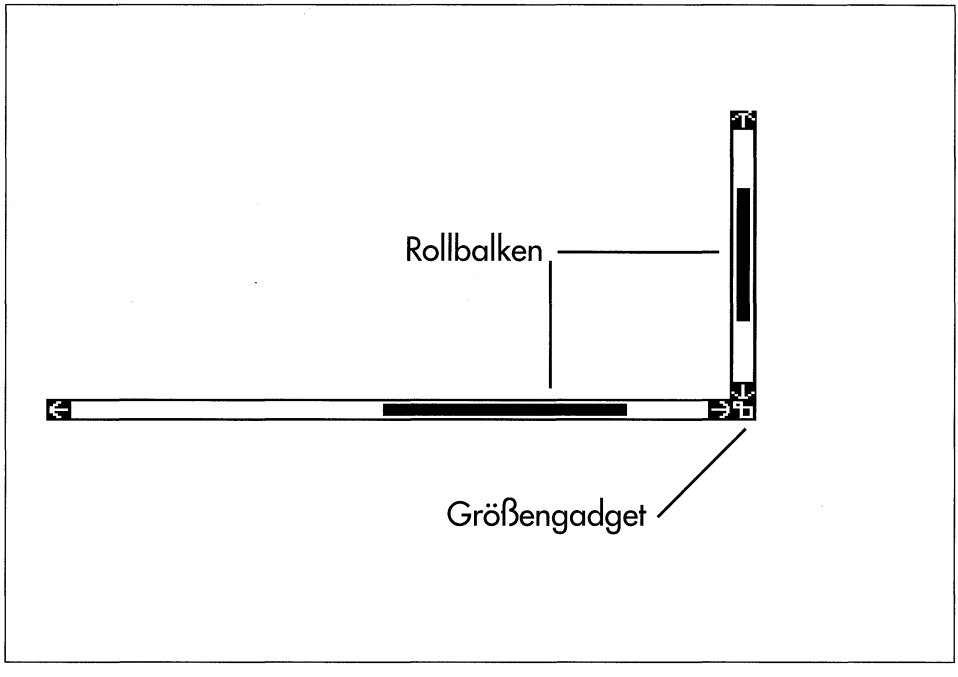



***Bild 3.7:*** *Rollbalken und das Größen-Gadget*

### 3.2.4 Der Aufbau eines Schubladenfenster

Wir kommen nun zu zwei weiteren sehr wichtigen Eigenschaften von Fenstern, die im vorigen Kapitel noch nicht erwähnt wurden. Diese Eigenschaften werden vor allem dann wichtig, wenn Fenster sehr viele andere Objekte enthalten.

Genauer gesagt, handelt es sich um zwei neue Arten von Gadgets (Bedienungselementen), die nun beschrieben werden sollen. Das erste davon ist das »Größen-Gadget«. Es ist das kleine Quadrat unten rechts im Rahmen eines Fensters, das ein kleines und ein etwas größeres Rechteck enthält. Wie der Name schon sagt, können Sie mit dem Größen-Gadget die Größe eines Fensters ändern.

Wenn Sie das Größen-Gadget mit der Maus ergreifen (das heißt, mit der linken Maustaste darauf klicken und die Taste dann festhalten), so folgt eine rechteckige Umrißlinie den Mausbewegungen. Das sieht ähnlich aus wie beim Verschieben eines Fensters. Anders als beim Verschieben bleibt aber die obere linke Ecke des Rechtecks unbeweglich und nur die untere rechte Ecke folgt (innerhalb der durch den Bildschirmrand gesetzten Grenzen) den Mausbewegungen. Sobald die Maustaste losgelassen wird, erhält das Fenster die neue, durch den flimmernden Rahmen angedeutete Größe.

Probieren Sie es aus: Öffnen Sie zuerst – am besten mit einem Doppelklick auf das Piktogramm *WB-Diskette* – das Fenster der Workbench-Diskette. Öffnen Sie dann (wieder mit Doppelklick) das Schubladenpiktogramm *Utilities*. Bewegen Sie die Spitze des Mauszeigers nun über die rechte untere Ecke des Fensters *Utilities* (also über das Größengadget). Drücken Sie die linke Maustaste und halten Sie sie fest. Bewegen Sie dann – bei gedrückter Maustaste – die Maus ein wenig hin und her. Lassen Sie die Maustaste dann los, wenn der dabei angezeigte Rahmen die gewünschte Größe hat.

Enthält ein Fenster nur sehr wenige Objekte, kann man es auf diese Art und Weise wesentlich platzsparender auf dem Bildschirm unterbringen. Liegen andererseits aber sehr viele Objekte darin, so kann man es entsprechend vergrößern. Durch Größenänderungen und geschicktes Verschieben ist es möglich, auch eine größere Menge von Fenstern einigermaßen übersichtlich am Bildschirm unterzubringen.

Was aber, wenn die Objekte in einem Fenster mehr Platz beanspruchen als das Fenster groß ist? In diesem Fall kann innerhalb des Fensters ja nur ein Teil der Fläche dargestellt werden, die eigentlich nötig wäre, um den ganzen Inhalt zu zeigen. Für solche Fälle gibt es die sogenannten »Rollbalken«.

### 3.2.5 Rollbalken

Um den Umgang mit Rollbalken zu üben, öffnen Sie jetzt bitte das Fenster der Workbench-Diskette (falls nötig) und dann die Schublade *Utilities*. Machen Sie dann das Fenster *Utilities* mit Hilfe des Größen-Gadgets so klein, daß nur noch zwei oder drei Piktogramme darin zu sehen sind und alle anderen Piktogramme verschwunden sind.

Keine Angst; diese Objekte verschwinden natürlich nicht wirklich. Sie werden bei der Verkleinerung nur vorübergehend unsichtbar. Man kann sich dies veranschaulichen, indem man sich vorstellt, daß alle Objekte, die innerhalb eines Fensters liegen, sich auf einer sehr großen Fläche verteilen, die Platz für (fast) beliebig viele Objekte bietet. Im Fenster selbst

kann aber nur ein (eventuell recht kleiner) Ausschnitt dieser Fläche gezeigt werden. Mit den Rollbalken kann man diesen Ausschnitt, der im Fenster sichtbar ist, auf verschiedene Arten ändern.

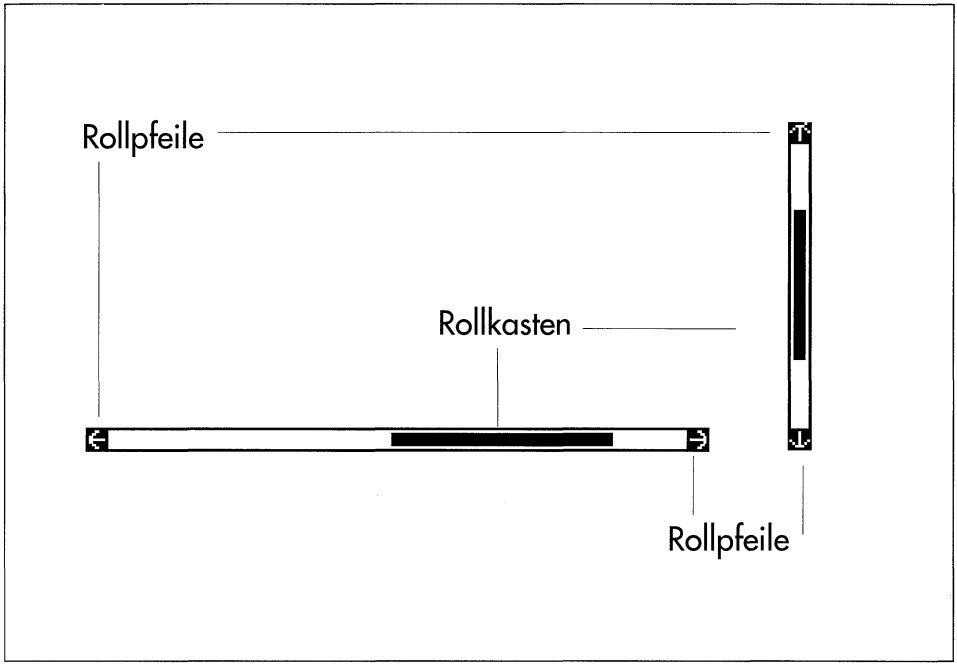



***Bild 3.8:*** *Anatomie eines Rollbalkens*

Was zuerst auffällt, wenn man ein Fenster kleiner macht »als nötig wäre«, sind die beiden Streifen, die Rollbalken am rechten und unteren Rand des Fensters, die ihr Aussehen ändern. Sie tragen an den beiden Enden zwei Pfeile, die in entgegengesetzte Richtungen zeigen. Die Fläche zwischen den Pfeilen ist leer und weiß, solange das Fenster »groß genug« ist. Sobald es aber zu klein wird, wird das weiße Rechteck ebenfalls kleiner und gibt eine »dahinterliegende« blaue Fläche frei.

Diese blaue Fläche symbolisiert eine große Fläche, auf der die in diesem Fenster liegenden Piktogramme verteilt sind und von der nur ein Teil innerhalb des Fensters gezeigt wird. Das weiße Rechteck entspricht dem gerade sichtbaren Teil davon. Das weiße Rechteck nennt man auch Rollkasten (übrigens ebenfalls ein Gadget). Wenn der sichtbare Ausschnitt eines Fensters kleiner wird, verkleinert sich der Rollkasten entsprechend. Dies ist der Grund dafür, daß der Rollkasten bei einem Fenster, in dem das gesamte Bild sichtbar ist, kaum auffällt. Er füllt dann nämlich den gesamten Raum zwischen den beiden Pfeilen an, weil ja auch das gesamte Bild zu sehen ist.

Die beiden Pfeile an den Enden jedes Rollbalkens sind die »Rollpfeile«. Klickt man auf einen davon, bewirkt dies, daß der im Fenster sichtbare Ausschnitt des Gesamtbildes ein Stück in die vom Pfeil angedeutete Richtung »rutscht«.

Probieren Sie es aus: Führen Sie die Spitze des Mauszeigers über den nach links zeigenden Pfeil in der unteren linken Ecke des *Utilities*-Fensters. Klicken Sie diesen Pfeil einmal an (drücken Sie also die linke Maustaste und lassen Sie sie sofort wieder los) und beobachten Sie was passiert. Klicken Sie dann auch einmal in den nach rechts zeigenden Pfeil im unteren Rollbalken (neben dem Größen-Gadget). Halten Sie jetzt die [Shift]-Taste fest und klicken Sie noch einmal auf die beiden Pfeile. Wie Sie sehen, sind die »Sprünge«, die der Fensterinhalt jetzt macht, bei weitem nicht so groß. Klicken Sie hingegen ohne die [Shift]-Taste auf die beiden Pfeile, so rutscht das Bild um fast ein Drittel der Fensterbreite zur Seite. Dabei springt auch der Rollkasten in die angegebene Richtung, um anzuzeigen, daß jetzt ein anderer Ausschnitt sichtbar ist.

Der Rollkasten in einem Rollbalken ist aber nicht nur eine Anzeige. Er kann auch selbst zur Verschiebung des sichtbaren Ausschnitts verwendet werden. Er läßt sich wie ein Piktogramm mit der Maus ergreifen und innerhalb des Rollbalkens (also nur horizontal oder nur vertikal) verschieben. Läßt man ihn (beziehungsweise die Maustaste) los, »springt« der im Fenster sichtbare Ausschnitt zur entsprechenden Stelle im Gesamtbild.

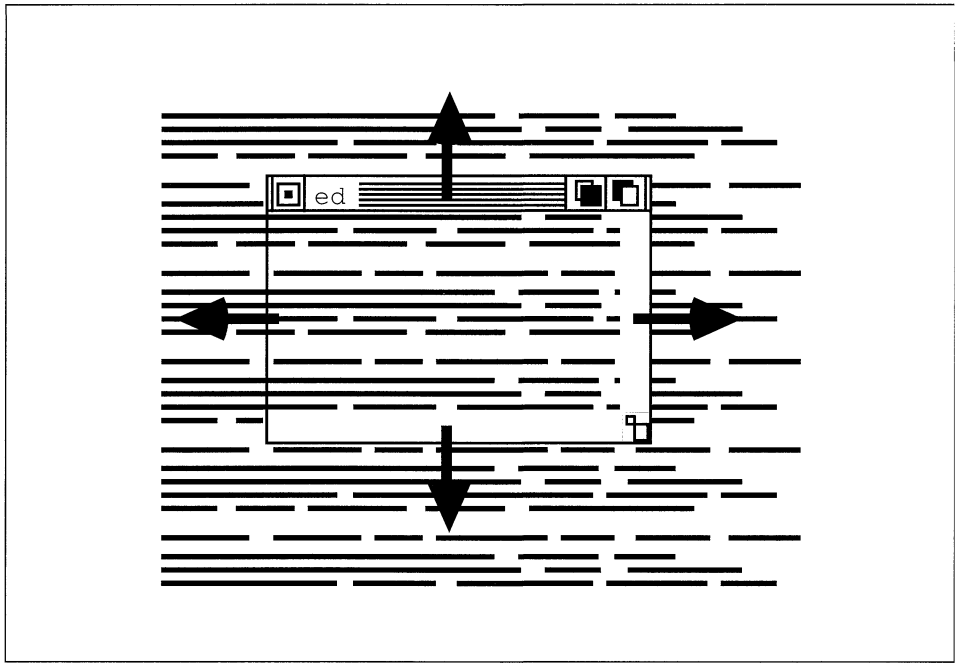



***Bild 3.9:*** *Funktionsweise von Fenstern und Rollbalken*

Probieren Sie das am besten gleich einmal aus: Bewegen Sie die Spitze des Mauszeigers über den Rollkasten (das weiße Rechteck) im unteren Rollbalken des *Utilities*-Fensters. Ergreifen Sie die Rollkasten, indem Sie die linke Maustaste niederdrücken und festhalten. Verschieben Sie den Rollkasten etwas nach rechts und lassen Sie ihn dann wieder los. Schieben Sie auch einmal den Rollkasten am rechten Rand des Fensters nach unten.

Die Rollbalken am rechten und unteren Rand eines Fensters verschieben den Ausschnitt vertikal und horizontal, verhalten sich aber ansonsten vollkommen gleich. Zusammen erlauben sie es, den sichtbaren Ausschnitt über einen beliebigen Bereich der »darunterliegenden« Fläche zu verschieben.

### 3.2.6 Eine Schublade bewegen

Nach diesem kurzen Exkurs über das Thema Rollbalken nun wieder zurück zu den Schubladen: Sie können Schubladen, genau wie Diskettenpiktogramme auch, mit der Maus bewegen. Die Bewegung von Piktogrammen – nicht nur von Schubladen – erhält durch Schubladen allerdings eine neue Qualität. Wenn Sie ein Objekt (Piktogramm) bewegen und die Maustaste loslassen, während sich das bewegte Piktogramm über dem Piktogramm oder dem Fenster einer Schublade befindet, so wird das Objekt in diese Schublade gelegt. Auf diese Weise kann man Objekte aus einer Schublade (oder aus dem Diskettenfenster) in eine andere Schublade legen.

Probieren Sie es aus: Führen Sie die Spitze des Mauszeigers über die Schublade *Empty* im Fenster der Workbench-Diskette. (Dazu müssen Sie dieses Fenster natürlich erst öffnen, falls es noch nicht offen ist.) Ergreifen Sie die Schublade, indem Sie die linke Maustaste niederdrücken und festhalten. Schieben Sie diese Schublade jetzt über das Piktogramm der Schublade *System* und lassen Sie die Maustaste los. Das Diskettenlaufwerk arbeitet dann einen Moment lang und die Schublade *Empty* ist verschwunden. Sie befindet sich jetzt in der Schublade *System*.

Davon können Sie sich sofort überzeugen, indem Sie die *System*-Schublade öffnen. Machen Sie dazu einen Doppelklick auf das entsprechende Piktogramm. Sollten Sie die Schublade *Empty* nicht sofort in dem Fenster *System*, das sich daraufhin öffnet, finden können, macht das gar nichts. Empty liegt dann in einem »unsichtbaren« Teil des Fensters. Sie müssen jetzt die Rollbalken verwenden – wie im letzten Abschnitt beschrieben – um den Fensterinhalt nach links und/oder nach oben zu verschieben. Spätestens nach ein oder zwei Klicks in die Rollpfeile sollten Sie dann die Schublade *Empty* sehen können. Ergreifen Sie sie wieder – wie gerade beschrieben – und legen Sie sie zurück in das Fenster *WB-Diskette*. Wie Sie sehen, ist es wirklich nicht schwer, ein Piktogramm von einer Schublade in eine andere zu legen!

Dieses Umlegen eines Piktogramms von einer Schublade in eine andere hat allerdings noch eine Besonderheit, wenn sich die beteiligten Schubladen auf verschiedenen Disketten befinden. Liegen beide an einem solchen Vorgang beteiligten Schubladen auf ein und derselben Diskette, läuft alles so ab wie oben beschrieben. Das Objekt verschwindet aus der einen Schublade und erscheint in der anderen.

Befindet sich die zweite Schublade aber auf einer anderen Diskette, so bleibt das verlegte Objekt in der alten Schublade, und in der Schublade auf der zweiten Diskette erscheint eine Kopie des Original-Objektes. Das Verlegen eines Objektes in eine Schublade auf einer zweiten Diskette ist also eine Möglichkeit, eine Kopie dieses Objektes auf einer anderen Diskette anzulegen.

Beim Verlegen von Objekten gelten übrigens auch Disketten als »eine Art« Schubladen! Man kann deshalb auch Objekte genauso in ein Diskettenpiktogramm oder in ein Disketten-Fenster wie in eine Schublade legen.

### 3.2.7 Eine Schublade duplizieren

Mit dem Befehl *Duplicate* können Sie ganze Schubladen mit allem, was sich darin befindet, duplizieren. Selektieren Sie dazu mit der Maus das Schubladenpiktogramm und wählen Sie dann *Duplicate* aus dem Workbench-Menü. Daraufhin erscheint eine neue Schublade schräg unterhalb der alten auf dem Bildschirm. Diese neue Schublade erscheint danach neben der alten und erhält zunächst einmal den Namen *copy of <alter Name>* (den man natürlich mit *Rename* ändern kann).

Probieren Sie den *Duplicate*-Befehl am besten gleich aus: Öffnen Sie dazu (falls nötig) das Fenster der Workbench-Diskette und selektieren Sie mit einem Mausklick die Schublade *Empty*. Wählen Sie nun den Befehl *Duplicate* aus dem Menü *Workbench*. Das Diskettenlaufwerk arbeitet dann einen Moment lang und schließlich erscheint ein neues Schubladenpiktogramm mit dem Namen *copy of Empty* am Bildschirm.

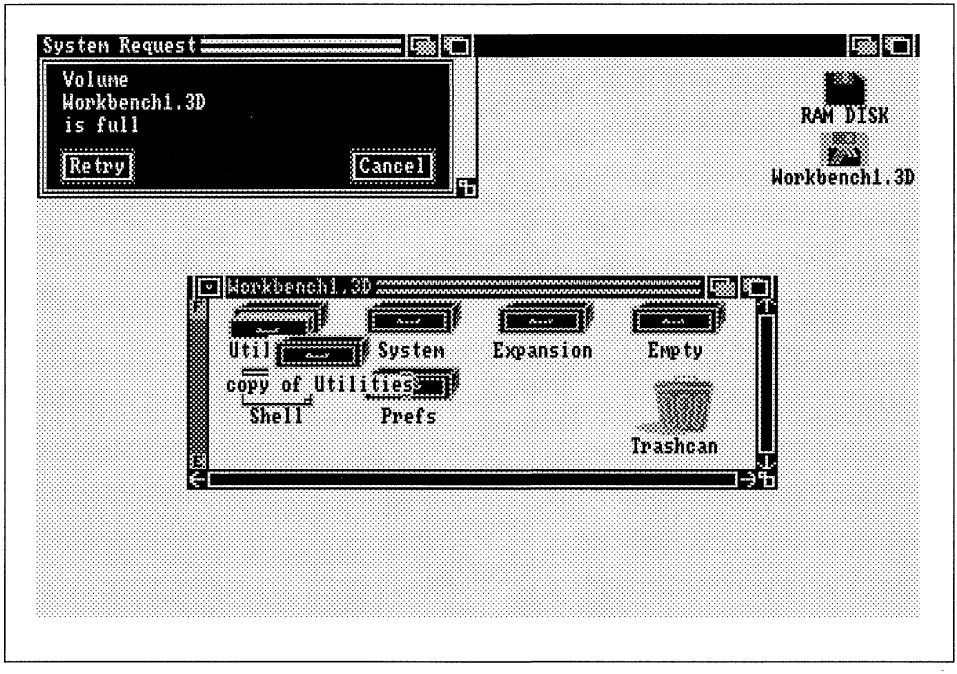



***Bild 3.10:*** *Die Fehlermeldung, die beim »Überlaufen« einer Diskette erscheint*

Der *Duplicate*-Befehl kann allerdings aus zwei verschiedenen Gründen fehlschlagen. Falls die Diskette, auf der sich die Schublade befindet, schreibgeschützt war, erscheint beim Aufruf des Befehls *Duplicate* ein neues Fenster mit einer Fehlermeldung darin auf dem Bildschirm. Wie Sie damit verfahren, wurde bereits weiter oben (im Abschnitt *Wie Fehlermeldungen aussehen*) beschrieben. Aber selbst wenn die Diskette nicht schreibgeschützt ist, kann eine Fehlermeldung auftauchen.

Wenn Sie eine Schublade duplizieren, wird ja nicht nur die Schublade selbst, sondern auch ihr gesamter Inhalt dupliziert (in dem obigen Beispiel war die Schublade allerdings leer). Von jedem Objekt, das sich in der alten Schublade befindet, gibt es danach zwei Versionen auf der Diskette: eine in der alten und eine in der neuen Schublade. Falls auf der Diskette aber nicht mehr genügend Platz für all die neuen Objekte ist, erscheint wieder eine Fehlermeldung, die besagt, daß die Diskette voll ist (engl. *Volume <name> is full*). Danach liegt zwar eine neue Schublade auf der Workbench. Dies ist aber keine vollständige Kopie der Originalschublade. Sie sollten diese unvollständige Kopie sicherheitshalber löschen (siehe unten). Wenn Sie das nicht tun, kann das später zu Fehlern führen.

### 3.2.8 Das Info-Fenster einer Schublade

Der Befehl *Info* liefert Ihnen detailliertere Informationen über eine Schublade. Selektieren Sie dazu die Schublade, die Sie interessiert, mit der Maus und wählen Sie dann den Befehl *Info* aus dem Menü Workbench. Das Fenster, in dem Ihnen diese Informationen dargeboten werden, sieht ein klein wenig anders aus als das Info-Fenster einer Diskette.

Probieren Sie es aus: Öffnen Sie dazu (falls nötig) das Fenster der Workbench-Diskette und selektieren Sie mit einem Mausklick die Schublade *Utilities*. Wählen Sie nun den Befehl *Info* aus dem Menü *Workbench*.

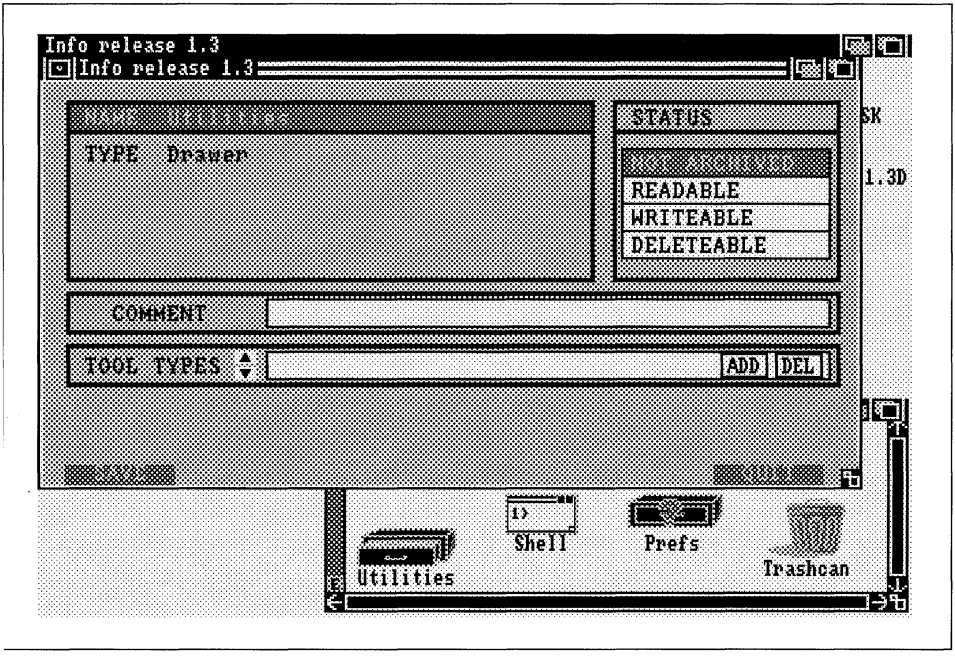



***Bild 3.11:*** *Das Info-Fenster einer Schublade*

Das Info-Fenster zeigt für eine Schublade im Gegensatz zu einer Diskette nicht den Speicherbedarf der Objekte in der Schublade an. Auch alle anderen Felder in diesem Fenster machen für Schubladen keinen Sinn. Diese Felder werden in Abschnitt 3.4 dieses Kapitels noch näher erläutert.

### 3.2.9 Eine neue Schublade erzeugen

Anders als bei einer wirklichen Werkbank können Sie der Workbench des Amiga jederzeit neue Schubladen und Schubladen in Schubladen hinzufügen. Das Erzeugen neuer Schubladen läuft dabei üblicherweise immer über das Vervielfältigen bereits vorhandener Schubladen ab. Dies ist ganz einfach. Wie bereits oben erwähnt, können Sie Schubladen duplizieren, indem Sie sie auswählen und dann den Befehl *Duplicate* aus dem Workbench-Menü aufrufen. Bei Schubladen bewirkt dieser Befehl das Anlegen einer Kopie der ursprünglichen Schublade und ihres gesamten Inhalts. Auf diese Art und Weise entsteht natürlich eine neue Schublade – eben die Kopie.

Benötigen Sie aber eine neue leere Schublade, um darin zum Beispiel einen Teil der Objekte aus einer anderen Schublade unterzubringen, so müssen Sie auch eine leere Schublade duplizieren. Auf der Standard-Workbench-Diskette ist deshalb eine leere Schublade untergebracht worden, die genau für diesen Zweck gedacht ist. (Sie trägt den Namen *Empty*; zu deutsch leer.)

Wenn Sie in irgendeiner Schublade eine neue Unterschublade unterbringen wollen, müssen Sie deshalb diese leere Schublade im Disketten-Fenster der Workbench-Diskette auswählen und duplizieren. Mit dem Befehl *Rename* (siehe oben) können Sie der Kopie danach den gewünschten Namen geben. Zuletzt können Sie sie dann in die Schublade legen, in der Sie eine neue Schublade benötigen. Sie können die Schublade *copy of Empty* aber natürlich auch zuerst dorthin legen, wo Sie sie benötigen und dann umbenennen.

### 3.2.10 Eine besondere Schublade: Der Mülleimer

Ein Symbol, das Ihnen bestimmt sofort aufgefallen ist, nachdem Sie das erste Mal ein Diskettenpiktogramm geöffnet haben, ist der Mülleimer (engl. *Trashcan*). Der Mülleimer dient zum Löschen anderer Objekte. Er verhält sich dabei im wesentlichen wie eine »Schublade«.

Wollen Sie ein beliebiges Objekt (zum Beispiel eine Schublade) auf der Workbench löschen, so nehmen Sie es einfach und legen es in den Mülleimer auf derselben Diskette. Das geht genauso wie das Verlegen eines Objektes in eine andere Schublade. Man ergreift das Objekt mit der Maus, bewegt es über das Mülleimerpiktogramm und läßt die Maustaste los.

Probieren Sie es aus: In Abschnitt 3.2.7 haben Sie eine Kopie der Schublade *Empty* erzeugt. Sie trägt den Namen *copy of Empty*. Ergreifen Sie jetzt diese Kopie mit der Maus und legen Sie sie in den Mülleimer.

Wenn man ein Objekt in den Mülleimer gelegt hat, ist es noch nicht sofort verloren. Wie »in Wirklichkeit« liegt es zunächst einmal im Mülleimer und kann wieder daraus hervorgeholt werden – solange die Müllabfuhr noch nicht da war. Hierzu kann man den Mülleimer öffnen wie jede andere Schublade auch. Daraufhin erscheint ein Schubladenfenster, in dem alle

Objekte auftauchen, die man bisher in den Mülleimer geworfen hat. Haben Sie es sich anders überlegt und wollen Sie eines davon nicht löschen, so können Sie es mit der Maus greifen und in eine andere Schublade legen.

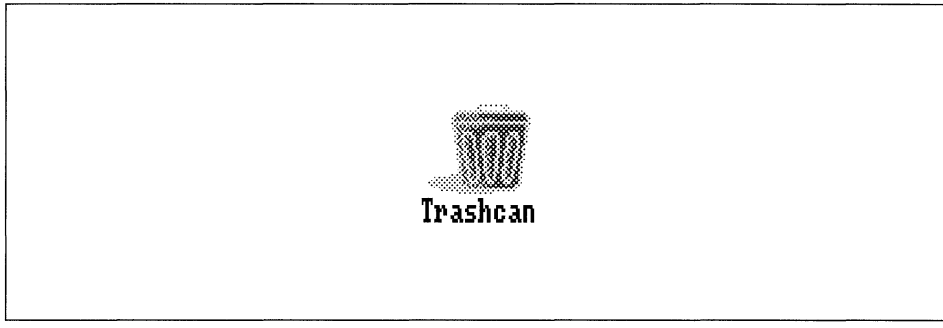



***Bild 3.12:*** *Das Mülleimerpiktogramm*

Öffnen Sie jetzt einmal den Mülleimer (mit einem Doppelklick auf das Piktogramm). Wie Sie sehen, befindet sich die Schublade *copy of Empty*, die Sie gerade hineingelegt haben, nach wie vor unversehrt – nicht einmal verbeult – im Mülleimer.

Erst wenn die Müllabfuhr gekommen ist, sind die Objekte, die in den Mülleimer geworfen wurden, unwiederbringlich verloren. Auf dem Amiga können Sie selbst bestimmen, wann die Müllabfuhr kommt. Wenn Sie ganz sicher sind, die Objekte im Mülleimer nicht mehr zu benötigen, selektieren Sie das Mülleimerpiktogramm und wählen danach *Empty Trash* aus dem *Disk*-Menü. (Der Befehl ist nur dann möglich, wenn Sie wirklich zuvor ein Mülleimerpiktogramm selektiert haben; sonst erscheint er in gerasterter Schrift und kann nicht ausgewählt werden.) Daraufhin werden die Objekte im Mülleimer wirklich von der Diskette gelöscht. Enthält der Mülleimer eines oder mehrere Objekte, die mit dem Info-Befehl vor Löschung geschützt wurden, so ergibt *Empty Trash* allerdings eine Fehlermeldung und die geschützten Objekte werden auch nicht entfernt. Es kann aber durchaus sein, daß zu dem Zeitpunkt, wo der Amiga diesen Fehler bemerkt hat, bereits Objekte, die im Mülleimer lagen und nicht geschützt waren, gelöscht wurden!

Probieren Sie diesen Befehl am besten sofort einmal aus: Schließen Sie das Fenster des Mülleimers, das Sie gerade geöffnet haben, klicken Sie auf das Mülleimerpiktogramm und wählen Sie den Befehl *Empty Trash* aus dem *Disk*-Menü. Wenn Sie den Mülleimer jetzt wieder (mit einem Doppelklick) öffnen, ist er leer. Die Schublade *copy of Empty*, die sich gerade noch darin befand, ist gelöscht.

Mülleimer sind aber nicht nur deshalb besondere Schubladen, weil man sie zum Löschen anderer Piktogramme verwenden kann. Sie können auch nicht aus dem Diskettenfenster entfernt werden. Sie können weder in eine andere Schublade noch in ein anderes Diskettenfenster gelegt werden. Auch können Mülleimer nicht mit dem Befehl *Discard* (siehe unten) gelöscht werden. Besitzt eine Diskette erst einmal ein Mülleimerpiktogramm, so kann dieses nicht wieder entfernt werden! (Zumindest nicht von der Workbench aus. Vom Shell-Fenster aus, das in den Kapiteln 6 bis 8 erläutert wird, kann auch der Mülleimer gelöscht werden.)

## 3.3 Disketten und Festplatten

Disketten und Festplatten (»Datenträger«) sind nach den Schubladen die zweitwichtigste Gruppe von Piktogrammen. In vieler Hinsicht können Disketten genauso behandelt werden wie Schubladen. Sie können zum Beispiel auf dieselbe Weise geöffnet und geschlossen, umbenannt und dupliziert werden. Noch wichtiger ist aber, daß Diskettenfenster und Diskettenpiktogramme sich ähnlich verhalten wie Schubladenfenster und Schubladenpiktogramme.

Ergreift man zum Beispiel ein Objekt mit der Maus und läßt es über dem Piktogramm einer Diskette oder Festplatte fallen (indem man dort die Maustaste löst), so verschwindet das Objekt an seiner alten Position und erscheint im Fenster dieser Diskette (sofern dieses geöffnet ist). Man muß Objekte aber nicht unbedingt in das Piktogramm einer Diskette, sondern kann sie auch direkt in das entsprechende Diskettenfenster legen.

Befand sich das »alte Objekt« vor seiner Verlegung in einer Schublade auf einer anderen Diskette, so findet auch beim Verlegen in ein Diskettenfenster oder -Piktogramm ein Kopiervorgang statt. Auf der zweiten Diskette erscheint eine Kopie des Objektes. Das alte Objekt selbst verbleibt in seiner »alten« Schublade.

In einer Hinsicht unterscheiden sich Festplatten und Disketten aber von Schubladen. Diese Datenträger können zwar Schubladen enthalten, umgekehrt gilt dies jedoch nicht. Versucht man ein Diskettenpiktogramm in ein anderes Fenster oder auf ein Schubladenpiktogramm zu legen, so erscheint eine Fehlermeldung. Disketten können allerdings in andere Diskettenpiktogramme gelegt werden. Dadurch wird der Inhalt der zweiten Diskette vollständig durch den der bewegten Diskette ersetzt. Das Legen eines Diskettenpiktogramms auf ein anderes bedeutet also ein Kopieren der Diskette. Diesen Vorgang haben Sie ja bereits im Kapitel 2 kennengelernt.

### 3.3.1 Der Aufbau eines Diskettenfenster

Wenn Sie einen Doppelklick (mit der linken Maustaste) auf ein Disketten- oder Festplattenpiktogramm machen, öffnet sich am Bildschirm ein neues Fenster, in dem der Inhalt des Datenträgers dargestellt wird. Öffnen Sie am besten jetzt gleich einmal das Piktogramm der Workbench-Diskette, um sich ein solches Fenster anschauen zu können.

Den Aufbau und die möglichen Operationen mit einem solchen Fenster haben Sie zum großen Teil ebenfalls schon im Kapitel 2 und im Abschnitt über Schubladen kennengelernt. Diskettenfenster und Diskettenpiktogramme verhalten sich, bis auf wenige Unterschiede, genauso wie Schubladenfenster und Schubladenpiktogramme. Hauptunterschied zwischen einem Disketten- und einem Schubladenfenster ist die Füllanzeige. Dies ist am linken Rand des Fensters, gegenüber dem vertikalen Rollbalken angebracht. Sie besteht aus einem schmalen senkrechten Balken, dessen unterer Teil orange und dessen oberer schwarz ist. Am unteren Ende des Balkens steht ein *E* für *empty* (engl. für leer), am oberen Ende ein *F* für *full* (engl. für voll). Der orangefarbene Teil kennzeichnet den Teil der Diskette, der bereits gefüllt ist, der schwarze den noch freien Teil. Ist der orangefarbene Balken also etwa halb so hoch wie das Fenster, ist die Diskette halb voll – oder halb leer; wie man es betrachtet. Ist der ganze Balken orange, ist die Diskette voll und kann keine neuen Daten mehr aufnehmen. Sie haben mit diesem Balken immer einen ungefähren Überblick über den

»Füllgrad« einer Diskette. Benötigen Sie genauere Daten, so können Sie dies mit dem *Info*-Befehl herausfinden, der im folgenden Abschnitt erläutert wird.

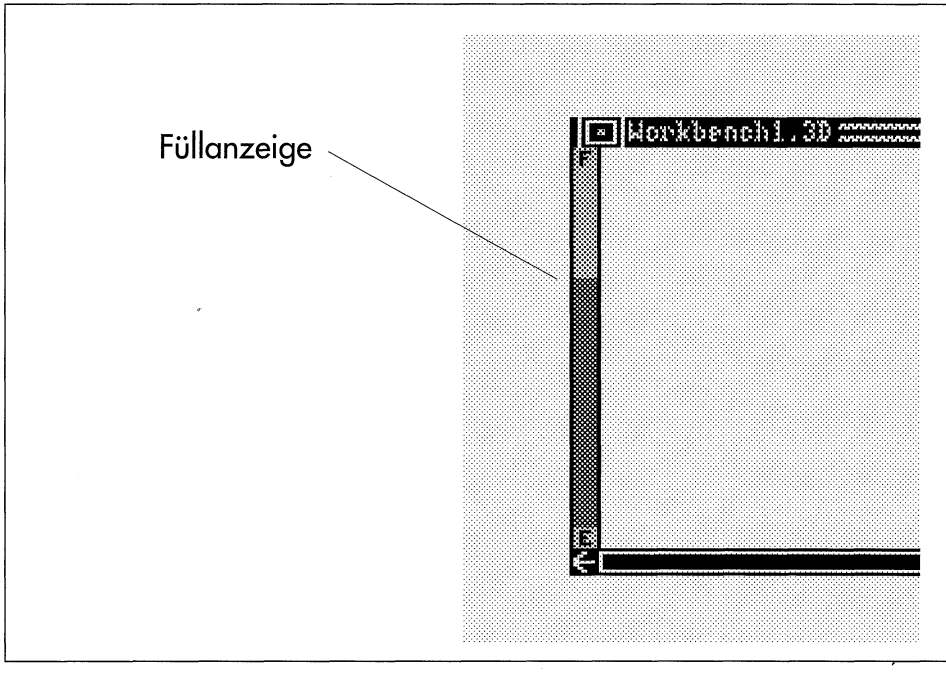



***Bild 3.13:*** *Die Disketten-Füllanzeige*

### 3.3.2 Das Info-Fenster einer Diskette

Wenn Sie mehr über eine Diskette wissen wollen, als nur ihren Namen, so müssen Sie, wie üblich, das Diskettenpiktogramm auswählen und dann den Befehl *Info* aus dem Menü Workbench wählen. Das Info-Fenster zeigt in der oberen linken Ecke den Typ (*TYPE Disk*) und, unter der Typangabe, einen Überblick über die exakte Platzausnutzung der Diskette. Hinter *Number of Blocks* erfahren Sie, in wieviele »physikalische Blocks« die Diskette aufgeteilt ist. Bei 3,5-Zoll-Disketten ist die Zahl immer gleich 1758; andere Diskettensorten und Festplatten verfügen auch über andere Zahlen von Blocks. Darunter finden Sie die Angaben darüber, wieviele dieser Blocks belegt (hinter *Number Used*) und wieviele noch frei sind (hinter *Number Free*). Die letzte Angabe schließlich (*Bytes per* Block) sagt Ihnen, wieviele Buchstaben beziehungsweise Bytes in jedem der Blocks Platz haben, von denen oben die Rede war. Bei einer Diskette sind es üblicherweise 512; falls Sie Besitzer einer Festplatte sind, kann hier aber durchaus eine andere Zahl stehen.

Wenn Sie diese Zahlen miteinander multiplizieren, kommen Sie darauf, daß eine 3,5-Zoll-Diskette maximal etwa 900 000 Byte aufnehmen (1758 Blocks * 512 Byte) aufnehmen kann. Das entspricht etwa 880 Kbyte (1 Kbyte = 1024 Byte).

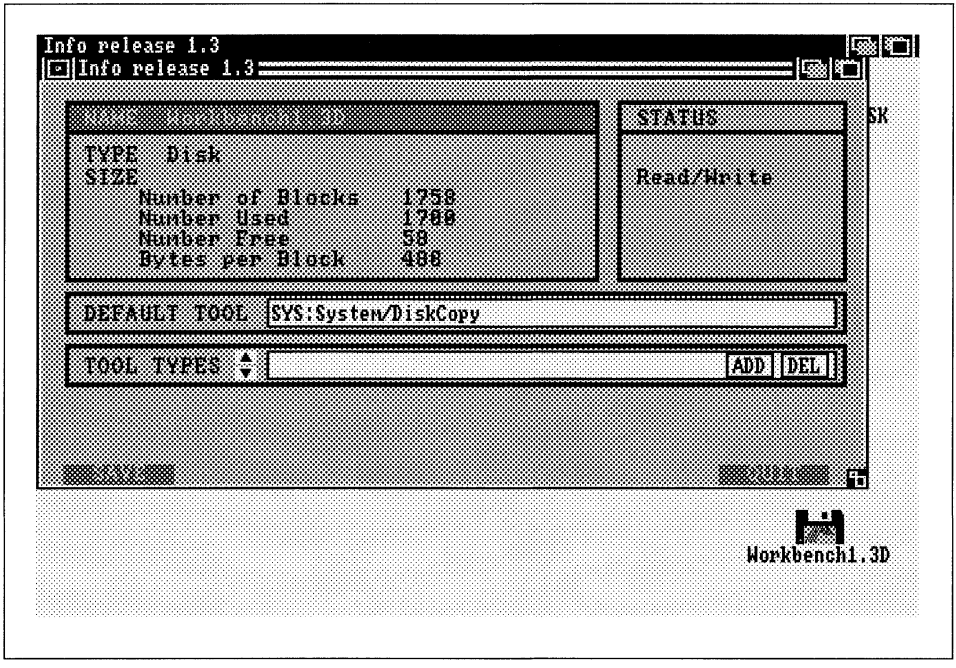



***Bild 3.14:*** *Das Info-Fenster einer Diskette*

Neben diesem Feld mit den vielen Zahlen befindet sich das Feld *STATUS*. Falls Sie die Diskette mit dem kleinen Plastikschieber schreibgeschützt haben, steht in diesem Feld *Read Only* (engl. für *nur lesen*) ansonsten immer *Read/Write* (engl. für *lesen und schreiben*). Die beiden Felder darunter (*Default Tool* und *Tool Types*) sollen uns vorläufig nicht interessieren. Wie Sie diese Felder bei bestimmten anderen Arten von Piktogrammen sinnvoll verwenden können, werden Sie im folgenden Kapitel erfahren.

### 3.3.3 Eine Diskette kopieren

Disketten können, wie alle anderen Piktogramme, natürlich auch kopiert beziehungsweise dupliziert werden. Hierzu gibt es sogar zwei Wege. Beide sind im Kapitel 2 bereits beschrieben worden. Hier nur noch einmal ein paar kurze Stichworte zur Erinnerung:

Wenn Sie nur ein Diskettenlaufwerk besitzen, müssen Sie das Piktogramm der Diskette selektieren und dann den Befehl *Duplicate* aus dem Workbench-Menü wählen, wenn Sie eine Diskette kopieren wollen. Danach erscheint nacheinander eine Reihe von Dialogfenstern (Requestern), die Sie auffordern, entweder die Originaldiskette (*FROM disk*) oder die Diskette, die die Kopie aufnehmen soll (*DESTINATION disk*), einzulegen. Folgen Sie dieser Aufforderung und drücken Sie dann jeweils auf den Knopf *Continue* im Dialogfenster.

Wenn Sie mit zwei Diskettenlaufwerken arbeiten, haben Sie es leichter: Legen Sie die schreibgeschützte Originaldiskette in ein Laufwerk, und eine neue Diskette, die die Kopie aufnehmen soll, in das zweite Laufwerk. Ergreifen Sie dann das Piktogramm der Original-

diskette mit der Maus und legen Sie es auf das Piktogramm der zweiten Diskette. Dann erscheint wieder ein Dialogfenster, mit den Knöpfen *Cancel* und *Continue*. Wenn Sie sicher sind, alles richtig gemacht zu haben, brauchen Sie nur noch auf *Continue* zu klicken, und der Kopiervorgang beginnt. *Cancel* bricht – wie üblich – den Befehl ab.

### 3.3.4 Eine Diskette löschen oder initialisieren

Wenn Sie auf eine der eben beschriebenen Arten eine ganze Diskette kopieren, so kann die Diskette, die die Kopie aufnehmen soll, frisch aus der Packung kommen. Sie brauchen sie nicht weiter vorzubereiten, damit sie Daten aufnehmen kann. Wenn Sie jedoch einzelne Piktogramme (zum Beispiel eine Schublade) auf eine neue Diskette legen wollen, muß die Diskette vorher erst einmal initialisiert werden. Dieser Vorgang teilt die zunächst unstrukturierte magnetische Oberfläche in einzelne Blöcke zu 512 Byte auf, die Sie ja schon vom Info-Befehl her kennen. Auch wird dabei Platz für das Dateiverzeichnis angelegt, eine Art Landkarte, die den Programmen sagt, wo welche Daten auf der Diskette zu finden sind.

Beim Kopieren einer ganzen Diskette wird die Diskette, die die Kopie aufnimmt, auch gleich initialisiert. Dies geschieht automatisch, ohne Ihr Zutun und ohne, daß Sie etwas davon bemerken.




***Bild 3.15:*** *Dialogfenster zur Bestätigung des Initialisierens einer Diskette*

Um eine neue Diskette zu initialisieren, müssen Sie diese erst einmal in ein Diskettenlaufwerk einlegen (wozu Sie eventuell eine andere Diskette herausnehmen müssen). Wenn es

sich wirklich um eine frische, noch unbenutzte Diskette handelt, erscheint das entsprechende Piktogramm unter dem Namen DF0:BAD oder DF1:BAD (je nachdem, ob Sie sie in das interne oder in ein externes Laufwerk gelegt haben). Markieren Sie dieses Piktogramm mit einem Mausklick und wählen Sie dann den Befehl *Initialize* im Menü *Disk* aus.

Die Workbench fordert nun die Workbench-Diskette an (*Please replace volume WB-Disk in any drive*), wenn Sie diese herausnehmen mußten, um die neue Diskette einlegen zu können. Nehmen Sie dann die neue Diskette wieder aus dem Laufwerk und legen Sie die Workbench-Diskette ein. Nun werden Sie von einem Requester aufgefordert, die Diskette, die Sie initialisieren wollen, einzulegen (*Please insert disk to be initialized in drive DF0:*). Tun sie das und klicken Sie auf den Knopf *Continue*. Nun erscheint noch ein Requester und fragt Sie, ob Sie sich Ihrer Sache auch sicher sind (*Ok to Initialize disk in drive DF0:?*). Klicken Sie in diesem Requester nur dann auf *Continue*, wenn Sie sich sicher sind, daß die richtige Diskette im Laufwerk liegt. Ansonsten können Sie den Vorgang mit einem Klick auf *Cancel* abbrechen.

Das Initialisieren kann nicht nur bei neuen Disketten nötig sein. Es ist auch manchmal sinnvoll, bereits verwendete Disketten, deren Inhalt Sie (vielleicht aus Geheimhaltungsgründen) löschen wollen, zu initialisieren. Das Initialisieren zerstört alle Daten, die sich eventuell vorher auf einer Diskette befanden! Achten Sie deshalb beim Initialisieren von schon benutzten Disketten darauf, daß es auch die »richtigen« sind. Sie können sich aber vor Fehlern schützen, indem Sie den Schreibschutz wichtiger Disketten aktivieren. Schreibgeschützte Disketten können nicht initialisiert werden!

### 3.3.5 Festplatten

Die meisten eben beschriebenen Operationen können Sie mit jedem Piktogramm eines Datenträgers durchführen – egal, ob es sich um eine 3,5-Zoll-Diskette, eine 5,25-Zoll-Diskette oder eine Festplatte handelt. In einigen wenigen Punkten unterscheiden sich Festplatten aber von Disketten. So können Sie Festplatten im allgemeinen nicht mit dem Befehl *Duplicate* oder durch Legen des Festplattenpiktogramms auf ein anderes Festplatten- oder Disketten-Piktogramm kopieren. Sie erhalten lediglich eine knappe Fehlermeldung in der Titelzeile des Bildschirm, wenn Sie das versuchen sollten.

Das liegt einfach daran, daß Festplatten und Disketten sehr unterschiedliche Speicherkapazitäten haben und die Daten einer Festplatte nicht auf eine Diskette passen. Auch der umgekehrte Kopiervorgang (eine Diskette auf eine Festplatte kopieren) funktioniert mit den auf der Workbench zur Verfügung stehenden Befehlen nicht. Sie werden in den Kapiteln über das sogenannte CLI noch erfahren, wie Sie den kompletten Inhalt einer Diskette in eine Schublade auf einer Festplatte legen können. Nur mit der Workbench geht es nicht.

Auch das Kopieren einer Festplatte auf eine andere funktioniert meist nicht – selbst dann nicht, wenn Sie zwei Festplatten (angeblich) gleicher Kapazität an Ihren Amiga angeschlossen haben. Denn auch zwischen zwei Festplatten, deren Kapazität mit 20 Mbyte angegeben wird, kann es Unterschiede geben (und gibt es in den allermeisten Fällen auch), die verhindern, daß eine exakte Kopie des Inhalts der einen Festplatte auf der anderen Festplatte angelegt wird.

Um trotz dieser Probleme Sicherungskopien des Inhalts einer Festplatte anlegen zu können, gibt es spezielle Programme, die den kompletten Inhalt der Festplatte auf mehrere Disketten

verteilen. Anhand dieser Disketten kann der alte Inhalt der Festplatte jederzeit wiederhergestellt werden, wenn die Festplatte einmal einen Fehler aufweisen sollte. Solche Programme liegen einigen Festplatten beim Kauf bei oder können auch als separate Software-Produkte gekauft werden. Fragen Sie im Zweifelsfall den Händler, bei dem Sie die Festplatte gekauft haben.

## 3.4 Dateien, Werkzeuge und Projekte

Alle Objekte, die Sie bis jetzt kennengelernt haben, waren mehr oder weniger Ordnungshilfen für Datenträger (Disketten und Festplatten). Die Piktogramme, die diese repräsentieren, sind Disketten, Schubladen und Mülleimer. Es gibt aber auch noch andere Piktogramme in den Fenstern – zum Beispiel der Workbench-Diskette und in ihren Schubladen. Bis jetzt wurde ja immer sehr abstrakt von »Objekten« gesprochen, womit alle möglichen Piktogramme in einem Schubladenfenster gemeint waren. Disketten, Schubladen und Mülleimer haben wir bereits kennengelernt. Was aber steckt hinter den anderen Piktogrammen auf der Workbench?

### 3.4.1 Werkzeuge und Projekte

Alle Objekte, die Sie in einem Disketten- oder Schubladenfenster sehen können, und bei denen es sich nicht um Schubladen oder Mülleimer handelt, sind »Werkzeuge« oder »Projekte.«

Getreu der Analogie, daß die Bedienung des Amiga abläuft wie die Arbeit auf einer Werkbank (Workbench), heißen die Programme des Amiga nämlich Werkzeuge (engl. *tools*) und die Dateien, die diese Programme erzeugen oder für ihre Arbeit benötigen, heißen Projekte (engl. *projects*). Ein Brief oder ein kurzes Memo, das Sie mit einem Textverarbeitungsprogramm Textcraft geschrieben haben, wäre also ein Projekt, das mit dem Werkzeug Textcraft bearbeitet wurde.

Jedes Werkzeug kann als nahezu beliebiges Piktogramm am Bildschirm erscheinen. Meist deutet das Aussehen des Piktogramms aber zugleich schon die Funktion des dahintersteckenden Programms an. Eine Textverarbeitung wird also vielleicht wie eine kleine Schreibmaschine aussehen und ein Grafikprogramm wie eine Palette mit Pinsel.

Zwischen Werkzeugen und den Projekten, die man damit bearbeitet, besteht ein enger Zusammenhang. Meist kann ein Projekt nur mit einem ganz bestimmten Werkzeug bearbeitet werden. Ein Textverarbeitungswerkzeug kann zum Beispiel keine Grafiken bearbeiten und umgekehrt (Ausnahmen kommen allerdings vor). Dieser Zusammenhang wird meist dadurch angedeutet, daß die Piktogramme von Projekten logisch zu den Werkzeugpiktogrammen passen. Texte werden also zum Beispiel wie ein Blatt Papier mit einem Eselsohr und ein paar angedeuteten Buchstaben aussehen und von der Aufmachung (zum Beispiel von den Farben) her dem Textverarbeitungsprogramm ähnlich sehen.

Wenn Sie jetzt gerne einmal ein solches Werkzeug sehen wollen, öffnen Sie einfach die Workbench-Diskette. Warten Sie einen Moment, bis das Diskettenlaufwerk zur Ruhe gekommen ist und öffnen Sie dann die *Utilities*-Schublade. In diesem Fenster sind eine ganze Reihe von Piktogrammen zu sehen, die weder wie Schubladen noch wie Disketten

oder Mülleimer aussehen. Bei allen diesen Piktogrammen handelt es sich um Werkzeuge (Tools). Das Programm *Clock* (engl. für Uhr) zeigt Ihnen zum Beispiel – wie der Name schon sagt – am Bildschirm eine Uhr und hinter *Calculator* (engl. für Rechner) verbirgt sich ein Taschenrechner, den Sie mit der Maus oder auch mit der Tastatur Ihres Amiga bedienen können.

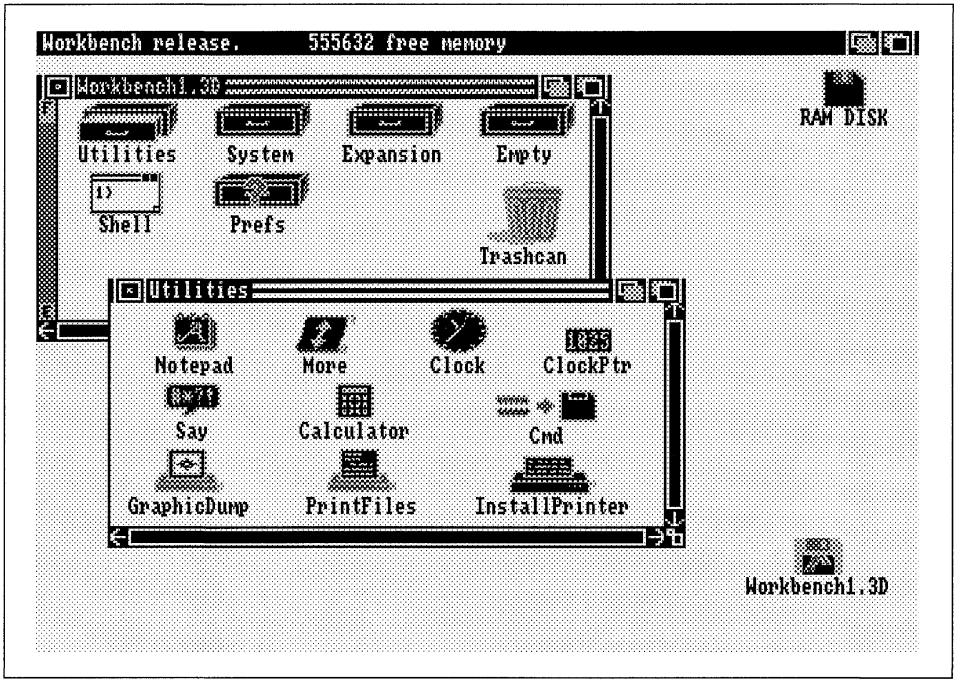



***Bild 3.16:*** *Einige Werkzeuge in der Schublade* Utilities

Damit sind die wesentlichen Eigenschaften von Werkzeugen und Projekten aber auch schon geklärt. Im Rest dieses Kapitels sollen Werkzeuge und Projekte aber nur aus der Sicht der Workbench betrachtet werden. Mehr zum Thema Werkzeuge und Projekte erfahren Sie im nächsten Kapitel.

### 3.4.2 Operationen für Werkzeuge und Projekte

Auf der Workbench können Werkzeuge und Projekte wie alle anderen Piktogramme behandelt werden. Die meisten Befehle und Operationen funktionieren wie bei Schubladen. Werkzeuge und Projekte können zum Beispiel mit der Maus ergriffen und bewegt werden. Sie können sie in Schubladen legen, indem Sie sie in das Schubladenpiktogramm oder in ein Schubladenfenster legen. Werden sie in eine Schublade, die sich auf einer anderen Diskette befindet, oder direkt in das entsprechende Diskettenfenster gelegt, kopiert die Workbench das Werkzeug beziehungsweise das Projekt auf diese andere Diskette – genauso, wie es bei einer Schublade der Fall wäre.

Aus dem Menü *Disk* können keine Befehle angewendet werden, wenn ein Werkzeug oder Projekt selektiert ist. Dafür sind fast alle Befehle des Menüs *Workbench* anwendbar.

Der Befehl *Open* wird ausführlich im nächsten Kapitel erläutert. (Sie sollten ihn deshalb vorläufig nicht auf Projekt- oder Werkzeugpiktogramme anwenden und auch keinen Doppelklick auf ein solches Piktogramm machen!)

Der Befehl *Close* kann nicht auf Werkzeuge oder Projekte angewendet werden. Mit dem Befehl *Duplicate* kann (auf derselben Diskette) eine Kopie eines Werkzeugs oder Projektes angelegt werden, die zunächst den Namen *copy of <alter Name>* trägt. Diesen Namen kann man danach mit dem Befehl *Rename* beliebig ändern. Ist nicht genügend Platz auf der Diskette, um die Kopie aufzunehmen, erhalten Sie statt des neuen Piktogramms eine Fehlermeldung in der Titelzeile des Bildschirms.

Mit dem Befehl *Info* bekommen Sie detailliertere Informationen über ein Werkzeug oder Projekt in einem neuen Fenster angezeigt, das ganz ähnlich aussieht wie das Info-Fenster einer Diskette oder einer Schublade. Näheres über die Möglichkeiten, die Ihnen im Info-Fenster eines Projekts oder eines Werkzeugs zur Verfügung stehen, erfahren Sie im nächsten Kapitel.

## 3.5 Weitere wichtige Workbench-Befehle

Die meisten Befehle der Workbench haben Sie nun – in diesem oder im Kapitel 2 – kennengelernt. Einige Befehle sind bisher aber unerwähnt geblieben, wie Ihnen vielleicht aufgefallen sein wird, wenn Sie die Menüs einmal aufmerksam durchblättert haben. Diese Befehle und einige zusätzliche Tips und Handgriffe werden Sie in den folgenden Abschnitten kennenlernen.

### 3.5.1 Löschen mit dem Discard-Befehl

Wenn Sie täglich mit dem Amiga arbeiten, wird es mit Sicherheit gelegentlich nötig werden, auf Ihren Disketten »aufzuräumen«. Damit ist natürlich unter anderem auch gemeint, daß alle Objekte ordentlich in Schubladen untergebracht werden, damit man sie später auch wiederfindet. Ab und zu werden Sie aber auch Objekte oder Schubladen, die durch eine Reorganisation überflüssig geworden sind, löschen müssen. Die sicherste Methode, ein Objekt zu löschen, besteht darin, es zunächst einmal in den Mülleimer zu legen und später, wenn der Platz auf der Diskette knapp wird, den Mülleimer leeren zu lassen. Der Vorteil des Mülleimers ist, daß Sie es sich auch »noch einmal anders überlegen können«. Objekte, die im Mülleimer sind, sind ja auch in Wirklichkeit noch nicht zerstört oder verloren, sondern können wieder hervorgeholt werden. Diese Löschmethode ist andererseits aber relativ umständlich und langwierig, wenn Sie etwas Bestimmtes sofort löschen wollen, und sich Ihrer Sache sehr sicher sind. Für solche Fälle gibt es den Befehl *Discard*.

Am besten, Sie probieren Ihn gleich aus: Erzeugen Sie dazu erst einmal eine Kopie der Schublade *Empty*, indem Sie diese selektieren und dann den Befehl *Duplicate* aufrufen. Dieses Duplikat, das unter dem Namen *copy of Empty* im Fenster der Workbench-Diskette auftaucht, werden Sie jetzt löschen. Selektieren Sie dazu das neue Piktogramm (mit einem Mausklick) und wählen Sie dann den Befehl *Discard* aus dem Menü *Workbench* aus.

Daraufhin erscheint ein Dialogfenster, das Sie über die Konsequenzen (unwiederbringlicher Verlust der Schublade und aller Objekte darin) informiert und Ihnen noch einmal die Chance gibt, es sich anders zu überlegen.

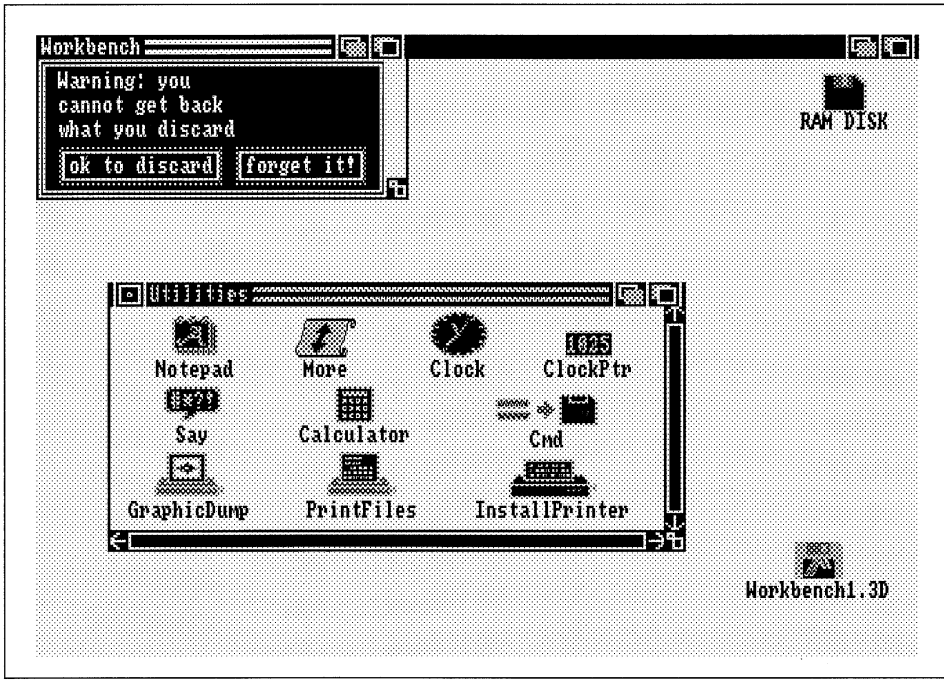



***Bild 3.17:*** *Das Dialogfenster zur Bestätigung des Discard-Befehls*

Dieses Fenster besitzt am unteren Rand zwei Knöpfe, *ok to discard* (engl. für ich bin damit einverstanden, dieses Objekt zu löschen) und *forget it!* (engl. vergiß es). Drückt man (mit Mauszeiger und Maustaste) auf *ok to discard*, so wird der Löschvorgang durchgeführt, und danach gibt es wirklich keine Möglichkeit mehr, das selektierte Objekt zu retten. Drückt man aber auf den Knopf *forget it!*, wird der Vorgang abgebrochen und es geschieht nichts weiter.

Mit *Discard* kann man natürlich nicht nur Schubladen, sondern (fast) alle Objekte auf einer Diskette oder Festplatte löschen. Das Löschen von Schubladen ist aber besonders gefährlich, da man oft nicht mehr genau weiß, was diese enthalten. Vor dem Löschen einer Schublade (egal ob mit *Discard* oder über den Mülleimer) sollte man diese deshalb sicherheitshalber öffnen und sich vergewissern, daß nur Objekte in ihr liegen, die nicht mehr benötigt werden!

Den Mülleimer sollten Sie aber der Verwendung des Befehls *Discard* im Zweifelsfall vorziehen. Legen Sie alle Objekte, die Sie löschen wollen, zunächst in den Mülleimer und leeren Sie diesen erst, wenn Sie dringend mehr Platz auf der entsprechenden Diskette benötigen! Dies läßt Ihnen die Chance, die so »weggeworfenen« Objekte bei einem Irrtum wieder zurückzuholen.

### 3.5.2 Eine Schublade aufräumen

Die Objekte, die in einem Schubladen- oder Diskettenfenster gezeigt werden, können Sie mit der Maus an jede beliebige Stelle schieben. Dadurch kann der Fensterinhalt manchmal etwas »unaufgeräumt« erscheinen. Falls Sie ein bißchen Ordnung im Diskettenfenster schaffen wollen, können Sie den Befehl *Clean up* (engl. für aufräumen) verwenden. Wählen Sie dazu das Disketten- oder Schubladenpiktogramm aus, das zu dem Fenster gehört (es reicht nicht, das Fenster anzuklicken!) und wählen Sie dann *Clean up* aus dem Menü *Special*. Die Piktogramme werden dann vom Amiga selbst in »Reih und Glied« gelegt.

Die Ergebnisse des *Clean up*-Befehls sind allerdings nicht immer so, wie man sich die Ordnung auf seiner Diskette vorstellt. Dies liegt unter anderem daran, daß die Piktogramme verschieden groß sein können. Sind Sie mit dem Ergebnis des Aufräumens nicht zufrieden, bleibt Ihnen nichts anderes übrig, als die Objekte selbst »von Hand« so zu legen, wie Sie es für richtig halten.

### 3.5.3 Einen Schnappschuß machen

Wenn Sie ein Piktogramm zu einer neuen Position legen, ein Fenster aufräumen (siehe oben), verschieben oder vergrößern, so »halten« alle diese Änderungen nur bis zum Abschalten des Amiga oder bis zum nächsten Neustart. Wenn Sie wollen, daß solche Änderungen auch einen Neustart überstehen, müssen Sie einen »Schnappschuß« machen, der diese Änderungen auf der Diskette abspeichert. Legen Sie dazu das Piktogramm mit der Maus an die gewünschte Position und wählen dann den Befehl *Snapshot* (engl. für Schnappschuß) aus dem Menü *Special*. Dieser Befehl speichert die aktuelle Position des Piktogramms auf der Diskette ab. Auch beim nächsten Start des Amiga erscheint das Piktogramm dann an dieser neuen Position. Ein Schnappschuß ist natürlich nur möglich, wenn die Diskette nicht schreibgeschützt ist. Wenn Sie versuchen, einen Schnappschuß auf einer schreibgeschützten Diskette zu machen, ergibt das eine Fehlermeldung.

Wenn es sich bei dem Piktogramm, von dem Sie einen Schnappschuß machen, um eine Diskette, Festplatte oder um eine Schublade handelt und deren Fenster gerade offen ist, wird nicht nur die Position des Piktogramms, sondern auch die Größe und Position des Fensters als Schnappschuß mit abgespeichert. Deshalb sollten Sie immer, wenn Sie den Befehl *Snapshot* auf Schubladen oder Disketten anwenden, darauf achten, daß Ihnen auch die Position und Größe, die das Fenster gerade hat, gefallen.

### 3.5.4 Mehr als ein Piktogramm mit einem Befehl bearbeiten

Wenn bislang in diesem Buch ein Befehl oder eine Operation vorgestellt wurde, so wurde immer nur beschrieben, wie Sie diesen Befehl auf genau ein Piktogramm anwenden können. Wenn Sie mehrere Piktogramme bearbeiten wollen, zum Beispiel zwei Schubladen auf einmal kopieren wollen oder einen Schnappschuß von drei Werkzeugen machen wollen, kann das wiederholte Aufrufen desselben Befehls lästig werden. Deswegen gibt es die erweiterte Selektion, durch die es möglich ist, mit einem Befehl gleichzeitig mehrere Piktogramme zu bearbeiten.

Probieren Sie diesen ausgesprochen praktischen Tip am besten sofort aus: Öffnen Sie dazu das Fenster der Workbench-Diskette (falls nötig) und klicken Sie das Piktogramm *Utilities* an. Die Schublade zeigt die Selektion durch eine Farb- oder Formänderung an. Drücken Sie dann eine der beiden [Shift]-Tasten rechts und links unten auf der Tastatur (die Umschalttasten, mit denen Sie sonst zwischen Klein- und Großbuchstaben wechseln) und halten sie fest. Klicken Sie nun bei gedrückter [Shift]-Taste das nächste Piktogramm *System* an. Auch dieses ist nun selektiert, während das erste selektiert bleibt. (Normalerweise führt die Selektion eines Piktogramms dazu, daß das zuletzt ausgewählte Piktogramm nicht mehr selektiert ist.) Fahren Sie nun mit dem Selektieren fort, und klicken Sie (immer noch bei gedrückter Shift-Taste) die Piktogramme *Prefs* und *Empty* an. Nun können Sie einen Menübefehl aufrufen, der alle selektierten Piktogramme nacheinander bearbeitet. Probieren Sie es am besten einmal mit *Snapshot* aus.

Sie können auf diese Weise auch eine Gruppe von Piktogrammen verschieben oder in eine andere Schublade legen. Halten Sie dazu einfach wieder die [Shift]-Taste fest und klicken Sie auf ein beliebiges der vorher ausgewählten markierten Piktogramme. Halten Sie die Maustaste fest und bewegen Sie die Maus. Alle selektierten Piktogramme folgen dann gleichzeitig den Mausbewegungen.

Sie können auf diese Weise unter anderem auch mehrere Schubladen auf eine andere Diskette kopieren, indem Sie mit der Spitze des Mauszeigers über das Piktogramm der anderen Diskette gehen und dann die Maustaste loslassen. Diese Vorgehensweise ist aber nur dann empfehlenswert, wenn Sie auch über zwei Diskettenlaufwerke verfügen. Ansonsten müssen Sie sehr oft die beiden Disketten austauschen.

### 3.5.5 Kopieren mit der RAM-Disk

Sie können sich das Leben – beziehungsweise die Arbeit – mit nur einem Diskettenlaufwerk allerdings wesentlich einfacher machen, indem Sie geschickt die »RAM-Disk« des Amiga nutzen. Diese RAM-Disk ist eine Diskette, die die Amiga-Software simuliert, indem Sie einen Teil des Arbeitsspeichers (RAM-Speichers) wie eine Diskette verwendet.

Das Piktogramm der RAM-Disk trägt sinnigerweise den Titel *RAM DISK* und taucht sofort nach dem Start in der rechten oberen Ecke des Bildschirms auf. Zumindest ist das bei den meisten Workbench-Disketten der Fall, die Commodore seit 1988 ausliefert. Sollte bei Ihnen nach dem Start kein Piktogramm namens *RAM Disk* auftauchen, ist das auch kein Beinbruch. Sie müssen sich dann nur ein wenig gedulden. In Kapitel 8 werden Sie erfahren, wie Sie die RAM-Disk »einschalten« können, wenn das beim Start nicht gleich passiert.

Sollte es die RAM-Disk auf Ihrer Workbench schon geben, können Sie sie sofort sinnvoll einsetzen, um die Anzal der nötigen Diskettenwechsel beim Kopieren mit einem Diskettenlaufwerk zu reduzieren. Gehen Sie dabei wie folgt vor: Legen Sie zunächst die Diskette ein, auf der sich die Daten befinden, die kopiert werden sollen. Markieren Sie dann alle Ordner und sonstigen Piktogramme, die Sie kopieren wollen (siehe Abschnitt 3.5.4), ergreifen Sie diese und legen Sie sie auf das Piktogramm *RAM DISK*. Die Workbench kopiert dann die ausgewählten Objekte auf diese simulierte Diskette. Wenn der Kopiervorgang abgeschlossen ist (was Sie daran erkennen können, daß das Diskettenlaufwerk nicht mehr arbeitet), nehmen Sie die Diskette aus dem Laufwerk und legen die zweite Diskette, die die Kopien aufnehmen soll ein. (Diese Diskette muß schon initialisiert sein; siehe Abschnitt 3.3.4.)

Öffnen Sie dann das Fenster der RAM-Disk, markieren Sie alle Piktogramme in diesem Fenster und legen Sie sie auf das Piktogramm der zweiten Diskette. Die Workbench kopiert nun die Daten von der RAM-Disk auf die zweite Diskette. Wenn dieser Vorgang abgeschlossen ist, sollten Sie die RAM-Disk am besten wieder leeren. Selektieren Sie dazu noch einmal alle Piktogramme im Fenster der RAM-Disk und wählen Sie den Befehl *Discard* aus dem Menü *Workbench*.

Auf diese Art und Weise können Sie, indem Sie die RAM-Disk als Zwischenspeicher verwenden, mehrere Schubladen von einer auf eine andere Diskette kopieren und brauchen dazu nur zweimal die Disketten zu wechseln. (Wenn Sie zwei Diskettenlaufwerke besitzen, benötigen Sie diesen Umweg über die RAM-Disk selbverständlich nicht.) Sie sollten sich beim Gebrauch der RAM-Disk aber immer darüber im klaren sein, das diese keine echte Diskette ist, sondern nur unter Gebrauch des Amiga-Arbeitsspeichers simuliert wird. Dieser Arbeitsspeicher hat nur eine begrenzte Größe und kann im allgemeinen nicht den Inhalt einer ganzen Diskette aufnehmen. Wenn Sie den Amiga 500 mit der Grundausstattung von 500 Kbyte RAM verwenden, stehen davon zum Beispiel nur knapp 350 Kbyte zur Verwendung als RAM-Disk zur Verfügung (den Rest benötigt die Workbench und andere Teile der Amiga-Systemsoftware).

Wenn Sie eine Speichererweiterung besitzen, verfügt Ihr Amiga mindestens über 1000 Kbyte (= 1 Mbyte) RAM-Speicher, wovon maximal etwa 840 Kbyte für die RAM-Disk zur Verfügung stehen. Das entspricht in etwa dem Fassungsvermögen einer Diskette. Beachten Sie aber, daß der RAM-Speicher des Amiga auch für andere Zwecke benötigt wird. Jedes Programm, daß Sie starten, benötigt zum Beispiel RAM-Speicher. Deshalb sollten Sie die RAM-Disk immer so schnell wie möglich leeren – besonders dann, wenn Sie große Mengen von Daten darauf untergebracht haben. Wieviel RAM-Speicher zu einem bestimmten Zeitpunkt noch frei sind, können Sie übrigens jederzeit in der Titelzeile des Workbench-Bildschirm sehen. Solange Sie die Menütaste der Maus nicht drücken, wird dort nämlich die Größe des noch verfügbaren RAM-Speichers in Bytes angegeben (vor *free memory*). Sinkt *free memory* unter etwa 150 000 Byte, können Sie nicht mehr besonders viel mit Ihrem Amiga machen. Die meisten Anwendungsprogramme laufen dann nicht mehr einwandfrei. Große, komplizierte Programme können noch viel mehr Speicher benötigen und unter Umständen schon bei 300 000 oder mehr *free memory* die Arbeitsaufnahme verweigern.

### 3.5.6 Die »Tastatur-Maus«

Sollten Sie einmal in die Verlegenheit kommen, Ihre Maus nicht gebrauchen zu können oder gebrauchen zu wollen, und müssen trotzdem den Mauszeiger bewegen, können Sie das übrigens auch mit der Tastatur machen. Daß diese Möglichkeit bislang noch nicht erwähnt wurde, liegt einfach daran, daß diese Art der Mausbewegung sehr unpraktisch und mühselig ist.

Zur Bewegung des Mauszeigers benötigen Sie die beiden Amiga-Tasten ([*A*] und [*A*]) links und rechts neben der Leertaste sowie die vier Pfeil-Tasten rechts auf Ihrem Amiga 500. Halten Sie einfach eine dieser beiden Tasten fest und drücken auf eine Pfeil-Taste. Das hat denselben Effekt, als hätten Sie die Maus auf ihrer Unterlage in die Richtung bewegt, in die der Pfeil zeigt.

Tippen Sie einmal auf eine Pfeil-Taste, und die Maus bewegt sich einen Punkt in die angezeigte Richtung. Wenn Sie die Pfeil-Taste längere Zeit festhalten, so bewegt sich die Maus mit wachsender Geschwindigkeit in die angegebene Richtung. Wenn Sie den Mauszeiger in größeren Schritten über den Bildschirm springen lassen möchten, müssen Sie gleichzeitig mit der Amiga-Taste und der entsprechenden Pfeil-Taste auch noch eine der beiden Shift-Tasten festhalten. Die dann zustande kommenden Sprünge sind allerdings recht groß.

Aber nicht nur die Mausbewegung läßt sich über die Tastatur simulieren. Auch die Maustasten können über die Tastatur simuliert werden. Dazu drücken Sie, nachdem Sie den Mauszeiger zur gewünschten Stelle bewegt haben, gleichzeitig auf eine Amiga-Taste und auf eine der beiden [Alt]-Tasten. Dabei entspricht die linke [Alt]-Taste der linken (Selektions-) Taste und die rechte [Alt]-Taste der rechten (Menü-) Taste auf der Maus.

Die Maus auf diese Art und Weise zu simulieren, ist sehr mühselig und verlangt einige Übung. Es ist im Grunde nur dann sinnvoll, wenn Ihre Maus einmal defekt sein sollte und Sie dringende Arbeiten zu erledigen haben, bevor sie repariert werden kann.

### 3.5.7 Ein völlig verdecktes Fenster nach vorne holen

Zum Abschluß dieses Kapitels noch ein Tip, der ganz hilfreich sein kann, wenn Sie die Möglichkeit, Schubladen zu bilden, sehr intensiv nutzen. Wenn allzu viele Fenster auf Ihrer Workbench liegen und Sie ein bestimmtes nicht mehr finden können, müssen Sie vielleicht alle anderen Fenster schließen oder mit dem Back-Gadget nach hinten legen, um das gesuchte zu finden. Sie können das gesuchte Fenster aber auch sehr leicht zum Vorschein bringen, wenn Sie zufällig die Schublade sehen können, aus der das Fenster hervorgegangen ist. Öffnen Sie einfach die Schublade noch einmal mit einem Doppelklick auf das Piktogramm, und das entsprechende Fenster erscheint oben auf dem Stapel. Ein kleiner Trick, mit dem man sich manchmal viel Arbeit erspart. (Wie Sie sehen, hat die elektronische Werkbank in diesem Fall deutliche Vorteile gegenüber einer wirklichen. Bei einer echten Werkbank können Sie eine schon offene Schublade natürlich nicht noch einmal öffnen.)

## 3.6 Die Menübefehle der Workbench

Im folgenden finden Sie eine Liste aller Menübefehle, die Sie auf der Workbench aus den Menüs wählen können. Jeder Befehl wird zunächst auf englisch aufgelistet, dann eine ungefähre deutsche Übersetzung oder Umschreibung. Die Menü-Namen (fettgedruckt) sind den einzelnen Befehlen (kursiv) jeweils vorangestellt.

**Workbench**

| | |
|---|---|
| *Open* | Öffnet ein Objekt; zum Beispiel eine Schublade. |
| *Close* | Schließt ein Disketten- oder Schubladenfenster. |
| *Duplicate* | Dupliziert ein beliebiges Objekt. |
| *Rename* | Zum Umbenennen eines Objektes. |
| *Info* | Zeigt das Info-Fenster eines Objektes. |
| *Discard* | Löscht ein Objekt unwiderruflich. |

**Disk**

| | |
|---|---|
| *Empty Trash* | Löscht unwiderruflich alle Objekte im Mülleimer. |
| *Initialize* | Bereitet eine neue Diskette zur Datenaufnahme vor und/oder löscht eine alte Diskette. |

**Special**

| | |
|---|---|
| *Clean Up* | Ordnet die Objekte in einer Diskette oder Schublade. |
| *Last Error* | Zeigt die letzte Fehlermeldung noch einmal. |
| *Redraw* | Neuzeichnen der Workbench nach Programmfehlern. |
| *Snapshot* | Speichert die aktuelle Position eines Piktogramms und die Größe und Position seines Fensters auf Diskette ab. |
| *Version* | Zeigt die Versionsnummer der Workbench-Software. |

# 4 Zusammenhänge: Werkzeuge und Projekte

Wenn Sie mit Ihrem Amiga wirklich etwas vollbringen wollen, so benötigen Sie dazu ein Programm. Dies ist bei keinem Computer anders und sollte inzwischen bereits ein recht vertrautes Prinzip für Sie sein. Denn auch bisher (in den ersten drei Kapiteln dieses Buches) haben wir uns ja bereits mit einem Programm beschäftigt, der Workbench. Die Workbench aber ist ein sogenanntes »Systemprogramm«, das von den meisten Anwendern als selbstverständliche Einrichtung mehr oder weniger übersehen wird. Spricht man von einem Programm, so meint man damit meist ein sogenanntes Anwendungsprogramm, wie zum Beispiel eine Textverarbeitung oder ein Buchhaltungsprogramm. Erst diese Anwendungsprogramme machen einen Computer zu einem wirklichen Hilfsmittel; ohne sie bleibt er ein teurer Türstopper.

## 4.1 Werkzeuge ohne Projekte: Uhr und Rechner

Auf der Workbench des Amiga heißen Anwendungsprogramme »Werkzeuge«, und genau das sollten gute Programme auch sein. Kein Programm leistet wirklich »von selbst« etwas. Es kann Ihnen aber, wie ein gutes Werkzeug, dabei helfen, eine Aufgabe zu erledigen, beziehungsweise die Erledigung dieser Aufgabe erst zu ermöglichen. Sie werden nun zwei sehr einfache Werkzeuge kennenlernen.

### 4.1.1 Wie Sie ein Werkzeug starten

Im letzten Kapitel wurden die meisten Operationen mit Werkzeugen und Projekten bereits vorgestellt. Die eine Operation, die noch nicht erläutert wurde, war das Öffnen von Werkzeugen – alle anderen Objekte auf der Workbench kann man ja schließlich mit dem Befehl Open oder einem Doppelklick öffnen. Bereits bei diesen anderen Objekten galt immer die Grundregel, daß das Öffnen eines Objektes immer zeigt, »was sich darin verbirgt«. Um zu sehen, was sich in einem Programm verbirgt, muß man es natürlich starten – und genauso wirkt der Befehl Open auch, er startet das »geöffnete« Programm.

Um ein Werkzeug (Programm) zu starten, selektieren Sie es einfach mit einem Mausklick und wählen dann den Befehl *Open* aus dem Menü *Workbench.* Alternativ können Sie dasselbe auch durch einen Doppelklick mit der Maus auf das Piktogramm des Werkzeugs erreichen. Probieren Sie es aus!

Öffnen Sie dazu die Schublade *Utilities* auf der Workbench-Diskette. In dem Fenster, das sich daraufhin öffnet, sehen Sie ein Piktogramm, das wie eine Uhr aussieht und auch den vielsagenden Namen *Clock* (engl. für Uhr) trägt. Dies ist ein Werkzeug. Starten Sie es mit einem Doppelklick auf dieses Piktogramm. Das Diskettenlaufwerk beginnt zu arbeiten; die Workbench liest das Programm von der Diskette in den RAM-Speicher des Amiga und startet es dann.

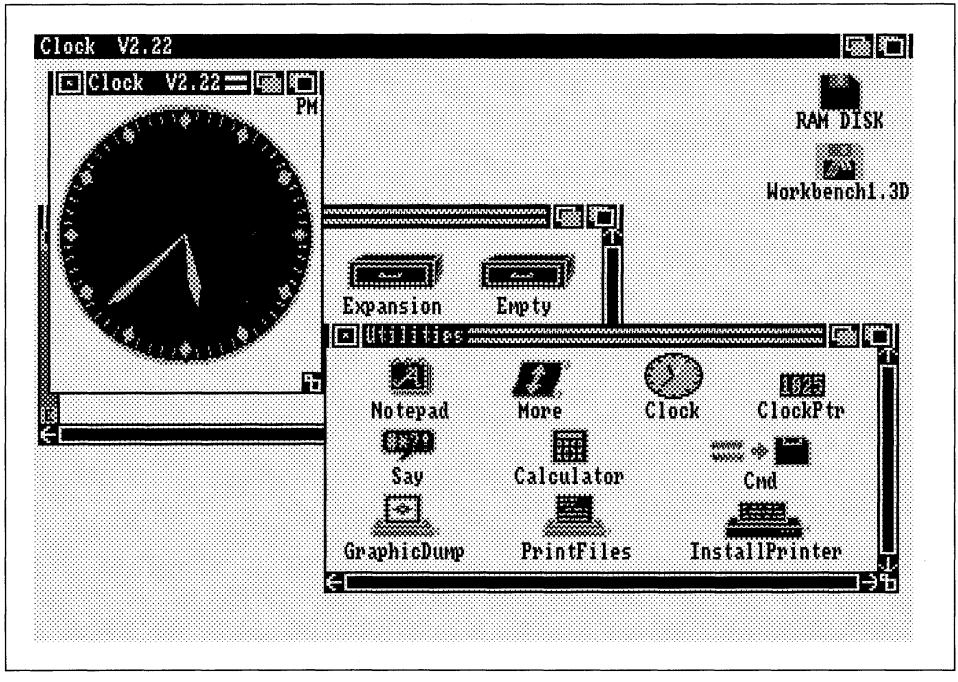



***Bild 4.1:*** *Die Uhr des Amiga*

Nach wenigen Sekunden erscheint ein neues Fenster links oben auf der Werkbank. Es trägt den Titel *Clock V2.2* und zeigt in seinem Innern ein wirklichkeitsgetreues Bild einer (analogen) Uhr. Uhrzeit und Datum stimmen vielleicht nicht, ansonsten geht die Uhr aber genau; das heißt, der Minutenzeiger bewegt sich jede Minute um einen Teilstrich weiter. Der Amiga 500 hat standardmäßig zwar eine ziemlich genau gehende Uhr eingebaut, aber keine Batterie, die dafür sorgt, daß diese Uhr auch dann noch weiterläuft, wenn Sie das Gerät abgeschaltet haben. Nur wenn Sie eine Amiga 501-Speichererweiterungskarte eingebaut haben, die auch eine batteriegepufferte Uhr enthält, stimmt die Uhrzeit, nachdem Sie sie einmal richtig eingestellt haben, immer. Falls Sie diese Speichererweiterungskarte nicht besitzen, ist es nach jedem Neustart nötig (und auch ratsam) die Uhr nachzustellen. Sie können dies mittels des Werkzeugs *Preferences* machen, das weiter unten noch erläutert wird.

Sie können die Uhr einfach in der Ecke des Bildschirms liegen lassen und mit Ihrer Arbeit auf der Workbench fortfahren. Die Uhr läuft dabei weiter (wie es sich für eine ordentliche Uhr gehört), und zeigt Ihnen weiter die (eventuell falsche) Zeit exakt an.

### 4.1.2 Das Fenster eines Werkzeugs

Sobald Sie das Fenster der Uhr anklicken, zum Beispiel um es zu bewegen, teilen Sie dem Programm *Clock* mit, daß Sie damit arbeiten beziehungsweise direkt mit ihm kommunizieren wollen. Dies wird durch zwei Änderungen auch am Bildschirm verdeutlicht. Zum einen wird das Fenster aktiv: der Titelstreifen wird besser lesbar (er enthält die gerasterte, schlecht lesbare »Geisterschrift«, solange das Fenster inaktiv ist). Zum anderen ändert sich die Titelleiste des Bildschirms. Statt *Workbench 1.3* und der Angabe des freien Speicherplatzes steht dort nun auch *Clock V2.2*.

Sobald Sie nämlich ein Werkzeug öffnen, wird es sehr wichtig, ob ein Fenster aktiv ist oder nicht. Solange das Fenster eines Programms nämlich inaktiv ist, kann das Programm zwar tätig sein und auch Ausgaben in seinem Fenster zeigen. Aber erst wenn das Fenster aktiv wird, können Sie Einfluß auf das Programm nehmen – zum Beispiel, ihm Befehle erteilen.

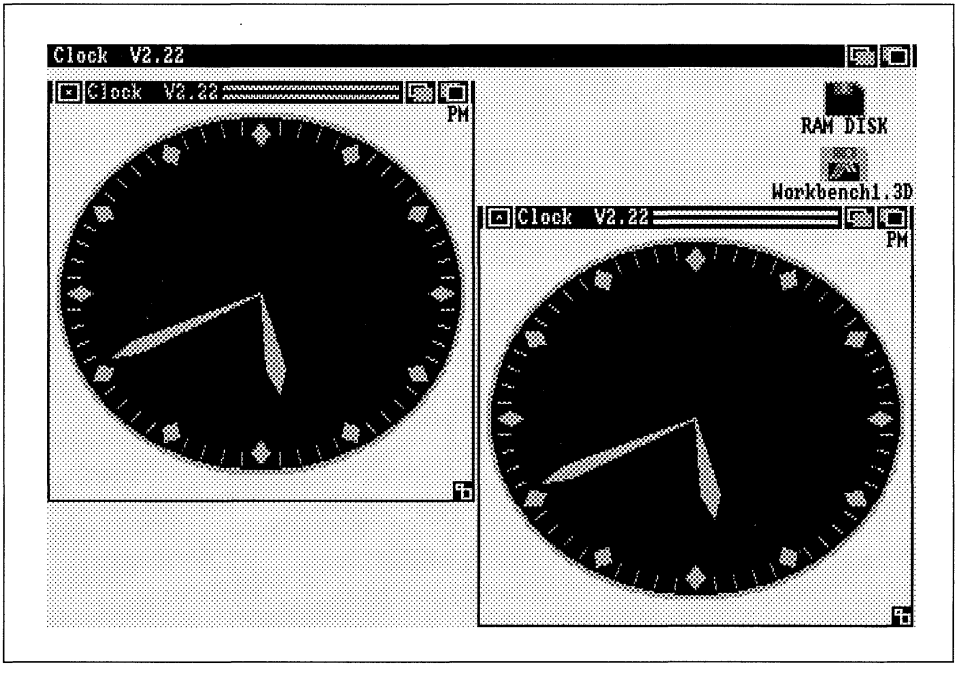



***Bild 4.2:*** *Verschiedene Fensterformen*

Da Sie sich nicht zweiteilen können, können Sie ja immer nur mit einem Programm zur selben Zeit kommunizieren. Nur dieses eine Programm erfährt von Ihren Aktionen (zum Beispiel Mausklicks und Tastendrücken). Indem Sie auf das Fenster eines bestimmten

Programms klicken (das Fenster also aktiv machen), schenken Sie diesem Programm Ihre Aufmerksamkeit. Das Programm, das vorher aktiv war, geht dann in eine Art Wartezustand über, in dem es ohne Ihre Eingaben auskommen muß. Das neue Programm empfängt nun Ihre Eingaben und reagiert darauf. Wie es darauf reagiert, entscheidet es allein. Deshalb kann auch ein Mausklick oder eine Bewegung der Maus in einem Fenster etwas völlig anderes bewirken als in einem anderen Fenster.

Die Systemsoftware des Amiga merkt sich zu jedem Fenster, das am Bildschirm sichtbar ist, zu welchem Programm es gehört. So kann sie jederzeit entscheiden, welches Programm die »Nachrichten«, die vom Anwender kommen, empfangen und verarbeiten soll.

Experimentieren Sie doch ein wenig mit dem Uhrfenster! Sie werden sehen, daß es sich ganz ähnlich verhält wie die Fenster der Workbench, die wir in den vorangegangenen Kapiteln schon kennengelernt haben. Das Uhrfenster enthält zum Beispiel ein Größen-Gadget, Back- und Front-Gadgets und ein Schließ-Gadget, mit dem Sie das Fenster schließen und damit das Programm (Werkzeug) *Clock* wieder beenden können. Versuchen Sie vor allem einmal das Größen-Gadget. Die Uhr paßt sich jeder neuen Fenstergröße an. Wenn Sie wollen, können Sie eine Uhr in Bildschirmgröße haben. Wird das Fenster rechteckig, so wird das Ziffernblatt entsprechend oval.

### 4.1.3 Die Menüleiste eines Werkzeugs

Wie bereits erwähnt wurde, wechselt jedesmal, wenn man auf das Fenster eines anderen Programms klickt, der Inhalt der Titelleiste des Bildschirms. Diese Titelleiste enthält aber nicht nur einen kurzen Namen wie *Workbench* oder *Clock*, sondern »darunter« auch die Menüs, die Ihnen aus den letzten Kapiteln ja schon bekannt sind. Auch diese Menüs werden jedesmal komplett ausgetauscht, wenn ein anderes Programm aktiv wird, weil Sie sein Fenster anklicken. Drücken Sie die Menütaste, während das Fenster der Uhr aktiv ist, und Sie werden fünf neue Menütitel sehen: *Type*, *Mode*, *Seconds*, *Date* und *Alarm*.

Im Menü *Type* stehen Ihnen drei Befehle zur Auswahl: *Digital1*, *Digital2* und *Analog*. Der Befehl *Analog* ist zusätzlich noch mit einem »Häkchen« versehen (»abgehakt«), das Ihnen sagt, daß die Uhr im Moment auf Analoganzeige geschaltet ist (was Sie natürlich auch ohne den Haken sehen würden). Dadurch lernen Sie auch gleich einen weiteren Aspekt von Menüs kennen: gibt es nämlich in einem Menü einen Befehl, der quasi ein Schalter ist, so kann die Stellung des Schalters durch ein Häkchen angezeigt werden. Ist der Schalter an (Analoganzeige sichtbar), wird der Befehl mit einem Haken versehen (abgehakt), ansonsten nicht. Wenn Sie den Befehl *Digital1* im *Type*-Menü wählen, wechselt die Anzeige der Uhrzeit von Zeigern nach Ziffern. Danach ist *Digital1* mit einem Häkchen versehen und *Analog* nicht mehr. Die Digitalanzeige wird allerdings in einem Fenster von fester Größe angezeigt, so daß es beim Wechsel von einer Anzeigeart zur anderen meist auch zu einem Sprung in der Fenstergröße kommt, der etwas überraschend sein kann.

Der letzte Befehl im *Type*-Menü, *Digital2*, zeigt eine andere, sehr kompakte Form der Digitalanzeige. Während die Digital-Uhr von *Digital1* noch zweizeilig ist und gleichzeitig Uhrzeit und Datum anzeigen kann (siehe unten), besteht sie bei *Digital2* nur noch aus der Titelleiste eines Fensters.

Im Menü *Mode* können Sie zwischen 24-Stunden-Anzeige und 12-Stunden-Anzeige für die Uhr wählen, was natürlich nur für die beiden Digitalanzeigen von Bedeutung ist. Die

Bedienung dieses Menüs funktioniert genauso wie bei *Type*. Der jeweils zuletzt gewählte – also gültige – Befehl wird mit einem Haken versehen. Genauso können Sie im Menü *Seconds* mit *Seconds On* den Sekundenzeiger beziehungsweise die Sekundenanzeige der Digitaluhr an- und mit *Seconds Off* wieder abschalten. Und im Menü *Date* schließlich können Sie in entsprechender Weise die Datumsanzeige an- und abschalten (mit *Date On* beziehungsweise *Date Off*). Probieren Sie alle diese Befehle am besten gleich einmal aus.

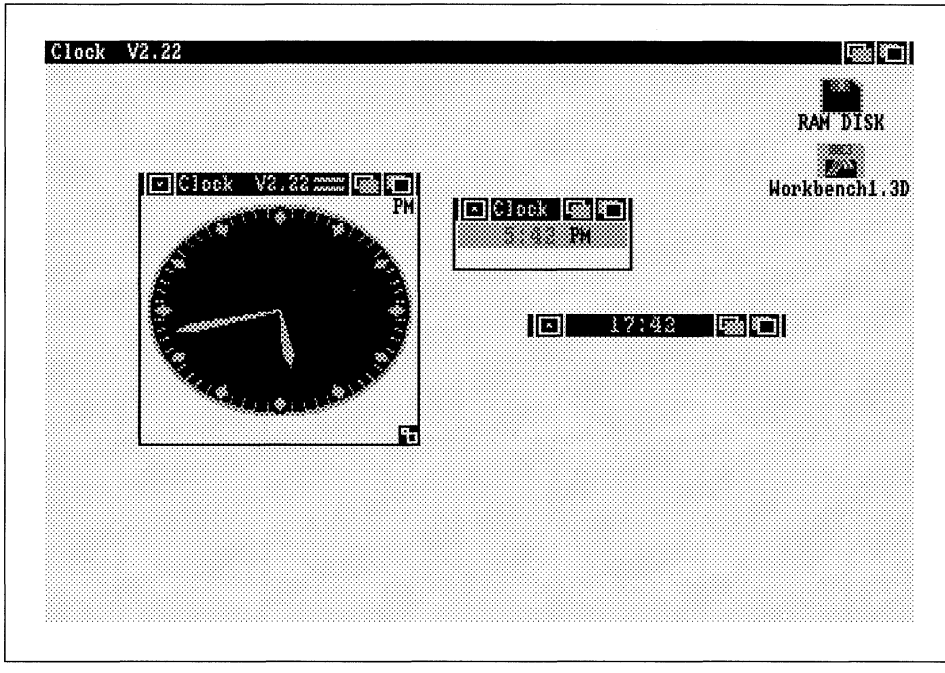



***Bild 4.3:*** *Die verschiedenen Anzeigeformate der Uhr des Amiga*

Im letzten Menü *Alarm* können Sie die Uhr des Amiga schließlich auch noch als Wecker einsetzen. Unten in diesem Menü gibt es die beiden Punkte *Alarm On* und *Alarm Off*, von denen wieder der jeweils gültige mit einem Häkchen versehen ist. Wie sich das für einen richtigen Wecker gehört, klingelt auch der Wecker des Amiga auf keinen Fall, solange *Alarm Off* gültig (abgehakt) ist. Nur wenn *Alarm On* abgehakt ist, kann der Wecker klingeln. Und mit dem ersten Befehl *Set* in diesem Menü können Sie die Alarm- beziehungsweise Weckzeit einstellen. Rufen Sie diesen Befehl auf und es erscheint ein Fenster mitten auf dem Bildschirm. Dies ist ein Dialogfenster oder Requester, ein Fenster, das eine Eingabe beziehungsweise Reaktion von Ihnen verlangt. Sie haben diese Fenstersorte ja schon beim Kopieren von Disketten kennengelernt.

In diesem Fenster sehen Sie unter anderem die Uhrzeit *12:00 Uhr* in digitaler Form. Klicken Sie auf die angezeigten Stunden (also die Zahl 12). Jetzt können Sie mit den danebenliegenden Pfeilsymbolen die Stunden vor- und zurücklaufen lassen. Wenn Sie auf den nach oben weisenden Pfeil klicken, werden die Zahlen größer. Wenn Sie auf den nach unten weisenden Pfeil klicken, werden die Zahlen kleiner. Um die Minuten der Alarmzeit einzustellen,

müssen Sie genauso vorgehen. Klicken Sie zunächst auf die Minutenanzeige (also die Zahl 00) und stellen Sie dann die gewünschte Minutenzahl mit den beiden Pfeilen ein.

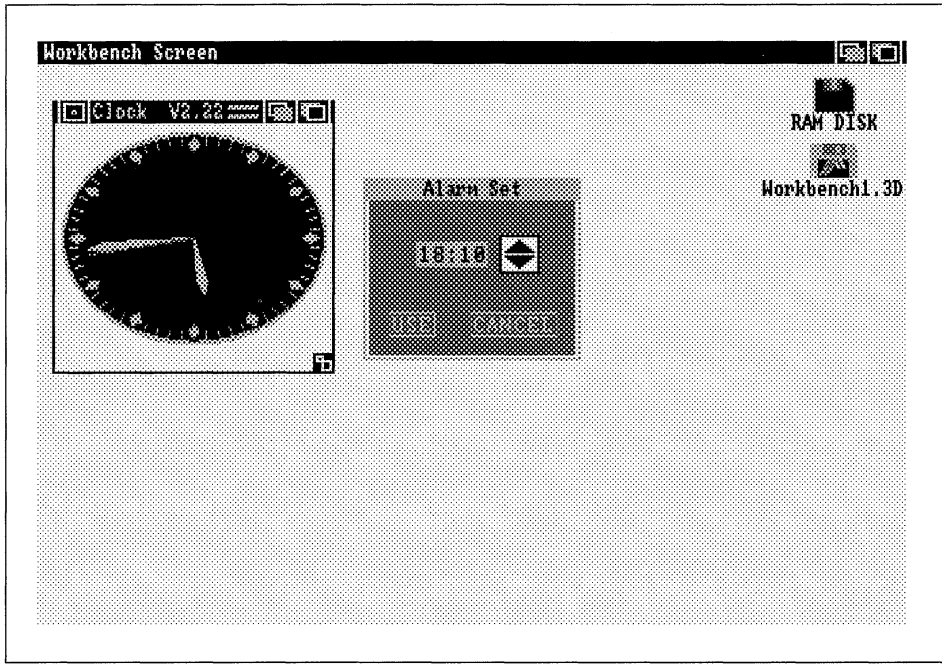



***Bild 4.4:*** *Einstellen der Alarmzeit*

Um das Fenster wieder verschwinden zu lassen, müssen Sie auf einen der Knöpfe *USE* (engl. für *Verwenden*) oder *CANCEL* (engl. für *Abbrechen*) klicken. Nur wenn Sie auf *USE* klicken, wird die eingestellte Weckzeit gültig, bei *CANCEL* ignoriert der Amiga die zuvor gemachten Eingaben. Beachten Sie, daß Sie auf diese Art und Weise lediglich die Alarmzeit einstellen aber nicht auch den Alarm automatisch aktivieren. Sie werden natürlich nur dann zur eingestellten Zeit mit einem Piepston »geweckt«, wenn *Alarm On* im Menü *Alarm* eingestellt ist.

### 4.1.4 Der Rechner

Die Amiga-Uhr ist nicht gerade ein überwältigendes, aber doch ein ganz nettes kleines Programm, für das Sie gelegentlich sicher Verwendung finden werden. Sie werden nun ein ähnlich einfaches Werkzeug kennenlernen, das Ihnen manchmal vielleicht eine echte Hilfe sein kann: den Taschenrechner.

Öffnen Sie dazu – falls nötig – die Schublade *Utilities* auf der Workbench-Diskette und suchen Sie das Piktogramm *Calculator* (engl. für Rechner). Machen Sie einen Doppelklick auf dieses Piktogramm und der eingebaute Taschenrechner des Amiga erscheint am Bildschirm.

Es handelt sich dabei um einen sehr simplen Taschenrechner, der nur die vier Grundrechenarten beherrscht und nicht einmal über einen Speicher verfügt. Dafür können Sie ihn sowohl mit der Maus als auch über die Tastatur bedienen. Die »Tasten« des Taschenrechners sind Gadgets, die Sie mit einem Mausklick betätigen können. Wollen Sie 2 * 3 berechnen, können Sie zum Beispiel die Taste [2], dann [*], dann [3] und schließlich die [=]-Taste anklicken. Wenn Ihnen das mit der Maus zu langsam geht, können Sie aber auch die Formel 2 * 3 auf der Tastatur eingeben und dann die [=]-Taste, die [Enter]-Taste oder die [Return]-Taste auf der Tastatur drücken. Wenn Sie dabei den Zehnerblock auf der rechten Seite der Tastatur benutzen, »fühlt« sich das fast wie das Arbeiten mit einem richtigen Taschenrechner an.

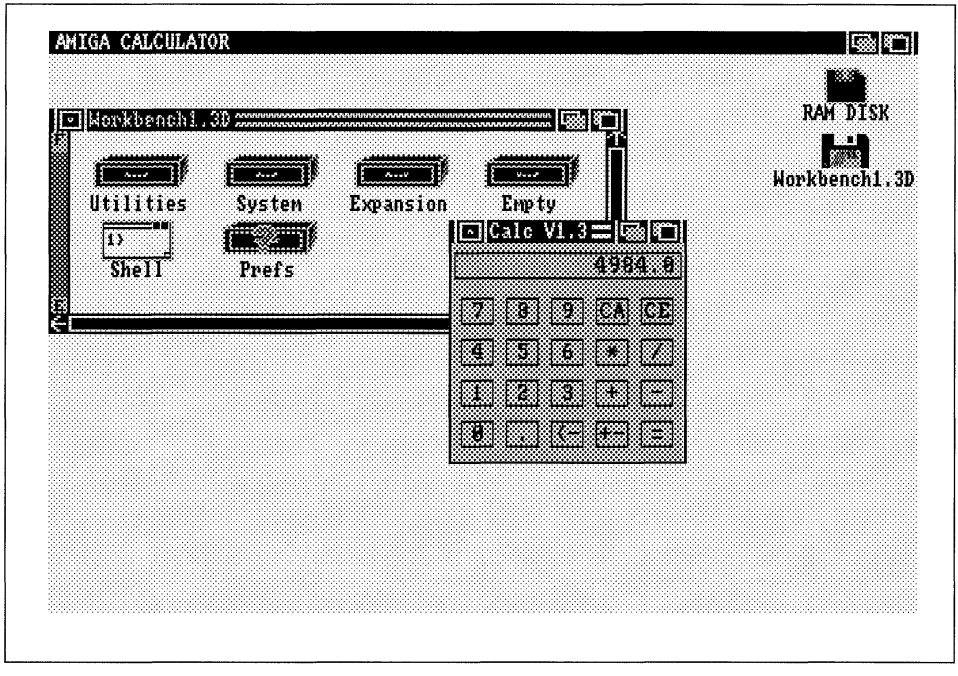



***Bild 4.5:*** *Der Taschenrechner des Amiga*

Abgesehen von der Möglichkeit, sowohl die Tastatur als auch die Maus zur Bedienung zu benutzen, verhält sich der Taschenrechner wie ein normaler Taschenrechner auf Ihrem Schreibtisch. Auch die restlichen »Tasten« sollten Ihnen bereits vertraut sein. [CA] löscht sämtliche Eingaben. [CE] löscht nur die zuletzt eingegebene Zahl, aber nicht ein vorher bereits errechnetes Zwischenergebnis. Und mit [<-] löschen Sie die zuletzt eingegebene Ziffer. (Diese Taste wirkt also wie die [Backspace]-Taste bei der Texteingabe.) Fast alle »Tasten« des Taschenrechners haben ein gleichbedeutendes Pendant auf der Tastatur. Die Taste [<-] im Rechner-Fenster entspricht zum Beispiel der [Backspace]-Taste. Für [CA] und [CE] müssen Sie zwei Tasten auf der Tastatur unmittelbar hintereinander drücken: erst [C] und dann [A] beziehungsweise [E].

Wenn Sie den Rechner nicht mehr benötigen, brauchen Sie nur sein Fenster zu schließen. Klicken Sie dazu in das Schließ-Gadget in der oberen linken Ecke des Fensters.

## 4.2 Ein Werkzeug für Projekte: Notepad

Die beiden Werkzeuge (Programme), die Sie bisher kennengelernt haben, erzeugen oder bearbeiten keine Dokumente beziehungsweise Dateien. (Welches Projekt wollte man auch mit dem »Werkzeug Uhr« bearbeiten?) In diesem Abschnitt werden Sie nun ein Werkzeug kennenlernen, das Dateien erzeugen und bearbeiten kann. Es handelt sich um das Werkzeug *Notepad* (engl. für *Notizblock*), ein kleines Textverarbeitungsprogramm. Sie verfügen mit *Notepad* zwar nicht über eine besonders leistungsfähige Textverarbeitung, können damit aber recht komfortabel viele kurze Texte (aber sicherlich keinen Roman) bearbeiten. Das ist für das Anlegen kleiner Notizen und zum Herumspielen mit den Möglichkeiten der Textverarbeitung auf dem Amiga aber völlig ausreichend.

Ein mit *Notepad* geschriebener Text ist ein »Projekt« von *Notepad.* Alle Dateien, die von einem Programm (Werkzeug) erzeugt werden, heißen auf der Workbench »Projekte«. In der Ihnen inzwischen ja schon bekannten Werkbank-Analogie kann man sich das so vorstellen, daß mit einem bestimmten Werkzeug an einem bestimmten (oder mehreren) Projekt(en) gearbeitet wird. Diese Verbindung zwischen Projekten und Werkzeugen wird meist auch durch die Piktogramme der entsprechenden Workbench-Objekte verdeutlicht. Das Piktogramm eines Projekts, das zu einem bestimmten Werkzeug gehört, sieht diesem Werkzeug meist ähnlich, wie Sie gleich erleben werden.

### 4.2.1 Wie Sie den Notizblock öffnen

Um an den Notizblock »heranzukommen«, müssen Sie zunächst wieder die Schublade *Utilities* auf der Workbench-Diskette öffnen. In dem Fenster, das sich daraufhin öffnet, sehen Sie dann das Piktogramm *Notepad*, hinter dem sich der Notizblock »verbirgt«. Sie können den Notizblock nun genauso einfach öffnen wie die Uhr. Dazu reicht ein Doppelklick mit der Maus auf das Piktogramm. Das Diskettenlaufwerk rotiert nun aber schon merklich länger als beim Öffnen der Uhr oder des Taschenrechners, bis endlich ein neues Fenster mit dem Namen *Notepad V2.2* erscheint. Im Gegensatz zur Uhr wird dieses Fenster sofort aktiv und auch die Titelleiste des Bildschirms zeigt gleich durch den neuen Inhalt *Notepad* an, daß dieses Programm nun Ihre Eingaben und/oder Befehle erwartet. Sie werden nun zunächst erfahren, wie Sie einen Text in diesen Notizblock eintragen und ändern können.

### 4.2.2 Wie Sie Text in den Notizblock eintragen

Nach dem Öffnen von *Notepad* wird Ihnen zunächst ein leeres weißes Blatt gezeigt, dessen Größe Sie mit Hilfe des Größen-Gadgets jederzeit ändern können. Oben links auf diesem weißen Blatt sehen Sie einen dünnen schwarzen Strich. Dieser Strich ist die Schreibmarke; oder auf Neudeutsch: der »Cursor«, der die Stelle markiert, an der neu eingegebener Text erscheint. Drücken Sie nun ein paar Tasten, so erscheinen die entsprechenden Buchstaben immer links neben der Stelle, an der die Schreibmarke gerade steht. Die Schreibmarke selbst

wandert dabei nach rechts. Eine etwas anders aussehende, aber sich fast identisch verhaltende Schreibmarke haben Sie ja schon beim Aufruf des Befehls *Rename* kennengelernt. Die Schreibmarke von *Notepad* steht – im Gegensatz zu der beim *Rename*-Befehl oder in einem Infofenster – immer zwischen zwei Buchstaben. Neu eingetippter Text erscheint immer zwischen diesen beiden Buchstaben und schiebt den Text rechts davon zur Seite.

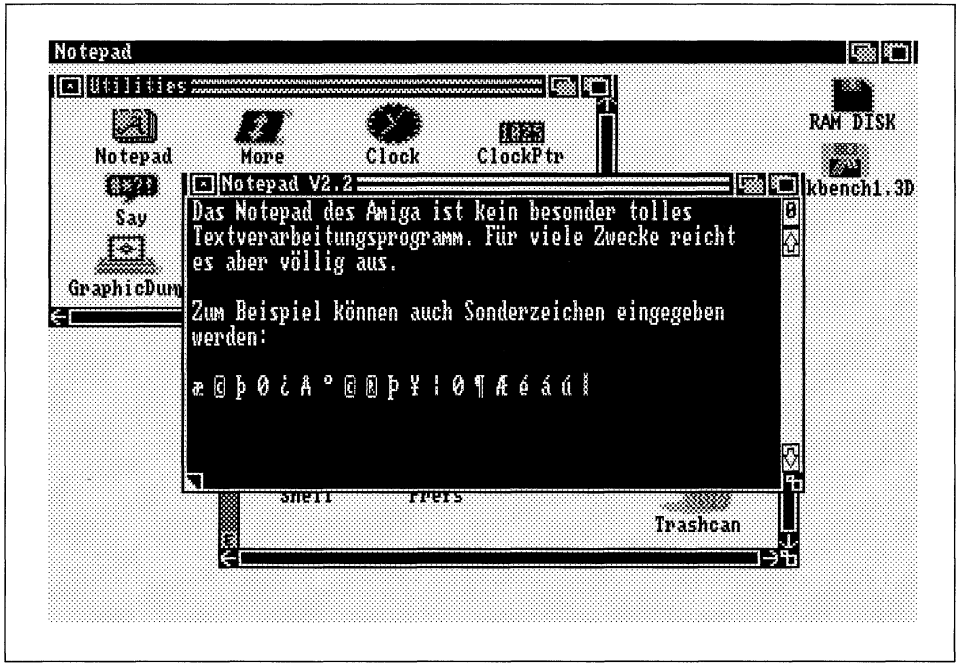



***Bild 4.6:*** *Eingabe eines Textes auf dem Notizblock des Amiga*

Wenn Sie sehen wollen, was der Amiga so alles an Buchstaben zu bieten hat, halten Sie einmal eine der beiden [Alt]-Tasten fest und tippen dann auf eine der »normalen« Buchstaben-Tasten – genauso wie Sie die [Shift]-Taste festhalten müssen, um Großbuchstaben zu bekommen. Sie können mit der [Alt]-Taste eine Vielzahl von speziellen Buchstaben und Sonderzeichen, wie ñ, œ, ç, µ und ∂, erzeugen, so daß Sie auch bei Briefen an Ihre französischen, spanischen oder norwegischen Freunde nie in die Verlegenheit kommen werden, einen Buchstaben der Landessprache nicht verwenden zu können. Wie es sich für einen benutzerfreundlichen Computer wie den Amiga gehört, brauchen Sie sich die dafür nötigen Tastenkombinationen natürlich auch nicht zu merken. Sie werden am Ende dieses Kapitels noch ein Werkzeug kennenlernen, das Ihnen jederzeit die komplette »Belegung« der Tastatur in übersichtlicher Form am Bildschirm zeigen kann.

Nun aber zurück zur Texteingabe und -bearbeitung: Die Schreibmarke beziehungsweise ihre Position ist für beide Aufgaben sehr wichtig. Sie wandert nicht nur »von selbst« weiter, während Sie Text eintippen oder löschen. Wenn Sie mit der Maus an die Stelle zwischen zwei Buchstaben klicken, so springt die Schreibmarke zu dieser Stelle. Text, den Sie danach

eintippen, erscheint dann dort. Während Sie gerade Ihre Hände auf der Tastatur haben, können Sie die Schreibmarke aber vielleicht noch bequemer mit den Pfeil-(Cursor-)Tasten bewegen. Jede dieser vier Tasten bewegt die Schreibmarke in die angegebene Richtung, »solange es geht«. Das heißt, Sie können nur so lange mit der [↓]-Taste nach unten gehen, bis das Ende des Textes erreicht ist, die [↑]-Taste »stößt« genauso in der ersten Zeile an.

### 4.2.3 Wie Sie kleine Fehler korrigieren

Machen Sie Fehler bei der Texteingabe, so können Sie diese mit der [Backspace]-Taste (oben rechts auf der Tastatur) wieder löschen. Die [Backspace]-Taste löscht immer den ersten Buchstaben links von der Schreibmarke. Die restlichen Buchstaben rechts davon rücken dabei nach links auf. Wenn Sie einen falschen Buchstaben eingeben und dies sofort bemerken, so brauchen Sie nur einmal auf [Backspace] zu drücken und der Fehler ist beseitigt. Wenn Sie den Fehler erst bemerken, wenn Sie schon weiteren Text eingegeben haben, müssen Sie die Schreibmarke bewegen. Klicken Sie mit der Maus hinter den fehlerhaften Buchstaben oder betätigen Sie die Cursortasten, bis die Schreibmarke hinter dem Fehler steht und drücken Sie dann [Backspace]. Durch mehrmalige Betätigung dieser Taste können Sie natürlich auch längere Textpassagen löschen.

Ähnlich wie die [Backspace]-Taste funktioniert die Taste [Del] (rechts neben [Backspace]). Sie löscht den Buchstaben rechts neben der Schreibmarke, sofern dort einer steht.

Probieren Sie jetzt am besten gleich einmal die Eingabe eines längeren Textes, um sich mit *Notepad* und den Korrekturfunktionen vertraut zu machen. Tippen Sie dazu einfach ein paar Sätze ein, die Ihnen gerade in den Sinn kommen – oder schreiben Sie einen Absatz aus diesem Buch ab.

### 4.2.4 Wie Sie im Text blättern

Wird das *Notepad*-Fenster voll, so können Sie auf ein neues Blatt überwechseln. Hierzu dienen die beiden Gadgets oben rechts und unten links im Fenster. Das Gadget unten links, das wie ein »Eselsohr« aussieht, blättert zur nächsten Seite, bis die letzte (zehnte) Seite erreicht ist. Das Gadget oben rechts, in dem eine Zahl steht, zeigt bei Betätigung (anklicken) die jeweils vorangehende Seite. In diesem Gadget wird auch immer die Nummer der aktuellen Seite angezeigt. Sie können sich also jederzeit vergewissern, auf welcher der zehn Seiten des Notizblocks (mit den Nummern 0 bis 9) Sie sich befinden.

Es ist auch möglich, mehr Text auf ein Blatt zu tippen, als ins Fenster paßt. Der Fensterinhalt wird dann nach oben verschoben und geht dabei auch nicht verloren. Sie können ihn jederzeit mit Hilfe des Rollbalkens am rechten Fensterrand wieder zum Vorschein bringen. Dieser funktioniert genauso wie der vertikale Rollbalken eines Schubladenfensters. Klicken Sie einmal auf den Pfeil am unteren Ende des Rollbalkens und der Fensterinhalt wird nach oben verschoben, wodurch eventuell am unteren Rand bislang verborgene Zeilen auftauchen. Klicken Sie einmal auf den Pfeil am oberen Ende des Rollbalkens und der Fensterinhalt wird nach unten verschoben, wodurch eventuell am oberen Rand verschwundene Zeilen wieder auftauchen. Wenn Ihnen dieses Hin- und Herschieben zu lästig ist, können Sie aber natürlich auch das Fenster des Notizblocks größer machen. Sie können dann einen größeren Teil Ihres Textes bearbeiten, ohne zur Maus greifen zu müssen.

Sie sollten nun schon wissen, wie Sie Texte eingeben, sich in einem größeren mehrseitigen Text zurechtfinden und kleine Fehler darin korrigieren können. Besonders die Hand-(beziehungsweise Maus-)Griffe für das Eingeben und Korrigieren von Texten sollten Sie sich unbedingt einprägen. Sie werden diese Fertigkeiten in fast allen anderen Programmen auch benötigen!

### 4.2.5 Wie Sie Textpassagen selektieren

Das Korrigieren größerer Fehler mit Hilfe der Tasten [Backspace] und [Del] ist recht umständlich. Deshalb gibt es die Befehle im *Edit*-Menü, mit denen Sie längere Textpassagen löschen, kopieren und innerhalb des Textes verschieben können. Um diese Befehle benutzen zu können, müssen Sie aber zunächst wissen, wie Sie eine Textpassage im Notizblock selektieren können. Das hat eine ähnliche Aufgabe wie die Selektion eines Piktogramms auf der Workbench: Sie teilen dem Programm dadurch mit, auf welchen Teil des Textes der nächste Befehl angewendet werden soll.

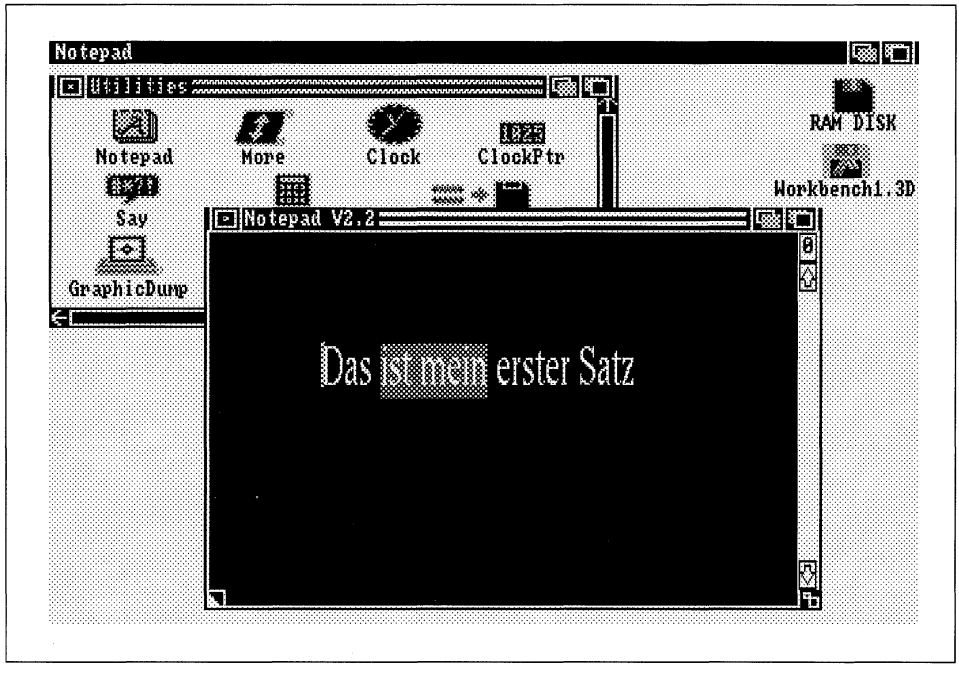



***Bild 4.7:*** *Hervorheben einer selektierten Textpassage im Notizblock*

Zum Selektieren müssen Sie die Schreibmarke zunächst vor den ersten Buchstaben der zu selektierenden Textpassage setzen und dann den Befehl *Mark Place* (markiere diese Stelle) aus dem Menü *Edit* wählen. Setzen Sie dann die Schreibmarke (mit der Maus oder den Cursortasten) an das Ende der Textpassage, die Sie selektieren wollen, und wählen Sie erneut *Mark Place* aus dem Menü *Edit*. Der so markierte Bereich wird nun »hervorgeho-

ben«, das heißt, er erhält eine andere Hintergrundfarbe. Normalerweise erscheint hervorgehobener Text im Notizblock schwarz auf orange statt schwarz auf weiß.

Probieren Sie es aus: Tippen Sie die Worte »Das ist mein erster Satz« ein und setzen Sie die Schreibmarke dann vor das Wort »ist«. Wählen Sie den Befehl *Mark Place* aus dem *Edit*-Menü und setzen Sie die Schreibmarke (mit den Cursortasten und mit der Maus) hinter das Wort »mein«. Wenn Sie nun noch einmal *Mark Place* aufrufen, werden die Worte «ist« und »mein« hervorgehoben.

Mark Place hat allerdings noch zwei Abkürzungen – es ist schließlich recht umständlich, zweimal die Schreibmarke zu versetzen und zwei Menübefehle auswählen zu müssen, um eine Textpassage zu markieren. Der Menübefehl *Mark Place* (und auch mancher andere Befehl) hat deshalb ein »Tastenäquivalent« oder eine »Tastenabkürzung«. Ein Tastenäquivalent ist eine Möglichkeit, durch das Drücken zweier bestimmter Tasten denselben Effekt zu erzielen wie durch die Auswahl eines Menübefehls. Wenn ein Menübefehl ein Tastenäquivalent besitzt, so wird dies auch im Menü sichtbar gemacht. Klappen Sie das *Edit*-Menü herunter und schauen Sie es sich näher an. Rechts neben allen Befehlen ist ein »A« auf dunklem Hintergrund und ein zweiter Buchstabe zu sehen. Das ist die Beschreibung eines Tastenäquivalents, das es für jeden Befehl im *Edit*-Menü gibt. Die Bedeutung dieser zwei Buchstaben ist ganz einfach zu verstehen: Neben dem Befehl *Mark Place* steht zum Beispiel »A M«. Das bedeutet, daß Sie die Amiga-Taste ([A]) rechts neben der Leertaste festhalten und dann die [M]-Taste drücken müssen, um diesen Befehl aufzurufen.




***Bild 4.8:*** *Tastenäquivalente in einem Menü*

Probieren Sie es aus: Verwenden Sie dazu wieder den Text »Das ist mein erster Satz«. Wählen Sie zunächst den Befehl *Cancel* aus dem *Edit*-Menü, wenn Sie bereits eine Textpassage hervorgehoben haben. Die Hervorhebung des selektierten Textes verschwindet dann. Setzen Sie dann die Schreibmarke vor das Wort »mein«. Drücken Sie nun auf die Amiga-Taste rechts neben der Leertaste und halten Sie sie fest. Betätigen Sie dann die Taste [M] und lassen Sie beide Tasten wieder los. Setzen Sie die Schreibmarke nun hinter das Wort »erster« und betätigen Sie noch einmal, wie eben beschrieben, die Tastenkombination [A]-[M]. Nun werden die beiden Worte »mein erster« hervorgehoben.

Wie Sie gerade gesehen haben, wirkt die Tastenkombination [A]-[M] also genauso wie die Auswahl von *Mark Place* aus dem *Edit*-Menü mit der Maus. Die dritte Möglichkeit, *Mark Place* zu »simulieren«, ist ein Doppelklick mit der Maus an eine bestimmte Stelle im Text.

Probieren Sie es aus: Wählen Sie zunächst wieder *Cancel* aus dem *Edit*-Menü, falls die Worte »mein erster« noch hervorgehoben sind. Bewegen Sie dann den Mauszeiger, so daß seine Spitze genau über oder etwas vor dem »m« des Wortes »mein« steht und drücken Sie schnell zweimal hintereinander auf die Selektionstaste (die linke Maustaste). Bewegen Sie dann die Mausspitze kurz hinter den Buchstaben »r« am Ende des Wortes »erster« und klicken Sie wieder zweimal auf die Selektionstaste. Die Worte »mein erster« werden nun wieder hervorgehoben.

Sie können den Bereich des selektierten Textes, die »Selektion«, aber mit *Mark Place* oder einem Doppelklick nicht nur festlegen, sondern auch erweitern oder verkleinern, nachdem Sie schon eine Textpassage selektiert haben. Setzen Sie dazu einfach die Schreibmarke vor den Anfang oder hinter das Ende der Selektion und wählen Sie wieder *Mark Place* aus dem *Edit*-Menü (oder machen Sie dort einen Doppelklick). Die Stelle, an der sich die Schreibmarke befand, wird dadurch zum neuen Anfang beziehungsweise Ende des selektierten Bereichs. Genauso können Sie mit einem Doppelklick (oder *Mark Place*) innerhalb des bereits hervorgehobenen Textbereichs die Selektion verkleinern.

Üben Sie nun ruhig ein wenig mit diesen Möglichkeiten. Geben Sie dazu ein paar längere Sätze ein. Sie werden dann bald »den Bogen raushaben«, wie Sie beliebige Textbereiche selektieren können. Am bequemsten ist dazu meist die »Methode Doppelklick«, wie Sie sicher bald merken werden.

### 4.2.6 Wie Sie Texte umstellen

Jetzt, da Sie wissen, wie Sie längere Textpassagen selektieren, können Sie Wörter, Absätze oder auch lange Textpassagen mit wenigen Schritten ändern oder umstellen. Wenn Sie zum Beispiel eine längere Textpassage löschen wollen, müssen Sie sie zunächst selektieren und dann den Befehl *Cut* (engl. *to cut* = ausschneiden) aus dem Menü *Edit* wählen. Die zuvor hervorgehobene Textpassage verschwindet daraufhin – allerdings nicht auf Nimmerwiedersehen. »Ausgeschnittener« Text landet in der sogenannten »Zwischenablage« (engl. *clipboard*). Dies ist ein kurzlebiger Zwischenspeicher, aus dem Sie versehentlich gelöschten Text wieder hervorholen können, den Sie aber auch dazu verwenden können, Texte umzustellen oder zu duplizieren.

Sobald Sie nämlich den Befehl *Paste* (engl. *to paste* = einkleben, einfügen) aus dem Menü *Edit* wählen, wird der Text, der sich in der Zwischenablage befindet, an der Stelle in den Text

eingefügt, an dem sich gerade die Schreibmarke befindet. Wenn Sie also eine Textpassage zu einer anderen Stelle im Text bewegen wollen, so müssen Sie diese Textpassage mit *Cut* ausschneiden, die Schreibmarke an die gewünschte andere Stelle setzen und *Paste* aufrufen. Der zuvor ausgeschnittene Text erscheint dann vor der Schreibmarke, als würde er dort Buchstabe für Buchstabe wieder eingetippt.

*Paste* leert aber die Zwischenablage nicht! Sie können also, indem Sie *Paste* zweimal hintereinander aufrufen, zwei identische Kopien der Textpassage aus der Zwischenablage in den Text einfügen. Zum Kopieren längerer Textpassagen wird normalerweise jedoch der Befehl *Copy* verwendet. Auch hierzu müssen Sie wieder den Bereich des Textes, den Sie kopieren wollen, hervorheben und dann *Copy* (engl. *to copy* = kopieren) aus dem Menü *Edit* wählen. Dadurch gelangt der markierte Text in die Zwischenablage (und ersetzt deren alten Inhalt), wird aber nicht aus der Seite gelöscht. Sie können diese Textpassage nun beliebig oft auf dieser oder auf den anderen Seiten des Notizblocks einfügen, indem Sie die Schreibmarke an die gewünschte Stelle setzen und *Paste* aufrufen.

### 4.2.7 Wie Sie Texte suchen und ersetzen

Neben diesen Möglichkeiten, Texte »von Hand« zu ändern, bietet *Notepad* aber – wie jedes Textverarbeitungsprogramm, das etwas auf sich hält, eine Suche-und-Ersetze-Funktion. Sie können damit nach einem Wort oder einem Wortstück suchen und es gegebenenfalls gleich durch eine andere Buchstabenfolge ersetzen lassen. Hierzu dienen die Befehle *Find* (engl. *to find* = finden), *Find Next* (= finde Nächstes), *Find Prev* (= finde Vorangehendes) und *Replace* (engl. *to replace* = ersetzen) im *Edit*-Menü. Mit dem Befehl *Find* legen Sie zunächst fest, wonach Sie suchen und wodurch Sie es gegebenenfalls ersetzen wollen, *Find Next* sucht das nächste Vorkommen des gesuchten Textes in Richtung auf das Textende und *Find Prev* sucht das nächste Vorkommen des gesuchten Textes in Richtung auf den Textanfang. Mit *Replace* schließlich kann man eine Textpassage, nachdem man sie gefunden hat, durch eine andere, die man zuvor mit *Find* festgelegt hat, ersetzen lassen.

Probieren Sie es aus: Geben Sie einen längeren Text ein, in dem mehrfach das Wort »Suche« vorkommt (der Text muß nicht unbedingt einen Sinn ergeben). Setzen Sie dann die Schreibmarke an den Anfang des Textes (oben links im Fenster) und wählen Sie *Find* aus dem *Edit*-Menü. Statt *Find* mit der Maus auszuwählen, können Sie natürlich auch sein Tastenäquivalent [Amiga]-[F] verwenden – wie das geht, wissen Sie ja jetzt. Daraufhin erscheint ein Dialogfenster am Bildschirm, in dem Sie nun die Angaben machen müssen, die *Notepad* für die Suche braucht.

Das Fenster enthält zwei weiß umrandete blaue Rahmen in der oberen, und drei Knöpfe, die man mit der Maus betätigen kann, in der unteren Hälfte. Diese Rahmen sind »Textgadgets«. Sie können einen Text darin eingeben oder – ganz ähnlich wie im *Notepad* – ändern. Ähnliche Textgadgets haben Sie bereits im Zusammenhang mit dem *Rename*-Befehl der Workbench und im Infofenster eines Piktogramms kennengelernt. In diesem Fall dienen die beiden Textgadgets dazu, einen Suchbegriff einzugeben und, sofern Sie das wünschen, einen Text, der diesen Suchbegriff ersetzen soll.

Klicken Sie dazu zunächst einmal mit der Maus in das Feld neben der Beschriftung *Find* und geben Sie das Wort »Suche« ein. Klicken Sie dann in das Textgadget neben der Beschriftung *Repl* (Abkürzung von *Replace*) und geben Sie »Ersetze« ein. Sie haben nun drei Möglich-

keiten, das Dialogfenster wieder zu schließen. Wenn Sie den Knopf *CANCEL* anklicken, verschwindet das Fenster einfach und nichts weiter geschieht. Klicken Sie jedoch *Next* an, so verschwindet das Fenster zwar auch. *Notepad* sucht aber zudem nach dem ersten Vorkommen des neben *Find* eingegebenen Begriffs. Wird ein solcher Begriff gefunden, wird die Schreibmarke danach direkt vor dem ersten Buchstaben dieses Begriffs gesetzt. Wenn Sie den Requester statt mit *Next* durch Anklicken von *Prev* (Abkürzung von *Previous* = vorangehend) schließen, sucht *Notepad* ebenfalls nach dem Suchbegriff, jedoch ausgehend von der augenblicklichen Position der Schreibmarke rückwärts in Richtung auf den Anfang des Textes.

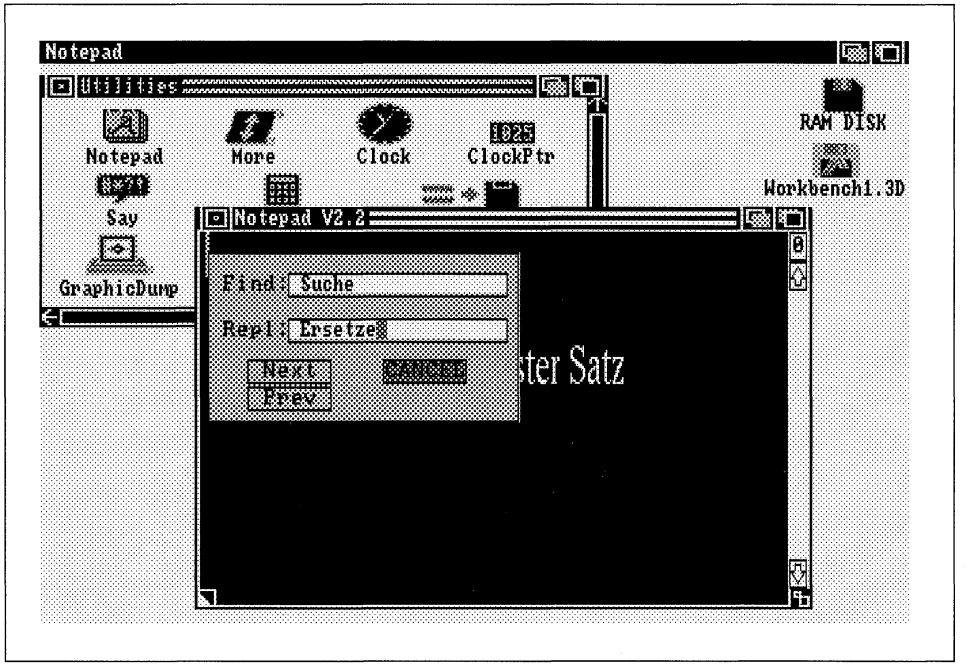



***Bild 4.9:*** *Der Find-Requester*

Klicken Sie nun auf *Next*. Die Schreibmarke müßte danach direkt vor dem ersten Vorkommen von »Suche« im Fenster des Notizblocks stehen. Wenn Sie nun den Befehl *Find Next* aus dem *Edit*-Menü wählen (oder dessen Tastenäquivalent drücken), springt die Schreibmarke an den Anfang des nächsten Vorkommens von »Suche«. Wählen Sie dann *Find Prev* (ious), springt die Schreibmarke zurück zum vorangehenden (also ersten) Vorkommen von »Suche«. So können Sie nacheinander sehr rasch alle Vorkommnisse eines bestimmten kurzen Textes auffinden und zwischen diesen Stellen hin- und herspringen. Steht die Schreibmarke nach Aufruf eines dieser Suchbefehle vor einem Vorkommen des Suchbegriffs im Text, können Sie diesen sofort gegen eine andere Buchstabenfolge austauschen.

Probieren Sie es aus: Wählen Sie *Find Next* oder *Find Prev* aus dem *Edit*-Menü, bis die Schreibmarke vor einem »Suche« steht. Rufen Sie nun den Befehl *Replace* aus dem *Edit*-Menü auf (oder drücken Sie [A]-[R]) und das Wort »Suche« verschwindet und statt dessen

taucht »Ersetze« auf. Der Suchbegriff, den Sie im oben beschriebenen Requester neben *Find* eingetragen haben, wird also durch die Buchstabenfolge ersetzt, die Sie neben *Repl* eingetragen haben, sobald Sie den Befehl *Replace* aufrufen. Beachten Sie aber bitte, daß *Replace* nur dann funktioniert, wenn Sie den Suchbegriff zuvor gesucht (und gefunden) haben. Wenn Sie die Schreibmarke mit der Maus vor »Suche« plazieren und dann *Replace* aufrufen, geschieht nichts.

### 4.2.8 Wie Sie Texte gestalten

Der Amiga-Notizblock (also das Programm *Notepad*) bietet wesentlich mehr Möglichkeiten, als nur Text einzugeben, zu löschen und umzustellen. Es steht auch eine Vielzahl verschiedener Methoden zur Wahl, wie Sie Texte gestalten oder auch bestimmte Textpassagen betonen können. Diese verbergen sich hinter den beiden Menüs *Font* und *Style*.




***Bild 4.10:*** *Ein Submenü des Font-Menüs*

Sie können zum Beispiel mit dem Menü *Font* (engl. *font* = Zeichensatz oder Schriftart) den Zeichensatz ändern, mit dem der von Ihnen eingegebene Text im Fenster erscheint. Dies wirkt, als würden Sie bei einer elektrischen Schreibmaschine den Kugelkopf oder das Typenrad tauschen. Dazu müssen Sie zunächst wissen, daß die Menüpunkte im Menü *Font* »Untermenüs« oder »Submenüs« sind. Die Namen von Untermenüs tauchen zwar in einem Menü auf, als wären sie Befehle. Es sind aber keine Befehle. Statt dessen bringen Sie ein neues Menü mit Befehlen darin zum Vorschein. In diesem Menü können Sie dann genauso

einen Befehl auswählen, wie Sie aus einem »normalen« Menü auswählen, das von der Menüleiste herunterklappt.

Probieren Sie es aus: Drücken Sie die Menütaste (die rechte Maustaste) und fahren Sie mit der Spitze des Mauszeigers über den Menütitel *Font*. Fahren Sie dann im Menü herunter, bis der Menüpunkt *Topaz* hervorgehoben wird. Rechts daneben taucht jetzt ein kleines Menü mit den Zahlen 8, 9 und 11 auf. Dies sind die Buchstabengrößen, in denen der Zeichensatz *Topaz* zur Verfügung steht. Sowohl der Menüpunkt *Topaz* als auch die Zahl *8* sind mit einem Haken versehen. Diese Art der Markierung von Menüpunkten kennen Sie ja bereits. Die beiden Häkchen besagen, daß im Augenblick der Zeichensatz *Topaz* in der Größe *8* verwendet wird. Fahren Sie nun im Fontmenü mit dem Mauszeiger auf und ab. Jedesmal, wenn ein anderer Zeichensatzname hervorgehoben wird, taucht ein anderes Submenü daneben auf. Einige Zeichensätze (zum Beispiel *Saphire*) können in bis zu vier Größen ausgegeben werden; andere gibt es nur in zwei Größen.




***Bild 4.11:*** *Text in der Schriftart Emerald*

Heben Sie nun den Namen *Emerald* hervor und gehen Sie mit dem Mauszeiger nach rechts. Heben Sie im Submenü die Zahl *20* hervor und lassen Sie dann die Menütaste los. Das Diskettenlaufwerk beginnt nun zu arbeiten (wie die Buchstaben eines Zeichensatzes aussehen, wird auf der Workbench-Diskette gespeichert) und danach ändert sich der Inhalt des *Notepad*-Fensters radikal. Der Text, den Sie zuvor eingegeben haben, erscheint jetzt in sehr großen, geschwungenen Lettern. Probieren Sie auch einmal andere Schriftarten und Schriftgrößen aus. Achten Sie aber darauf, daß es nicht ausreicht, nur den Namen der Schrift

im Hauptmenü hervorzuheben und dann die Menütaste loszulassen. Das bewirkt nichts. Sie müssen immer auch eine Schriftgröße im Submenü neben dem Schriftnamen hervorheben und dürfen erst dann die Menütaste loslassen – auch wenn Sie nur die Schriftart ändern wollen.

Im Menü *Style* (engl. *style* = in diesem Fall Buchstabenart, -stil oder Auszeichnung) können Sie wählen, ob der Text normal (*Plain*), kursiv (*Italic*), fett (*Bold*) oder unterstrichen (*Underline*) ausgegeben werden soll. Im Gegensatz zur Einstellung für die Schriftart (Zeichensatz), die immer für den ganzen Text gilt, können Sie auf einem Notizblatt durchaus Textpassagen haben, die einen unterschiedlichen Zeichenstil haben. Es können auch mehrere Stilmittel miteinander kombiniert werden. So sind Passagen möglich, die fett gedruckt sind, andere, die kursiv und wieder andere, die unterstrichen und kursiv gedruckt sind. (Im folgenden Abschnitt werden Sie sehen, daß es auch möglich ist, mehrere verschiedene Schriftarten auf einem Notizblatt zu verwenden.)

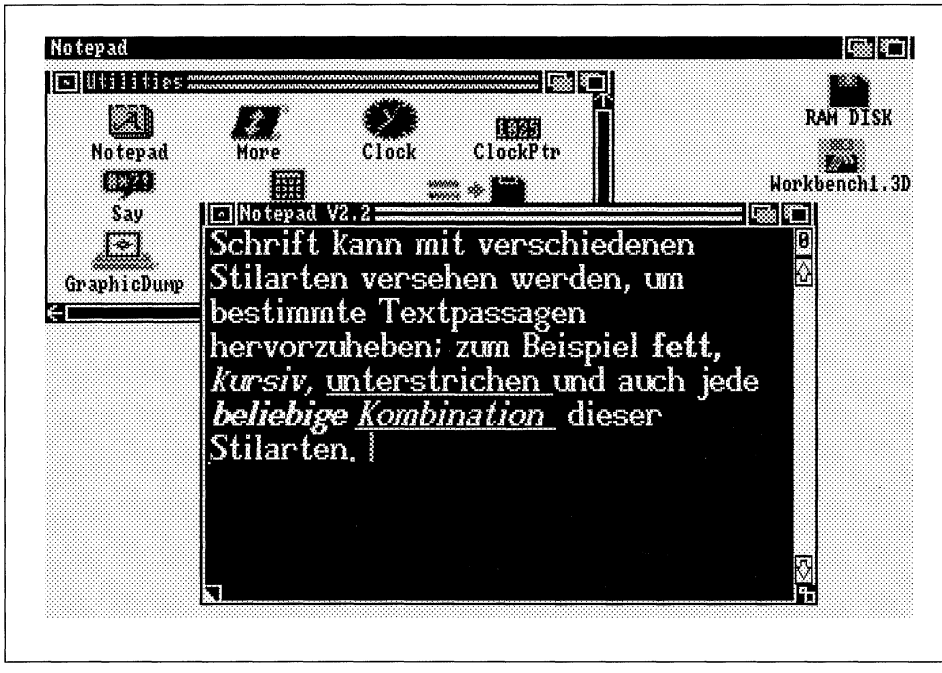



***Bild 4.12:*** *Text mit verschiedenen Stilarten*

Probieren Sie es aus: Geben Sie ein paar Zeilen ein, falls Sie das noch nicht getan haben, und setzen Sie die Schreibmarke mitten in den Text. Wählen Sie dann den Befehl *Bold* aus dem *Style*-Menü (oder drücken Sie [A]-[B]). Der gesamte Textbereich, von der Schreibmarkenposition bis zum Textende, erscheint dadurch fettgedruckt. Setzen Sie die Schreibmarke nun ein Stückchen weiter und wählen Sie *Italic* aus dem *Style*-Menü. Die Buchstaben bis zum Textende sind nun fett und kursiv, die Buchstaben zwischen der alten und der neuen Position der Schreibmarke werden hingegen nur fett angezeigt. Setzen Sie die Schreibmarke wieder ein Stück weiter und wählen Sie den Befehl *Plain* (engl. für einfach oder normal) aus dem

*Style*-Menü. Alle Buchstaben hinter der Schreibmarke bis zum Textende sehen nun wieder normal aus. Alle Befehle, die Sie aus dem *Style*-Menü wählen, gelten nämlich immer nur bis zu dem Punkt, an dem Sie den nächsten Stilwechsel befehlen!

### 4.2.9 Wie Sie den Notizblock nach Ihrem Geschmack gestalten

Das letzte Menü *Format* schließlich bestimmt, wie sich der Notizblock dem Betrachter darbietet. Sie können damit den Notizblock nach Ihrem Geschmack gestalten. Die ersten beiden Menüpunkte bestimmen, in welcher Farbe das »Papier«, also der Hintergrund des Fensters, angezeigt werden soll (*Paper Color*; zu deutsch Papier-Farbe) und in welcher Farbe die Buchstaben darauf erscheinen sollen (*Pen Color*; zu deutsch Stift-Farbe). Beide Menüpunkte bringen ein Submenü mit vier Farben zum Vorschein, von denen Sie jeweils eine auswählen müssen. (Wie Sie einen Punkt aus einem Submenü auswählen, wissen Sie ja schon aus dem vorangegangenen Abschnitt.) Die Grundeinstellung ist – wie könnte es auch anders sein – schwarzer Text auf weißem Hintergrund. Sie können aber auch weiße Schrift auf weißem Hintergrund wählen – und sehen dann nichts mehr.

Mit dem Menüpunkt *Word Wrap* legen Sie fest, was passiert, wenn Sie beim Tippen an das Zeilenende beziehungsweise an den rechten Fensterrand kommen. *Notepad* beginnt dann automatisch, ohne daß Sie auf die [Return]-Taste drücken, eine neue Zeile. Entweder kommt dann der folgende Buchstabe in die neue Zeile, wobei die neue Zeile auch mitten im Wort beginnen kann, oder das Wort, das Sie gerade schreiben, kommt komplett in die nächste Zeile. Ist *Word Wrap* mit einem Haken versehen, werden Zeilen nur zwischen den Wörtern getrennt – es gibt also keine Wörter, die in einer Zeile beginnen und in der nächsten fortgeführt werden. Hat *Word Wrap* jedoch keinen Haken, können neue Zeilen gegebenenfalls auch mitten im Wort beginnen. Das sieht meist häßlich aus und ist wirklich nur für ganz besondere Anwendungen geeignet.

Der nächste Menüpunkt *Global Font* (globale Schriftart) ist zunächst mit einem Haken versehen. Dies bewirkt, daß, wie oben beschrieben, jede Änderung der Schriftart über das Menü *Font* immer den gesamten Text betrifft. Wenn Sie jedoch einmal *Global Font* aus dem Menü *Format* auswählen, verschwindet der Haken davor und das *Font*-Menü verhält sich danach wie das *Style*-Menü. Das heißt, wenn Sie eine andere Schriftart aus dem *Font*-Menü wählen, so ändern nur die Buchstaben hinter der aktuellen Schreibmarke ihr Aussehen. Sie können dann also auch erreichen, daß nur ein einzelnes Wort in einer anderen Schriftart erscheint, indem Sie die Schreibmarke davorsetzen, die gewünschte Schriftart und -größe aus dem *Font*-Menü wählen, die Schreibmarke hinter das Wort setzen und eine andere Schriftart wählen.

Die beiden letzten Befehle im *Format*-Menü dienen dazu, die Formatänderungen, die Sie mit dem *Font*- und dem *Style*-Menü durchgeführt haben, wieder rückgängig zu machen. Wenn Sie *Remove Fonts* (zu deutsch etwa Schriftarten entfernen) aus dem *Format*-Menü wählen, erscheint der gesamte Text wieder in einer Schriftart. Alle Schriftartenwechsel, die Sie vorgenommen haben, werden entfernt und statt dessen erscheinen alle Buchstaben in derselben Schriftart wie der erste Buchstabe des Blattes. Diesen Befehl dürfen Sie aber nicht mit *Global Font* verwechseln. Nach dem Aufruf von *Remove Fonts* können Sie wieder von neuem beginnen, in einzelnen Passagen eines Textes andere Schriftarten zu verwenden. Ist hingegen *Global Font* abgehakt, erscheinen immer alle Buchstaben des Textes in der zuletzt gewählten Schriftart.

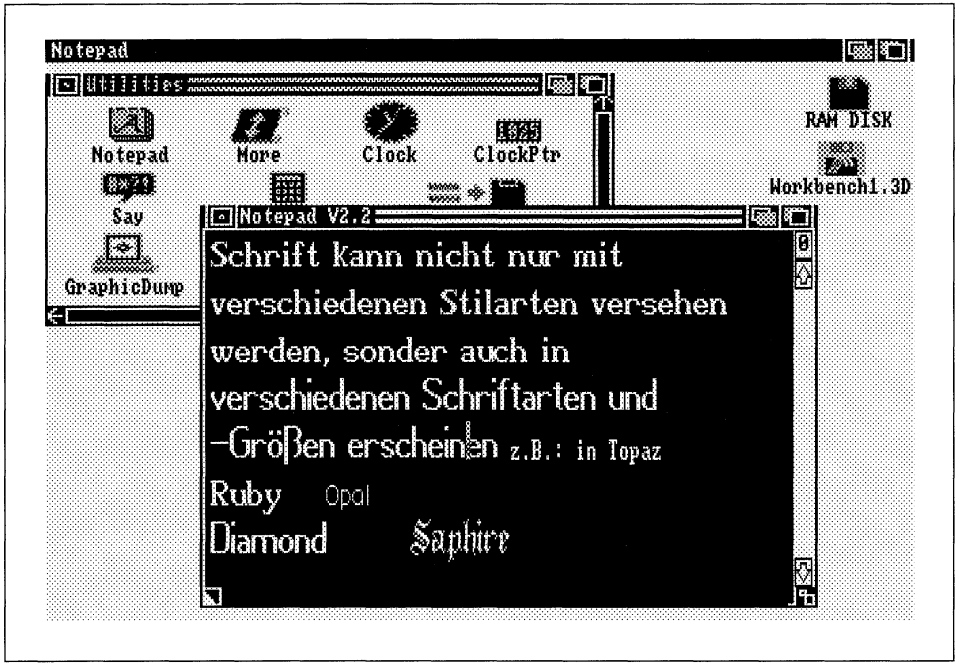



***Bild 4.13:*** *Text mit verschiedenen Schriftarten*

Genauso wie *Remove Fonts* auf die Schriftarten, wirkt *Remove Styles* (zu deutsch etwa Stilwechsel entfernen) auf die mit dem *Style*-Menü festgelegten Schriftstile. Nach dem Aufruf von *Remove Styles* erscheint der gesamte Text *Plain* (also nicht fett, nicht kursiv und nicht unterstrichen).

## 4.3 Wichtige Eigenschaften von Projekten

Nun haben Sie bereits die Grundzüge eines ziemlich leistungsfähigen und komplexen Werkzeugs kennengelernt (eine echte Textverarbeitung ist allerdings noch um einiges komplizierter). Worum es jedoch eigentlich in diesem Abschnitt geht, ist bislang noch zu kurz gekommen: die Projekte beziehungsweise Dokumente. Der Notizblock ist ja – anders als die Uhr zum Beispiel – ein Programm, mit dem Sie wirklich ein Dokument bearbeiten können.

Den Text, den Sie im Notizblock erstellt haben, verlieren Sie normalerweise in dem Augenblick, in dem Sie das Fenster des Programms *Notepad* wieder schließen – wenn Sie ihn nicht vorher sichern beziehungsweise speichern. Die meisten Programme – so auch *Notepad* – besitzen für Operationen mit Dokumenten (die Workbench nennt sie Projekte) ein spezielles Menü: das *Project*-Menü. Mit den Befehlen aus diesem Menü kann ein Projekt üblicherweise auf der Diskette abgespeichert, wieder von Diskette gelesen und ausgedruckt werden. Weiterhin befindet sich bei den meisten Programmen auch der Befehl zum Beenden des Programms im *Project*-Menü.

### 4.3.1 Das Project-Menü

Das erste Menü ganz links in der Menüleiste ist üblicherweise dieses *Project*-Menü. Das ist bei *Notepad* nicht anders. Dieses Menü ist in gleicher oder ähnlicher Form bei vielen anderen Programmen auch zu finden. Es lohnt sich deshalb für Sie, sich ein wenig näher damit zu beschäftigen.

Der erste Befehl, *New* (engl. für neu), löscht das *Notepad*-Fenster und beginnt so ein neues Projekt. Aller Text, den Sie eingegeben haben, geht dadurch verloren. Vermeiden Sie es also nach Möglichkeit, diesen Befehl aus Versehen auszuwählen.

Der Befehl *Save* (engl. *to save* = sichere) dient zum Abspeichern des gerade erstellten Textes auf Diskette. Wenn Sie den Text das erste Mal mit diesem Befehl abspeichern, müssen Sie ihm dazu natürlich einen Namen geben, wofür ein Dialogfenster erscheint.




***Bild 4.14:*** *Der Save-Requester*

Der *Save*-Requester besteht aus einem Text-Gadget, in das Sie einen Namen eingeben können, und zwei Knöpfen. Klicken Sie das Text-Gadget neben der Bezeichnung *Name* an und tippen Sie den gewünschten Namen des Dokuments (Projekts) ein. Schließen Sie den Namen (wie jede Texteingabe) mit der [Return]-Taste ab. Betätigen Sie dann, falls nötig, mit der Maus den Knopf *OK*. (Dieser Knopf wird bei allen neueren Versionen automatisch betätigt, sobald Sie auf die [Return]-Taste drücken.) Ihr Text wird nun unter dem eingegebenen Namen auf die Diskette geschrieben und somit »gesichert«. Die Bezeichnung »Sichern« für diesen Vorgang kommt einfach daher, daß die Daten dort (auf der Diskette)

wesentlich sicherer sind als im RAM-Speicher des Amiga. Im Gegensatz zu den Daten im RAM-Speicher überstehen die Daten auf der Diskette zum Beispiel das Ein- und Ausschalten des Geräts.

Falls Sie es sich anders überlegen, wenn der Save-Requester erschienen ist, und den Text nicht mehr abspeichern wollen, brauchen Sie nur auf den Knopf *CANCEL* zu klicken. Der Requester verschwindet dann und nichts weiter geschieht – insbesondere werden keine Daten auf der Diskette abgespeichert.

Sobald Sie den Text einmal gesichert haben, hat er einen Namen und *Notepad* merkt sich diesen Namen auch. Wenn Sie das nächste Mal *Save* aufrufen erscheint deshalb kein Dialogfenster mehr. Statt dessen speichert *Notepad* dann sofort den neuen Text – mit sämtlichen Änderungen, die Sie nach dem letzten Sichern durchgeführt haben, auf der Diskette ab. Dabei wird die letzte gespeicherte Version allerdings zerstört!

Wenn Sie den geänderten Text unter anderem Namen abspeichern wollen, müssen Sie den Befehl *Save As* (engl. für Sichere als/unter dem Namen) verwenden. Der Befehl *Save As* wirkt genau wie *Save*, nur, daß Sie immer einen Namen für den zu sichernden Text eingeben müssen – auch dann, wenn Sie ihn schon einmal gesichert hatten und ihm dabei einen Namen gegeben haben.

*Save* und *Save As* sind (auch in den meisten anderen Programmen) die Befehle, mit denen neue Projektdateien auf der Diskette erzeugt werden. Sie können das leicht überprüfen: Starten Sie *Notepad*, falls Sie das noch nicht getan haben, und geben Sie einen kurzen Text auf dem ersten Blatt ein. Sichern Sie diesen dann mit *Save* unter einem beliebigen Namen auf der Diskette. Daraufhin erscheint nach kurzer Zeit im Fenster *Utilities* (in dem ja das Werkzeug *Notepad* liegt) ein neues Piktogramm, das wie ein Blatt Papier aussieht und unter dem der Name steht, den Sie im Save-Requester angegeben haben. Dies ist ein Projekt, wie es *Notepad* erzeugen und bearbeiten kann.

Mit dem Befehl *Open* können Sie ein solches Projekt wieder im *Notepad*-Fenster anzeigen lassen und bearbeiten, wenn Sie es einmal abgespeichert haben. Nachdem Sie *Open* aufgerufen haben, erscheint wieder das Dialogfenster, das Sie ja schon von den Befehlen *Save* und *Save As* her kennen. Sie müssen jetzt den Namen der gewünschten Projektdatei eingeben. Drücken Sie danach wie üblich auf die [Return]-Taste und dann mit der Maus auf *OK*. Der Text (die Projektdatei) mit dem vorher angegebenen Namen wird eingelesen und angezeigt. Dabei verschwindet aber der alte Text, den Sie zuvor im *Notepad* bearbeitet hatten. Wenn Sie ihn nicht zuvor gesichert hatten, ist er verloren.

Mit den beiden Menüpunkten *Print* (engl. *to print* = drucken) und *Print As* (engl. für *drucke als*) können Sie den Inhalt des Notizblocks zu Papier bringen. Beide Menüpunkte bringen Untermenüs zum Vorschein, die bestimmen, wie der Ausdruck stattfindet beziehungsweise aussieht.

Zunächst müssen Sie im Submenü Print As auswählen, ob beim Ausdruck versucht werden soll, den Text so erscheinen zu lassen, wie er auch auf dem Bildschirm erscheint. Sie haben hier die Auswahl zwischen *Graphic* und *Draft* (engl. für Entwurf). Diese beiden Punkte des Untermenüs sind Schalter, die sich gegenseitig ausschließen. (Solche Menü-Schalter sind Ihnen ja bereits bekannt.) Ist der Schalter *Graphic* an, so werden bei einem Ausdruck des Textes alle Besonderheiten und speziellen Effekte, die in den vorangegangenen Abschnitten

vorgestellt wurden, auf dem Papier ausgegeben. Hierzu gehören zum Beispiel die verschiedenen Schriftarten und Schriftstile. Wenn Sie einen Mehrfarbdrucker verwenden, werden auch die auf dem Bildschirm zu sehenden Farben beim Ausdruck verwendet. Sie erhalten dann einen Ausdruck, der weitgehend dem Bild entspricht, das Sie am Bildschirm sehen.

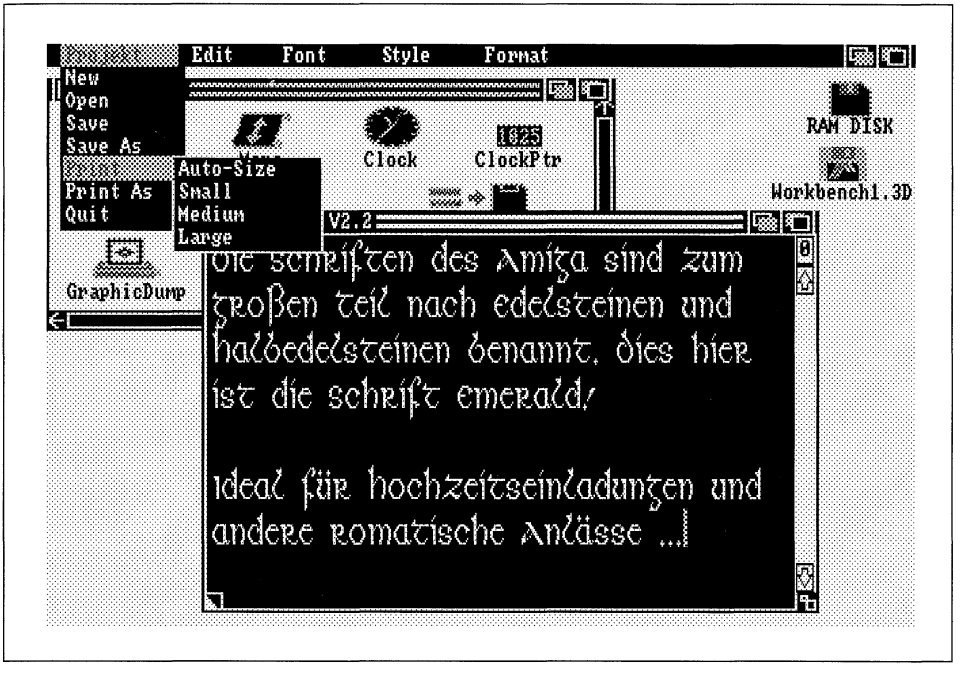



***Bild 4.15:*** *Das Untermenü Print*

Das funktioniert natürlich nur dann, wenn Sie einen Drucker verwenden, der grafikfähig ist. Wenn Sie einen nicht grafikfähigen Drucker anschließen oder den Schalter *Draft* im Submenü *Print As* aktivieren (indem Sie diesen Befehl aufrufen), werden zwar alle Zeichen des Textes korrekt ausgegeben, die verschiedenen Schriftarten und Zeichenstile und Farben werden ignoriert. Der letzte Punkt im Submenü *Print As* ist nur dann wichtig, wenn Sie einen sehr langen Text ausdrucken – wozu der Notizblock eigentlich nicht geeignet ist. Wenn Sie *Form-Feeds* (engl. für Seitenvorschübe) auswählen und damit auch abhaken, werden beim Ausdruck am oberen und unteren Rand des Papiers einige Zeilen frei gelassen. Ist *Form-Feeds* nicht abgehakt, wird es bei längeren Texten vorkommen, daß der Drucker über das Seitenende hinweg druckt. Das kann bei der Verwendung von Endlospapier sinnvoll sein, ist bei einzelnen Blättern aber sicherlich nicht erwünscht.

Nachdem Sie mit *Print As* die Ausdruckoptionen gewählt haben, können Sie mit *Print* den Druckvorgang starten. *Print* ist ebenfalls ein Untermenü mit den Unterpunkten *Auto-Size* (engl. für automatische Größe), *Small* (engl. für klein), *Medium* (engl. für mittel) und *Large* (engl. für groß). Alle diese Befehle bewirken einen Ausdruck des Notizblockinhalts auf dem

Drucker (sofern Sie einen besitzen und korrekt angeschlossen haben). Wählen Sie *Auto-Size, entscheidet* das Programm, in welcher Größe Ihr Text ausgegeben wird. Bei *Small, Medium* und Large bestimmen Sie selbst die Größe des Ausdrucks (klein, mittel oder groß). Experimentieren Sie am besten selbst ein wenig mit den verschiedenen Möglichkeiten herum! Es kostet ein wenig Papier, ist aber die beste Methode, um zu wirklich (für Sie) brauchbaren Ergebnissen zu kommen.

Der letzte Befehl im *Project*-Menü ist ein besonders »schwerwiegender«. Mit *Quit* verlassen Sie das Werkzeug *Notepad*. Sämtliche Texte beziehungsweise alle Änderungen, die Sie zu diesem Zeitpunkt nicht gesichert haben, sind verloren. Denselben Effekt wie mit *Quit* können Sie aber auch erreichen, indem Sie das Schließ-Gadget des *Notepad*-Fensters betätigen. Auch dadurch wird *Notepad* beendet.

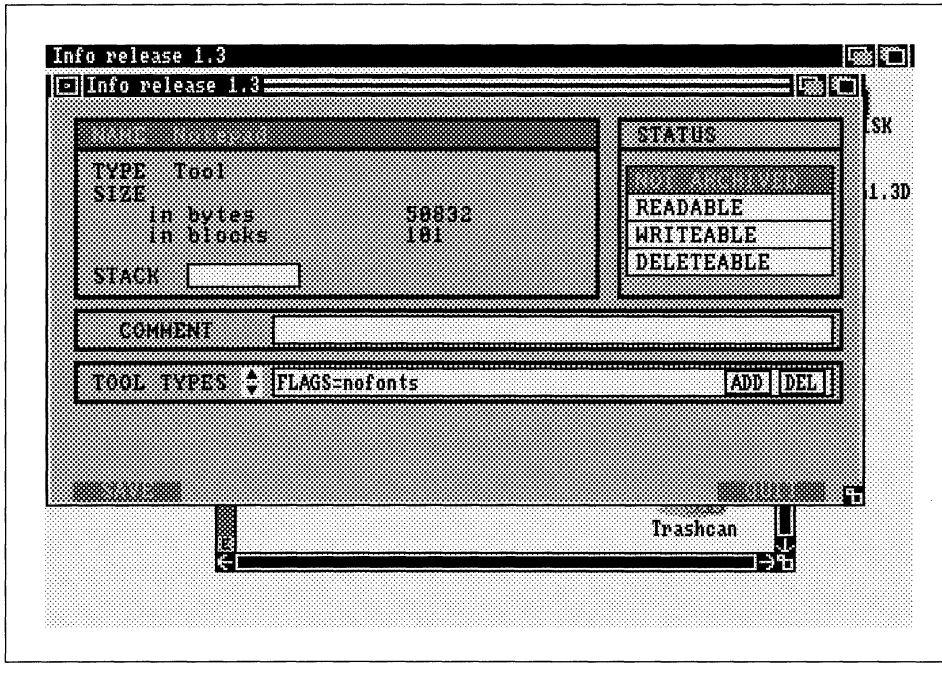



***Bild 4.16:*** *Das Infofenster eines Werkzeugs*

## 4.3.2 Das Infofenster eines Werkzeugs

Nachdem Sie jetzt ja die typische Arbeitsweise eines Werkzeugs andeutungsweise kennengelernt haben, sollten Sie sich als nächstes anschauen, was die Workbench an Informationen über Projekte und Werkzeuge zu vermelden hat. Dafür ist wie üblich der *Info*-Befehl der Workbench zuständig. Verlassen Sie deshalb den Notizblock mit dem Befehl *Quit*. Selektieren Sie dann das *Notepad*-Piktogramm (mit einem Mausklick auf das Piktogramm) und wählen Sie den Befehl *Info* aus dem *Workbench*-Menü.

Dieses Fenster dürfte Ihnen bereits von den anderen Workbench-Objekten her vertraut sein. Es enthält nur einige zusätzliche Komponenten. Links in diesem Fenster steht unter dem Namen der Objekt-Typ, in diesem Fall *Tool* (engl. für Werkzeug). Darunter steht die Größe in Byte (Buchstaben) und Blöcken (die jeweils 512 Byte enthalten). Ähnliche Angaben gibt es ja auch bei Disketten-Infos. Darunter befindet sich nun außerdem ein neues Feld namens *Stack*. Hinter dem Wort *Stack* können Sie eine Zahl eingeben, die bestimmt, wieviel einer besonderen Sorte Speicherplatz das Werkzeug beim Start im RAM des Amiga reserviert bekommt. Diese Angabe ist für Sie erst dann interessant, wenn Sie ein wenig mehr vom »Innenleben« des Amiga kennengelernt haben; sie sei aber der Vollständigkeit halber erwähnt. (Falls Sie bei der Anwendung eines Programms einmal eine Fehlermeldung bekommen, in der der Begriff *stack overflow* vorkommt, können Sie versuchen, diesen Wert zu ändern (also zu vergrößern). Normalerweise beträgt er 4000 (Byte), wird aber nicht angezeigt. Tragen Sie schrittweise größere Werte ein, und versuchen Sie den Vorgang, der zum Fehler führte, noch einmal.)

Die meisten der anderen Felder im Infofenster sollten Ihnen zwar bereits vom Infofenster einer Schublade oder Diskette bekannt sein. Diese Felder haben aber nur bei den Piktogrammen von Projekten oder Werkzeugen eine sinnvolle Funktion und nicht bei Schubladen oder Disketten!

### 4.3.4 Ein Piktogramm vor versehentlichem Löschen schützen

Sie können im Infofenster zum Beispiel ein Programm vor unbeabsichtigtem Löschen schützen. Klicken Sie dazu einfach im Rechteck mit dem Titel *STATUS* (rechts oben) das Feld an, in dem das Wort *DELETEABLE* (engl. für löschbar) steht. An dieser Stelle erscheint nach dem Mausklick nun *NOT DELETEABLE* (engl. für nicht löschbar). Wenn Sie dann später versuchen, *Notepad* zu löschen, erhalten Sie nur eine Fehlermeldung. Bevor das Programm wirklich gelöscht werden kann, muß mit dem Info-Befehl dann zunächst der »Schalter« unter *STATUS* wieder auf *DELETEABLE* umgelegt werden.

Dieser Schutz für Programme (und andere Dateien) ist allerdings nicht allzu sicher. Er verhindert zum Beispiel nicht ein Löschen der gesamten Diskette, und damit natürlich auch des »geschützten« Programms, mit dem *Initialize*-Befehl. Wirklich wichtige Disketten sollten Sie deshalb immer mit dem Schreibschutzschieber der Disketten sichern.

### 4.3.3 Ein Piktogramm mit einem Kommentar versehen

In einem weiteren Feld des Infofensters können Sie zu jedem Programm oder Projekt einen zusätzlichen Kommentar vermerken. Klicken Sie zum Anbringen eines Kommentars einfach in das blaue Rechteck hinter *COMMENT* (engl. für Kommentar). Sie können in diesem Feld dann einen beliebigen kurzen Text abspeichern und bekommen ihn dann beim nächsten Aufruf des *Info*-Befehls wieder angezeigt.

Probieren Sie es aus: Klicken Sie einmal in das Feld hinter *COMMENT* und geben Sie »Das ist ein Kommentar« ein. Schließen Sie dann die Eingabe mit der [Return]-Taste ab. Zur Fehlerkorrektur innerhalb des Rechtecks stehen Ihnen übrigens genau dieselben Möglichkeiten zur Verfügung wie beim Befehl *Rename* (siehe oben).

### 4.3.5 Das Feld TOOL TYPES im Infofenster

Unter dem *COMMENT*-Feld befindet sich das Feld *TOOL TYPES*, das es nur im Infofenster von Werkzeugen gibt. Sie können hier einen oder mehrere Texte eintragen (wie, erfahren Sie gleich), die dem Programm beim Start mitgeteilt werden. Diese Texte nennt man »Parameter« oder »Argumente« und Sie stellen praktisch Befehle an das Programm dar – nur, daß diese Befehle nicht mit der Maus erteilt werden, sondern dem Programm beim Start automatisch mitgeteilt werden. Indem Sie hier im Infofenster diese Parameter eintragen, erreichen Sie ferner, daß Sie sie nur einmal angeben müssen und daß sie trotzdem bei jedem Start wirksam werden. *Notepad* können Sie zum Beispiel mit Hilfe solcher Parameter mitteilen, wie groß das Fenster des Programms beim Start sein soll, welche Schriftart verwendet werden soll, solange Sie nichts anderes angeben, und so weiter. Typischerweise dienen Parameter dazu, die Grundeinstellungen (der Fachmann sagt »Default-Einstellungen« oder einfach »Defaults« dazu) zu verändern, die in einem Programm nach dem Start gelten.

Wenn Sie einen neuen Parameter eintragen wollen, müssen Sie zunächst auf das ADD (engl. to add = hinzufügen/ ergänzen) am rechten Rand des Feldes *TOOL TYPES* klicken. Klicken Sie dann in das blaue Rechteck links daneben (ein Text-Gadget) und tippen Sie den Parameter ein. Ein solcher Parameter besteht immer aus einem Wort in Großbuchstaben, gefolgt von einem Gleichheitszeichen, hinter dem weitere Informationen folgen, deren Aufbau abhängig vom Namen des Parameters (dem ersten Wort) ist.

Wenn Sie mehr als einen Parameter eintragen wollen, brauchen Sie einfach nur erneut auf ADD zu klicken und das Textgadget links daneben wird wieder frei. Sie können dann den Text des nächsten Parameters eintippen. Sie können sich auch alle Parameter, die Sie oder ein anderer schon eingegeben haben, nachträglich anschauen oder auch ändern. Hierzu dienen die beiden Pfeile rechts neben der Beschriftung *TOOL TYPES*. Indem Sie diese anklicken, können Sie zwischen den Parametern vor- und zurückblättern. Wenn ein Parameter erst einmal in dem blauen Rechteck neben den beiden Pfeilen erschienen ist, können Sie ihn auch ändern. Klicken Sie dazu einfach mit der Maus auf eine beliebige Stelle im Text des Parameters. Sie können ihn dann mit der [Del]-, der [Backspace]- und den Cursor-Tasten abändern, wie Sie wünschen. Ihnen stehen dazu dieselben Möglichkeiten zur Verfügung wie in jedem Text-Gadget (diese Möglichkeiten werden vollständig in Kapitel 6 beschrieben). Wenn Sie einen Parameter nicht nur ändern, sondern komplett löschen wollen, brauchen Sie nur auf den Knopf *DEL* (engl. *to delete* = löschen) am rechten Rand des Feldes *TOOL TYPES* zu klicken. Der Parameter verschwindet daraufhin und der vorangehende (falls es einen gibt) erscheint am Bildschirm.

Dies war zunächst einmal nur eine grundsätzliche Beschreibung der Funktion des Feldes *TOOL TYPES* im Infofenster eines Projekts. Eine vollständige Liste der Parameter, die Sie beim Programm *Notepad* verwenden können, erhalten Sie im folgenden Abschnitt.

### 4.3.6 Das Infofenster schließen

Das Infofenster verschwindet (wird geschlossen), wenn Sie in das Schließ-Gadget den Knopf *QUIT* oder den Knopf *SAVE* klicken. Alle Änderungen im Infofenster, die gerade beschrieben wurden, sind nur dann dauerhaft, wenn Sie zum Abschluß mit der Maus den Knopf *SAVE* drücken. Schließen Sie das Fenster mit dem Schließ-Gadget oder indem Sie auf

den Knopf *QUIT* unten rechts in der Ecke drücken, so werden sämtliche Änderungen, die Sie im Infofenster durchgeführt haben, ignoriert. Wenn Sie das Infofenster das nächste Mal öffnen, erhalten Sie wieder die alten Einstellungen. Wenn Sie also den Kommentar einer Schublade geändert oder den Schalter im Kasten *STATUS* umgelegt haben, sollten Sie das Fenster unbedingt mit *SAVE* verlassen. Nur dann werden Ihre Änderungen auch wirklich berücksichtigt und dauerhaft gespeichert.

Beachten Sie bitte auch, daß Änderungen im Feld TOOL TYPES erst beim nächsten Start des entsprechenden Programms (Öffnen des Projekt-Piktogramms) Wirkung zeigen. Änderungen im Infofenster haben keine Wirkung auf vorher gestartete, laufende Programme.

### 4.3.7 Die Parameter von Notepad

Nachdem Sie nun wissen, wie Sie einem Programm über das Infofenster Befehle (Parameter) zukommen lassen können, sollen Sie auch gleich die vollständige Liste der Parameter, die *Notepad* versteht, erhalten. Sie können diese neue Möglichkeit dann sofort nach Herzenslust ausprobieren.

Der Parameter *WINDOW* bestimmt die Größe des *Notepad*-Fensters direkt nach dem Start des Programms. Hinter dem Gleichheitszeichen (=) müssen bei diesem Parameter vier von Kommata getrennte dreistellige Zahlen stehen. Diese Zahlen geben nacheinander an, welchen Abstand der linke Fensterrand vom linken Bildschirmrand, welchen Abstand der obere Fensterrand vom oberen Bildschirmrand, welche Breite und welche Höhe das Fenster hat. (Alle diese Angaben müssen dreistellig sein und werden in Bildschirmpunkten oder Pixeln gemessen.) Wenn Sie also wollen, daß das *Notepad*-Fenster beim Start 300 Punkte breit und 100 Punkte hoch ist und die linke obere Ecke 10 Punkte vom linken und 40 Punkte vom oberen Bildschirmrand entfernt ist, müssen Sie Taste ADD im Feld *TOOL TYPES* anklicken und dann daneben »WINDOW=010,040,300,100« eintragen. (Beachten Sie die führende Null bei 010 und 040. Diese Null oder eine Leerstelle müssen Sie eingeben, wenn eine Zahl weniger als drei Stellen hat!)

Der Parameter *FONT* bestimmt, in welcher Schriftart die Buchstaben, die Sie eingeben, erscheinen, solange Sie keine Schriftart aus dem *Font*-Menü wählen (also die Default-Schriftart). Hinter dem Gleichheitszeichen müssen Sie den Namen der gewünschten Schriftart (derselbe wie im *Font*-Menü), einen Punkt und die gewünschte Größe angeben. Wenn Sie also wollen, daß sofort nach dem Start eingegebene Buchstaben in der Schriftart *Diamond* und der Größe *20* erscheinen, müssen Sie »FONT=diamond.20« im Feld *TOOL TYPES* eintragen.

Der Parameter FLAGS erlaubt es Ihnen, einige der als Schalter wirkenden Menüpunkte im *Project*- und *Format*-Menü schon vor dem Start zu bestimmen. So können Sie im Submenü *Print As* zum Beispiel festlegen, ob Sie beim Drucken nur den »nackten Text« ohne Schriftart-und Stilwechsel (*Draft*) oder den Text, so wie er auf dem Bildschirm erscheint (*Graphic*), ausgeben wollen und ob Sie am Seitenende ein paar Leerzeilen wünschen. Voreingestellt ist nach dem Start Graphic; die »Schalter« Draft und Form-Feeds sind aus. Indem Sie »FLAGS=draft« im Feld TOOL TYPES eintragen, können Sie dafür sorgen, daß immer im Draft-Modus gedruckt wird (sofern Sie nicht explizit noch einmal *Graphic* wählen) und mit dem Parameter »FLAGS=formfeed« können Sie genauso den Schalter *Form-Feeds* im Submenü *Print As* aktivieren.

Nach dem Start von *Notepad* ist der Schalter *Global Font* im Menü *Format* zunächst an (abgehakt). Jede Auswahl einer neuen Schriftart aus dem *Font*-Menü betrifft deshalb immer den gesamten Text. Wenn Sie jedoch im Feld TOOL TYPES des Infofensters von *Notepad* »FLAGS=local« eintragen, ist *Global Font* gleich nach dem Start von *Notepad* ausgeschaltet und Sie können mehrere verschiedene Schriftarten auf einem Blatt verwenden. Mit »FLAGS=global« können Sie in ähnlicher Form *Global Font* wieder aktivieren. Genauso können Sie den Schalter *Word wrap* im Menü *Format* beeinflussen. »FLAGS=nowrap« macht *Word wrap* inaktiv.

Und schließlich können Sie mit dem Parameter »FLAGS=nofonts« verhindern, daß *Notepad* beim Start auf der Workbench-Diskette nachsieht, welche Schriftarten vorhanden sind. Der Start von *Notepad* geht dann wesentlich schneller und das *Font*-Menü bleibt bis auf die Schrift *Topaz* leer. Wenn Sie sowieso keinen Wert auf die Verwendung verschiedener Schriftarten legen, sollten Sie diesen Parameter sofort eintragen. Sie sparen dadurch bei jedem Start von *Notepad* einige Sekunden. Falls Sie später doch mehrere Schriftarten verwenden wollen, können Sie das *Font*-Menü jederzeit mit dem Befehl *Read Fonts* (engl. für Schriftarten einlesen) aus dem *Project*-Menü nachträglich aufbauen.

Wenn Sie eine frühe Version des Programms *Notepad* besitzen, beziehungsweise zusammen mit Ihrem Amiga 500 erhalten haben, kann es übrigens sein, daß einige der hier beschriebenen Parameter (insbesondere eine Kombination mehrerer FLAGS-Parameter) nicht funktionieren. Bei der von mir getesteten *Notepad*-Version war dies der Fall. Die Firma Commodore versichert jedoch, daß die endgültige Version von *Notepad* alle hier beschriebenen Parameter erkennen wird.

## 4.4 Zusammenhänge von Werkzeugen und Projekten

Projekte und Werkzeuge gehören stets eng zusammen. Oft gibt es zu jedem Projekt nur genau ein Werkzeug, mit dem es bearbeitet werden kann, und alle Projekte, die mit einem bestimmten Werkzeug bearbeitet werden können, besitzen auch dasselbe Piktogramm.

### 4.4.1 Öffnen eines Projektes

Der Zusammenhang zwischen Projekten und Werkzeugen wird aber nicht nur durch solche Äußerlichkeiten bestimmt. Sie haben statt dessen auch die Möglichkeit, ein Projekt zu öffnen wie ein Werkzeug. Das heißt, wenn Sie durch einen Doppelklick oder den Befehl *Open* ein Projekt statt eines Werkzeuges öffnen, so wird Ihnen – wie von anderen Objekten bekannt – gezeigt, was es enthält. Im Falle eines Textes muß also dieser Text in einen Fenster dargestellt werden. Hierzu wird das Projekt natürlich ein Werkzeug benötigen, das das Projekt (in diesem Fall den Text) dann anzeigt. Idealerweise sollte das das Werkzeug sein, mit dem das Projekt erstellt wurde, denn nur dieses Werkzeug »weiß« wahrscheinlich wie man das Projekt richtig darstellt.

Wenn Sie ein Projekt öffnen, prüft die Workbench deshalb zunächst, zu welchem Werkzeug das Projekt gehört, und startet dieses. Gleichzeitig wird dem Werkzeug mitgeteilt, welche Projekt der Anwender geöffnet hat. (Dies geschieht auf ähnlichem Wege wie die Mitteilung der oben beschriebenen Parameter.) Das Werkzeug liest dieses Projekt daraufhin ein und

zeigt es am Bildschirm. Dies ist wesentlich praktischer, als immer zuerst ein Werkzeug starten zu müssen und dann das Projekt einzulesen, wenn man es bearbeiten oder nur betrachten will. Hier geschieht also beim Öffnen eines Projektes genau das, was Sie aus Ihrer bisherigen Erfahrung mit anderen Objekten erwarten können.

Probieren Sie das am besten gleich einmal aus: Starten Sie dazu *Notepad*, tippen Sie ein paar Sätze ein und sichern Sie diesen Text dann (mit dem oben beschriebenen Befehl *Save*) unter einem beliebigen Namen ab. Sie können das *Notepad*-Fenster dann schließen. Daraufhin sollte ein neues Piktogramm im *Utilities*-Fenster erschienen sein. Falls nicht, schließen Sie bitte das *Utilities*-Fenster und öffnen Sie dann die Schublade *Utilities* erneut. Machen Sie jetzt einen Doppelklick auf das neue Piktogramm. Die Workbench startet daraufhin automatisch *Notepad* und zeigt gleich nach dem Start den als Projekt abgespeicherten Text im *Notepad*-Fenster.

### 4.4.2 Das Infofenster eines Projektes

Die Verbindung zwischen einem Projekt und dem zugehörigen Werkzeug wird auch dann sichtbar, wenn man das Infofenster des Projektes öffnet (also dem Projekt den Befehl *Info* erteilt). Um das auszuprobieren, benötigen Sie ein Projekt, also einen Text, den Sie mit *Notepad* erfaßt und abgespeichert haben. Ein solches Projekt sollte noch – von Ihren diversen Experimenten mit *Notepad* her – im *Utilities*-Fenster liegen. Selektieren Sie dieses neue Piktogramm nun und wählen Sie den Befehl *Info* aus dem Menü *Workbench*.

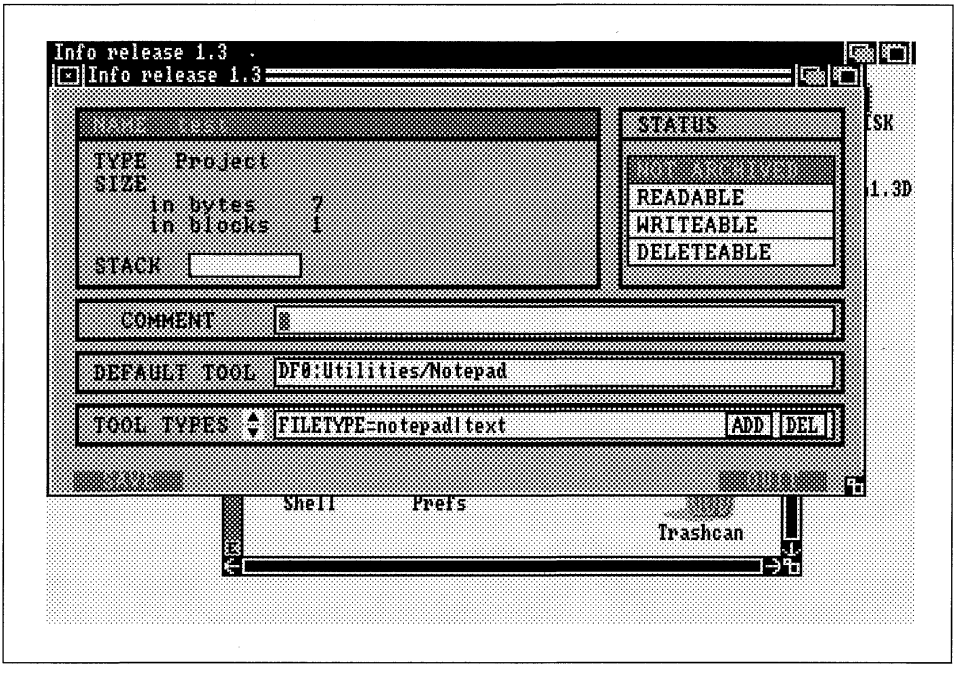



***Bild 4.17:*** *Das Infofenster eines Projektes*

Das Infofenster eines Projektes sieht fast genauso aus wie das eines Werkzeugs (Programms). Es enthält nur ein zusätzliches Text-Gadget namens *DEFAULT TOOL*. Dieses Text-Gadget stellt die Verbindung zwischen einem Projekt und dem zugehörigen Werkzeug her. Im Text-Gadget *DEFAULT TOOL* kann nämlich der Name eines Werkzeuges angegeben werden, das aufgerufen werden soll, wenn das Projekt geöffnet wird. Beim ersten Ablegen eines neuen Projektes auf der Diskette hinterlegt das erzeugende Programm hier üblicherweise seinen eigenen Namen.

Betrachten Sie zum Beispiel das Infofenster eines *Notepad*-Projektes, so steht dort zum Beispiel »Workbench1.3D:Utilities/Notepad«. »Workbench1.3D:« bedeutet, daß das Werkzeug auf der Workbench-Diskette zu finden ist, »Utilities/«, daß es in der Schublade *Utilities* liegt und »Notepad« schließlich ist natürlich der Name des Werkzeugs selbst. (Mehr zu dieser ungewöhnlichen Schreibweise eines Dateinamens können Sie erfahren, wenn Sie das Kapitel Dateien und Dateiverzeichnisse lesen.) Sie können hinter *DEFAULT TOOL* theoretisch einen anderen Namen eintragen und diesen dann abspeichern, indem Sie auf *SAVE* klicken. Wenn Sie zum Beispiel einmal eine leistungsfähigere Textverarbeitung erwerben, können Sie auf diese Weise erreichen, daß beim Öffnen von Texten, die mit *Notepad* erstellt wurden, dieses neue Textverarbeitungsprogramm gestartet wird und nicht mehr *Notepad*.

### 4.4.3 Parameter eines Projektes

Eine andere Verbindung zwischen einem Projekt und »seinem« Werkzeug stellt das Feld *TOOL TYPES* dar, das Ihnen ja schon vom Infofenster eines Werkzeugs her bekannt ist. Wenn Sie bei einem Werkzeug in diesem Feld einen oder mehrere Befehle (Parameter) eintragen, so werden diese Befehle dem Werkzeug beim Start mitgeteilt. Genauso verhält es sich, wenn Sie im Infofenster eines Projektes hinter TOOL TYPES Parameter eintragen. Auch diese werden dem Werkzeug mitgeteilt, wenn Sie es – natürlich »indirekt« über das Projekt-Piktogramm – starten. Sie können bei dem Piktogramm eines mit *Notepad* erstellten Projektes also dieselben Parameter verwenden wie beim Piktogramm von *Notepad* selbst. Diese werden aber nur dann gültig, wenn Sie *Notepad* durch Öffnen dieses Projekt-Piktogramms starten und nicht dann, wenn Sie *Notepad* direkt oder über ein anderes Projekt-Piktogramm öffnen.

## 4.5 Wie man sich zerreißt: Multitasking

Am Beispiel des Werkzeugs *Notepad* und seiner Projekte haben Sie die wichtige Beziehung zwischen Werkzeugen und Projekten kennengelernt. Unbedingt einprägen sollten Sie sich auch die Bedeutung, die das Infofenster sowohl für Werkzeuge wie auch für Projekte hat. Damit soll die Beschreibung des Verhältnisses von Werkzeugen und Projekten und der verschiedenen Werkzeuge der Workbench aber auch abgeschlossen werden. Im nächsten Kapitel werden Sie auch noch die anderen Werkzeuge näher kennenlernen, die sich außer der Uhr, dem Rechner und dem Notizblock noch auf der Workbench-Diskette befinden.

Bevor wir uns aber mit diesen anderen Werkzeugen näher beschäftigen, sollen zunächst noch zwei weitere wichtige Konzepte vorgestellt werden, mit denen jeder Amiga-Anwender vertraut sein muß: das »Multitasking« und die sogenannten »Screens«.

Besonders das Prinzip des Multitaskings sollte jeder Amiga-Anwender kennen und auch verstehen. Es kann einem bei der täglichen Arbeit nämlich viel helfen. Wer hat nicht schon einmal den Stoßseufzer gehört oder selbst ausgesprochen: »Ich kann mich doch nicht zerreißen«? Das soll im allgemeinen bedeuten, daß jemand aufgefordert wird, mehrere Aufgaben gleichzeitig zu erledigen, was natürlich unmöglich ist – zumindest für einen Menschen. Der Amiga kann sich zerreißen. Oder anders formuliert: Der Amiga kann mehrere Programme (scheinbar) gleichzeitig bearbeiten!

Sie haben diese Fähigkeit auch schon in Aktion gesehen – nur vielleicht nicht näher darüber nachgedacht. Wenn Sie das Programm *Notepad* gestartet haben und dieses Programm sein Fenster geöffnet hat, hält Sie nämlich niemand davon ab, in ein anderes Fenster zu klicken und dort mit einem anderen Programm zu arbeiten – oder ein paar Piktogramme von einer Schublade in eine andere zu legen.

Das heißt aber nichts anderes, als daß zwei Programme mehr oder weniger gleichzeitig aktiv sind: die Workbench (die ja auch ein Programm ist) und *Notepad*. Beide Programme können sogar (fast) gleichzeitig etwas tun, wie man am Beispiel der Uhr sehr gut sehen kann (speziell dann, wenn man den Sekundenzeiger eingeschaltet hat). Die Uhr tickt, nachdem Sie einmal gestartet wurde, weiter, auch wenn man auf der Workbench andere Aufgaben erledigt.

Auf anderen Computern, von denen Sie möglicherweise auch den einen oder anderen kennen, ist ein solches Verhalten durchaus nicht selbstverständlich. Startet man dort ein Programm, so »übernimmt es« den ganzen Computer, beansprucht den ganzen Speicher für sich, löscht unter Umständen alles, was sich darin befindet und beansprucht auch den gesamten Bildschirm für sich. Auf dem Amiga üben die Programme hingegen (weitgehend) friedliche Koexistenz, wozu unter anderem auch die Fenster beitragen, die dafür sorgen, daß den Programmen genau abgegrenzte Bereiche des Bildschirms zugeteilt werden können, ohne sie dadurch allzusehr einzuschränken.

Der Fachmann nennt dieses Verhalten (die gleichzeitige Aktivität mehrerer Programme) eines Computers »Multitasking«, weil der Computer gleichzeitig an mehreren Aufgaben (engl. *tasks*) arbeiten kann. Dies ist natürlich nur eine scheinbare Gleichzeitigkeit. Auch ein Computer kann immer nur ein Programm zur gleichen Zeit bewältigen. Der Amiga arbeitet aber immer nur ein kleines Stück eines Programms ununterbrochen ab. Danach »springt« er zu einem anderen und bearbeitet dieses für einen kleinen Augenblick und so weiter. Er wechselt aber so schnell von Programm zu Programm, daß für den langsamen Menschen der Eindruck entsteht, er würde diese Programme gleichzeitig bearbeiten.

### 4.5.1 Vorteile des Multitasking

Eventuell werden Sie sich nun fragen, wozu Multitasking denn überhaupt gut sein soll. Denn obwohl der Computer scheinbar schnell genug ist, mehrere Aufgaben gleichzeitig zu bewältigen, kann ja kaum von einem Menschen verlangt werden, blitzschnell zwischen der Bedienung von zwei oder drei Programmen hin und her zu wechseln. Ganz davon abgesehen, daß für einen solchen ständigen Wechsel wohl kaum Bedarf besteht.

Dies ist auch völlig richtig, und auch das Multitasking des Amiga ist nicht dafür gedacht, daß Sie simultan einen Text tippen und Ihre Buchhaltung erledigen. Es gibt aber bei vielen Programmen Vorgänge, die sehr viel Zeit in Anspruch nehmen – einige Minuten oder gar eine Stunde und mehr. Das Ausdrucken langer Dokumente und aufwendige Kalkulations-

aufgaben sind zum Beispiel solche Vorgänge. Wenn Ihr Computer immer nur ein Programm bearbeiten kann, müssen Sie nach dem Start eines solchen langwierigen Vorgangs immer erst darauf warten, daß ein Programm fertig wird, bevor Sie mit der Arbeit mit diesem oder einem anderen Programm fortfahren können. Oder Sie müssen sich einen zweiten Computer kaufen. Beherrscht Ihr Computer aber das Multitasking, können Sie, während sich das eine Programm seiner langwierigen (und wahrscheinlich auch langweiligen) Tätigkeit hingibt, ein anderes Programm für eine andere Aufgabe nutzen.

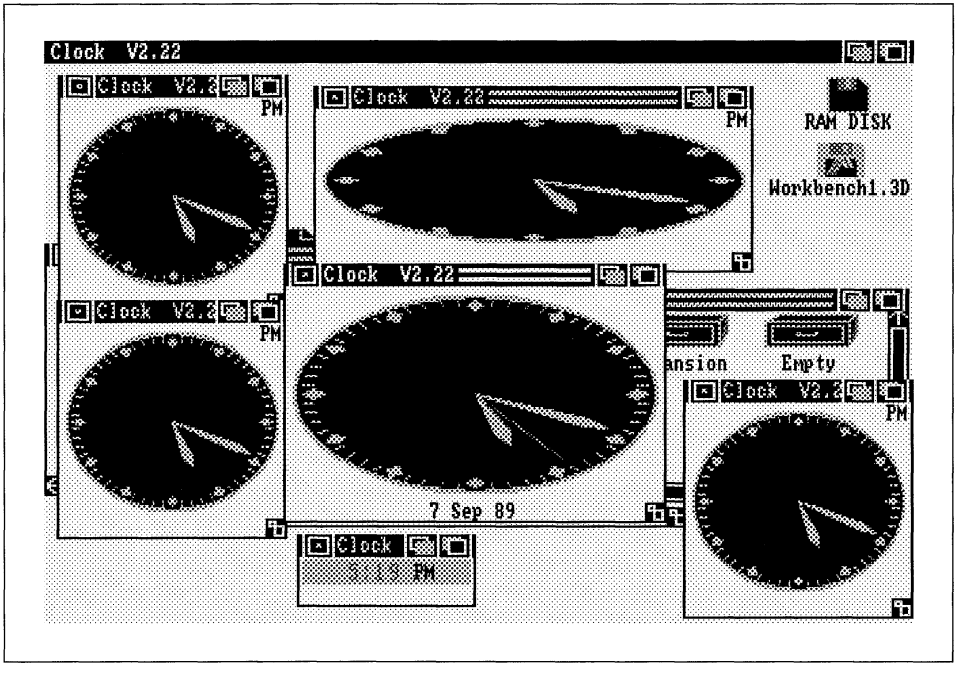



***Bild 4.18:*** *Mehrere Uhren, die gleichzeitig ticken*

Aber auch ein schneller Wechsel zwischen verschiedenen Programmen kann hilfreich sein. Angenommen, Sie wollen einen Bericht schreiben und verwenden dazu ein Textverarbeitungsprogramm. Plötzlich merken Sie, daß Ihnen einige Daten fehlen. Diese Daten verwalten Sie normalerweise mit einem Datenbankprogramm. Um sie jetzt sehen zu können, müßten Sie dieses Programm nun starten. Auf der Workbench des Amiga können Sie das sofort tun und in der Datenbank nachsehen, ohne die Textverarbeitung zu verlassen (und den Text abzuspeichern). Danach sind Sie blitzschnell wieder zurück in der Textverarbeitung. Dazu genügt ein Mausklick in das Fenster der Textverarbeitung. Gebe es kein Multitasking, müßten Sie hingegen das Datenbankprogramm verlassen, die Textverarbeitung wieder starten und den gerade bearbeiteten Text einlesen. Gerade das Starten und Verlassen von Programmen sind aber Vorgänge, die auf einem (bei aller Leistungsfähigkeit doch vergleichsweise kleinen) Computer wie dem Amiga 500 recht zeitraubend sein können.

Die Möglichkeit des Multitasking kann Ihnen also in vielerlei Hinsicht Zeit und Mühe ersparen und nutzt zugleich den Computer auch noch viel besser aus. Selbst ein kleiner Computer wie der Amiga 500 ist nämlich immer noch so schnell, daß er einen Großteil seiner Zeit nichts anderes tut, als auf Ihre Eingaben zu warten. In dieser Zeit könnte er genausogut schon sinnvollere Dinge tun – zum Beispiel drucken oder rechnen. Der Amiga tut das, andere Computer nicht!

Wenn Sie sich davon überzeugen wollen, daß auf dem Amiga wirklich mehrere Programme gleichzeitig laufen können, probieren Sie es am besten gleich einmal aus. Öffnen Sie dazu die Schublade *Utilities* und starten Sie das Programm *Clock*, das sich in dieser Schublade befindet. Schalten Sie die Sekundenanzeige ein und verschieben Sie das Fenster in die Mitte des Bildschirms. Starten Sie jetzt ein zweites Mal *Clock*. Daraufhin erscheint ein neues Fenster mit der Uhr darin. Schalten Sie wieder die Sekundenanzeige an, wählen Sie als Darstellungsart eine Digitaluhr und legen Sie auch dieses Fenster zur Seite. Starten Sie noch einmal *Clock* und sie erhalten ein drittes Uhren-Fenster. In allen drei Fenstern ticken jetzt die verschiedenen Uhren. Das Programm Clock läuft dafür in drei verschiedenen »Inkarnationen«. Und daß diese drei Programme nahezu gleichzeitig laufen, können Sie an den Sekundenzeigern (beziehungsweise der Sekundenanzeige der Digitaluhr sehen), die sich im Gleichtakt ändern. Das ist Multitasking live!

### 4.5.2 Das Problem Speicherplatz

Multitasking ist aber auch nicht ganz ohne Probleme. Wenn zwei identische Programme (Werkzeuge) gleichzeitig arbeiten, benötigen sie auch den doppelten Speicherplatz im Computer. Gerade bei großen und leistungsfähigen Programmen werden Sie rasch feststellen, daß selbst die 1 Million Zeichen, die der Amiga 500 mit Speichererweiterung in seinem RAM-Speicher ablegen kann, nicht allzuviel sind. Während Sie auf der Workbench sind, können Sie dies sehr schön beobachten. Der noch freie Speicherplatz wird dann nämlich in der Titelleiste des Bildschirms immer angezeigt. Jedes Öffnen einer Schublade (aber vor allem eines Werkzeugs oder Projekts) reduziert diesen freien Speicherplatz. Und genau für diesen Zweck ist die Speicheranzeige nämlich auch gedacht. Sie gibt Ihnen die Möglichkeit, recht genau den Speicherbedarf eines Programms zu ermitteln.

Wollen Sie häufig mit mehreren Werkzeugen gleichzeitig arbeiten, also das Multitasking so richtig nutzen, ist es sinnvoll, sich zu jedem Programm zu notieren, welchen Speicherbedarf es ungefähr hat. Zur Ermittlung dieses Wertes reicht meist ein Probelauf des Programms und die Kontrolle der Anzeige des freien Speichers in der Workbench-Titelleiste. Sie können dann jederzeit abschätzen, ob für ein weiteres Werkzeug, das sie gerade öffnen wollen, noch genügend Platz ist.

Ein weiteres Problem des Multitasking ist natürlich, daß sich ein Programm, welches gleichzeitig mit anderen Programmen läuft, auch die Leistung des Computers mit diesen anderen Programmen teilen muß. Bei zwei oder drei Programmen, die gleichzeitig laufen, kann dies noch ganz erträglich sein, besonders dann, wenn diese Programme nicht sehr »rechenintensiv« sind, das heißt, den Computer nicht allzusehr beanspruchen. Je mehr Programme Sie darüber hinaus noch starten, desto langsamer erscheinen die Ergebnisse der Programme auf dem Bildschirm und desto langsamer werden auch die Reaktionen des Amiga auf Ihre Eingaben.

### 4.5.3 Das Problem Bildschirm

Der Speicher und die Leistungskapazität der Rechenchips ist aber nicht das einzige Problem, das das Multitasking mit sich bringt. Genauso wie den Speicher müssen sich mehrere parallel arbeitende Programme natürlich auch den Bildschirm (und die restlichen »Ressourcen«) des Amiga 500 teilen. Eine erste Lösung für dieses Problem stellen schon die Fenster dar. Mit der Verwendung eines Fensters hat ein Programm ja die Möglichkeit, recht unabhängig von dem zu werden, was die anderen Programme gerade tun. Das Programm kann zum Beispiel »so tun«, als hätte es den ganzen Bildschirm oder eine noch viel größere Fläche allein zur Verfügung und kann auf dieser ganzen Fläche Grafiken, Texte usw. zeigen. Sichtbar wird davon aber nur ein Teil, der so groß wie das Fensterinnere des Fensters ist, das dem Programm beim Start zugeteilt wurde. Der Programm-Anwender, nicht der Programmierer, entscheidet dann darüber, wie groß dieser Ausschnitt ist und welcher Teil des Gesamtbildes gezeigt wird.

Die Werkzeuge des Amiga, die Sie bisher kennengelernt haben, nutzen seine grafischen Fähigkeiten kaum. Sie öffnen alle nur ein Fenster auf der Workbench und begnügen sich mit den vier Farben, die die Workbench standardmäßig zur Verfügung stellt (Schwarz, Weiß, Blau und Orange, wenn Sie diese Farben noch nicht mit dem Programm *Preferences* geändert haben). Was aber geschieht, wenn einem Programm die Bildschirm-Attribute, wie sie die Workbench standardmäßig zur Verfügung stellt, »nicht gefallen«? Ein Programm kann zum Beispiel eine größere Anzahl oder eine andere Kombination von Farben bevorzugen (oder benötigen) als die Workbench. Der Amiga bietet ja noch wesentlich mehr grafische Möglichkeiten als die Workbench nutzt.

Ein solches Programm könnte nun natürlich einfach in den Interna des Amiga »herumpfuschen« und die gewünschten Einstellungen für die Bildschirm-Attribute vornehmen. Alle anderen Programme, die gerade aktiv sind, würden davon aber beeinträchtigt. Es könnte sein, daß deren Fenster auf einmal schwer oder gar nicht mehr lesbar wären. Es könnte aber auch vorkommen, daß diese Programme gar nicht mehr lauffähig sind oder nur noch völliger Quatsch auf dem Bildschirm erscheint. Der Amiga 500 beziehungsweise die Amiga-Software bietet aber auch für diese Problematik eine elegante Lösung, die sogenannten »Screens«.

## 4.6 Screens, die Superfenster

Glücklicherweise erlaubt es die Video-Hardware des Amiga 500, jedem Programm seinen eigenen »virtuellen« Bildschirm zur Verfügung zu stellen. (Statt virtuellen Bildschirm werde ich meist den Fachausdruck »Screen« verwenden; er ist einfach kürzer.) Diese Möglichkeit heißt deshalb virtueller Bildschirm, weil es natürlich nur einen wirklichen, physikalischen Bildschirm gibt, den Sie ja vor sich sehen.

Einem Programm kann es aber egal sein, ob es einen virtuellen Bildschirm verwendet oder einen echten; die Programmierung ist dieselbe. Nur Sie als Anwender müssen sich entscheiden, welchen virtuellen Bildschirm Sie auf Ihrem wirklichen Bildschirm sehen wollen. Wollen Sie (Teile von) mehrere(n) virtuelle(n) Bildschirme(n) sehen, müssen Sie den wirklichen Bildschirm entsprechend »aufteilen«. Das geht einfacher als Sie vielleicht annehmen.

### 4.6.1 Wie Sie Screens bewegen

Vielleicht ist es Ihnen auf der Workbench schon öfter einmal passiert, daß Sie aus Versehen mit der linken Maustaste in die Titelleiste des Bildschirms geklickt haben, statt die rechte (Menü-)Taste zu verwenden. Wenn Sie bei einem solchen Fehler die Maus bewegen, kann es passieren, daß sich das Bild zu bewegen beginnt. Klickt man nämlich mit der linken Maustaste in die Titelleiste eines (virtuellen) Bildschirms, so kann man diesen ergreifen – wie ein Piktogramm oder ein Fenster – und auf und ab bewegen. Liegt »hinter« dem Workbench-Bildschirm noch ein anderer Bildschirm, so kommt dieser zum Vorschein, wenn Sie den Workbench-Bildschirm nach unten ziehen. Da nach dem Start keine anderen Programme laufen, wird meist nur der blaue Streifen am oberen Bildschirmrand größer, wenn man aus Versehen die Titelzeile der Workbench nach unten zieht.

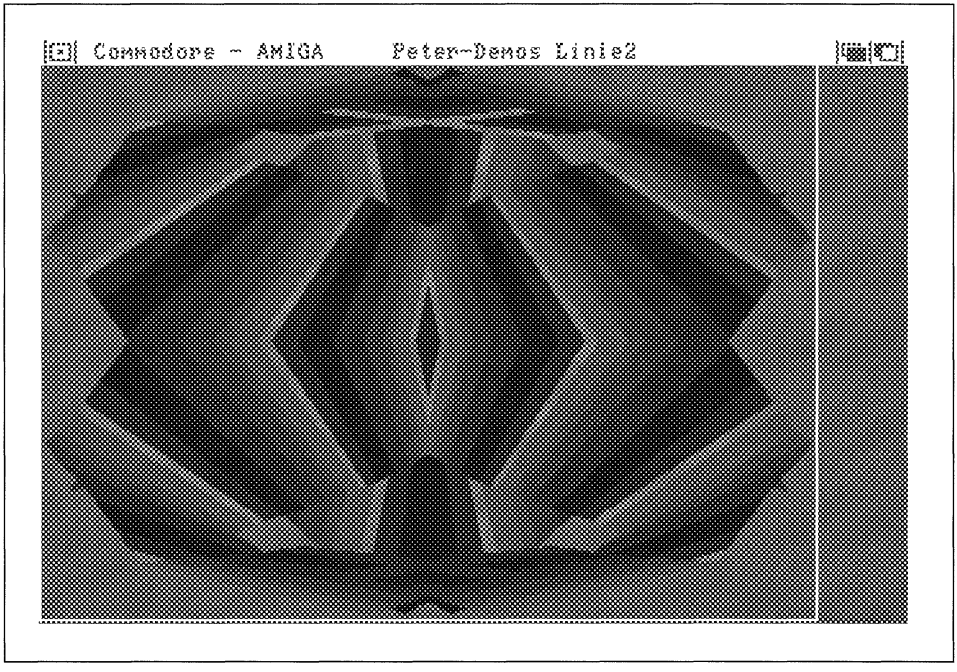



***Bild 4.19:*** *Das Programm Linie2*

Das ist recht langweilig. Sie werden deshalb nun ein Programm starten, das einen zweiten Screen (virtuellen Bildschirm) erzeugt, mit dem Sie dann etwas herumexperimentieren können. Legen Sie dazu die Extras-Diskette in das Diskettenlaufwerk (wenn Sie ein zweites Diskettenlaufwerk besitzen, können Sie die Diskette auch in dieses schieben). Öffnen Sie das Disketten-Piktogramm, sobald es auf der Workbench erscheint (unter dem Namen *Extras1.3D*) und suchen Sie das Piktogramm *Linie2*. Dies ist ein Programm, das Sie starten können, indem Sie es öffnen (zum Beispiel mit einem Doppelklick darauf). Starten Sie dieses Programm nun und warten Sie eine Weile, bis sich auf dem Bildschirm nichts mehr ändert.

Der Bildschirm müßte dann etwa so aussehen wie das folgende Bild. Daß es sich um einen neuen, vom Workbench-Screen verschiedenen Screen handelt, können Sie ganz leicht daran feststellen, daß in den kaleidoskopähnlichen Bildern, die dieses Programm erzeugt, wesentlich mehr als die vier auf der Workbench üblichen Farben vorkommen.

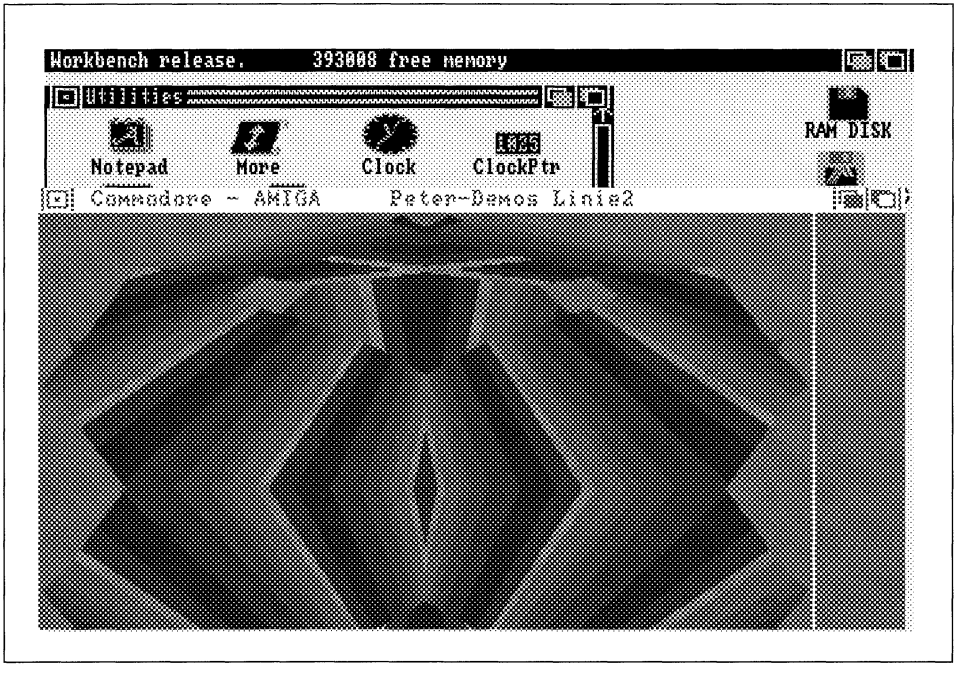



***Bild 4.20:*** *Zwei Screens gleichzeitig am Bildschirm*

Der Umgang mit einem Screen ist ganz einfach. Ein Screen verhält sich im Prinzip genauso wie ein Fenster. Sie können ihn an seiner Titelleiste ergreifen und auf- und abbewegen – jedoch nicht nach links oder rechts. Probieren Sie es aus: Gehen Sie mit dem Mauszeiger an den rechten Rand der Titelleiste. Führen Sie die Spitze des Mauszeigers links neben die beiden Gadgets in der oberen rechten Ecke des Bildschirms, drücken Sie die linke Maustaste und halten Sie sie fest. Wenn Sie nun die Maus auf- und abbewegen, folgt das gesamte sichtbare Bild (mit einer gewissen »Trägheit«) dieser Bewegung und hinter diesem Bild wird der blaue Workbench-Bildschirm sichtbar. Bei diesem Programm müssen Sie leider recht exakt eine Stelle zwischen dem rechten Rand des Kaleidoskops und dem linken Rand der beiden Gadgets treffen, um den Screen zu bewegen. So präzise brauchen Sie meistens nicht zu arbeiten. Bei anderen Screens genügt es, mit der Selektionstaste in die Titelzeile zu klicken, um den Screen zu »ergreifen«.

### 4.6.2 Den »Screen-Stapel« umschichten

Die Reihenfolge, in der Screens »im Stapel übereinanderliegen«, können Sie mit den Back- und Front-Gadgets am rechten Rand der Titelleiste eines Screens ändern. Sie funktionieren

wie die gleich aussehenden Gadgets eines Fensters. Schieben Sie, um das auszuprobieren, den obenliegenden vielfarbigen Screen des Programms Linie2 an den oberen Bildschirmrand und klicken Sie in sein Back-Gadget (das linke der beiden kleinen Symbole am rechten Rand der Titelleiste). Daraufhin verschwindet dieser Screen sofort und der Workbench-Screen kommt nach oben. Auf diesem ist ein zusätzliches Fenster namens *Linie2* zu sehen.

Ergreifen Sie nun auch den Workbench-Screen an seiner Titelleiste mit der Maus und bewegen Sie in auf und ab. Wenn Sie ihn nach unten schieben, müßte darunter wieder der mehrfarbige Screen des Programms *Linie2* zum Vorschein kommen.

Klicken Sie nun in das Back-Gadget des Workbench-Screens (oder in das Front-Gadget des Linie2-Screens) und sofort kommt wieder der Linie2-Screen »nach oben«. Probieren Sie diese beiden Gadgets und das Verschieben der Screens ruhig noch ein bißchen aus, um ein Gefühl dafür zu bekommen. Sie können fast nichts dabei falsch machen. Und wenn Sie einmal nicht mehr weiter wissen sollten oder die Workbench nicht wiederfinden, starten Sie Ihren Amiga 500 einfach neu (mit [Ctrl]+[*A*]+[*A*]). Meistens ist eine solch radikale Maßnahme aber überhaupt nicht nötig. Wenn Sie den Workbench-Screen zum Beispiel nicht wiederfinden können, so liegt das wahrscheinlich daran, daß Sie ihn bis ganz an den unteren Bildschirmrand gezogen haben. Legen Sie in diesem Fall einfach den anderen Screen nach hinten (mit dem Back-Gadget), gehen Sie mit der Maus auf dem Bildschirm ganz nach unten, drücken Sie die Selektionstaste und fahren Sie bei gedrückter Taste langsam nach oben.

In diesem Zusammenhang könnten vielleicht noch zwei weitere Tastenkombinationen wichtig für Sie sein. Diese Tastenkombinationen holen den Workbench-Screen ganz nach vorn oder legen ihn ganz nach unten hinter alle anderen Screens (es können auch mehr als nur ein anderer Screen sein). Diese Tastenkombinationen sind immer dann sehr praktisch, wenn Sie die Titelleiste des Workbench-Screens nicht finden oder wenn ein schlecht programmiertes Programm »seinen eigenen« Screen nach oben legt und dieser keine Titelleiste und kein Back-Gadget hat. Halten Sie in so einem Fall einfach die linke Amiga-Taste (links neben der Leertaste) fest und drücken Sie auf [N]. Der Workbench-Screen kommt dann nach vorn. Genauso können Sie den Workbench-Screen auch wieder nach hinten legen, indem Sie die Amiga-Taste festhalten und dann auf [M] drücken. Probieren Sie diese beiden Tastenkombinationen auch am besten gleich aus – und prägen sie sich ein. (Zur Wiederholung deshalb noch einmal: [*A*]+[N] holt den Workbench-Screen nach vorn und [*A*]+[M] legt ihn nach hinten.) Besonders beim Herumprobieren mit neuen unbekannten Programmen sind diese »Tastentricks« oft die Rettung in letzter Not.

Wenn Sie zum Abschluß Ihrer Experimente das Programm *Linie2* wieder verlassen wollen, klicken Sie einfach einmal in seinen Screen (mit der bewegten Grafik darin). Gehen Sie dann mit dem Mauszeiger in die Titelleiste, drücken Sie die Menütaste und wählen Sie den Befehl Ende aus dem Menü *Symmetrie*. Falls Sie dieses Menü nicht finden sollten, oder wenn Sie es besonders eilig haben, tut es ein Neustart mit [Ctrl]+[*A*]+[*A*] natürlich auch.

Nachdem Sie nun auch die Screens kennengelernt haben, können Sie jetzt mit allen wichtigen »Bausteinen« der Amiga-Software – soweit Sie diese Bausteine sehen können – umgehen. Damit ist der kleine »Amiga-Grundkurs«, der aus den ersten 5 Kapiteln dieses Buches besteht, fast abgeschlossen. Im nächsten Kapitel werden Sie nun noch die restlichen Werkzeuge (Programme) kennenlernen, die Sie beim Kauf des Amiga 500 dazu erhalten.

# 5 Werkzeuge der Workbench

Auf den beiden Disketten, der Workbench-Diskette und der Extras-Diskette, die Sie beim Kauf des Amiga 500 erhalten, befinden sich neben Uhr (*Clock*), Rechner (*Calculator*) und Notizblock (*Notepad*) noch eine ganze Reihe weiterer Werkzeuge (Programme). Mit diesen Programmen wird sich dieses Kapitel schwerpunktmäßig befassen. Als erstes wird das Programm *Preferences* beschrieben – eines der wichtigsten Programme, das Sie als Amiga-Anwender umbedingt kennen müssen.

## 5.1 Preferences – das wichtigste Werkzeug überhaupt

Mit dem Programm *Preferences* (zu deutsch etwa »Vorlieben«; besser könnte man den Namen aber vielleicht mit »Einstellungen« oder »Grundeinstellungen« übersetzen) können Sie viele »grundsätzliche« Eigenschaften der Hard- und Software des Amiga ändern. *Preferences* dient zum Beispiel dazu, Uhrzeit und Datum einzustellen, die Farben zu ändern, in denen Ihnen die Workbench am Bildschirm gezeigt wird, und der Systemsoftware mitzuteilen, welcher Drucker an welchem Stecker angeschlossen ist (das muß man den Programmen nämlich erst extra sagen; sie merken es nicht von selbst).

### 5.1.1 Preferences starten

Sie finden *Preferences* in der Schublade *Prefs* auf der Original-Workbench-Diskette. Diese Schublade ist durch ein kleines Fragezeichen auffällig hervorgehoben. Wenn Sie diese Schublade öffnen, sollten insgesamt fünf Piktogramme sichtbar werden: *Preferences*, *CopyPrefs*, *Pointer*, *Printer* und *Serial*. Alle diese Piktogramme gehören zu *Preferences*. Zunächst einmal wollen wir uns aber nur um das Programm *Preferences* selbst kümmern. Die restlichen Piktogramme werden später noch beschrieben.

*Preferences* können Sie – wie jedes andere Programm auch – starten, indem Sie einen Doppelklick auf das entsprechende Piktogramm machen. Danach erscheint ein bildschirmfüllendes Fenster, das geradezu überläuft von einer Ansammlung der verschiedensten Gadgets. Dies ist das »Hauptfenster« von *Preferences*. Beim Klick auf einige der verschie-

denen Knöpfe in diesem Fenster erscheinen noch bis zu vier andere Fenster, die aber nie gleichzeitig mit dem Hauptfenster zu sehen sind, sondern es immer (vorübergehend) verdecken. Diese Fenster werden in den folgenden Abschnitten noch näher beschrieben.

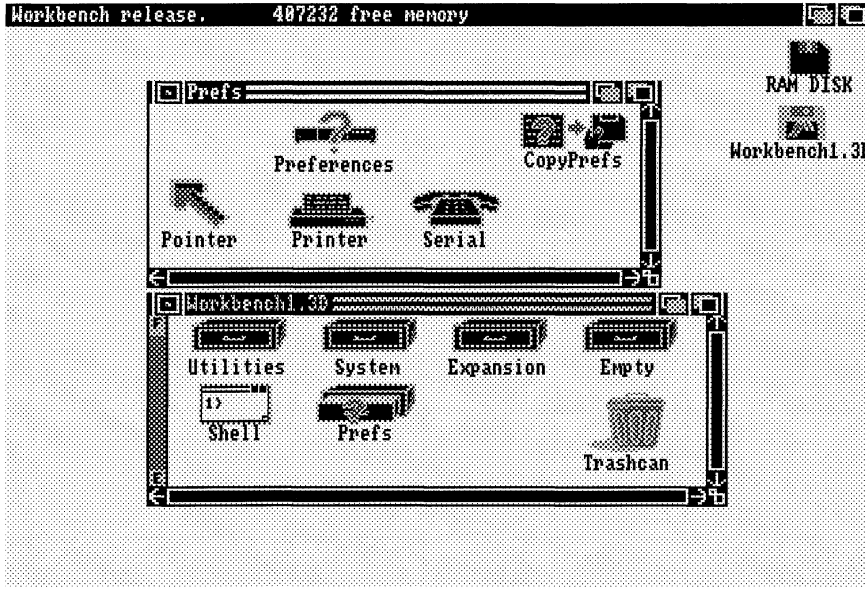



***Bild 5.1:*** *Der Inhalt der Schublade Prefs*

Nachdem Sie *Preferences* aufgerufen haben, können Sie das Programm nur von diesem Fenster aus wieder verlassen. Unten rechts im Fenster ist eine Gruppe von fünf Knöpfen zu sehen: *Change Printer*, *Edit Pointer*, *Save*, *Use* und *Cancel*. Wenn Sie drei davon (mit einem Mausklick) betätigen, so wird *Preferences* verlassen, die anderen Knöpfe öffnen zwei der bereits erwähnten anderen Fenster von *Preferences*.

### 5.1.2 Wichtige Tasten im Preferences-Hauptfenster

Drücken Sie auf *Cancel* (engl. für »Abbrechen« oder »Vergiß es«), so wird *Preferences* verlassen und alle Änderungen, die Sie eventuell zuvor durchgeführt haben, werden ignoriert. Drücken Sie auf *Save*, so werden die Änderungen erstens gültig und zweitens auf Ihrer Workbench-Diskette gespeichert. Sie sind dann auch nach dem nächsten Neustart beziehungsweise Einschalten noch gültig. Betätigen Sie hingegen *Use* (engl. *to use* = verwenden/benutzen), so werden Ihre Änderungen und Einstellungen zwar gültig, sie werden aber nicht auf der Diskette gespeichert. Das heißt, nach dem nächsten Neustart oder Aus- und Einschalten des Amiga 500 gelten wieder die alten Einstellungen, die Sie beim Start von *Preferences* vorgefunden haben.

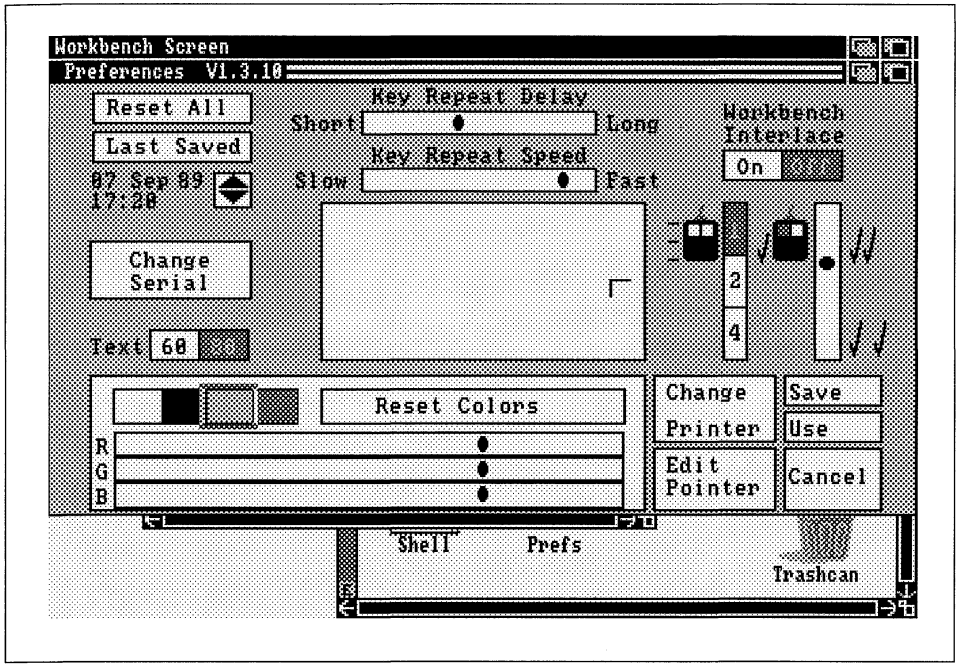



***Bild 5.2:*** *Das Hauptfenster von Preferences*

*Save* funktioniert natürlich nur, wenn der Schreibschutz der Workbench-Diskette (der kleine Plastikriegel in einer Ecke der Diskette) nicht aktiviert ist. Ist das der Fall, erscheint eine Fehlermeldung. Sie können dann die Diskette herausnehmen, den Schreibschutz entfernen (den Plastikriegel verschieben) und die Diskette wieder einlegen. Haben Sie es sich anders überlegt und wollen Ihre Änderungen lieber doch nicht auf der Diskette abspeichern, klicken Sie einfach auf den Knopf *Cancel* in dem Fenster, das Ihnen die Fehlermeldung zeigt.

In der oberen linken Ecke des Hauptfensters befinden sich zwei weitere wichtige Tasten: *Reset All* und *Last Saved*. Wenn Sie *Reset All* (auf deutsch etwa »alles zurücksetzen«) anklicken, gelten für alle Einstellungen (Bildschirmfarben und so weiter), die Sie mit *Preferences* verändern können, Standardwerte, die in der Systemsoftware des Amiga 500 festgelegt sind. Dies sind die Einstellungen, mit denen Sie wahrscheinlich beim ersten Einschalten Ihres Amiga gearbeitet haben. Klicken Sie auf *Last Saved* (auf deutsch etwa »zuletzt gesichert«), so gelten die Einstellungen, die Sie das letzte Mal gesichert haben, als Sie *Preferences* mit einem Klick auf *Save* verlassen haben. Wenn Sie noch nie mit *Preferences* gearbeitet haben, ergeben *Reset All* und *Last Saved* dasselbe Ergebnis – eben die Standardeinstellungen.

### 5.1.3 Ganz nach Geschmack: Die Workbench-Farben

Zu den Einstellungen, die Sie in *Preferences* ändern können, gehören auch die vier Farben, die der Amiga 500 im Workbench-Bildschirm verwendet. Alle Programme, die keine

eigenen Screens erzeugen, müssen mit diesen vier Farben auskommen. Zum Ändern dieser vier Farben dienen die drei »Schieberegler« unten links im Fenster.

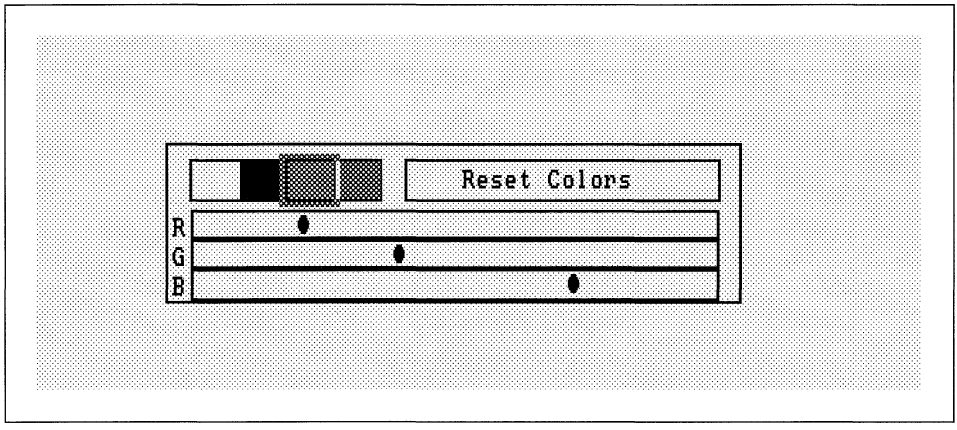



***Bild 5.3:*** *Einstellen der Workbench-Farben*

Wollen Sie die Farben ändern, müssen Sie zunächst eine aussuchen. Dies geschieht, indem Sie eines der vier farbigen Quadrate links neben der Taste *Reset Colors* anklicken. Die so selektierte Farbe wird dann mit einem Rahmen versehen. Sie können dann die kleinen Knöpfe in den Reglern darunter mit der Maus ergreifen (wie ein Piktogramm) und nach links und rechts verschieben, wobei sich die zuvor ausgewählte Farbe sofort ändert. Die Buchstaben *R*, *G* und *B* neben den Reglern stehen für Rot, Grün und Blau. Aus diesen drei Grundfarben werden die am Bildschirm sichtbaren Farben nämlich »zusammengemischt«. Sind alle Regler am rechten Anschlag, ergibt dies die Farbe Weiß. Sind alle am linken Anschlag, entsteht Schwarz. (Auf dieselbe Weise – durch additive Farbmischung aus Rot, Grün und Blau – entstehen übrigens auch die Farben auf Ihrem Farbfernseher.)

Es ist nicht ganz einfach, eine Farbe, die man sich vorstellt, auf diese Weise zusammenzumischen. Ein wenig Experimentieren wird Ihnen aber bestimmt bald ein Gefühl dafür vermitteln. Sie sollten ruhig ein wenig mit den drei Reglern herumspielen! Ähnliche Regler werden in fast jedem Mal- und Grafik-Programm für den Amiga verwendet. Falls es Ihnen gelungen sein sollte, so scheußliche Farben zusammenzumischen, daß Ihre Augen es nicht mehr aushalten oder Ihnen übel wird, können Sie mit dem Knopf *Reset Colors* die Standard-Kombination (Schwarz, Weiß, Orange und Blau) jederzeit wiederherstellen.

### 5.1.4 Wem die Stunde schlägt: Einstellen von Uhrzeit und Datum

Während die Verstellung der Workbench-Farben eher in die Kategorie Spielerei fällt, erfüllt die folgende Funktion von *Preferences* durchaus ernsthafte Aufgaben. Oben links im Fenster (unter den Tasten *Reset All* und *Last Saved*) sehen Sie die aktuelle Uhrzeit und das Datum – oder besser, was der Amiga dafür hält. Wenn Sie Wert darauf legen, daß seine Vorstellung von Zeit und Datum mit der Ihren übereinstimmt, so können Sie die Anzeige mit den beiden Pfeilen daneben korrigieren.

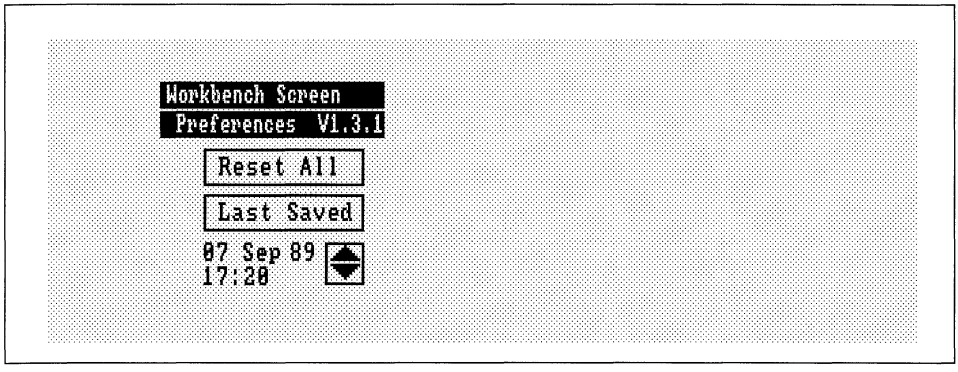



***Bild 5.4:*** *Einstellen von Uhrzeit und Datum*

Klicken Sie dazu erst auf die Ziffer beziehungsweise die Monatsbezeichnung, die Sie verstellen wollen. Danach können Sie diese mit einem Klick auf die danebenstehenden Dreiecke verändern. Ein Klick auf das Dreieck, dessen Spitze nach oben zeigt, erhöht die zuvor selektierte Ziffer um eins, das nach unten zeigende Dreieck macht sie entsprechend kleiner. So können Sie nacheinander alle angezeigten Zahlen auf die korrekten Werte stellen. Wenn Sie damit fertig sind, geht die Uhr Ihres Amiga 500 richtig. Solange Sie ihn eingeschaltet lassen und keinen Neustart machen, behält die Uhr auch die richtige Zeit bei.

Wenn Sie den Amiga 500 jedoch abschalten, »vergißt« er die richtige Uhrzeit – sofern Sie nicht die Speichererweiterung A 501 (oder ein ähnliches Fabrikat von einer Drittfirma) mit integrierter Quarzuhr eingebaut haben. Diese Uhr ist mit Batterie versehen und wird deshalb auch dann mit Strom versorgt, wenn Sie den Netzstecker herausziehen.

### 5.1.5 Das Kleingedruckte: Einstellen von Schriftgröße und Interlace

Die nächste Einstellungsmöglichkeit betrifft die Größe (und damit auch Lesbarkeit) von Text auf dem Bildschirm. Sie können zwischen zwei Größen wählen: 60 Zeichen pro Zeile und 80 Zeichen pro Zeile. Text mit 80 Zeichen pro Zeile ist auf preiswerteren Bildschirmen oder gar auf einem Fernseher meist unscharf und deshalb schwer zu lesen. Sechzigspaltiger Text kann hingegen fast immer scharf dargestellt werden.

Je nachdem, welchen Bildschirm Sie an Ihren Amiga angeschlossen haben, wie nahe Sie vor dem Bildschirm sitzen und wie gut Ihre Augen sind, wird eine dieser beiden Einstellungen ideal für Sie sein. Klicken Sie einfach auf eine der beiden Zahlen neben dem Wort *Text*. Die so ausgewählte Zahl erscheint (sofern Sie nicht an den Farbreglern gespielt haben) schwarz auf orange, während die andere weiß auf blau zu lesen ist. Der Wechsel von einer Schriftbreite zur anderen findet übrigens nicht sofort dann statt, wenn Sie eine der beiden Zahlen anklicken. Sie müssen erst *Preferences* mit einem Klick auf *Save* oder *Use* verlassen, um den neuen Bildschirmaufbau sehen zu können.

Wenn Sie einen sehr guten Bildschirm besitzen (und recht wiederstandsfähig gegenüber Kopfschmerzen sind), können Sie noch mehr Informationen auf den Bildschirm packen als mit dem 80-Zeichen-Modus allein. Oben rechts im Hauptbildschirm von *Preferences*

befinden sich unter der Überschrift *Workbench Interlace* zwei Knöpfe mit den Beschriftungen *On* und *Off* (Ein und Aus). Wenn Sie den Knopf *On* anklicken, *Preferences* dann mit einem Klick auf *Save* verlassen und schließlich auch noch einen Neustart machen, wird die Workbench im sogenannten »Interlace-Modus« gezeigt.

Der Vorteil dieses Modus ist, daß dann doppelt soviele Zeilen auf den Bildschirm passen als im »normalen« Bildschirm-Modus. Dafür flimmert das Bild aber auch verhältnismäßig stark. Dieses Flimmern können Sie aber durch geschickte Wahl der Workbench-Farben reduzieren. (Mehr zum Interlace-Modus, der Grafik-Hardware des Amiga und den Gründen für das starke Flimmern erfahren Sie im letzten Teil dieses Buches.)

Beachten Sie bitte, daß ein Wechsel zwischen Interlace-Modus und dem normalen Bildschirm-Modus nur nach einem Neustart stattfindet (vorher müssen Sie natürlich *Preferences* durch Anklicken von *Save* verlassen. Alle anderen in *Preferences* durchgeführten Änderungen werden dagegen sofort nach Anklicken von *Use* oder *Save* gültig.

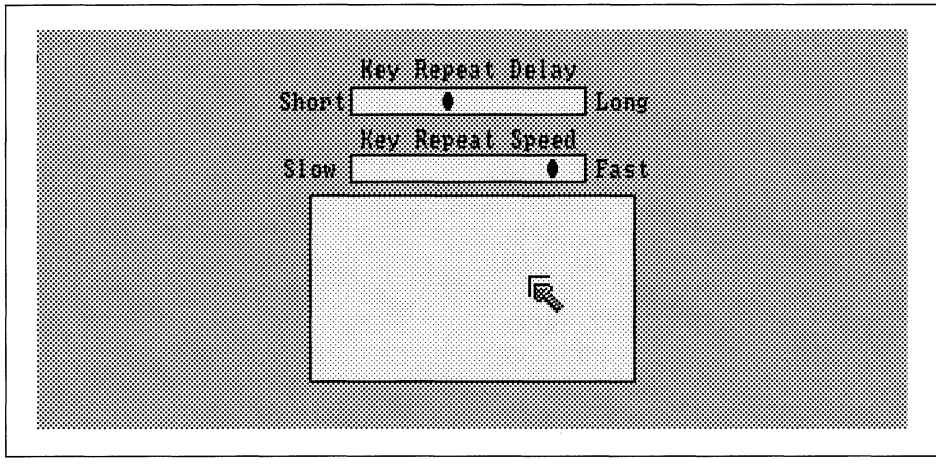



***Bild 5.5:*** *Zentrierung des Bildes mit der Maus*

### 5.1.6 Nichts für Schwindelige: Zentrieren des Bildes

Einen recht spektakulären Effekt können Sie mit dem großen blauen Rechteck in der Mitte des Hauptfensters von *Preferences* erzielen. Dieses Rechteck dient zur Zentrierung des Bildes auf dem Bildschirm. Durch kleine Unterschiede bei den Bauteilen des Bildschirms kann es nämlich leicht dazu kommen, daß das Bild, das der Amiga 500 erzeugt, nicht genau in der Mitte des Bildschirms zu sehen ist. Manchmal sind sogar Teile des Bildes durch den Rand verdeckt.

»Ergreifen« Sie einfach den kleinen weißen Winkel in der blauen Fläche mit der Maus und bewegen sie bei gedrückter Selektionstaste. Dann folgt nicht nur dieser Winkel der Mausbewegungen, sondern das komplette Bild. Auf diese Weise ist es problemlos möglich das Bild innerhalb des Bildschirms dahin zu legen, wo man es haben möchte.

### 5.1.7 Einstellen der automatischen Tastenwiederholung

Mit den beiden »Schiebereglern« am oberen Rand des Hauptfensters können Sie die Geschwindigkeit der automatischen Tastenwiederholung und die Pause bis zu deren Beginn einstellen. Halten Sie eine der Tasten fest, während Sie einen Text eingeben oder bearbeiten, so wirkt dies – nach einer kleinen Pause so – als würden Sie dieselbe Taste schnell hintereinander immer wieder niederdrücken; die Taste wiederholt sich also automatisch.

Die Geschwindigkeit, mit der die simulierten Tastendrücke ablaufen, können Sie mit dem Regler unter der Überschrift *Key Repeat Speed* (Tastenwiederholungsgeschwindigkeit) einstellen. Je näher der Knopf in diesem Regler dem rechten Anschlag (engl. *Fast* = schnell) kommt, desto höher ist die Wiederholungsrate. Der Regler *Key Repeat Delay* (Tastenwiederholungsverzögerung) bestimmt darüber, wie lange Sie eine Taste niederhalten müssen, bis die automatische Wiederholung beginnt. Je näher der Knopf in diesem Regler dem rechten Anschlag (engl. *Long* = lange) kommt, desto länger ist diese Pause.

Wie Sie diese beiden Regler einstellen, ist weitgehend persönlicher Geschmack. Falls Sie aber kein »Zehn-Finger-Tipper« sind, sollten Sie den Regler *Key Repeat Delay* nicht auf zu kleine Werte (nahe bei *Short* = kurz) einstellen, da sonst fast jede Taste gleich zwei Buchstaben erzeugt. Auch zu hohe Einstellungen für *Key Repeat Speed* führen in vielen Programmen zu verwirrenden Effekten und sind deshalb nicht zu empfehlen.

### 5.1.8 Einstellen der »Mausübersetzung«

Zwei andere wichtige Einstellungen ähnlicher Art sind die »Übersetzung« der Mausbewegungen und die Zeitspanne, während der der Amiga zwei Mausklicks als einen Doppelklick auffaßt. Diese beiden Werte können Sie mit den zwei Reglern am rechten Rand des Hauptfensters von *Preferences* einstellen, neben denen jeweils ein kleines Maussymbol zu sehen ist.

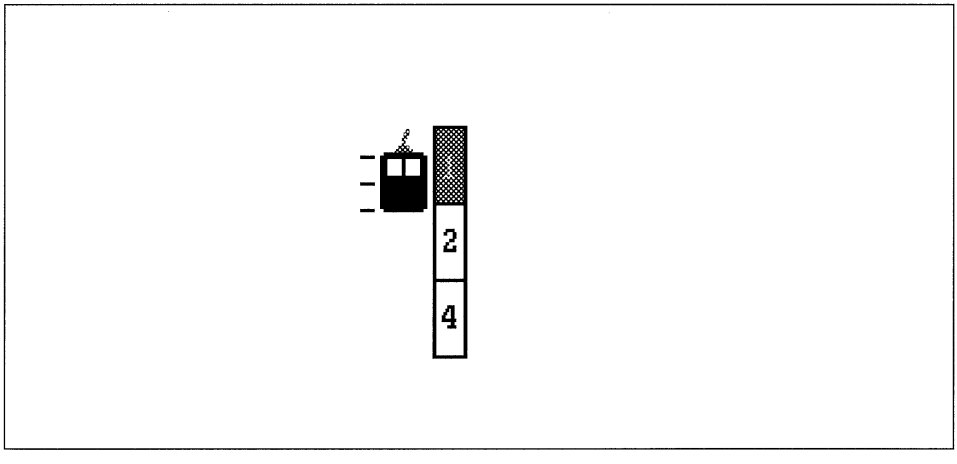



***Bild 5.6:*** *Einstellung der Mausübersetzung*

Die Übersetzung der elektrischen Impulse, die von der Maus kommen, in die Bewegung des Mauszeigers am Bildschirm geschieht in der Systemsoftware des Amiga. Diese Umsetzung kann deshalb auch in der Software (durch *Preferences*) beeinflußt werden. Zu diesem Zweck dient der linke der beiden senkrechten Regler am rechten Bildschirmrand. Sie haben hier die Wahl zwischen den Einstellungen *1*, *2* und *4*, die Sie aktivieren können, indem Sie das entsprechend beschriftete Feld mit der Maus anklicken.

Je höher der Wert ist, den Sie hier einstellen, desto weiter müssen Sie die Maus auf dem Tisch bewegen, um mit dem Mauszeiger eine bestimmte Strecke auf dem Bildschirm zurückzulegen. Ich selbst bevorzuge die Einstellung *1*, weil damit große Strecken »aus dem Handgelenk« zurückgelegt werden können. Wenn Sie mit der Maus noch nicht so gut vertraut sind, sollten Sie die Einstellung *2* wählen. Die Einstellung *4* ist eigentlich nur dann sinnvoll, wenn Sie die Maus sehr präzise bewegen wollen – zum Beispiel in einem Mal- oder Zeichenprogramm.

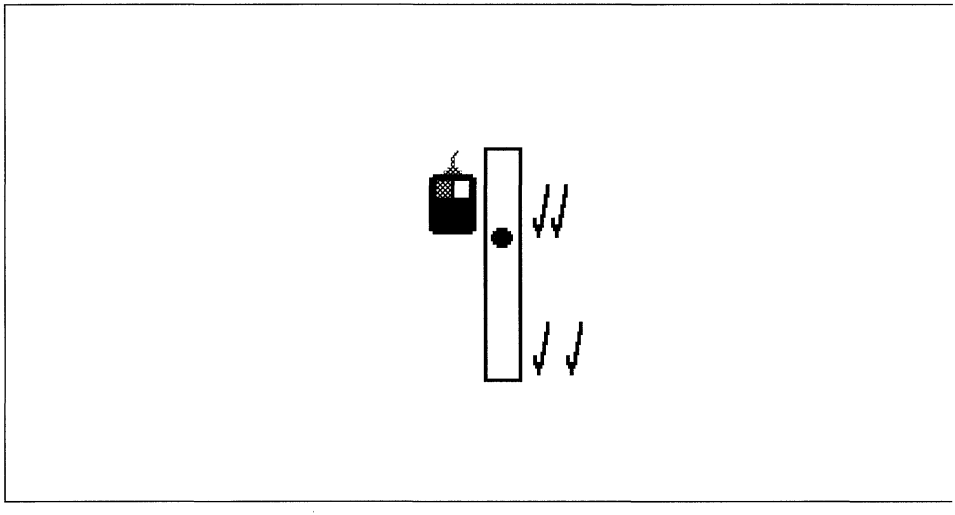


***Bild 5.7:*** *Einstellung der Doppelklick-Zeitspanne*

Für viele Operationen auf der Workbench (besonders für das Öffnen von Schubladen und Starten von Programmen) ist es sehr wichtig, ob zwei Mausklicks, die kurz hintereinander erfolgen, als zwei einzelne Mausklicks oder ein »Doppelklick« aufgefaßt werden. Ist die Zeitspanne, während der die Amiga-Software noch zwei Klicks zu einem Doppelklick zusammenfaßt, zu groß, kann es sehr leicht vorkommen, daß Objekte versehentlich geöffnet werden – was Zeit kostet. Ist diese Zeitspanne zu kurz, kann es besonders für Anfänger sehr schwierig werden, überhaupt einen Doppelklick durchzuführen. Mit dem rechten der beiden Schieberegler am rechten Rand des Hauptfensters von *Preferences* kann sich jeder Anwender diese Zeitspanne so einstellen, wie er es am angenehmsten empfindet. Je näher der Knopf in diesem Regler dem oberen Anschlag kommt, desto kürzer wird der bewußte Zeitraum. Ist dieser Knopf am unteren Rand, kann zwischen den zwei Mausklicks eines Doppelklicks eine größere Zeitspanne liegen. (Eine kurze Zeitspanne ist allerdings nur für geübte Benutzer geeignet.)

### 5.1.9 Ändern des Mauszeigers

Klickt man im Hauptfenster von *Preferences* auf den Knopf mit der Bezeichnung *Edit Pointer* (nahe der unteren rechten Ecke des Fensters), so erscheint ein neues Fenster, in dem man das Aussehen des normalerweise pfeilförmigen Mauszeigers ändern kann. Auch für dieses Fenster gilt, daß man es durch Anklicken von OK verlassen muß, damit die Änderungen wirksam werden. Verläßt man es hingegen mit *Cancel* (= Abbrechen) werden alle Modifikationen am Mauszeiger ignoriert.

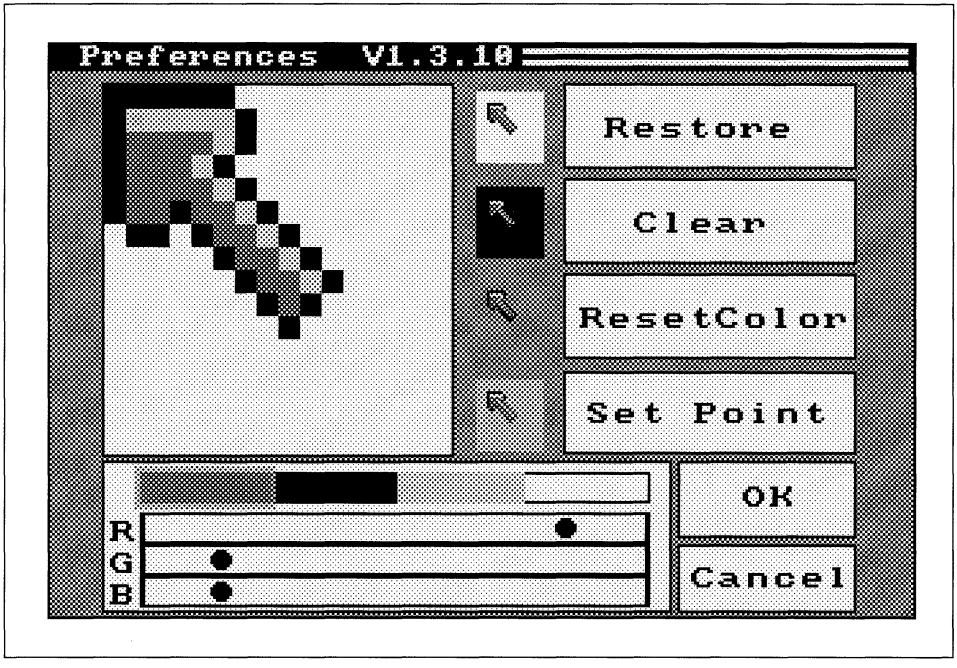



***Bild 5.8:*** *Ändern des Mauszeigers*

In einem großen Rechteck oben links im Fenster können Sie den Mauszeiger Punkt für Punkt ändern. Er wird dazu in etwa achtfacher Vergrößerung gezeigt. Wollen Sie einem Punkt im Mauszeiger eine andere Farbe geben, so klicken Sie diese neue Farbe in der Palette (über den drei Schiebereglern am unteren Rand) einfach an und klicken danach den entsprechenden Punkt in der vergrößerten Darstellung an. Dadurch erhält dieser Punkt die zuletzt gewählte Farbe. Wollen Sie einen Punkt löschen, müssen Sie einfach auf die vierte Farbe in der Palette klicken (normalerweise ist sie blau) und dann den zu löschenden Punkt anklicken. Diese Farbe gilt als »transparent«. Die drei anderen Farben können zur Gestaltung des Mauszeigers verwendet werden.

Dabei sind Sie aber nicht auf die drei anfänglich vorgeschlagenen Farben (Rot, Schwarz und Beige) angewiesen, sondern können sich selbst drei beliebige andere Farben »anmischen«. Die Änderung der Mauszeiger-Farben erfolgt in ähnlicher Weise wie die Änderung der Farben für die Workbench (siehe oben). Sie klicken einfach die Farbe an, die Sie ändern

wollen und verschieben dann die drei Regler am unteren Rand des Fensters. Achten Sie darauf, daß der Mauszeiger immer gut erkennbar ist, egal über welchen Hintergrund er sich befindet. Wählen Sie also keine zu blassen Farben oder Dunkelblau. Wie dieser neue Mauszeiger aussieht, wenn er über den vier möglichen Farben der Workbench erscheint, können Sie rechts neben dem Vergrößerungsfeld gleich kontrollieren. Dort wird der Mauszeiger in vier kleinen Quadraten gezeigt, die mit den entsprechenden Farben gefüllt sind.

Zum Schluß müssen Sie noch den »heißen Punkt« des Zeigers bestimmen. Dieser »heiße Punkt« ist der Punkt, mit dem Sie »wirklich« zeigen, wenn Sie die Maustaste niederdrücken. Der ganze Zeiger ist ja für präzise Positionsbestimmungen zu vage. Beim normalen pfeilförmigen Zeiger ist die Spitze der heiße Punkt. Um diesen Punkt zu verschieben müssen Sie einmal auf den Knopf *Set Point* (engl. für Punkt setzen) klicken und dann in der Vergrößerung den Punkt anklicken, der der heiße Punkt werden soll. Nun brauchen Sie dieses Fenster nur noch durch Betätigung der Taste *OK* verlassen. Sie kehren dann in das Hauptfenster zurück und haben schon einen neuen Mauszeiger.

### 5.1.10 Einstellen der seriellen Schnittstelle

Klickt man im Hauptfenster von *Preferences* auf den Knopf mit der Bezeichnung *Change Serial* (nahe der oberen linken Ecke des Fensters), so erscheint ein neues Fenster, in dem man die Parameter (Eigenschaften) der seriellen Schnittstelle ändern kann. Sie sollten das immer tun, wenn Sie ein neues Gerät an die serielle Schnittstelle des Amiga 500 angeschlossen haben (zum Beispiel einen Drucker oder ein Modem). Legen Sie dazu am besten das Handbuch dieses Geräts neben den Amiga. Die Einstellung einer seriellen Schnittstelle ist eine Kunst, da es dabei sehr viele Möglichkeiten gibt.

Alle Einstellungen in diesem Fenster können Sie mit wenigen Mausklicks erledigen. Entweder sind es Zahlen, dann können Sie diese mit einem Klick auf einen der beiden Pfeile neben der Zahl erniedrigen oder erhöhen. Oder es handelt sich um die Wahl zwischen mehreren sich ausschließenden Möglichkeiten, dann genügt der Klick auf die gewünschte Möglichkeit.

Am wichtigsten ist die Einstellung *Baud Rate* (Baudrate) oben links in diesem Fenster. Die Baudrate bestimmt die Geschwindigkeit der Datenübertragung durch eine serielle Schnittstelle. Übliche Werte für den Anschluß eines Modems sind 300 und 1200 Baud. Sehr schnelle Modems (die leider auch sehr teuer sind – zumindest dann, wenn Sie von der Post zugelassen wurden) schaffen auch 2400 und 4800 Baud. Serielle Drucker erhalten ihre Daten (Buchstaben) meist mit 1200 oder 9600 Baud. Wenn Sie einen anderen Computer direkt über ein kurzes Kabel an den Amiga anschließen, werden Sie meist mit 9600 oder 19 200 Baud arbeiten. Dabei können aber schon recht häufig Übertragungsfehler auftauchen, wenn das zur Datenübertragung verwendete Programm keine Fehlererkennungs- und -Korrekturroutinen verwendet.

Solche Übertragungsfehler kann man unter Umständen durch eine Erhöhung der Puffergröße (*Buffer Size* rechts oben im Fenster) vermeiden. Ein größerer Datenpuffer gestaltet die Datenübertragung oft schneller und fehlerfreier. Besonders, wenn Sie mit einer hohen Baudrate arbeiten und bei *Handshaking* die Option *None* gewählt haben (siehe unten), kann der Puffer gar nicht groß genug sein.

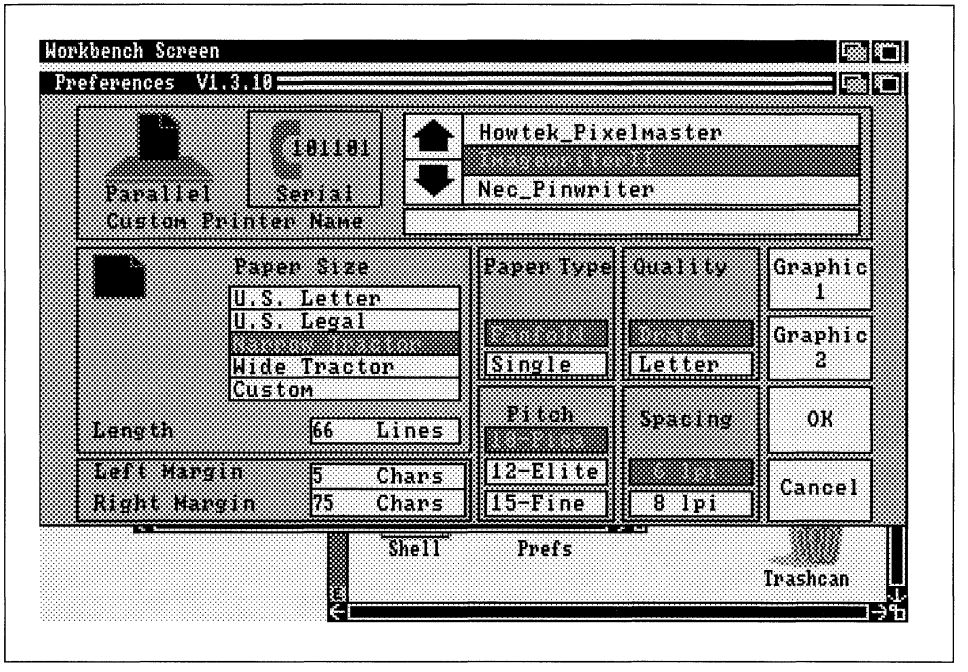



***Bild 5.10:*** *Einstellen der seriellen Schnittstelle*

*Read Bits* und *Write Bits* in der Mitte des Fensters bestimmen, wieviele Bits eines Bytes über die serielle Schnittstelle gesendet (*Write Bits*) und empfangen (*Read Bits*) werden. Da ein Byte aus acht Bit besteht, sollten Sie es bei beiden Einstellungen zunächst einmal mit der Option 8 versuchen. Besonders bei der Datenfernübertragung kann es jedoch Fälle geben, bei denen nur jeweils 7 Bit übertragen werden dürfen. Dies betrifft aber nur den Anschluß eines Modems; moderne Drucker benötigen fast immer 8 Bit. Von wenigen, sehr seltenen, Ausnahmen abgesehen, sollten Sie bei *Read Bits* und *Write Bits* immer dieselbe Einstellung wählen.

Zu den Schaltern *Parity*, *Stop Bits* und *Handshaking* kann man sehr wenig sagen, ohne ausführlich zu beschreiben, wie serielle Datenübertragung vonstatten geht. Sie sollten die nötigen Einstellungen im Handbuch des Gerätes nachschlagen, das Sie an die serielle Schnittstelle angeschlossen haben oder so lange probieren, bis es klappt. Weit verbreitet sind heutzutage die Einstellungen *1* bei *Stop Bits* und *None* bei *Parity*. Bei *Handshaking* sollten Sie immer zunächst *xON/xOFF* oder *RTS/CTS* versuchen, bevor Sie *None* wählen. Die Option *None* erhöht gerade bei hohen Baudraten die Fehlerwahrscheinlichkeit bei der Datenübertragung, die sich nicht immer durch Erhöhung der Puffergröße (siehe oben) verringern läßt.

Wenn Sie alle diese Einstellungen vorgenommen haben oder zumindest einen ersten Versuch gemacht haben, sollten Sie dieses Fenster mit einem Klick auf *OK* verlassen. Ein Klick auf *CANCEL* bringt Sie zwar auch ins Hauptfenster zurück, bewirkt aber – wie üblich – ansonsten nichts.

### 5.1.11 Die Druckeranpassung

Wenn Sie Ihren Drucker an die serielle Schnittstelle des Amiga angeschlossen haben, ist natürlich auch die eben beschriebene Einstellung der seriellen Schnittstelle für den Drucker wichtig. Darüber hinaus benötigt der Amiga aber noch eine ganze Reihe weiterer Angaben, um sich mit einem Drucker einwandfrei verständigen zu können. *Preferences* dient auch dazu, Ihren Amiga an die verschiedensten Drucker anzupassen, die Sie an ihn anschließen können. Nachdem Sie den Drucker mit einem passenden Kabel hinten an Ihren Amiga angeschlossen haben, betätigen Sie dazu im Hauptfenster von *Preferences* die Taste *Change Printer* (engl. für »Drucker wechseln« beziehungsweise »Drucker anpassen«).

Auch für das Druckeranpassungsfenster, das daraufhin auftaucht, gilt, daß man es mit einem Klick auf *Cancel* (abbrechen) oder mit *OK* wieder schließen und die Anpassung damit beenden kann. *Cancel* bedeutet dabei Ignorieren aller Änderungen und *OK* Bestätigen der durchgeführten Anpassung, die dadurch gültig wird.

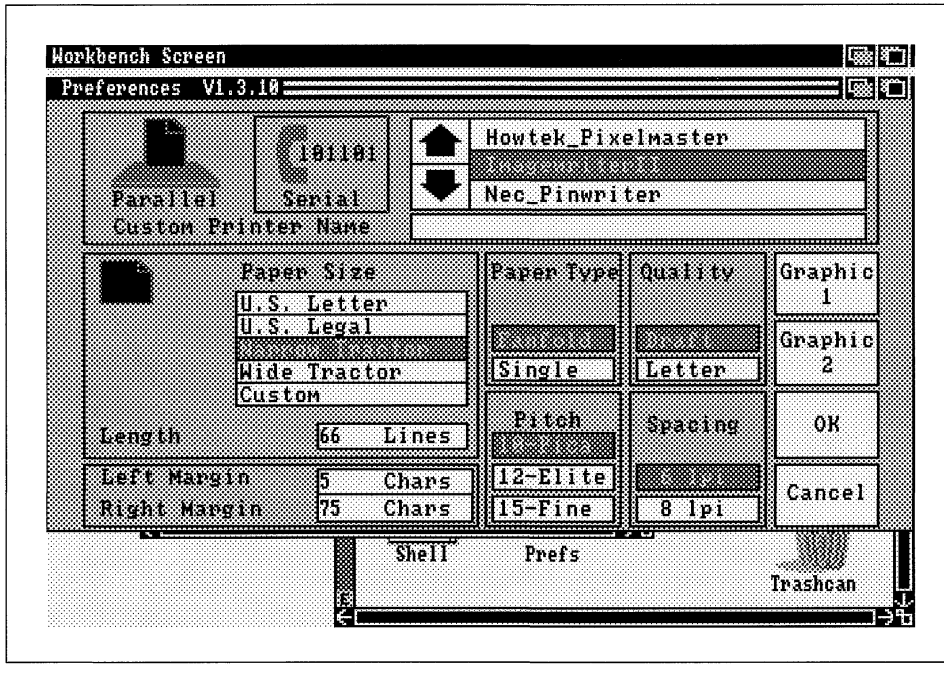



***Bild 5.11:*** *Auswahl des angeschlossenen Druckers*

Welche Einstellungen Sie hier vornehmen müssen, hängt in starkem Maße von dem Drucker ab, den Sie verwenden. Konsultieren Sie deshalb das entsprechende Handbuch, das dem Drucker beilag. Falls Ihnen die darin auftauchenden Fachausdrücke alle oder teilweise nichts sagen, können Sie die Druckeranpassung auch durch Ihren Händler vornehmen lassen und ein für allemal auf Ihrer Workbench-Diskette abspeichern. Sie brauchen sie im allgemeinen nie wieder zu verändern.

Die wichtigsten Einstellungen befinden sich in diesem Fenster am oberen Rand. Links können Sie bestimmen, an welchen Anschluß an der Rückseite des Amiga 500 Sie das Druckerkabel gesteckt haben. Klicken Sie dazu auf das Symbol *Parallel* oder *Serial*. Die beiden Symbole ähneln denen, die unter den entsprechenden Anschlüssen aufgedruckt sind. Rechts daneben müssen Sie angeben, welchen Druckertyp Sie angeschlossen haben. Hierzu müssen Sie solange auf die beiden nach oben und nach unten weisenden Pfeile klicken, bis der Name Ihres Druckers daneben hervorgehoben wird (das ist immer der mittlere der drei Namen).

Bei der aktuellen Version der Workbench-Diskette haben Sie aber leider nur die Wahl zwischen den Druckertypen *Custom* und *generic*. Wenn Sie *generic* wählen, sollte es möglich seine, auf einer Vielzahl von Druckern Texte in einer einzigen Schriftart auszugeben. Der Ausdruck von Grafiken oder von Texten mit unterschiedlichen Schriftarten und Schriftgrößen ist damit nicht möglich. Damit Sie die Fähigkeiten eines Druckers wirklich ausnutzen können, muß es auf Ihrer Workbench-Diskette einen sogenannten »Druckertreiber« geben, der auf den von Ihnen verwendeten Drucker zugeschnitten ist. Es gibt für den Amiga eine ganze Reihe solcher Druckertreiber für nahezu alle gebräuchlichen Druckertypen. Aus Platzgründen konnte man diese Treiber aber nicht mehr auf der Workbench-Diskette unterbringen. Sie befinden sich statt dessen auf der Extras-Diskette.

Um einen Drucker verwenden – beziehungsweise dessen Fähigkeiten voll ausnutzen – zu können, müssen Sie deshalb mit Hilfe des Programms *InstallPrinter*, das später in diesem Kapitel noch beschrieben wird, den gewünschten Druckertreiber von der Extras- auf die Workbench-Diskette kopieren. Die im folgenden beschriebenen Einstellungen können Sie aber auch schon vorher durchführen. Dazu muß nicht unbedingt der richtige Druckertreiber vorhanden sein. Sie sollten aber nie vergessen, noch einmal *Preferences* aufzurufen, nachdem Sie einen neuen Druckertreiber mit *InstallPrinter* auf die Workbench-Diskette kopiert haben. Damit der Amiga den neuen Druckertreiber auch verwendet, muß nämlich dessen Name in der Liste oben rechts im Druckerfenster von *Preferences* an zweiter Stelle stehen und durch Farbumkehrung hervorgehoben werden.

Die Gadgets im unteren Teil des Druckeranpassungsfensters sind nicht ganz so wichtig wie die beiden im oberen Teil. Sie bestimmen, wie der Amiga Text und Grafik ausdruckt. Hierzu müssen aber zunächst einmal die beiden Einstellungen im oberen Teil korrekt sein, sonst funktioniert gar nichts. Unter *Paper Size* können Sie die Größe des Papiers einstellen, die Sie in Ihrem Drucker verwenden. Hierzu gibt es wieder einige (in den USA) übliche Längen zur Auswahl. Das bei uns übliche Papierformat DIN A4 ist leider nicht dabei. Zur Angabe solcher (in den USA) unüblichen Längen gibt es die Option *Custom*. Klicken Sie auf diesen Knopf und dann in das Text-Gadget darunter (links daneben steht *Length*). Tragen Sie dann in diesem Text-Gadget ein, wie lang das von Ihnen verwendete Papier ist. Für normales DIN-A4-Papier müssen Sie *69* eintragen und für das übliche perforierte Endlospapier *72*.

Beim Ausdruck längerer Texte merken Sie sofort, ob diese Längenangabe stimmt oder nicht. Wenn Sie eine zu große Länge eintragen, beginnt der Ausdruck der zweiten Seite zu tief unten auf dem Papier. Bei einer zu geringen Längenangabe beginnt der Text der zweiten Seite zu früh – vielleicht noch auf der ersten Seite des Papiers. Rechts neben der Auswahl der Papiergröße sollten Sie noch auswählen, ob Sie einzelne Blätter (*Single*) oder Endlospapier (*Fanfold*) verwenden. Klicken Sie das entsprechende Gadget an.

Links unten im Druckeranpassungsfenster können Sie angeben, ob Sie beim Ausdruck links und rechts auf dem Papier unbedruckte Ränder (engl. *margins*) haben wollen. Tragen Sie neben *Left Margin* ein, wieviel Platz am linken Rand freibleiben soll, und neben *Right Margin*, in der wievielten Spalte jeder Zeile der Ausdruck spätestens aufhören soll. Beide Angaben müssen in Buchstabenbreiten (*Chars*) gemacht werden, wobei Sie sich ungefähr daran orientieren können, daß auf die volle Papierbreite etwa 80 Buchstaben passen.

Unter der Überschrift *Pitch* können Sie die »Buchstabendichte« bestimmen. Beim Ausdruck von Grafiken bestimmt dies unter Umständen die Breite und die Proportionen des Bildes, beim Ausdruck von Texten die Breite (und manchmal auch das Aussehen) der Buchstaben. *10-Pica* ergibt die breitesten Buchstaben beziehungsweise Bilder und *15-Fine* die schmalsten; *12-Elite* liegt dazwischen.

Genauso können Sie unter *Spacing* die »vertikale Dichte« des Drucks bestimmen. Die Option *6 lpi* (***l**ines **p**er **i**nch* = Zeilen pro Zoll) ergibt 6 Druckzeilen pro Zoll Papierhöhe, während bei *8 lpi* 8 Druckzeilen auf ein Zoll (und damit natürlich auch mehr Zeilen auf eine Seite) passen.

Unter der Überschrift *Quality* können Sie noch weiteren Einfluß auf die Druckqualität nehmen. Wählen Sie *Draft* (engl. für Entwurf) für schnelle Ausdrucke niedriger Qualität und *Letter* für (meist langsamere) Ausdrucke mit höherer Qualität. *Letter* steht dabei für »Near Letter Quality« (NLQ), wozu man auf Deutsch auch »Korrespondenzqualität« sagt.

Ob hinterher wirklich ein Unterschied zwischen *Draft* und *Letter* besteht, hängt von dem verwendeten Drucker ab. Manche Drucker können nur in einer Qualität drucken. Auch, ob die anderen Einstellungen, die Sie unter den Punkten *Pitch* und *Spacing*, machen können, etwas bewirken, hängt in starkem Maße von Ihrem Drucker ab. So haben manche Drucker nur eine Schriftqualität, andere nur eine oder zwei (und nicht drei) Schriftweiten. Genauso bewirken manche Einstellungen zwar bei einigen Druckern etwas beim Ausdruck von Texten, aber nicht beim Grafikdruck. Auch hier gilt: Probieren geht über Studieren – wie immer beim Anschluß zusätzlicher Geräte!

### 5.1.12 Einstellungen für Grafikdruck 1

Vom Druckeranpassungsfenster aus können Sie über die beiden Tasten *Graphic 1* und *Graphic 2* noch in zwei weitere Fenster von *Preferences* gelangen, mit denen Sie den Ausdruck in noch wesentlich größerem Maß beeinflussen können, als das mit den Angaben im Druckeranpassungsfenster möglich ist. Das betrifft vor allem den Ausdruck von Grafiken. Der Druck von Seiten, die lediglich Text enthalten, wird von den in diesen beiden Fenstern gemachten Angaben meist nicht beeinflußt. Eine Ausnahme stellen in dieser Hinsicht lediglich Programme dar, die Text behandeln, als wären die Buchstaben kleine Grafiken. Speziell die leistungfähigeren Textverarbeitungsprogramme sowie die sogenannten Desktop-Publishing-Programme für den Amiga gehen bei der Behandlung von Texten in dieser Weise vor und werden deshalb auch von den Druckereinstellungen betroffen, die »eigentlich« nur für den Grafikdruck gedacht sind.

Wenn Sie einen grafikfähigen Drucker besitzen, sollten Sie sich deshalb die beiden entsprechenden Fenster zumindest einmal anschauen. Sie wissen dann zumindest, daß es diese zusätzlichen Einstellungsmöglichkeiten gibt und welche Beeinflussungen des Aus-

drucks dort möglich sind. Drücken Sie dazu im Druckeranpassungsfenster auf die Taste *Graphic 1*.

In dem Fenster, das daraufhin auftaucht, können Sie insgesamt vier Einstellungen verändern: *Shade*, *Aspect*, *Image* und *Treshold*. Bei *Shade* (engl. für Schattierung oder Farbe) haben Sie drei Möglichkeiten zur Wahl: *Black and White*, *Gray Scale* und *Color*. Die jeweils gültige davon wird durch Farbumkehrung hervorgehoben. *Black and White* (engl. für Schwarzweiß) bedeutet, daß der Drucker beim Ausdruck jedes Bildschirm-Punktes die Entscheidung trifft, ob dieser schwarz oder weiß ausgedruckt wird. Welche Bildschirmfarben schwarz und welche weiß gedruckt werden, können Sie im Detail dann mit dem Schieberegler *Treshold* einstellen.

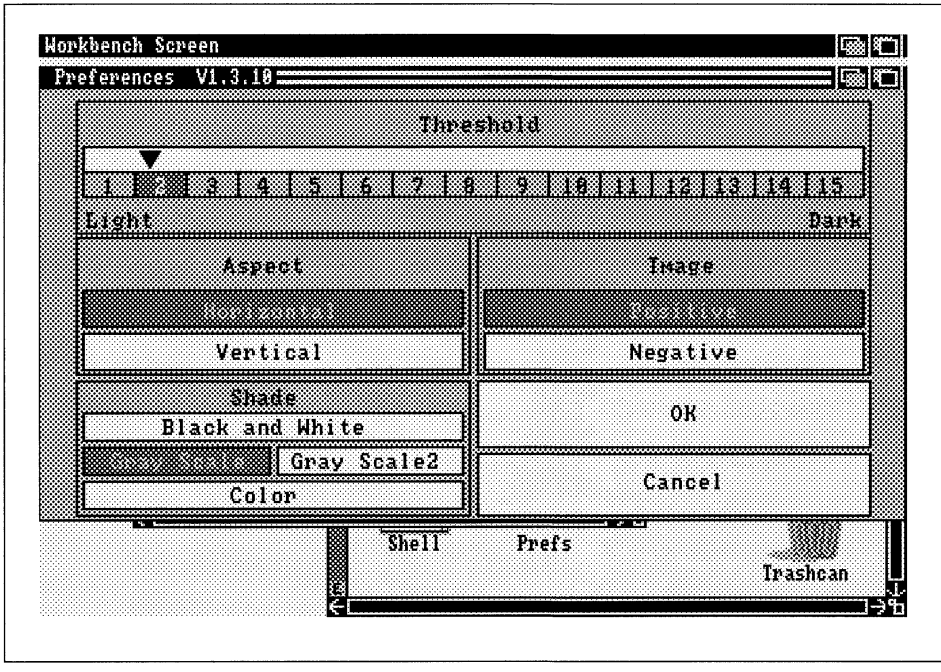



***Bild 5.11:*** *Einstellungen für den Grafikdruck 1*

Bei der Einstellung *Gray Scale* (engl. für Graustufung) versucht der Drucker, die verschiedenen Farben durch Grauschattierungen anzunähern. Hierbei werden aber mehrere Druckerpunkte pro Bildpunkt benötigt, weshalb bei vielen Druckern Schärfe verlorengeht und feine Details verschwimmen. Bei Bildern, bei denen es auf feine Details und Schärfe ankommt, sollten Sie also *Black and White* wählen und etwas mit der Justierung für *Treshold* experimentieren. Wenn Ihnen aber eine Annäherung der Bildschirmfarben wichtiger ist, müssen Sie *Gray Scale* wählen.

Die Einstellung *Color* (Farbe) ist nur dann sinnvoll, wenn Sie einen farbfähigen Drucker angeschlossen und in der Druckeranpassung korrekt ausgewählt haben. Die Farben auf dem Bildschirm erscheinen dann auch wirklich als verschiedene Farben auf dem Papier. Sie

sollten jedoch selbst bei einem guten (und teuren) Farbdrucker nicht erwarten, auch auf Papier ein ebenso brillantes Bild zu erhalten, wie es auf dem Bildschirm zu sehen ist. Die heutige Drucker-Technologie ist dazu leider in keiner Weise fähig. Zudem leuchtet der Bildschirm selbst, während die Farbe auf dem Papier nur Licht reflektiert, was immer einen anderen Effekt ergibt.

*Aspect* (Orientierung des Ausdrucks) bestimmt, ob ein Bild so ausgedruckt wird, wie es im Bildschirm zu sehen ist (*Vertical*), oder ob es, um 90° gedreht, quer aus dem Drucker kommt (*Horizontal*) – was bei großen Bildern, die breiter als hoch sind, durchaus sinnvoll sein kann.

Unter der Überschrift *Image* können Sie zwischen positivem (*Positive*) und negativem (*Negative*) Ausdruck wählen. Wird positiver Audruck gewählt, so werden helle Flächen bei einem Ausdruck in *Black and White* auch hell gedruckt (weiß oder hellgrau) und dunkle dunkel. Bei negativem Ausdruck werden schwarze und weiße (beziehungsweise hellgraue und dunkelgraue) Punkte ausgetauscht. Farbige Ausdrucke beeinflußt *Image* nicht.

Der Schieberegler *Treshold* (Schwarzweiß-Grenze) bestimmt bei Schwarzweiß-Druck, welche Farben weiß und welche schwarz ausgedruckt werden. Je näher der Knopf in diesem Schieberegler dem linken Anschlag kommt, desto weniger (nur sehr dunkle) Farben werden schwarz auf das Papier gedruckt, desto heller wird also das gesamte Bild. Je weiter der Regler nach rechts geschoben wird, desto hellere Farben werden noch schwarz ausgedruckt und um so weniger Farben werden weiß ausgegeben. Das Bild wird also insgesamt dunkler. Dies gilt zumindest, wenn die Einstellung für Image positiv ist. Ist sie negativ, bedeuten kleine Werte für *Treshold* ein dunkles Bild und große ein helles. Besitzen Sie einen grafikfähigen Schwarz-weiß-Drucker, so müssen Sie wahrscheinlich eine Weile mit *Treshold* experimentieren, um zufriedenstellende Ausdrucke von Bildern mit mehr als zwei Farben zu erreichen. Für unterschiedliche Bilder können sogar verschiedene *Treshold*-Werte nötig sein. Farbige Ausdrucke werden von *Treshold* nicht beeinflußt.

### 5.1.13 Einstellungen für Grafikdruck 2

Sollten Sie alle diese Einstellungsmöglichkeiten verwirrend finden, können Sie sich damit trösten, daß das fast jedem so geht, der sich das erste Mal mit diesem Thema beschäftigt. Man bekommt sehr leicht die Bedeutungen der verschiedenen Einstellungen heraus, wenn man einige Probedrucke mit unterschiedlichen Einstellungen ausprobiert.

Dasselbe gilt auch für die Einstellungen, die Sie im dritten Druckeranpassungsfenster vornehmen können, das erscheint, wenn Sie die Taste Graphic 2 im ersten Druckeranpassungsfenster anklicken. In diesem Fenster können Sie vor allem festlegen, wie Farben ausgedruckt werden und mit welcher Qualität Linien und bestimmte Texte zu Papier gebracht werden. Ich werde mich im wesentlichen darauf beschränken, die Konsequenzen zu beschreiben, die unterschiedliche Einstellungen beim Ausdruck haben. Eine Erklärung der Gründe für diese Unterschiede würde – im Rahmen dieses Buches – sicherlich zu weit gehen. (Dafür müßten zuvor sehr viele Grundlagen der Computergrafik erläutert werden.)

#### Smoothing

Unter der Überschrift *Smoothing* links oben im Fenster können Sie eine Art Kippschalter ein- und ausschalten (indem Sie auf ON oder OFF klicken). Ist *Smoothing* an, so werden diagonale Linien beim Ausdruck oft »glatter« aussehen. Oder anders formuliert: die Punkte,

aus denen der Drucker diese Linien aufbaut und die bei schrägen Linien in Form kleiner Treppchen deutlich zutage treten, werden kleiner und die Linie wird dadurch etwas geglättet (engl. *smooth* = glatt/sanft). Dafür dauert der Ausdruck einer Seite aber auch doppelt so lange! Auch die schrägen Linien in großen Buchstaben können unter Umständen besser aussehen, wenn Sie *Smoothing* anschalten. (Bei einigen Programmen – zum Beispiel vielen Malprogrammen – hat diese Einstellung leider keine Auswirkungen.)

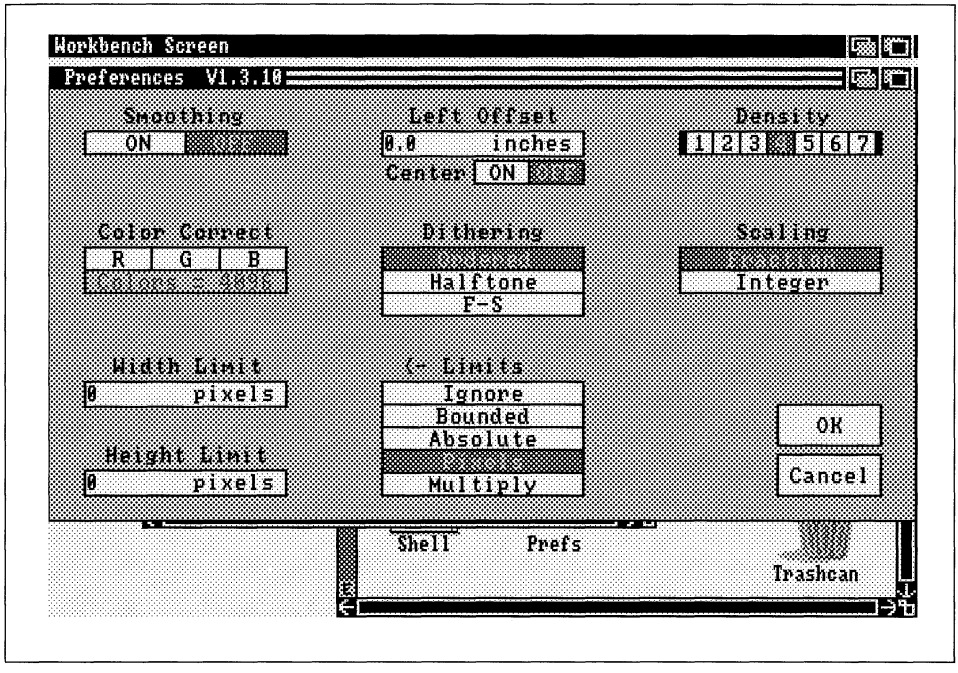



***Bild 5.12:*** *Einstellungen für den Grafikdruck 2*

## Density

Unter der Überschrift *Density* (Dichte) in der rechten oberen Fensterecke können Sie die gewünschte »Druckdichte« zwischen *1* und *7* festlegen. Diese Einstellung darf nicht mit der Schriftdichte verwechselt werden, die Sie unter der Überschrift *Pitch* im ersten Druckeranpassungsfenster festlegen können. *Pitch* betrifft vor allem die Breite der ausgedruckten Buchstaben und damit auch, wieviele Buchstaben in eine Zeile passen. *Density* hingegen bezieht sich auf die Druckdichte von Grafiken und damit vor allem um die Größe der Punkte, mit denen der Drucker eine Grafik zu Papier bringt. Als Faustregel kann man sagen, daß Grafiken um so besser aussehen, je größer die gewählte Druckdichte ist. Bei vielen Nadeldruckern werden farbige (auch schwarze) Flächen erst dann richtig »satt«, wenn man eine größere Druckdichte wählt. Vorher wirken Sie blaß und flau. Der Nachteil einer höheren Druckdichte ist aber üblicherweise, daß man auf den Ausdruck länger warten muß. Also: Druckdichte (*Density*) 1 ergibt sehr schnell einen Ausdruck niedriger Qualität; Druckdichte 7 ergibt einen Ausdruck mit der höchstmöglichen Qualität, auf den man dann aber

auch am längsten warten muß. Die wenigsten Drucker können aber mit sieben verschiedenen Druckdichten arbeiten. Meist sind nur zwei oder drei verschiedene Druckdichten möglich; alle anderen Einstellungen von *Density* ergeben dieselben Ergebnisse, wie eine der von der Druckerhardware unterstützten Druckdichten.

### Color Correct und Dithering

Die Einstellungen unter den beiden Überschriften *Color Correct* und *Dithering* (in der Mitte des Fensters) erlauben es, festzulegen, wie der Amiga beim Ausdruck die Bildschirmfarben simuliert. Um zu verstehen, wieso überhaupt eine solche Simulation nötig ist, muß man wissen, daß fast alle Drucker nur mit maximal vier Farben arbeiten. Der Amiga kann aber bis zu 4096 verschiedenen Farben am Bildschirm darstellen. Zur »Vortäuschung« einer größeren Anzahl von Farben setzt der Drucker deshalb mehrere Punkte in jeweils einer der vier Druckerfarben dicht nebeneinander, bildet also eine Art mehrfarbiges Muster. Diese Muster verschwimmen dann – bei größerem Betrachtungsabstand – zu neuen (Misch-) Farben, die der Drucker »eigentlich« gar nicht beherrscht. Auf ähnliche Art und Weise kann auch ein reiner Schwarzweißdrucker zumindest Grautöne simulieren, indem er verschieden große Anteile einer Fläche mit schwarzen oder weißen Punkten füllt.

Es gibt eine ganze Reihe verschiedener Methoden, aus wenigen Grundfarben »Muster« zu bilden, mit denen weitere Mischfarben simuliert werden. Sie können zwischen dreien dieser Methoden wählen, indem Sie einen der drei Schalter unter der Überschrift *Dithering* aktivieren. Welche der drei entsprechenden Methoden die »beste« ist, kann man nicht eindeutig sagen. Das ist zum großen Teil Geschmackssache und hängt zum Teil auch vom jeweils verwendeten Drucker ab.

Einfache Nadeldrucker dürften die besten Ergebnisse oft mit der Methode *Ordered* erreichen. Wenn Sie hingegen einen Laserdrucker, Tintenstrahl- oder einen hochauflösenden Thermotransferdrucker verwenden, werden die Druckergebnisse nach der Methode *Halftone* und *F-S* meist besser aussehen.

Allerdings kann eine Methode bei dem einen Bild und eine andere bei einem anderen Bild »optimal« sein. Zudem hat die *F-S*-Methode den Nachteil, daß Sie nicht mit dem linienglättenden Smoothing kombiniert werden kann. (Sobald *F-S* gewählt wird, geht der Schalter unter Smoothing automatisch auf *OFF*.) Auch hier bleibt Ihnen nichts anderes übrig, als alle Möglichkeiten einmal auszuprobieren.

Das gilt auch für den Punkt *Color Correct*. Wenn Sie hier einen oder mehrere der drei Schalter *R*, *G* und *B* (die für Rot, Grün und Blau stehen) aktivieren, führt der Amiga bei Ausdruck farbiger Grafiken auf einem farbfähigen Drucker eine »Farbkorrektur« durch. Diese Farbkorrektur kann getrennt für jede der drei Komponenten Rot, Grün und Blau durchgeführt werden, aus denen die jeweilige Farbe auf dem Bildschirm »zusammengesetzt« wird. Sie sollte im Idealfall dafür sorgen, daß die auf dem Papier sichtbaren Farben den Bildschirmfarben besser entsprechen, als dies ohne Korrektur der Fall wäre. Ob das wirklich der Fall ist oder ob nicht sogar die nichtkorrigierten Farben besser aussehen, kann man wiederum nur durch Ausprobieren herausfinden.

Die Wirkung der restlichen, in diesem Fenster noch möglichen Einstellungen ist hingegen recht gut vorhersehbar und kontrolliert im wesentlichen, wie groß die ausgedruckte Grafik wird und wo sie auf dem Papier erscheint.

### Left Offset

In dem Textgadget unter der Überschrift *Left Offset* können Sie beispielsweise eintragen, welchen Abstand der linke Rand der Grafik zum linken Rand des Papiers haben soll. Sie müssen diesen Abstand in Zoll (*inches*) angeben. Die Umrechnung in Zentimeter ist allerdings nicht allzu schwierig: 1 Zoll = 2,54 Zentimeter. Wenn Sie sich das Kopfrechnen ersparen wollen, klicken Sie am besten einfach auf den Schalter *ON* unter dem Textgadget *Left Offset*. Die Grafik wird dann bei Ausdruck auf dem Papier zentriert.

### Limits

Mit den Einstellungen im unteren Drittel des Fensters können Sie Einfluß darauf nehmen, wie groß die Grafiken beim Ausdruck werden. Dazu müssen Sie zunächst einmal eine der fünf Möglichkeiten unter der Überschrift *Limits* (Grenzen) wählen. Solange *Ignore* eingeschaltet (durch Farbumkehrung hervorgehoben) ist, überlassen Sie die Auswahl der Größe dem Amiga beziehungsweise der Druck-Software. Wählen Sie eine der anderen vier Möglichkeiten, können Sie in den beiden Textgadgets *Width Limit* und *Height Limit* (neben den Limits-Schaltern) Angaben zur Höhe und/oder Breite des Ausdrucks machen.

Wählen Sie zum Beispiel *Bounds* (Grenzen), so können Sie unter *Width Limit* die maximale Breite und unter *Height Limit* die maximale Höhe des Bildes (in Zoll) eingeben. Wohlgemerkt: es handelt sich dabei um Maximalwerte; das Bild kann also nicht größer jedoch kleiner werden. Anders verhält es sich, wenn Sie die Option *Absolute* wählen. Dann versucht der Amiga, das Bild exakt so groß auszudrucken, wie Sie es in den beiden Textgadgets *Width Limit* und *Height Limit* angegeben haben. Genau das Selbe bewirkt auch die Option *Pixels*. Der Unterschied zwischen *Absolute* und *Pixels* ist lediglich der, daß im ersten Fall (*Absolute*) die gewünschte Höhe und Breite in Zoll (Inch) und im zweiten Fall in Pixels (Druckerpunkten) angegeben werden muß. Die letzte Option, *Multiply*, bewirkt schließlich, daß das Bild beim Ausdruck exakt um ein ganzzahliges Vielfaches seiner ursprünglichen Größe vergrößert wird.

Wenn Sie nur die Breite oder nur die Höhe des Bildes festlegen beziehungsweise begrenzen wollen, brauchen Sie nur in einem der beiden Textgadgets *Width Limit* und *Height Limit* die gewünschte Zahl und in dem anderen eine Null eintragen. Der Amiga errechnet die zweite Zahl dann selbsttätig. Diese Zahl wird so gewählt, daß das Bild beim Ausdruck seine Proportionen behält.

Die Wirkung der Vielzahl von Optionen unter der Überschrift *Limits* kann man am besten mit einem Beispiel verdeutlichen. Angenommen, Sie wollen ein Bild mit den Schubladen, Fenster und Diskettenpiktogrammen der Workbench ausdrucken (ein Programm für genau diesen Zweck wird gleich noch beschrieben), dann besteht dieses Bild aus 256 Zeilen mit je 640 Punkten (sofern Sie im Hauptfenster von *Preferences* den Interlace-Modus nicht aktiviert haben). Wenn Sie wollen, daß dieses Bild mit genau doppelt so vielen Druckerpunkten auf das Papier kommt, müssen Sie die Option *Multiply* wählen und in den beiden Textgadgets *Width Limit* und *Height Limit* jeweils eine *2* eintragen. Dabei wird das Bild jedoch auf dem Papier merkwürdig in die Breite gezogen erscheinen.

Das liegt daran, daß der Drucker mit annähernd quadratischen Punkten arbeitet, wohingegen die von der Workbench auf dem Bildschirm verwendeten Punkte doppelt so hoch wie breit (also nicht quadratisch) sind. Um diesen Unterschied zu kompensieren, können Sie im Feld *Height Limit* eine *4* eintragen. Die Proportionen stimmen dann wieder – sofern der Drucker

wirklich quadratische Punkte verwendet. Ein identisches Ergebnis sollten Sie auch dann erhalten, wenn Sie unter Limits die Option Pixels wählen und dann unter *Width Limit* 1280 (640*2) und unter *Height Limit* 1024 (256*4) eintragen.

Wenn Sie sich nicht um Punkte kümmern wollen, sondern eine exakte Größe benötigen, müssen Sie die Option *Absolute* wählen und die gewünschte Größe in Zoll angeben. (Ein Zoll entspricht 2,54 Zentimetern.)

Dabei kann es allerdings zu unschönen Effekten kommen, speziell dann, wenn ein Bild feinstrukturierte Muster enthält. Diese »unschönen Effekte« – Moirés und Unregelmäßigkeiten – treten immer dann auf, wenn das Verhältnis zwischen Bildschirmpunkten und Druckerpunkten nicht ganzzahlig ist. Solange die Druck-Software jeden Bildschirmpunkt auf einen, zwei, vier oder mehr Druckerpunkte abbilden kann, gibt es keine Probleme. Die Ursachen dafür zu beschreiben, würde an dieser Stelle aber zu weit führen.

Um solche Effekte zu vermeiden, sollten Sie entweder immer mit der Option *Multiply* arbeiten oder die Option *Bounds* wählen und unter der Überschrift *Scaling* den Schalter *Integer* aktivieren. Mehr zum Thema *Scaling* erfahren Sie im nächsten Abschnitt.

Zuvor noch ein Tip, wie Sie möglichst große Ausdrucke erhalten können: Wählen Sie die Option *Absolute* und geben Sie in den beiden Textgadgets *Width Limit* und *Height Limit* jeweils ein Null ein. Die Druck-Software gibt dann alle Bilder so breit wie möglich aus und justiert die Druckhöhe so, daß die Proportionen (Verhältnis Höhe zu Breite) des ursprünglichen Bilds erhalten bleiben. Dabei können jedoch wieder die eben schon beschriebenen unschönen Effekte und Verzerrungen auftreten. Wenn Sie das vermeiden wollen, müssen Sie die Option *Multiply* anwählen und ebenfalls in den beiden Textgadgets *Width Limit* und *Height Limit* jeweils eine Null eintragen. In den meisten Fällen wird das dann ausgedruckte Bild aber etwas kleiner sein, als das mit *Absolute* ausgedruckte.

## Scaling

Ist der Schalter *Integer* (ganzzahlig) unter der Überschrift *Scaling* eingeschaltet, sorgt die Druck-Software automatisch dafür, das ein Bildschirmpunkt immer in eine ganzzahlige Anzahl von Druckerpunkten übersetzt wird – solange Sie der Software die Freiheit lassen, über die Abmessungen des Ausdrucks zu entscheiden. Die Freiheit besteht nur dann, wenn Sie unter der Überschrift *Limits* die Optionen *Ignore* oder *Bounds* wählen. Alle anderen Optionen unter Limits legen ja die Größe des Ausdrucks exakt fest. (Bei Benutzung der Option *Multiply* wird ja sowieso dafür gesorgt, daß aus einem Bildschirmpunkt immer eine bestimmte Anzahl von Druckerpunkten wird; siehe oben.)

Wenn Sie wollen, daß die Größe des ausgedruckten Bildes stufenlos verändert werden kann, müssen Sie den Schalter *Fraction* unter der Überschrift *Scaling* aktivieren.

Beachten Sie bitte, daß die Einstellungen, die Sie unter den Punkten *Limits* und *Scaling* vornehmen können, unter Umständen zu Bildern führen, die von Ihren Proportionen her nicht mit dem Aussehen des Bildes am Bildschirm übereinstimmen. Mit der *Limits*-Option *Absolute* können Sie zum Beispiel ohne weiteres ein Ausdruckformat erzwingen, bei dem ein Bild höher als breit ist, obwohl es auf dem Bildschirm breiter als hoch war. Derselbe Effekt kann natürlich auch mit den Optionen *Pixels, Multiply* und *Bounded* erreicht werden, mit denen ebenfalls ein Strecken beziehungsweise Stauchen des ausgedruckten Bildes möglich ist. Sie können solche groben Verzerrungen vermeiden, indem Sie in eines der beiden Textgadgets *Width Limit* und *Height Limit* eine Null eintragen und nur die Höhe oder

Breite festlegen. Die Druck-Software legt dann die Abmessung auf der anderen Bildachse automatisch so fest, daß es nicht zu groben Verzerrungen der Proportionen kommen kann.

Zu Proportionsänderungen kleineren Umfangs kann es jedoch immer dann kommen, wenn Sie unter *Scaling* die Option *Integer* aktivieren und einen Drucker verwenden, der nicht mit quadratischen Punkten arbeitet. Die Ursache dafür ist leicht einzusehen: Angenommen, die Punkte mit denen Ihr Drucker arbeitet, haben in der horizontalen einen Abstand, der 20% größer ist als der vertikale Punktabstand. Wenn dann ein Bild so ausgedruckt wird, daß aus einem Bildschirmpunkt immer genau ein Druckerpunkt wird, ist dieses Bild selbstverständlich auch 20% zu breit. Das könnte die Druck-Software des Amiga automatisch ausgleichen (und tut es auch), wenn Sie die Option *Bounded* oder *Ignore* unter der Überschrift *Limits* und die Option *Fraction* unter der Überschrift *Scaling* anwählen. Nur dann hat die Druck-Software die Freiheit, die Höhe oder Breite des gedruckten Bildes zu ändern, um die Unterschiede zwischen Bildschirm- und Druckerpunkten auszugleichen.

Andererseits wird es bei dieser Anpassung oft zu den schon erwähnten Moiré-Mustern (und anderen unschönen Effekten) in den ausgedruckten Bildern kommen. Wenn Sie das als unangenehmer empfinden als kleine Proportionsänderungen, sollten Sie auf die automatische Proportionsanpassung, die bei *Fraction* möglich ist, verzichten und statt dessen besser die Option *Integer* aktivieren!

## 5.2 Werkzeuge in der Utilities-Schublade

Neben *Preferences* enthalten die Workbench- und die Extras-Diskette aber noch eine ganze Reihe weiterer »kleiner« Werkzeuge, die Sie zum Teil vielleicht einmal gebrauchen werden, vielleicht aber auch nicht. Diese Programme werden in den folgenden Abschnitten dieses Kapitels zumindest alle kurz beschrieben werden, wobei die Programme in der *Utilities*-Schublade auf der Workbench-Diskette den Anfang machen.

### 5.2.1 Alte Bekannte: Clock, Notepad Calculator

Drei dieser Programme kennen Sie ja bereits. Es sind *Clock*, die Schreibtischuhr, *Notepad*, eine einfache Textverarbeitung und *Calculator*, der »Taschenrechner« des Amiga. Diese drei Programme gehören in die Kategorie des nützlichen Zubehörs. Ihr Funktionsumfang ist nicht gerade überwältigend – dafür sind sie umsonst. Je nachdem, was Sie mit Ihrem Amiga machen wollen, werden Sie das eine oder andere dieser Programme aber vielleicht häufiger benutzen. *Notepad* eignet sich zum Beispiel sehr gut für die Bearbeitung kurzer Texte und Notizen (engl. *notepad* = Notizblock), selbst wenn das Programm keine ernsthafte Konkurrenz für die meisten der auf dem Markt erhältlichen professionellen Textverarbeitungsprogramme darstellt.

### 5.2.2 Einen Drucker installieren: InstallPrinter

Ein besonders wichtiges Werkzeug in der Utilities-Schublade ist das Programm *InstallPrinter*. Dieses Programm müssen Sie zumindest einmal verwenden, bevor Sie von Ihrem neuen Amiga aus drucken können. *InstallPrinter* hat die Aufgabe, einen der sogenannten

»Druckertreiber« (siehe Abschnitt 4.1.11), die sich auf der Extras-Diskette befinden, auf die Workbench-Diskette zu kopieren. Dieses Programm ist erst seit der Workbench-Version 1.3 nötig. Bei den früheren Workbench-Versionen befanden sich alle Druckertreiber auf der Workbench-Diskette und der gewünschte konnte mit dem Programm *Preferences* ausgewählt werden. Inzwischen ist die Zahl der Druckertreiber aber angestiegen und gleichzeitig auch der Umfang der restlichen, auf der Workbench-Diskette ausgelieferten Software angewachsen. Deshalb befinden sich jetzt nahezu alle Druckertreiber (einer für jeden Druckertyp, den der Amiga benutzen kann) auf der Extras-Diskette und Sie müssen einen davon (den für Ihren Drucker geeigneten) mit *InstallPrinter* auf der Workbench-Diskette installieren.

Starten Sie dazu *InstallPrinter* mit einem Doppelklick auf das entsprechende Piktogramm in der *Utilities*-Schublade. Daraufhin öffnet sich zunächst ein großes Fenster und dann werden Sie – mit einem Requester – dazu aufgefordert, die Extras-Diskette einzulegen. Wenn Sie zwei Diskettenlaufwerke besitzen, sollten Sie jetzt am besten die Extras-Diskette in das zweite externe Laufwerk einlegen. Ansonsten bleibt Ihnen nichts anders übrig, als die Workbench-Diskette aus dem internen Laufwerk des Amiga herauszunehmen und dann die Extras-Diskette einzulegen.

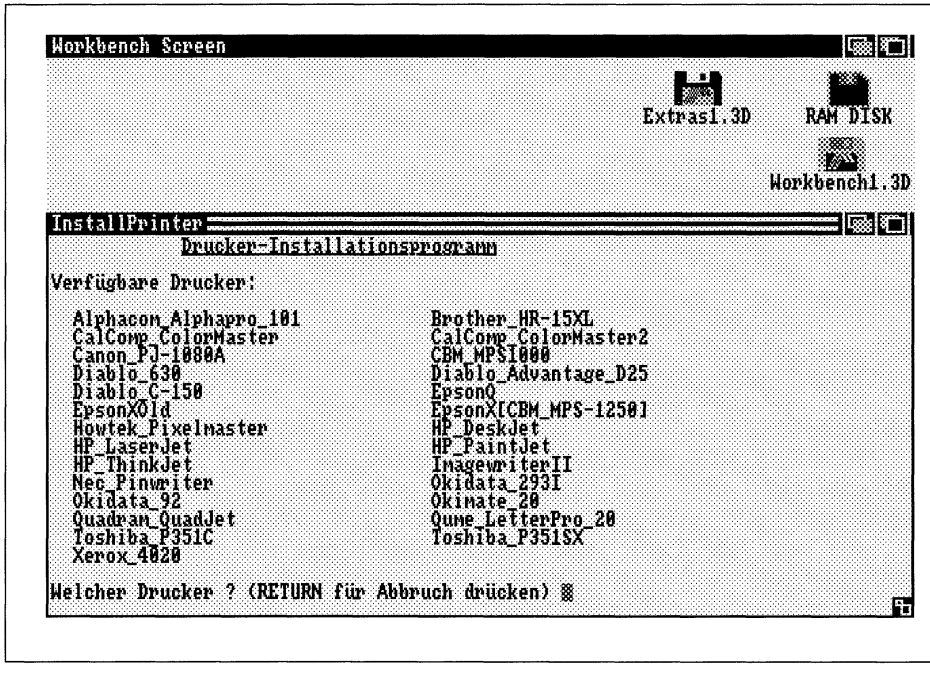



***Bild 5.13:*** *Die Liste der auf der Extras-Diskette verfügbaren Drucker*

Wenn Sie das getan haben, listet *InstallPrinter* die Namen der auf der Extras-Diskette vorhandenen Druckertreiber am Bildschirm auf. (Amiga-Besitzer ohne zweites Laufwerk müssen vorher eventuell noch ein- oder zweimal die Disketten wechseln.) Die Namen der Druckertreiber ähneln meist den Namen der entsprechenden Drucker. Sie sollten den von Ihnen benötigten Drucker deshalb problemlos finden können.

*InstallPrinter* fragt Sie am Ende der Liste »Welcher Drucker ? (RETURN für Abbruch drücken)«. Dahinter erscheint die Schreibmarke, die Sie ja schon von anderen Gelegenheiten her kennen. Geben Sie nun möglichst sorgfältig den Namen des gewünschten Druckertreibers ein. Beachten Sie, daß in den Namen statt eines Leerschritts immer das Unterstreichungszeichen verwendet wird. Vergleichen Sie dann noch einmal den von Ihnen eingegebenen Namen mit der Schreibweise in der Liste darüber und drücken Sie auf die [Return]-Taste.

*InstallPrinter* kopiert nun den gewünschten Druckertreiber von der Extras-Diskette in die dafür vorgesehene (unsichtbare) Schublade der Workbench-Diskette. Amiga-Besitzer mit nur einem Laufwerk müssen dabei wieder mehrmals die Disketten austauschen. Ist der Kopiervorgang beendet, wird das zuvor geöffnete Fenster geschlossen und Sie kehren wieder zur Workbench zurück.

Damit der soeben kopierte Druckertreiber beim Ausdrucken von Texten und Grafiken auch verwendet wird, müssen Sie jetzt aber auch dafür sorgen, daß die Druck-Software von dem neuen Treiber erfährt. Rufen Sie dazu *Preferences* auf und klicken Sie im Hauptfenster auf die Taste *Change Printer*. Klicken Sie dann in der Liste in der oberen rechten Ecke des Druckeranpassungsfensters auf die beiden nach oben und nach unten zeigenden Pfeile, bis der Name des neuen Druckertreibers in der Mitte der danebenstehenden Liste zu sehen ist und durch Farbumkehrung hervorgehoben wird. Verlassen Sie dann das Druckeranpassungsfenster mit einem Klick auf *OK* und dann *Preferences* mit einem Klick auf *Save* im Hauptfenster.

### 5.2.3 Den Amiga reden lassen: das Programm Say

Während *InstallPrinter* ein notwendiges Werkzeug ist, das fast jeder Amiga-Besitzer zumindest einmal benötigen wird, handelt es sich bei dem nächsten Programm eher um eine kleine Spielerei. Das Programm *Say* (engl. *to say* = etwas sagen/sprechen) dient zur Demonstration der Sprachsynthese, zu der jeder Amiga fähig ist. Sie können Texte eingeben und dann zuhören, wie Sie der Amiga »ausspricht«. Starten Sie dazu *Say* (mit einem Doppelklick auf das Piktogramm) und warten Sie ab, bis zwei neue Fenster erschienen sind. Das eine heißt *Input Window* (Eingabe-Fenster), das andere *Phoneme Window* (Phonem-Fenster).

Im Eingabe-Fenster tippen Sie die Sätze ein, die der Amiga aussprechen soll. (Klicken Sie vorher einmal mit der Maus in dieses Fenster.) Jedesmal, wenn Sie einen Satz mit [Return] abschließen, wird er ausgesprochen und es erscheinen parallel dazu die »Phoneme«, aus denen er besteht, im Phonem-Fenster. Phoneme sind die Bausteine, aus denen gesprochene Wörter bestehen. Ein Phonem entspricht in etwa einer Silbe. Während ein und dieselbe Silbe aber in verschiedenen Wörtern verschieden ausgesprochen werden können, ist die Aussprache eines Phonems immer identisch und vom Zusammenhang unabhängig. Die »Phonemschrift« ist im Gegensatz zu allen anderen Schriften nicht sprachabhängig. Das heißt, die gleiche Folge von Phonemen wird – abgesehen von einem leichten Akzent – in allen Sprachen gleich ausgesprochen. Der Amiga hat übrigens – entsprechend seiner Herkunft – einen amerikanischen Akzent.

Sollten Ihre Versuche mit *Say* keine hörbaren Ergebnisse zustande bringen, hat das Problem wahrscheinlich technische Ursachen. Der Amiga 500 besitzt keinen eingebauten Lautsprecher und bleibt deshalb sprachlos, solange Sie seine Ton-Ausgänge (die beiden runden

Cinch-Buchsen an der Rückseite) nicht mit einem Lautsprecher verbinden. Das müssen nicht unbedingt große HiFi-Boxen sein. In viele Monitore sind ein oder zwei kleine Lautsprecher eingebaut. Passende Kabel für Ihren Monitor beziehungsweise Ihre Boxen erhalten Sie im Fachhandel. Sie brauchen lediglich darauf zu achten, daß das Kabelende, das für den Amiga gedacht ist, mit Cinch-Steckern versehen ist. (Wie das andere Ende der Kabel aussehen muß, hängt vom Monitor beziehungsweise den Boxen ab, den/die Sie verwenden.)

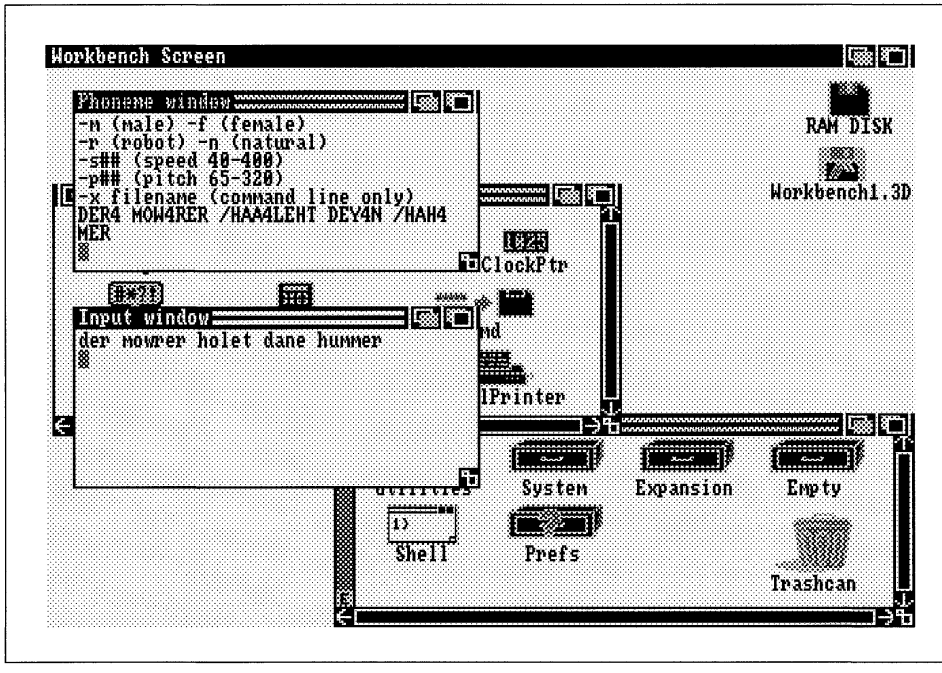



***Bild 5.14:*** *Die beiden Fenster des Sprachausgabeprogramms Say*

Welches Phonem wie ausgesprochen wird, erfahren Sie im Amiga-Basic-Handbuch. Sie müssen das aber nicht unbedingt wissen, um *Say* sprechen zu lassen. Englische Sätze bewältigt *Say* sehr gut. Und wenn Sie Ihren Amiga deutsch sprechen lassen wollen, so überlegen Sie sich einfach, welches englische Wort so ausgesprochen würde wie das gewünschte deutsche. Den Satz »der mowrer holet den hummer« spricht *Say* zum Beispiel als »Der Maurer holt den Hammer« aus.

Neben Sätzen, die der Amiga aussprechen soll, können Sie aber auch noch ein paar Befehle eingeben, die die Aussprache des Amiga 500 beeinflussen. Hierzu müssen Sie einen neuen Satz einfach mit einem Bindestrich (Minuszeichen) beginnen. »–f« (für female = weiblich) schaltet auf eine weibliche Stimme um, »–m« (für male = männlich) auf eine männliche. Mit »–n« (für natürlich) erreicht man eine einigermaßen normale Aussprache, wobei der Amiga seine Stimme hebt und senkt und verschiedene Silben verschieden stark betont. Dagegen schaltet »-r« auf eine roboterhafte Stimme ohne jede Betonung um. Mit den beiden Befehlen »–p« (für *pitch* = Tonhöhe) und »–s« (für *speed* = Geschwindigkeit) kann man

Tonhöhe und Geschwindigkeit der Aussprache ändern. Hierzu muß hinter »–p« eine Zahl zwischen 65 (sehr tief/dunkel) und 320 (sehr hoch/hell) und hinter »–s« eine Zahl zwischen 40 (sehr langsam) und 400 (sehr schnell) folgen. Die Zahlen müssen direkt, ohne eine Leerstelle dazwischen, hinter »p« oder »s« folgen!

| Befehl | Bedeutung |
|---|---|
| -f | weibliche Stimme |
| -m | männliche Stimme |
| -n | natürliche Betonung |
| -r | tonlose, roboterhafte Aussprache |
| -p<Zahl> | Tonhöhe entsprechend <Zahl> verändern |
| -s<Zahl> | Sprechgeschwindigkeit entsprechend <Zahl> verändern |

***Tabelle 5.1:*** *Die Befehle zur Änderung der Sprachausgabe von Say*

Noch ein Tip zu dieser Tabelle: Um von einer männlichen zu einer weiblichen Stimme zu wechseln (oder umgekehrt), reicht »–f« meist nicht aus. Gleichzeitig sollten Sie die Tonhöhe ändern, um eine Frauenstimme zu erhalten. »–f« und »–p250« ergibt zum Beispiel eine realistische Frauenstimme und »–m« und »–p100« eine realistische Männerstimme.

Wenn Sie der merkwürdigen Laute, die Ihr Amiga ausstößt, dann endlich müde sind, können Sie *Say* beenden, indem Sie das Eingabe-Fenster anklicken, nichts eingeben und ein- oder zweimal auf die [Return]-Taste drücken.

### 5.2.4 ClockPtr

Ebenfalls in die Kategorie »Spielerei« gehört – wie *Say* – das Programm *ClockPtr*, das Sie ebenfalls in der Schublade *Utilities* finden. Wenn Sie dieses Programm (wie üblich mit Doppelklick) starten, scheint zunächst einmal überhaupt nichts zu passieren. Das Diskettenlaufwerk wird einen kurzen Moment lang tätig und danach ist wieder alles beim Alten. Kein Fenster öffnet sich und auch die Menüzeile bleibt unverändert.

Wenn Sie aber jetzt einmal mit dem Mauszeiger in die freie (meist blaue) Fläche des Workbench-Hintergrunds klicken, ändert der Mauszeiger radikal seine Form. Er wird auf einmal zu einer kleinen Digitaluhr, die die aktuelle Zeit anzeigt. Sobald Sie in ein Fenster oder auf ein Piktogramm klicken, verschwindet die Uhr wieder und der Mauszeiger bekommt wieder seine ursprüngliche Form.

Diese winzige Uhr – die gegenüber *Clock* den Vorteil der deutlich geringeren Größe hat – kann aber nicht nur Stunde und Minuten anzeigen. Sie können Sie auch dazu bringen, eine Sekundenanzeige ticken zu lassen oder das aktuelle Datum zu zeigen. Schieben Sie den Mauszeiger dazu mit eingeschalteter Digitalanzeige an den linken Rand des Bildschirms. Die kleine Digitaluhr wechselt dann auf Datumsanzeige um. Dabei wird das englische Datumsformat verwendet, bei dem der Monat zuerst kommt. Der erste Juni würde beispielsweise als »601« angezeigt. Wenn Sie nun (am linken Rand des Bildschirms) die Maus nach oben schieben, verschwindet die Datumsanzeige wieder. Kurz unter der Menüzeile, also in

der oberen linken Ecke des Bildschirms taucht statt dessen die dritte Form der Digitaluhr auf, die Minuten und (laufende) Sekunden zeigt.

Nachdem Sie *ClockPtr* gestartet haben, wird der Mauszeiger immer zur Digitaluhr, sobald Sie in den Workbench-Hintergrund klicken – auch dann, wenn Sie zum Beispiel ein Diskettenpiktogramm anklicken. Wenn Sie das vermeiden wollen, können Sie *ClockPtr* auch wieder abschalten. Starten Sie das Programm dazu einfach ein zweites Mal.

### 5.2.5 Graphic Dump

Keine Spielerei ist im Gegensatz zu *Say* und *ClockPtr* das Programm *Graphic Dump*. Es dient dazu, den Bildschirminhalt, wie Sie ihn in einem bestimmten Moment sehen, auf einem (grafikfähigen) Drucker auszugeben. Dieser Drucker muß dazu angeschlossen, eingeschaltet und mit *Preferences* »angemeldet« worden sein. Computerfachleute sagen zu einem solchen Bildschirmausdruck »Dump« oder »Screendump«. Da es sich beim Amiga immer um eine Bildschirmgrafik handelt, spricht man von einem grafischen Screendump oder »Graphic Dump«.

Wenn Sie einen solchen Screendump ausdrucken wollen, brauchen Sie *Graphic Dump* nur zu starten (mit einem Doppelklick auf das Piktogramm). Nach dem Start bleiben Ihnen noch ungefähr 10 Sekunden, den Bildschirm so zu gestalten, daß Sie das sehen, was ausgedruckt werden soll, dann beginnt *Graphic Dump* zu arbeiten (drucken). Sie können in dieser Zeit zum Beispiel mit [Ⓐ]+[M] schnell den Workbench-Screen nach hinten legen. Ausgedruckt wird dann der in diesem Moment vorne liegende Screen, ohne einen eventuell darüber liegenden Mauszeiger. (Es gibt keinen Weg, den Mauszeiger auszudrucken.) Wenn in diesem Moment jedoch ein Menü sichtbar ist, wird es mit eventuell hervorgehobenen Menüpunkten (aber ohne Mauszeiger) ausgedruckt.

Ist der Drucker nicht oder nicht korrekt angeschlossen und mit *Preferences* angemeldet worden, erscheint eventuell nach Ablauf von etwa 20 Sekunden eine Fehlermeldung. Klicken Sie in dem dabei auftauchenden Requester dann einfach auf *CANCEL*, schalten Sie den Drucker ein (oder beheben den Fehler auf andere Weise) und versuchen Sie es noch einmal.

### 5.2.6 More

Das Programm *More* ist ideal für das Lesen größerer Textdateien, die zu umfangreich für *NotePad* sind. Wohlgemerkt: Sie können diese Texte mit *More* nicht ändern sondern nur betrachten. Probieren Sie *More* am besten einfach einmal aus, um ein Gefühl dafür zu bekommen, wie das Programm funktioniert.

Starten Sie es dazu mit einem Doppelklick. Daraufhin öffnet sich ein neues Fenster am Bildschirm, in dem am oberen Rand einige kurze Meldungen erscheinen. *More* fragt Sie nun nach dem Namen der Textdatei, die angezeigt werden soll. Tippen Sie daraufhin den folgenden Namen exakt ein »S:startup-sequence«. Schließen Sie die Eingabe wie üblich mit [Return] ab.

*More* zeigt Ihnen jetzt die »erste Seite« der Datei Startup-Sequence« (über die Sie in einem späteren Kapitel noch mehr erfahren werden) in seinem Fenster. Als eine Seite faßt *More*

24 Zeilen auf. (Sie können das Fenster aber auch größer oder kleiner machen. *More*s Seitenlänge ändert sich dann entsprechend.) Je nachdem, wie groß die Textdatei ist, die Sie gerade betrachten, haben Sie jetzt mehrere Möglichkeiten. Um More einen Befehl zu erteilen, müssen Sie jeweils eine Taste drücken. (Das Programm hat keine Menüs.)

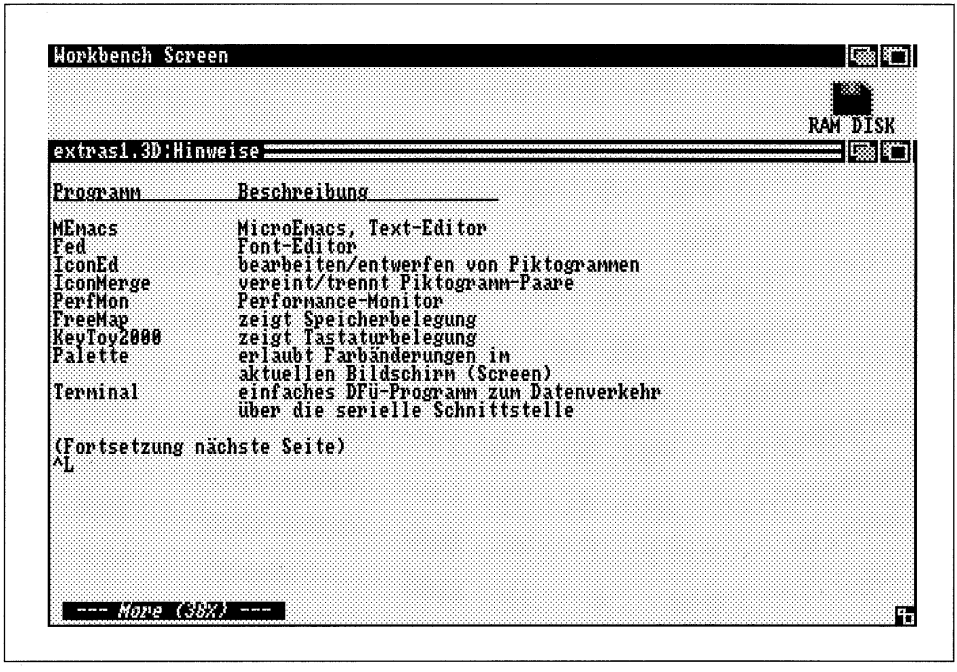



***Bild 5.15:*** *Das Programm More*

Wenn Sie auf die [Return]-Taste drücken, zeigt *More* Ihnen die nächste Zeile der Textdatei. Wollen Sie gleich die nächste »Seite« sehen, müssen Sie die Leertaste drücken. Wenn Sie eine Seite zurückblättern wollen, müssen Sie [Backspace] drücken und mit den Tasten [<] und [>] gelangen Sie zur ersten beziehungsweise letzten Seite der Datei. Wenn Sie die verschiedenen Befehle vergessen haben, brauchen Sie nur auf [H] (Hilfe) zu drücken. *More* beschreibt dann selbst (leider auf englisch) die zur Verfügung stehenden Befehle.

Wie Sie dieser Übersicht unschwer entnehmen können, gibt es noch einige Befehle mehr, als bisher beschrieben wurden. Der wichtigste davon ist sicherlich die Taste [Q], mit der Sie das Programm jederzeit abbrechen können – auch bevor das Ende erreicht ist. Testen Sie diesen Befehl am besten sofort einmal aus, indem Sie *More* durch Betätigung der Taste [Q] verlassen.

Das Ausprobieren der *More*-Befehle ist anhand der sehr kurzen Datei Startup-Sequence leider nicht besonders gut möglich. Sie sollten *More* deshalb auch noch einmal anhand einer größeren Textdatei ausprobieren. Eine solche Datei befindet sich zum Beispiel auf der Extras-Diskette. Sie heißt *Hinweise* und liegt mitten im Diskettenfenster.

Um sich diese Datei anzuschauen, gehen Sie am besten wie folgt vor: Öffnen Sie zunächst einmal die *Utilities*-Schublade (falls nötig), so daß Sie das *More*-Piktogramm sehen können. Legen Sie dann die Extras-Diskette ein und öffnen Sie sie mit einem Doppelklick auf das entsprechende Piktogramm (Amiga-Besitzer mit einem Laufwerk müssen dazu selbstverständlich erst die Workbench-Diskette aus dem Laufwerk nehmen).

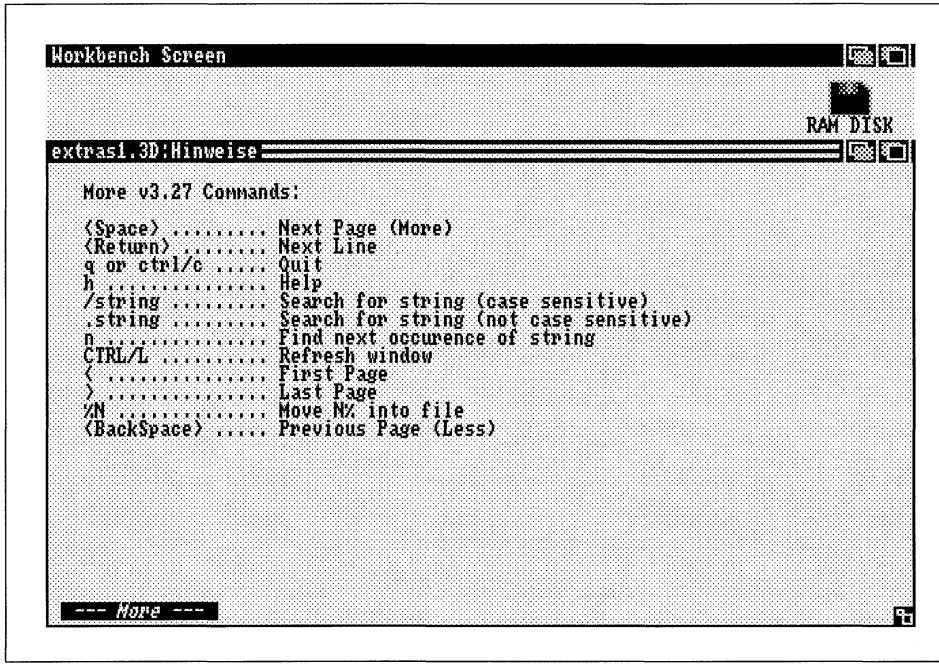



***Bild 5.16:*** *Die Hilfe-Funktion von More*

Im Diskettenfenster der Extras-Diskette sehen Sie jetzt das Piktogramm *Hinweise*. Klicken Sie es einmal an, damit es durch Farbumkehr hervorgehoben wird. Halten Sie jetzt die [Shift]-Taste fest und klicken Sie auch das Piktogramm More einmal an. Beide Piktogramme sollten jetzt hervorgehoben (und damit ausgewählt) sein. Wählen Sie jetzt den Befehl *Open* aus dem Menü *Workbench*.

Sofern Sie nur ein Laufwerk verwenden, werden Sie jetzt aufgefordert, die Workbench-Diskette wieder einzulegen, damit die Workbench-Software das Programm *More* in den Speicher laden kann. Ist das geschehen, wird wieder die Extras-Diskette angefordert.

*More* zeigt dann die erste Seite der Datei Hinweise. Das es sich dabei um eine größere Datei handelt, können Sie am unteren Rand des Fensters erkennen. Die Meldung »-- More (19%) --« besagt, daß unterhalb des gerade sichtbaren Teils noch 81% der Datei folgen. Jedesmal, wenn Sie auf die Leertaste oder die [Return]-Taste drücken, bekommen Sie eine Seite beziehungsweise eine Zeile dieser restlichen 81% zu sehen und die Zahl in der Meldung am unteren Fensterrand wird entsprechend größer.

Lesen Sie jetzt am besten erst einmal diese Datei – sie enthält wichtige Hinweise über den Inhalt der Extras-Diskette – und benutzen Sie dazu die Leertaste und die [Return]-Taste. Probieren Sie aber auch einmal die Tasten [Backspace], [<] und [>] aus, mit denen Sie sich innerhalb der Datei beliebig hin- und herbewegen können.

Sie brauchen eine längere Textdatei aber nicht unbedingt mühselig zu durchblättern, um eine bestimmte Stelle eines Textes zu finden. *More* bietet eine Möglichkeit, nach einem bestimmten Word oder einer Textpassage zu suchen. Dazu gehen Sie am besten mit [<] an den Anfang der Datei und drücken dann die Taste [/] oder [.]. Dahinter geben Sie den Suchbegriff ein und drücken zum Abschluß die [Return]-Taste. Der Unterschied zwischen [/] oder [.] liegt in der Behandlung der Groß- beziehungsweise Kleinschreibung beim Suchen. Verwenden Sie [.] spielt die Schreibweise des Suchbegriffs keine Rolle. Bei [/] hingegen muß die Schreibweise des Suchbegriffs exakt mit einer Passage im Text übereinstimmen, damit diese Stelle aufgefunden wird.

Probieren Sie die Suchbefehle am besten einmal aus: Drücken Sie dazu auf die Taste [<], dann auf [.] und geben Sie dahinter »drucker« ein. Wenn Sie jetzt auf [Return] drücken, zeigt Ihnen *More* die erste Zeile der Datei, in der das Wort »Drucker« vorkommt. Wenn Sie die nächste Stelle finden wollen, an der der Suchbegriff auftaucht, brauchen Sie nur die Taste [N] zu drücken. Dabei spielt es keine Rolle, ob dieses Wort »drucker«, »Drucker« oder »DRUCKER« geschrieben wird, da Sie die Suche mit [.] begonnen haben. Hätten Sie den Befehl [/] verwendet, würden Stellen, die »Drucker« oder »DRUCKER« enthalten, nicht gefunden.

Leider funktioniert *More* nicht korrekt mit Umlauten. Um einen Begriff zu finden, der Umlaute enthält, muß seine Schreibweise exakt angegeben werden. »Ö« und »ö« sind für More immer – auch hinter [.] – zwei völlig verschiedene Buchstaben.

Wenn Sie nicht nach einem bestimmten Begriff suchen, sondern möglichst schnell zu einer bestimmten Stelle in einer Datei gelangen wollen, können Sie die Taste [%] verwenden. Drücken Sie dazu diese Taste und geben Sie dahinter (als Prozentwert) die Stelle der Datei an, die Sie sehen wollen. Um den Beginn des hinteren Drittels einer Textdatei zu sehen, müßten Sie beispielsweise »%66« eingeben und dann die [Return]-Taste drücken.

Damit soll die Beschreibung des *More*-Programms dann auch abgeschlossen werden. Die restlichen, bisher nicht beschriebenen Befehle sind eher »esoterischer« Natur. Sie werden sie selten oder nie benötigen.

### 5.2.7 PrintFiles

Ähnlichen Zwecken wie *More* dient *PrintFiles*. Aber – wie sein Name schon andeutet (engl. *to print* = drucken) – gibt *PrintFiles* Textdateien nicht am Bildschirm, sondern auf einem Drucker aus. *PrintFiles* kann darüber hinaus gleich mehrere Dateien nacheinander zum Drucker schicken.

Die Auswahl dieser Dateien ist denkbar einfach. Sie läuft ähnlich ab, wie das im letzten Abschnitt beschriebene Starten von *More* über das Piktogramm einer Textdatei: Klicken Sie das Piktogramm der ersten Datei an, die Sie ausgedruckt wünschen, halten Sie dann die [Shift]-Taste fest und klicken Sie nacheinander die weiteren Dateien an. Klicken Sie dann

zu guter Letzt (wieder mit gedrückter [Shift]-Taste) auf das Piktogramm von *PrintFiles* und wählen Sie den Befehl *Open* aus dem Menü *Workbench*. Die Workbench-Software startet dann *PrintFiles* und teilt dem Programm auch gleich mit, welche weiteren Piktogramme vor dem Start markiert wurden. *PrintFiles* gibt die Dateien dann nacheinander auf dem Drucker aus – sofern er angeschlossen und eingeschaltet ist. Wenn das nicht der Fall ist, erscheint nach kurzer Zeit ein Requester mit einer Fehlermeldung auf dem Bildschirm.

Beachten Sie unbedingt, daß *PrintFiles* nur Text-Dateien ausdrucken kann und Sie deshalb (außer *PrintFiles* selbst) auch nur Piktogramme von Textdateien markieren dürfen, bevor Sie den Befehl *Open* aufrufen. *PrintFiles* kann keine Bilder, die Sie mit einem Malprogramm erstellt haben und auch keine Datenbanken ausdrucken, sondern wirklich nur Texte.

### 5.2.8 Cmd

Das letzte noch nicht beschriebene Piktogramm in der *Utilities*-Schublade ist *Cmd*. Mit diesem Programm können Sie den Strom von Daten, der normalerweise zum Drucker geschickt würde in einer Datei »umleiten«. Das kann für verschiedene Zwecke sinnvoll sein. Der häufigste Fall ist sicherlich, ein Programm erst einmal »auf die Diskette drucken« zu lassen (was sehr schnell geht) und die dabei entstandene Datei dann mit einem anderen Programm und/oder zu einem späteren Zeitpunkt zum Drucker zu senden (was sehr lange dauert).

Eine Diskette mit einer solchen Druckdatei könnte man sogar zu einem anderen Amiga tragen (an dem vielleicht ein besserer Drucker angeschlossen ist) und dort ausdrucken. Programmierer können *Cmd* darüber hinaus auch noch dazu verwenden, um nachzuprüfen, welche Befehle beim Drucken an den Drucker geschickt werden.

Um *Cmd* sinnvoll nutzen zu können, müssen Sie aber leider über einige Kenntnisse mehr verfügen, als Sie bisher bei der Arbeit mit der Workbench erworben haben. Sie erhalten die entsprechenden Informationen in den folgenden Kapiteln (über das sogenannte CLI), in denen *Cmd* dann auch noch einmal ausführlicher beschrieben wird.

Vermeiden Sie es vorläufig – sofern Sie nicht genau wissen, was Sie tun – Cmd zu starten. Unter Umständen kann Ihr Amiga dann nämlich nicht mehr drucken! Sollte Ihnen das versehentlich einmal passieren, hilft ein Neustart ([Ctrl]+[*A*]+[*A*]).

## 5.3 Werkzeuge in der System-Schublade

Zusätzliche Hilfsprogramme finden sich nicht nur in der *Utilities*-Schublade sondern auch in der *System*-Schublade der Workbench. Die Programme in der *System*-Schublade sind aber keine typischen Anwendungsprogramme und Hilfsmittel, wie es zum Beispiel *Notepad* und *Calculator* darstellen. Es handelt sich statt dessen um sogenannte »Dienstprogramme«.

Einige dieser Programme werden Sie vielleicht nie benötigen. Andere davon werden Sie vielleicht nur ein- oder zweimal starten und dann wieder vergessen. Das heißt aber nicht, daß diese Programme unwichtig wären. Manche davon werden in gewissen Situationen automatisch gestartet, ohne daß Sie etwas davon merken. Löschen Sie also bitte keines der Piktogramme in der *System*-Schublade, ohne zuvor seine Bedeutung verstanden zu haben.

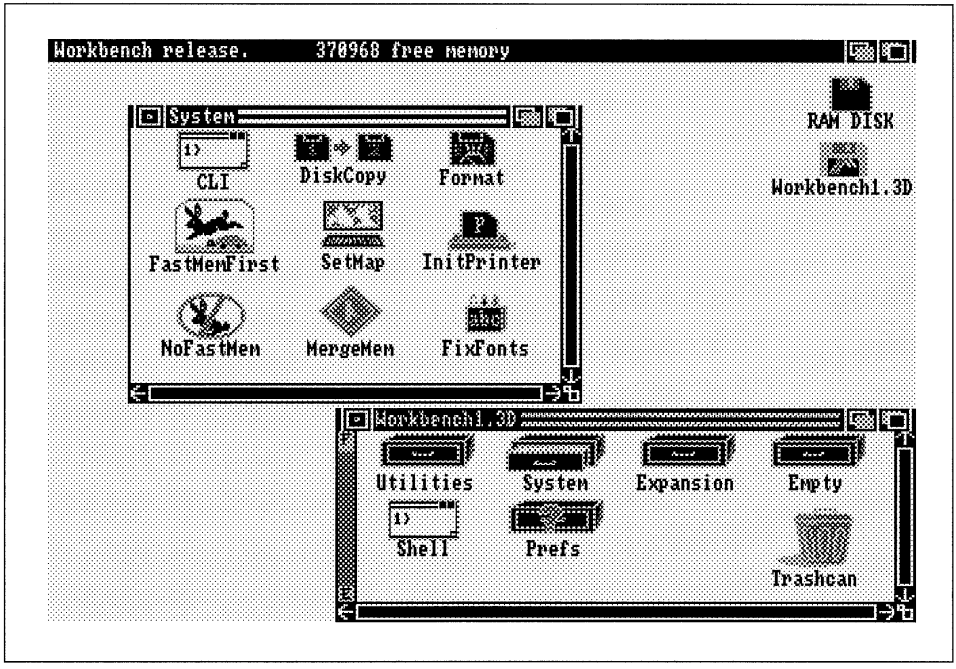



***Bild 5.17:*** *Inhalt der System-Schublade*

### 5.3.1 Format und DiskCopy

Zu den eben erwähnten Dienstprogrammen, die weitgehend »unbemerkt gestartet werden, gehören zum Beispiel *Format* und *DiskCopy*. Wenn Sie eines dieser beiden Programme starten, erscheint lediglich eine Fehlermeldung, die Sie mit einem Klick auf *Ok* bestätigen müssen. Diese Fehlermeldungen besagen nichts anderes, als daß *DiskCopy* und *Format* automatisch von der Workbench aufgerufen werden, wenn Sie ein Diskettenpiktogramm selektieren (also mit der Maus anklicken) und dann den Befehl *Duplicate* oder *Initialize* aus dem *Workbench*- oder *Disk*-Menü aufrufen. *DiskCopy* dient zum Kopieren und *Format* zum Initialisieren beziehungsweise Formatieren kompletter Disketten (siehe Kapitel 3).

Obwohl Sie diese beiden Programme nicht mit einem Doppelklick starten können, sollten Sie sie deshalb auf keinen Fall löschen. Wenn Sie das trotzdem tun, können Sie danach keine Disketten mehr kopieren und/oder initialisieren.

### 5.3.2 NoFastMem

Das Programm *NoFastMem* in der *System*-Schublade der Workbench-Diskette dient dazu, bestimmte alte Programme auf Ihrem Amiga 500 laufen zu lassen, die zu einer Zeit entwickelt wurden, als es nur den Amiga 1000 gab. Der Amiga 1000 verfügte zunächst nur über 512 Kbyte RAM-Speicher. Später kamen Speichererweiterungen auf den Markt, die

ihn oder auch seine Nachfolgermodelle 500 und 2000 mit wesentlich mehr RAM-Speicher ausstatteten. Dieser über 512 Kbyte hinausgehende Speicher unterscheidet sich allerdings von den ersten 512 Kbyte. Er wird auch »FastMem« (schneller Speicher) genannt. Für bestimmte Aufgaben im Zusammenhang mit Grafik- und Klangerzeugung ist er jedoch nicht geeignet. Dafür benötigt man den sogenannten »Chip-RAM«, der seinen Namen daher hat, daß die Spezial-Chips des Amiga nur diesen RAM-Bereich benutzen können. Clevere Programme wissen das und benutzen die beiden verschiedenen Speichersorten entsprechend ihrer Eignung. Einige sehr alte Programme sind jedoch so programmiert worden, als gäbe es nur Amiga-Modelle mit 512 Kbyte RAM.

Solche Programme laufen oft auf dem Amiga 500 dann nicht mehr, wenn Sie die Speichererweiterung A 501 oder eine vergleichbare Speichererweiterung eines anderen Herstellers eingebaut haben. Damit Sie mit einem solchen Programm arbeiten können, ohne dazu jedesmal die Speichererweiterung ausbauen zu müssen, gibt es *NoFastMem*. Wenn Sie mit einem solchen Programm arbeiten wollen, müssen Sie zunächst *NoFastMem* starten und dann erst das »Oldtimer-Programm«. *NoFastMem* belegt nämlich den gesamten RAM-Speicher, der sich in der Speichererweiterung befindet, und läßt nur die ersten 512 Kbyte frei. So wird verhindert, daß das fehlerhafte Programm diesen Speicherbereich benutzen kann. Wenn Sie später wieder den vollen Speicher nutzen wollen, brauchen Sie *NoFastMem* nur noch einmal starten. Sie können nach dem Start von *NoFastMem* jeweils in der Titelleiste der Workbench verfolgen, wie der freie Speicher ab- und wieder zunimmt.

### 5.3.3 FastMemFirst

Das Werkzeug *FastMemFirst* ist fast das genaue Gegenteil von *NoFastMem*. Nachdem Sie *FastMemFirst* das erste Mal starten, werden Programme und Daten bevorzugt in den RAM-Bereich oberhalb der ersten 512 Kbyte gelegt. (Ein erneuter Start von *FastMemFirst* hebt diesen Effekt wieder auf.) Der Chip-RAM-Bereich wird nur dann verwendet, wenn der FastMem-Bereich oberhalb von 512 Kbyte voll ist oder wenn ein Programm ausdrücklich auf Chip-RAM »besteht«. Das kann dazu führen, daß der Amiga insgesamt etwas schneller wird, da Programme, die im FastMem-Bereich liegen, unter gewissen Umständen wirklich etwas schneller laufen. Mehr darüber im letzten Teil dieses Buches.

### 5.3.4 MergeMem

*MergeMem* gehört in eine ähnliche Kategorie wie *FastMemFirst* und *NoFastMem*. Sie brauchen dieses Programm aber nur dann zu starten, wenn Sie mehrere Speichererweiterungen besitzen, wenn Ihr Amiga also mehr als 1 Mbyte RAM-Speicher besitzt. *MergeMem* versucht, die in den verschiedenen Speichererweiterungen liegenden RAM-Speicher zu einem großen zusammenhängenden Speicherbereich zusammenzufassen. Danach kann ein Programm einen Speicherbereich benutzen, der so groß ist, daß er sich über mehrere Speichererweiterungen erstreckt. (Ohne Aufruf von *MergeMem* ist das nicht möglich.)

Da aber die wenigsten Amiga-500-Besitzer über mehrere Speicherweiterungen verfügen dürften und darüber hinaus wahrscheinlich auch kaum Programme verwenden werden, die solche riesigen Speicheranforderungen stellen, werden Sie *MergeMem* wahrscheinlich nie benötigen. Anspruchsvolle, speicherhungrige Programme dieser Art fühlen sich ohnedies

auf dem Amiga 2000 oder Amiga 2500 wesentlich wohler. Diese Amiga-Modelle können auch ohne größere Probleme mit mehreren Speicherweiterungen ausgestattet werden.

### 5.3.5 InitPrinter

Das Werkzeug *InitPrinter* dient zur »Initialisierung« des Druckers, also zur Vorbereitung des Druckens. Es wird nicht unbedingt bei jedem Drucker benötigt. Wenn Sie den von Ihnen verwendeten Drucker aber korrekt mit *Preferences* angemeldet haben und trotzdem nur »Quatsch« aus dem Drucker kommt, sollten Sie *InitPrinter* einmal starten und dann noch einmal versuchen, zu drucken.

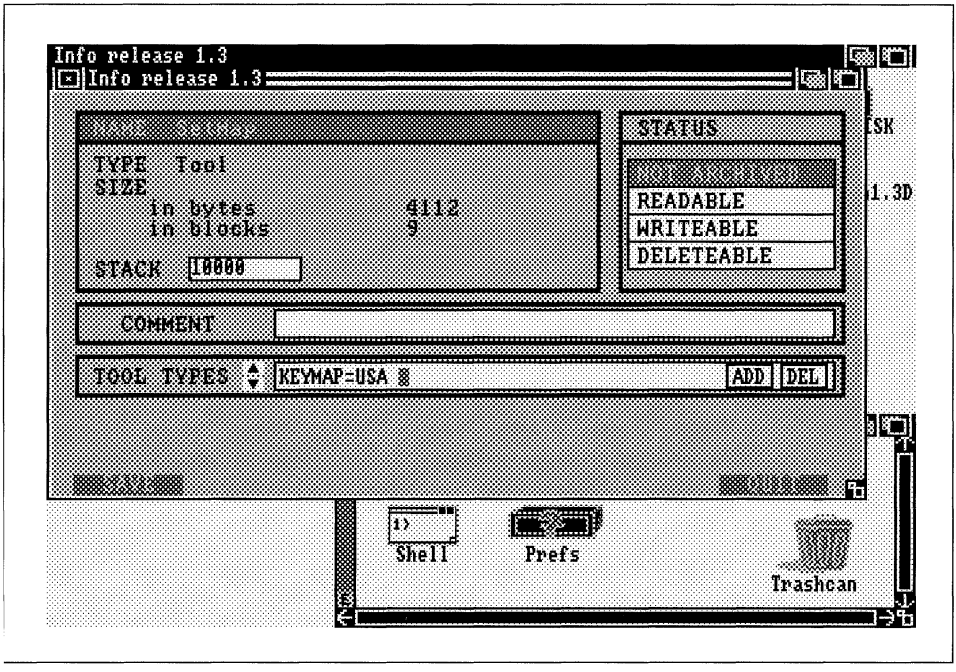



***Bild 5.18:*** *Das Info-Fenster von SetMap*

### 5.3.6 SetMap

Das Programm *SetMap* werden Sie wahrscheinlich relativ selten benötigen. Es dient dazu, die »Tastaturbelegung« des Amiga zu ändern. Diese Tastaturbelegung legt fest, welcher Buchstabe auf dem Bildschirm erscheint, wenn Sie eine Taste drücken. Es ist keineswegs selbstverständlich, daß ein »a« am Bildschirm erscheint, wenn Sie *Notepad* starten und die Taste [A] drücken. Wenn Sie eine Taste betätigen, geht eine Folge elektrischer Impulse von einem bestimmten Chip im Amiga an einen anderen. Wie diese Impulse interpretiert werden, also die Zuordnung der Impulsfolge zu einem bestimmten Buchstaben, bestimmt die

Systemsoftware des Amiga. Sie verwendet dazu eine Tabelle, die sogenannte »keymap« (Tastenkarte). Diese Tastenkarte können Sie mit *SetMap* »setzen«, also verändern.

Sie werden sich nun vielleicht fragen, wozu denn das gut sein soll. Schließlich wollen Sie ja, daß ein »a« erscheint, wenn Sie die Taste [A] drücken. Die Antwort darauf ist relativ einfach: Der Amiga wird nicht nur nach Deutschland verkauft und in anderen Ländern ergeben dieselben Tasten andere Buchstaben. Die beiden Tasten, die bei uns das »Ü« und das »+« ergeben, dienen auf einem englischen Amiga zur Eingabe der eckigen Klammern.

Mit *SetMap* können Sie auf die Tastaturbelegung eines anderen Landes umschalten. Selektieren Sie hierzu das Piktogramm *SetMap* und wählen Sie *Info* aus dem Menü *Workbench*. Daraufhin erscheint das Info-Fenster von *SetMap*. Klicken Sie nun einmal auf den Knopf *Add* am rechten Fensterrand (auf gleicher Höhe mit den Worten *TOOL TYPES*). Klicken Sie dann in das Text-Gadget *TOOL TYPES* und geben Sie den folgenden Text ein »KEYMAP=<xx>« (natürlich ohne die Anführungsstriche), wobei Sie <xx> durch das Kürzel eines Landes ersetzen müssen, dessen Tastaturbelegung Sie wünschen.

Probieren Sie zum Beispiel einmal »KEYMAP=gb« aus. Schließen Sie die Eingabe im Feld TOOL TYPES durch Betätigung von [Return] ab und klicken Sie dann auf den Knopf SAVE unten links im Info-Fenster. Wenn Sie nun *SetMap* (mit einem Doppelklick auf das Piktogramm) starten, geschieht scheinbar nichts. Wenn Sie dann aber *Notepad* starten und ein paar Sätze eingeben, werden Sie feststellen, daß die Beschriftung der Tasten nicht mehr mit den Buchstaben übereinstimmt, die sie ergeben. Sie arbeiten nun mit einer englischen Tastatur!

| **Kürzel** | **Land** |
|---|---|
| cdn | Kanada, französische Sprache |
| ch1 | Schweiz, französische Sprache |
| ch2 | Schweiz, deutsche Sprache |
| d | Deutschland/Österreich |
| e | Spanien |
| f | Frankreich |
| gb | Großbritannien |
| i | Italien |
| is | Island |
| n | Norwegen |
| s | Schweden |
| usa | USA |
| usa0 | USA (alt; wie Workbench Version 1.1) |
| usa2 | USA Dvorak-Belegung |

***Tabelle 5.2:** Länderkürzel für SetMap*

Sie können das aber leicht wieder rückgängig machen. Die brutalste Methode ist zweifellos ein Neustart. Eleganter ist es natürlich, im Feld TOOL TYPES von *SetMap* »KEYMAP=gb«

durch »KEYMAP=d« zu ersetzen und *SetMap* noch einmal zu starten. Sie brauchen dazu nicht einmal *Notepad* zu verlassen.

Löschen Sie *SetMap* bitte auch dann nicht, wenn Sie meinen, dieses Programm nicht zu brauchen. *SetMap* wird bei jedem Neustart automatisch aufgerufen, um die deutsche Tastenbelegung zu aktivieren. Wenn *SetMap* dann nicht vorhanden ist, behält der Amiga die voreingestellte (amerikanische) Tastenbelegung.

### 5.3.7 FixFonts

Das Programm *FixFonts* dient dazu, den Amiga beziehungsweise seine Systemsoftware davon zu informieren, daß Sie Änderungen an den auf der Workbench-Diskette verfügbaren Schriften vorgenommen haben. In der Beschreibung des Programms *NotePad* haben Sie ja gesehen, daß Ihnen auf dem Amiga eine ganze Reihe verschiedener Schriften (engl. *fonts*) in verschiedenen Schriftgrößen zur Verfügung stehen. Diese Schriften sind – bis auf die Schrift Topaz – auf der Workbench-Diskette gespeichert. Für jede Größe einer Schrift gibt es eine Datei, in der beschrieben wird, wie die Buchstaben dieser Schrift aussehen.

Sie haben jederzeit die Möglichkeit, den vorhandenen Schriften weitere hinzuzufügen oder überflüssige zu löschen. Dazu brauchen Sie lediglich die entsprechenden Dateien (die die Buchstaben enthalten) auf die Workbench-Diskette zu kopieren oder solche Dateien zu löschen. Da diese Dateien auf der Workbench unsichtbar sind, müssen Sie allerdings das CLI oder die Shell verwenden (siehe auch die nächsten beiden Abschnitte), um das zu tun!

Nach allen Änderungen an den Schriftdateien müssen Sie das Programm *FixFonts* aufrufen. Es vermerkt die durchgeführten Änderungen in speziellen Dateien, in denen die auf der Workbench-Diskette vorhandenen Schriften und Schriftgrößen aufgelistet werden. Programme, die wissen wollen, welche Schriften zur Verfügung stehen, brauchen dann lediglich diese Dateien zu lesen und nicht die ganze Diskette nach Schriftdateien durchsuchen – was ein recht langwieriger Vorgang ist.

Daß die Arbeit von *FixFonts* geraume Zeit in Anspruch nimmt, können Sie leicht feststellen, indem Sie das Programm einfach einmal starten (auch wenn Sie keine Änderungen an den Schriftdateien vorgenommen) haben. Dann geschieht scheinbar nichts – es öffnet sich zum Beispiel auch kein Fenster. Wenn Sie aber auf Ihr Diskettenlaufwerk beziehungsweise dessen Kontrollampe achten, werden Sie sehen, daß es ein ganze Zeit lang heftig arbeitet, während *FixFonts* die Schriftdateien inspiziert.

### 5.3.8 CLI

Unter Umständen finden Sie in der *System*-Schublade Ihrer Workbench-Diskette noch ein weiteres Werkzeug namens CLI. Die Buchstaben »CLI« stehen für ***C**ommand **L**ine **I**nterface*. Dahinter verbirgt sich eine zweite, von der Workbench unabhängige Art, den Amiga zu bedienen. Diese Methode arbeitet mit Befehlen, die Sie über die Tastatur eintippen müssen, und Rückmeldungen, die in reiner Textform, ohne Grafiken und Symbol, auf den Bildschirm ausgegeben werden.

Nähere Erläuterungen zum Thema CLI finden Sie im zweiten Teil dieses Buches, der sich ausschließlich mit diesem Thema beschäftigt (ab Kapitel 6). Solange Sie diesen Buchteil

nicht durchgearbeitet haben, sollten Sie das Werkzeug CLI nicht starten. Sollte es Ihnen aber doch einmal passieren, daß Sie CLI aus Versehen starten, so warten Sie bitte, bis ein neues Fenster auf dem Bildschirm erscheint. Tippen Sie dann das Wort (den Befehl) *endcli* ein und drücken Sie die [Return]-Taste. Das Fenster müßte daraufhin wieder vom Bildschirm verschwinden (es besitzt kein Schließ-Gadget und kann nur auf die eben beschriebene Weise wieder geschlossen werden).

Das CLI wird in den Workbench-Versionen 1.3 und darüber jedoch kaum noch verwendet. Sein Piktogramm befindet sich nur noch aus »historischen« Gründen in der *System*-Schublade. Wer den Amiga mit Befehen bedienen möchte, die über die Tastatur eingegeben werden (für manche Aufgaben ist das nötig), wird ab der Workbench-Version 1.3 meist die Shell verwenden, deren Piktogramm im Disketten-Fenster der Workbench-Diskette liegt.

### 5.3.9 Shell

Für die Shell gilt im Prinzip ähnliches wie für das CLI. Auch dieses Programm erlaubt es, nachdem man es gestartet hat und es ein Fenster geöffnet hat, den Amiga allein über die Tastatur zu steuern. Shell und CLI sind sich sehr ähnlich und wer das CLI einmal kennengelernt hat, wird keine Schwierigkeiten mit der Shell haben. Alle Befehle, die das CLI versteht, können zum Beispiel auch zusammen mit der Shell verwendet werden. Die Shell ist aber wesentlich komfortabler als das CLI und wird deshalb üblicherweise dem wesentlich älteren CLI vorgezogen.

Mehr zu diesem Thema können Sie in den folgenden Kapiteln erfahren. Bevor Sie diese gelesen haben, sollten Sie die Shell nicht starten.

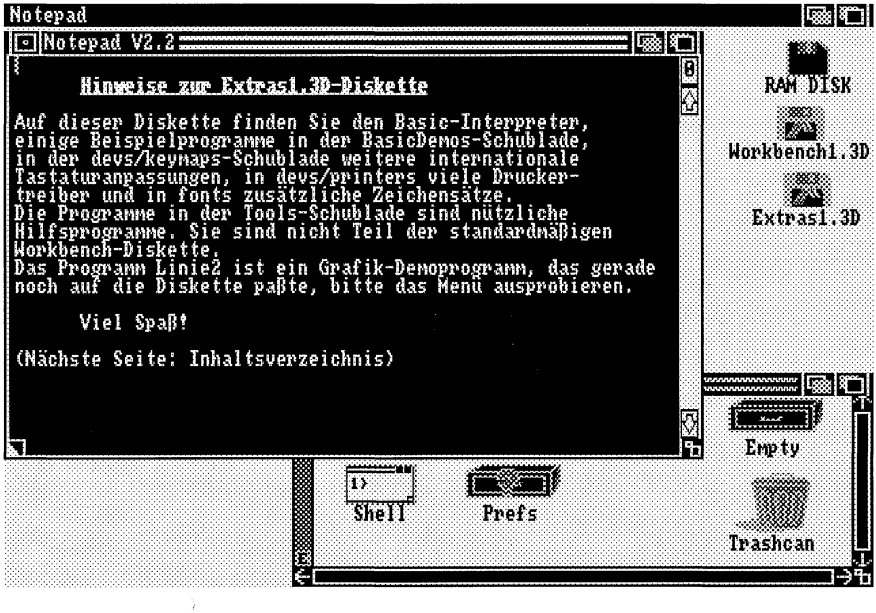
## 5.4 Werkzeuge auf der Extras-Diskette

Nicht nur auf der Workbench-Diskette, sondern auch auf der Extras-Diskette, die Sie beim Kauf des Amiga ebenfalls erhalten, befinden sich einige Werkzeuge. Wenn Sie das Diskettenpiktogramm der Extras-Diskette öffnen, sehen Sie zunächst eine Reihe von Schubladen, die Piktogramme *Amiga-BASIC*, *Hinweise* und *Linie2*, sowie den Mülleimer, dessen Funktion ja schon bekannt ist. Für Sie sind – im Augenblick – wahrscheinlich vor allem die kleinen Hilfsprogramme von Interesse, die sich in der Schublade Tools befinden. Die meisten dieser Hilfsprogramme sind für den fortgeschrittenen Anwender gedacht. Auf eines dieser Hilfsmittel, den Texteditor *MEmacs*, wird an anderer Stelle in diesem Buch noch ausführlich eingegangen. Die restlichen werden in den folgenden Abschnitten dieses Kapitels kurz beschrieben.

Die anderen Schubladen auf der Extras-Diskette enthalten Dateien, die Sie erst dann verwenden können, wenn Sie das CLI beziehungsweise die Shell kennengelernt haben. Diese Dateien sind von der Workbench aus deshalb auch unsichtbar. Lediglich die Schublade *Basic Demos* enthält eine ganze Reihe von sichtbaren Piktogrammen, bei denen es sich um kleine Beispielprogramme in der Programmiersprache Amiga-Basic handelt.

Wenn Sie mehr über die Werkzeuge erfahren wollen, die sich auf Ihrer Extras-Diskette befinden, brauchen Sie nur einen Doppelklick auf das Piktogramm *Hinweise* zu machen, das sich im Diskettenfenster der Extras-Diskette befindet. *Hinweise* ist ein Notepad-Projekt, das

lle Informationen enthält, die Sie für den Umgang mit den anderen Werkzeugen auf der :xtras-Diskette benötigen. *Notepad* wird bei einem Doppelklick auf Hinweise deshalb .utomatisch von der Workbench-Diskette gestartet und die erste Seite des Textes erscheint m Bildschirm. Die weiteren Seiten können Sie betrachten, indem Sie auf das Eselsohr in ler unteren linken Ecke des Fensters klicken.




*3ild 5.19: Betrachten der Datei »Hinweise«*

## 5.4.1 Basic

Beim Kauf eines Amiga erhalten Sie auch immer eine Programmiersprache. Diese verbirgt ich hinter dem Piktogramm *Amiga-BASIC* auf der Extras-Diskette. Amiga-Basic ist der pezielle »Amiga-Dialekt« der beliebten Basic-Sprache, die es für fast alle Computer gibt. Sie kann hier aus Platzgründen nicht näher beschrieben werden. Eine kurze Einführung in Amiga-Basic erhalten Sie im letzten Kapitel dieses Buches.

Wenn Sie wollen, können Sie Amiga-Basic aber auch sofort einmal ausprobieren und sich einen Eindruck von den Möglichkeiten dieser Programmiersprache verschaffen. Starten Sie lazu (mit einem Doppelklick) das Basic-Programm *Linie2*, das sich im Diskettenfenster hrer Extras-Diskette befindet. Dieses Programm demonstriert einen Teil der grafischen Fähigkeiten des Amiga und zeigt ein sich bewegendes, kaleidoskopähnliches Bild. (Sie ollten dieses Programm noch aus Kapitel 4 kennen, wo es zur Demonstration der ogenannten »Screens« verwendet wurde.) Wenn Sie es wieder verlassen wollen, brauchen Sie lediglich den Befehl *Ende* aus seinem einzigen Menü auszuwählen.

### 5.4.2 Für Verspielte: Das Werkzeug IconEd

Während *Preferences* ein sehr wichtiges Werkzeug ist, mit dem jeder Besitzer des Amiga umgehen können sollte, ist das in diesem Abschnitt beschriebene Werkzeug mehr etwas für Verspielte: *IconEd* ist ein Programm, mit dessen Hilfe Sie die Piktogramme (*Icons*) von Disketten, Schubladen, Werkzeugen und Projekten abändern (editieren) können.

Sie finden den Piktogramm-Editor in der Schublade *Tools* auf der Extras-Diskette (bei einigen früheren Versionen der Workbench befand er sich auch auf der Workbench-Diskette). Es ist ein relativ kleines Piktogramm mit dem Namen *IconEd*. Sie können dieses Programm wie jedes andere Werkzeug mit einem Doppelklick auf das entsprechende Piktogramm starten. Daraufhin erscheint zunächst ein Requester mit einigen weniger (englischen) Informationen über das Programm. Klicken Sie auf den Knopf *OK* und dieser Requester verschwindet und gibt den Blick frei auf das Hauptfenster von *IconEd*.

Rechts sehen Sie neun kleine Quadrate, in denen sich das Piktogramm des Programms *IconEd* befindet. Sie werden später noch sehen, daß Sie in jedes dieser Quadrate ein anderes Piktogramm laden können und dann schnell von der Bearbeitung eines Piktogramms zum nächsten wechseln können. Eines der neun kleinen Piktogramme ist dadurch etwas hervorgehoben, indem es einen anderen Rahmen besitzt (schwarz statt weiß, wenn Sie die Workbench-Farben nicht geändert haben). Dieses Piktogramm erscheint in vergrößerter Form links daneben (der Vergrößerungsfaktor beträgt etwa 4). In dieser vergrößerten Form können Sie das Piktogramm nun – ähnlich wie den Mauszeiger in *Preferences* – ändern. Der Mauszeiger wird dabei zu Ihrem Pinsel.

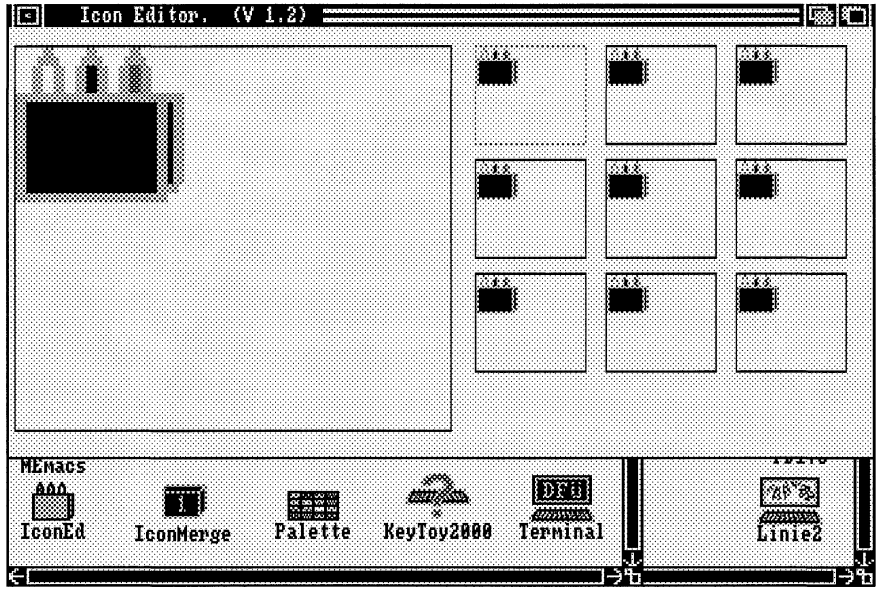



***Bild 5.20:*** *Der Piktogramm-Editor*

Zunächst aber müssen Sie eines der neun Piktogramme zur Bearbeitung auswählen. Dazu brauchen Sie nur in das Quadrat mit dem gewünschten Piktogramm zu klicken. Das ausgewählte Piktogramm wird dann schwarz umrahmt und erscheint in der Vergrößerung. Da nach dem Start alle neun Piktogramme gleich aussehen, bemerken Sie davon aber vielleicht nichts. Sie werden weiter unten noch erfahren, wie Sie andere Piktogramme von Diskette laden und in ein beliebiges der neun Quadrate plazieren können.

### Malen im Piktogramm

Sie können jedem Punkt des ausgewählten Piktogramms eine andere Farbe geben. Dazu brauchen Sie nur diesen Punkt anklicken. Das ist allerdings trotz der vergrößerten Darstellung nicht ganz leicht und verlangt einige Übung. Wenn Sie einen Punkt anklicken, erhält er die »aktuelle Farbe«. Klappen Sie einfach das Menü *Color* herunter, wenn Sie sehen wollen, was die aktuelle Farbe ist. Das Menü *Color* präsentiert die vier Workbench-Farben als schmale Streifen. Die aktuelle Farbe ist mit einem Haken versehen. Wenn Sie in einer anderen Farbe malen wollen, brauchen Sie diese andere Farbe nur wie einen Befehl aus dem Menü auszuwählen. Sie wird dann die aktuelle Farbe und wird auch abgehakt.

Sie müssen mit der Maus aber auch nicht jedesmal klicken, wenn Sie die Farbe eines Punktes ändern wollen. Wenn Sie die Selektionstaste (die linke) innerhalb der Vergrößerung niederdrücken und festhalten, können Sie mit der Zeigerspitze wie mit einem Stift oder Pinsel malen. Gerade Linien, Kreise, Rechtecke und so weiter sind auf diese Weise aber nur sehr schlecht zu zeichnen. Entsprechende Hilfsmittel wie in einem richtigen Malprogramm fehlen *IconEd*. Sie können allerdings die Farbe einer größeren Fläche recht rasch ändern. Wählen Sie dazu die gewünschte neue Farbe aus dem *Color*-Menü, wählen Sie dann aus dem Menü *Misc* den Befehl *Flood Fill* und klicken Sie einmal in die Fläche, deren Farbe Sie ändern wollen. Genauso schnell können Sie mit dem Befehl *Clear This Frame* in demselben Menü das gesamte Piktogramm löschen und »bei Null« beginnen. Probieren Sie es aus:

Klicken Sie zunächst rechts im Fenster in ein Quadrat mit einem Piktogramm, das Sie noch nicht geändert haben. Wählen Sie dann die oberste Farbe (Blau) aus dem *Color*-Menü und rufen Sie den Befehl *Flood Fill* auf. Klicken Sie nun in der vergrößerten Darstellung in das Rechteck, das den größten Teil des Piktogramms von *IconEd* ausmacht, und mit einem Schlag ist es blau. Klicken Sie nun noch einmal in dieses Rechteck und halten Sie die Maustaste fest. Bewegen Sie die Maus langsam (!) nach rechts, bis eine durchgezogene blaue Linie den Rand des Piktogramms durchbricht. Füllen Sie nun das Piktogramm mit der Farbe *Orange*. Wählen Sie dazu diese Farbe (die unterste) aus dem *Color*-Menü, rufen Sie *Flood Fill* auf und klicken Sie in die Fläche, die Sie eben blau gefärbt haben. Wie Sie sehen, läuft die Farbe bei *Flood Fill* sehr leicht aus!

### Piktogramme verschieben, austauschen und mischen

Sie müssen für größere Änderungen nicht jeden Punkt des Piktogramms einzeln bearbeiten. *IconEd* bietet auch einige Befehle für größere Manipulationen an Piktogrammen. Mit dem Befehl *In Frame* im Menü *Move* können Sie das Piktogramm zum Beispiel als Ganzes bewegen. Wenn Sie diesen Befehl aus dem Menü auswählen, erscheint rechts statt der neun kleinen Piktogramme eine Art Requester. In dessen oberer Hälfte befinden sich vier Pfeile. Sie können nun das Piktogramm in der Vergrößerung in eine beliebige Richtung schieben, indem Sie auf den Pfeil klicken, der in diese Richtung zeigt. Wenn Sie auf das Quadrat in der Mitte der vier Pfeile klicken, springt das Piktogramm wieder dahin zurück, wo es am Anfang lag.

Normalerweise bewegt sich das Piktogramm bei jedem Mausklick auf einen der vier Pfeile nur einen Punkt in die angegebene Richtung. Sie können diesen Vorgang beschleunigen indem Sie auf den Knopf *Repeat* (engl. für wiederholen) darunter klicken. Die Bewegung des Piktogramms hält dann, nachdem Sie auf einen der Pfeile geklickt haben, so lange an wie Sie die Selektionstaste der Maus festhalten. Wenn Sie den alten Zustand (der schrittweisen Bewegung) wiederherstellen wollen, brauchen Sie nur einmal auf den Knopf *Single* zu klicken. Bei jedem Mausklick bewegt sich das Piktogramm dann wieder nur einen Punkt in die Richtung, in die der angeklickte Pfeil weist.

Sie müssen den Verschiebevorgang schließlich mit einem Klick auf *OK* oder *Cance* abbrechen. *OK* bestätigt die Verschiebung, während ein Klick auf *Cancel* den alten Zustand des Piktogramms vor dem Aufruf von *Move In-Frame* unverändert beläßt.

Neben dieser Verschiebung bietet das *Move*-Menü noch einen zweiten Befehl. *Exchang With Frame* bringt ein Submenü zum Vorschein. Wenn Sie aus diesem Menü eine Zahl wählen, wird das gerade bearbeitete Piktogramm mit einem anderen der neun Piktogramm ausgetauscht. Die Anordnung der neun Zahlen im Submenü *Exchange With Frame* entspricht dabei exakt den Positionen der Piktogramme in den neun Quadraten in der rechten Fensterhälfte.

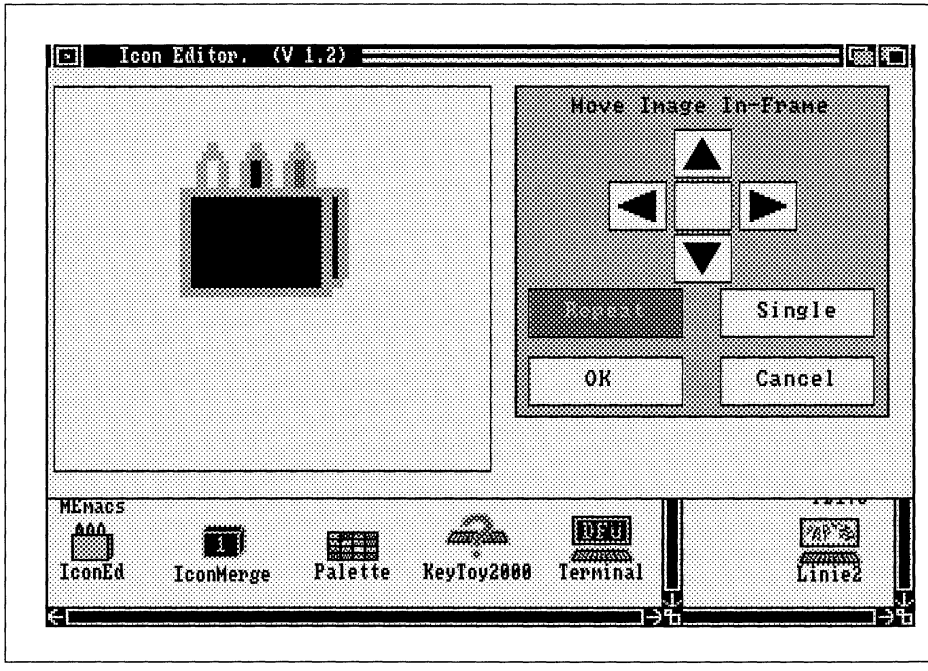



***Bild 5.21:*** *Verschieben eines Piktogramms*

Genauso einfach können Sie auch ein Piktogramm durch ein anderes ersetzen (hierbei geht Ihnen im Gegensatz zum gerade beschriebenen *Exchange With Frame* eines des neun Piktogramme verloren). Wählen Sie dazu zuerst dasjenige der neun kleinen Piktogramm

ırch Anklicken aus, das Sie ersetzen wollen. Gehen Sie dann mit der Maus auf den Befehl *·om Frame* aus dem Menü *Copy*, der wieder ein Submenü zum Vorschein bringt. Wählen e aus diesem Submenü dann die Nummer des Piktogramms, das das eben ausgewählte setzen soll.

ne interessante Alternative zum Kopieren eines ganzen Piktogramms erlaubt es Ihnen, ktogramme zu »mischen«, das heißt, die eingefärbten Punkte von zwei Piktogrammen ısammenzulegen. Klicken Sie dazu zuerst auf dasjenige der neun kleinen Piktogramme, ıs eines der beiden zu mischenden Piktogramme enthält. Wählen Sie dann aus dem *Copy*-enü *Merge with Frame* und aus dem daraufhin auftauchenden Submenü die Nummer des ıderen Piktogramms. Diese beiden Piktogramme »zusammen« ersetzen dann das aktuelle ktogramm (das mit dem schwarzen Rahmen).

ie restlichen beiden Befehle im *Copy*-Menü ermöglichen es, Ihre eigenen Fehler wieder ckgängig zu machen; oder besser: es sich nach einer größeren Änderung noch einmal ıders zu überlegen. Wählen Sie dazu vor jeder größeren Änderung an einem komplizierten ktogramm den Befehl *Snapshot Frame* aus dem Menü *Copy*. *IconEd* legt dann eine exakte opie des aktuellen Piktogramms (mit dem schwarzen Rahmen) in einen Pufferspeicher. ühren Sie nun Ihre Änderungen aus. Gefallen Ihnen die Ergebnisse Ihrer Bemühungen cht, können Sie die letzte Version des Piktogramms nun mit *Undo Frame* wieder zurückholen«. Jedesmal, wenn Sie *Undo Frame* aus dem Menü *Copy* wählen, ersetzt das it *Snapshot Frame* abgespeicherte Piktogramm das gerade bearbeitete – auch wenn sich eses an einer anderen Position befindet als das mit *Snapshot Frame* abgespeicherte!

### urze Texte in ein Piktogramm setzen

*·onEd* besitzt auch eine Möglichkeit, kurze Texte in ein Piktogramm zu plazieren. Das kann ım Beispiel ganz praktisch sein, wenn Sie ein Piktogramm erstellen wollen, das ein oder wei verfremdete Buchstaben enthält. Ansonsten sollten Sie mit Texten in Piktogrammen ɔarsam umgehen. Zum Beispiel sollte der Name eines Piktogramms nie im Piktogramm ɛlbst stehen. Er wird von der Workbench ja daruntergesetzt und kann mit dem *Rename*-efehl geändert werden.

Wenn Sie aber trotzdem einen Text in ein Piktogramm setzen wollen, brauchen Sie nur den efehl *Write Into Frame* aus dem Menü *Text* aufzurufen. Daraufhin wird das rechte Drittel es Bildschirms von einer Art Requester eingenommen. Sie müssen hier nun zunächst n Text-Gadget am oberen Rand (direkt unter Icon Text) den gewünschten Text eintragen. ı den beiden Feldern darunter können Sie die Vorder- und Hintergrundfarbe (Foreground nd Background) wählen, mit der Text im Piktogramm erscheinen wird. Sie können diese eiden Farben einfach ändern, indem Sie in das entsprechende farbige Rechteck neben oreground oder Background klicken. Im Feld Font (Schriftart) können Sie nur die chriftgröße wählen, obwohl der Name Font auch eine Wahlmöglichkeit der Schriftart uggeriert. Ihnen steht nur die Schriftart Topaz (Sie kennen sie aus dem Notepad) in den rößen 8 und 9 zur Verfügung. Die Größe 8 heißt hier Topaz_Eighty und die Größe 9 opaz_Sixty. Sie können auch wieder von einer zur anderen Größe wechseln, indem Sie auf en Schriftnamen klicken (Topaz_Eighty oder Topaz_Sixty).

Nun kommt noch die Wahl des Modus (engl. *mode*), der bestimmt, wie die neuplazierte chrift auf die Punkte im Piktogramm wirkt, die dort schon stehen, wo die Schrift erscheinen oll. Ihnen stehen vier *verschiedene* Einstellungen für Mode zur Verfügung: *JAM1*, *JAM2*, *OMPLEMENT* und INVERSVID. Im Modus *JAM1* erscheinen nur die Punkte der

Buchstaben selbst (in der Vordergrundfarbe) im Piktogramm und ersetzen die Punkte, die vorher dort standen.

Im Modus *JAM2* erscheinen sowohl die Punkte der Buchstaben (der »Vordergrund«) im Piktogramm als auch das Buchstabenumfeld (der »Hintergrund«) in der gewählten Hintergrundfarbe. Beide Gruppen von Punkten ersetzen (überdecken) die Punkte, die vorher dort standen.

Im Modus *COMPLEMENT* werden die Punkte der Buchstaben (ohne Hintergrund) mit den schon im Piktogramm stehenden Punkten kombiniert. Die Punkte, aus denen die Buchstaben bestehen, bewirken bei den Punkten des Piktogramms eine Farbänderung. Diesen Modus verstehen Sie am einfachsten durch Ausprobieren (siehe unten)! Und im Modus INVERSVID erscheint nur das Umfeld der Buchstaben in der eingestellten Hintergrundfarbe im Piktogramm. Die Buchstaben selbst sind »durchscheinend« und lassen die alten Punkte des Piktogramms stehen.

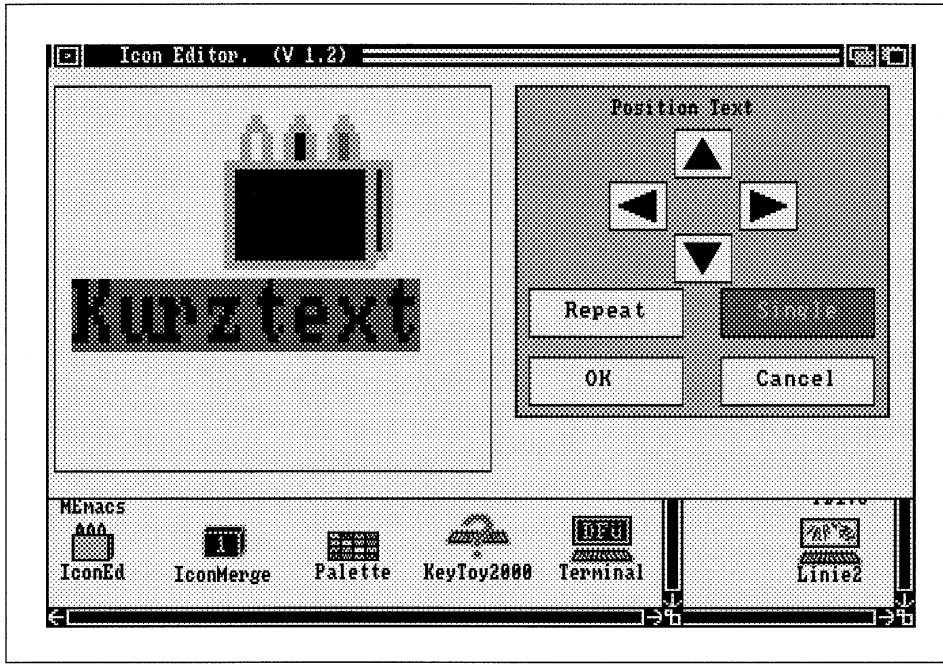



***Bild 5.22:*** *Positionieren eines Textes in einem Piktogramm*

Am besten probieren Sie die verschiedenen Modi nun, damit Sie ein Gefühl dafür bekommen. Rufen Sie dazu *Write Into Frame* aus dem Menü *Text* auf und geben Sie einen kurzen Text ein. Ändern Sie dann gegebenenfalls die beiden Farben (keine der Farben sollte gleich der Farbe des Bildschirmhintergrunds sein) und klicken Sie auf den Knopf *Position* (engl. *to position* = positionieren). Es erscheint nun ein *Requester*, der dem ähnlich sieht, mit dem Sie auch das Piktogramm im Rahmen verschieben können (*Move In-Frame*). Gleichzeitig erscheint der eingegebene Text im Piktogramm. Sie können ihn nun mit den vier Pfeilen

bewegen. Dabei können Sie verfolgen, wie die Punkte der Buchstaben beziehungsweise ihr Umfeld auf das alte Piktogramm wirken. Klicken Sie nun auf *Cancel* und brechen Sie den Vorgang damit ab. Ändern Sie nun den Modus (*Mode*) und klicken Sie erneut auf Position. Bewegen Sie nun wieder den Text über das Icon und beachten Sie die Wirkung von Buchstaben und Buchstabenhintergrund. Klicken Sie dann wieder auf *Cancel* und probieren Sie die beiden anderen Modi auch noch aus.

Wenn Sie den Requester *Position Text*, der nach dem Klick auf *Position* erscheint, mit einem Klick auf *OK* beenden, bleibt der Text genau an der Stelle im Piktogramm stehen, an der er zuletzt zu sehen war. Auf diese Weise plazieren Sie also den vorher eingegebenen Text im Piktogramm und kehren zurück zum Requester mit dem Titel *Icon Text*. Falls Ihnen der Text dann nicht oder an dieser Stelle nicht gefallen sollte, können Sie diese Änderung nun immer noch mit einem Klick auf *Cancel* im Requester *Position Text* rückgängig machen und unverrichteter Dinge zum Hauptfenster zurückkehren. Nur wenn Sie im Requester *Position Text* auf *OK* klicken, werden die Eingaben und Änderungen, die Sie nach dem Aufruf des Befehls *Write Into Frame* gemacht haben, wirksam.

### Ein Piktogramm zum Ändern einlesen

Nun wissen Sie schon, wie Sie alle möglichen Dinge mit einem Piktogramm anstellen können, aber noch nicht, wie Sie beliebige Piktogramme von der Diskette einlesen, damit Sie überhaupt etwas zum Ändern haben. Das werde ich deshalb nun schleunigst nachtragen.

Bevor Sie ein Piktogramm einlesen, klicken Sie zuerst auf eines der neun kleinen Quadrate in der rechten Bildschirmhälfte. In diesem Quadrat wird das neu eingelesene Piktogramm erscheinen. (Sie können auf diese Weise mit bis zu neun verschiedenen Piktogrammen gleichzeitig arbeiten.) Wählen Sie dann den Befehl *Load Data* aus dem Menü *Disk*. Daraufhin erscheint ein Requester in der rechten Bildschirmhälfte, der die Überschrift *Load Icon Image Data* trägt. Er enthält am oberen Rand ein Text-Gadget, in das Sie den Namen des Piktogramms eintragen müssen, das Sie einlesen wollen. (Sie können den alten Namen, der schon dort steht, mit der Tastenkombination [Ctrl]+[X] löschen.)

In den meisten Fällen genügt es jedoch nicht, hier nur den Namen einzutragen, der unter dem Piktogramm steht. Sie müssen *IconEd* auch mitteilen, in welcher Schublade sich das Piktogramm befindet. Nur wenn sich das Piktogramm in derselben Schublade befindet wie *IconEd* selbst, reicht der Name des Piktogramms.

Für Piktogramme, die in anderen Schubladen liegen, müssen Sie zunächst einen Doppelpunkt (:) eingeben und dann hintereinander die Namen der Schubladen angeben, die geöffnet werden müßten, um das gewünschte Piktogramm zu finden. Diese Namen müssen durch Schrägstriche getrennt werden. Auch hinter dem Namen der letzten Schublade muß noch ein Schrägstrich gesetzt werden und dahinter dann schließlich der Name des gewünschten Piktogramms. Um das Piktogramm *Müller* in der Schublade *Briefe*, die selbst wieder in der Schublade *Texte* liegt, einzulesen, müßten Sie in diesem Textrequester also den Namen *:Texte/Briefe/Müller* eintragen. (Das ist ein sogenannter »Pfadname«, über den Sie im nächsten Kapitel noch mehr erfahren werden.)

Auf diese Art können Sie das Piktogramm jeder Schublade, jedes Werkzeugs, Projektes und auch des Mülleimers einlesen. Das Schubladenpiktogramm *Briefe* aus dem obigen Beispiel hätte zum Beispiel den Namen *:Texte/Briefe* und der Mülleimer auf derselben Diskette den Namen *:Trashcan*. Wenn Sie jedoch das Piktogramm einer Diskette modifizieren wollen,

müssen Sie etwas anders vorgehen. Das Piktogramm der Diskette, auf der sich der Piktogrammeditor selbst befindet, trägt den Namen *:Disk* (achten Sie auf den Doppelpunkt am Anfang). Das Piktogramm einer anderen Diskette hat den Namen *<Diskettenname>:Disk*, wobei Sie *<Diskettenname>* durch den Namen ersetzen müssen, der unter dem Piktogramm dieser Diskette auf der Workbench erscheint.

Wenn Sie den Namen des Piktogramms eingetragen haben, können Sie auf den Knopf *Load Icon* Image klicken (wenn Sie es sich anders überlegt haben, auf den Knopf *Cancel The Load*). Das neue Piktogramm sollte dann im vorher aktivierten (angeklickten) Quadrat und in der Vergrößerung in der linken Bildschirmhälfte erscheinen. Sie können es nun nach Herzenslust ändern.

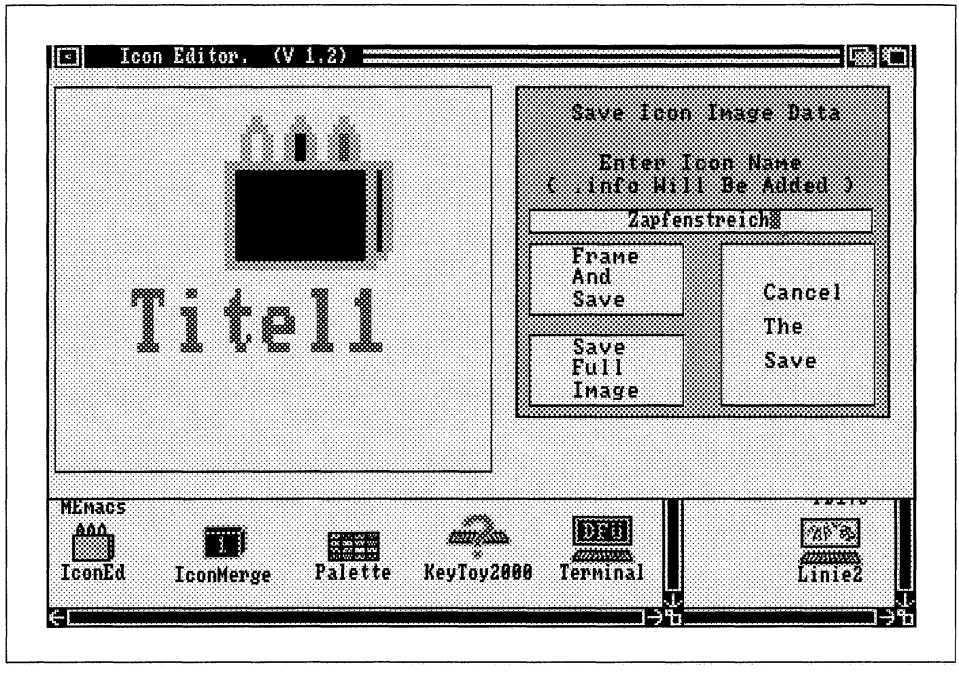



***Bild 5.23:*** *Abspeichern eines Piktogramms*

Wenn Sie bei der Eingabe des Namens einen Fehler gemacht haben, erscheint allerdings kein neues Piktogramm. Statt dessen steht in der Titelzeile *Icon Editor - Loading error (Press mouse button)*. Sie sollten dann noch einmal *Load Data* im *Disk*-Menü aufrufen und versuchen, den richtigen Namen, wie oben beschrieben, einzugeben. Falls Sie nicht verstanden haben, wie der lange Name mit dem Doppelpunkt und den Schrägstrichen zustande kommt, sollten Sie Ihre *IconEd*-Experimente vielleicht erst einmal abschließen und Kapitel 6 lesen. Eine andere Alternative ist es, zunächst einmal *IconEd* zu verlassen (indem Sie das Hauptfenster schließen) und dann das Piktogramm *IconEd* in dieselbe Schublade zu legen, in dem auch das Piktogramm liegt, das Sie einlesen wollen. Oder Sie können dieses Piktogramm auch in dieselbe Schublade legen, in der sich *IconEd* befindet.

Wenn Sie dann noch einmal den *IconEd* starten und *Load Data* im *Disk*-Menü aufrufen, brauchen Sie im Requester *Load Icon Image Data* nur den Namen einzutragen, der auf der Workbench unter dem Piktogramm erscheint, das Sie ändern wollen.

### Ein Piktogramm abspeichern

Wenn Sie alle Änderungen an einem Piktogramm abgeschlossen haben, wollen Sie es natürlich auch wieder abspeichern, beziehungsweise gegen das alte Piktogramm ersetzen. Klicken Sie in diesem Fall auf dasjenige der neun kleinen Piktogramme in der rechten Fensterhälfte, das Sie abspeichern wollen, und rufen Sie dann den Befehl *Save Data* im Menü *Disk* auf. Es erscheint ein Requester, der dem sehr ähnlich sieht, der nach *Load Data* auftaucht. Auch hier müssen Sie wieder in dem Text-Gadget am oberen Rand den Namen eintragen, unter dem das Piktogramm gesichert werden soll. Wie Sie solche Namen für Piktogramme bilden, die nicht in derselben Schublade liegen wie der *IconEd*, können Sie im letzten Abschnitt nachlesen. Achten Sie sehr sorgfältig auf Tippfehler, sonst kann es Ihnen passieren, daß Sie ein neues Piktogramm abspeichern, obwohl Sie eigentlich ein altes überschreiben wollten.

Wenn Sie den Namen eingetragen haben, können Sie das Piktogramm mit einem Klick auf *Save Full Image* oder *Frame And Save* auf die Diskette schreiben. Wenn Sie auf *Save Full Image* klicken, wird das Piktogramm in seiner vollen Größe, wie es im Rahmen rechts daneben gezeigt wird, abgespeichert. Dies ist meist nicht das, was Sie wollen. Das »eigentliche Piktogramm« macht ja meist nur einen kleinen Teil des gesamten Quadrats aus, das bearbeitet werden kann. Klicken Sie in solchen Fällen auf den Knopf *Frame And Save* (zu deutsch etwa »umrahmen und dann sichern«). Sie können nun mit der Maus den Bereich markieren, der wirklich als Piktogramm abgespeichert werden soll. Klicken Sie dazu in der Vergrößerung einmal in die linke obere Ecke des Rechtecks, das das »eigentliche Piktogramm« vollständig umschließt. Gehen Sie dann mit der Mausspitze in die rechte untere Ecke dieses Rechtecks. Ein dünner Rahmen markiert daraufhin die gewünschte Größe. Wenn er genau den Bereich umschließt, den Sie abspeichern wollen, müssen Sie ein zweites Mal auf die Selektionstaste drücken. Das Piktogramm wird dann abgespeichert. Wenn Sie *IconEd* nun verlassen wollen, brauchen Sie nur das Hauptfenster zu schließen (mit dem Schließ-Gadget).

Falls es aus irgendeinem Grund nicht möglich ist, das Piktogramm unter dem von Ihnen angegebenen Namen abzuspeichern, erscheint wieder eine Fehlermeldung in der Titelleiste des Fensters. Die häufigsten Gründe dafür sind der aktivierte Schreibschutz einer Diskette und ein Tippfehler in dem langen Namen eines Piktogramms aus einer anderen Schublade.

Sie können beim Ändern von Piktogrammen aber noch ein paar andere Fehler machen, die nicht so leicht zu entdecken sind. Sie können diese Fehler leicht vermeiden, wenn Sie die folgenden Punkte beachten:

Sichern Sie ein Piktogramm eines bestimmten Typs niemals unter einem Namen ab, der zu einem anderen Typ gehört! Es gibt insgesamt fünf Typen von Objekten, denen fünf verschiedene Arten von Piktogrammen entsprechen: Disketten, Schubladen, Mülleimer, Projekte und Werkzeuge. Wenn Sie ein Piktogramm eines dieser Typen von Objekten ändern wollen, sollten Sie zunächst ein Piktogramm desselben Typs laden (vielleicht sogar genau das, das Sie auch ändern wollen), es modifizieren und dann unter demselben oder einem anderen Namen abspeichern. Der Typ des Objekts, dessen Piktogramm Sie laden, sollte immer dem Typ des Objekts gleich sein, unter dessen Namen Sie das Piktogramm

wieder abspeichern. Sie dürfen also zum Beispiel kein Schubladen-Piktogramm einlesen und es dann unter dem Namen eines *Notepad*-Projektes (Notizblocks) abspeichern.

### Das Erscheinungsbild des Piktogramms bestimmen

Damit haben Sie nun schon fast alle Möglichkeiten des Piktogramm-Editors *IconEd* kennengelernt. Sie können zum Beispiel vorhandene Piktogramme von der Diskette einlesen, abändern und dann wieder unter demselben oder einem anderen Namen abspeichern. *IconEd* bietet zusätzlich noch zwei Befehle, mit denen man das Erscheinungsbild selbsterstellter oder modifizierter Piktogramme auf der Workbench abändern kann.

Die erste dieser Möglichkeiten verbirgt sich hinter dem Befehl *Set Bottom Border* im Menü Misc. Dies ist ein Submenü, das Ihnen die beiden Möglichkeiten 0 und 1 zur Wahl bietet. Die gerade gültige Einstellung ist wie üblich mit einem Haken versehen. Wenn Sie 1 wählen und dann das gerade bearbeitete Piktogramm abspeichern, läßt die Workbench zwischen dem unteren Rand des Piktogramms und dem darunterstehenden Namen immer eine dünne Linie frei. Das sieht besonders bei Piktogrammen mit einer geraden unteren Kante besser aus. Wählen Sie hingegen im Submenü *Set Bottom Border* die Möglichkeit *0*, so geht der obere Rand des Namens bis an den unteren Rand des Piktogramms.

Das Menü *HiLite* (engl. *to hilite* = aufhellen/hervorheben) bietet schließlich noch die Wahl zwischen *Inverse* und *Backfill*. Ihre Wahl in diesem Menü bestimmt, wie ein Piktogramm, das Sie abgespeichert haben, auf der Workbench hervorgehoben wird, wenn Sie es anklicken. Wählen Sie *Inverse*, wird es invertiert, das heißt, die vier Farben, aus denen es besteht, werden untereinander ausgetauscht. Genau dasselbe passiert auch bei *BackFill*. Im Unterschied zu Inverse ändern bei Backfill jedoch nicht die Gebiete eines Piktogramms ihre Farbe, die erstens die Hintergrundfarbe (die vierte im *Color*-Menü) tragen und zweitens an den Rand des Rechtecks grenzen, das das Piktogramm umschließt. Das sieht aus, als würde dieser Rand in der Hintergrundfarbe gar nicht zum Piktogramm gehören (soll er meist auch nicht). Das ist die am häufigsten verwendete Art, ein Piktogramm hervorzuheben.

Wenn Sie diese beiden Arten der Hervorhebung einmal direkt vergleichen wollen, brauchen Sie nur die *System*-Schublade auf der Workbench-Diskette zu öffnen. Das Piktogramm *SetMap* darin verwendet zur Hervorhebung die *Inverse*-Methode, während alle anderen Piktogramme, zum Beispiel *NoFastMem* die *Backfill*-Methode verwenden.

## 5.4.3 IconMerge

Es gibt noch eine dritte Methode ein Piktogramm »aufleuchten« zu lassen beziehungsweise hervorzuheben. Bei dieser Methode, die Sie mit *IconEd* nicht festlegen können, wird das markierte Piktogramm komplett durch ein anderes ersetzt. Wenn Sie ein Beispiel dafür sehen wollen, brauchen Sie nur das Piktogramm *IconMerge* in der *Tools*-Schublade anzuklicken. Auch Schubladen selbst zeigen ein ähnliches Verhalten, wenn man sie anklickt, und sie dann »aufgehen«.

Die Aufgabe des Programms *IconMerge* ist es, genau solche Piktogramme zu schaffen, deren Aussehen sich bei der Selektion radikal ändert. Dazu werden zunächst einmal zwei einzelne Piktogramme benötigt, die Sie mit *IconEd* erzeugen können. Diese Piktogramme können Sie dann zu einem zweigesichtigen Doppel-Piktogramm verschmelzen. Die genaue Funktionsweise von *IconMerge* ist aber mit Worten allein schlecht zu beschreiben. Probie-

ren Sie es am besten einfach einmal aus. Dazu brauchen Sie sich nicht einmal die Mühe machen, zwei neue Piktogramme zu zeichnen. Sie können auch zwei beliebige schon vorhandene Piktogramme zu einem neuen verschmelzen.

Rufen Sie dazu *IconMerge* auf. Es erscheint dann ein neues Fenster am Bildschirm, indem Sie zunächst einmal gefragt werden »Merge or split [m or s]:« antworten Sie auf diese Frage durch Eingabe eines »m« und drücken Sie die [Return]-Taste, denn Sie wollen zwei Piktogramme verschmelzen (engl. *to merge* = verschmelzen, mischen).

Sie werden nun nacheinander nach den Namen von insgesamt drei Piktogrammen gefragt (*Icon1* bis *Icon3*). Die ersten sind die Namen der beiden Piktogramme, die zu einem verschmolzen werden sollen, den dritten Namen erhält das neue Piktogramm. Dieser dritte Name muß kein neuer Name sein. Es kann sich auch um den Namen eines schon existierenden Piktogramms handeln, dessen Aussehen geändert werden soll. Geben Sie beispielsweise einmal die folgenden drei Namen ein:

```
Icon1: FEd
Icon2: PerfMon
Icon3: Neues Pikt
```

Dann arbeitet das Diskettenlaufwerk einen Moment lang und das von *IconMerge* geöffnete Fenster wird wieder geschlossen. Es gibt jetzt ein neues Piktogramm namens *Neues Pikt* in der Schublade *Tools*. Es ist jedoch noch nicht sichtbar. Um es sichtbar zu machen, brauchen Sie aber nur das Fenster Tools zu schließen und dann wieder zu öffnen. Das neue Piktogramm liegt dann an der Stelle, an der vorher *FEd* lag. Genauer gesagt, liegt es über *FEd*. Ergreifen Sie es deshalb jetzt bitte und legen Sie es beiseite.

Schon beim Ergreifen können Sie sehen, wie es seine Form ändert. Es ist entweder ein kleiner Kasten mit den Buchstaben a, b und c darin oder ein Häschen. So einfach ist es, ein Piktogramm zu erzeugen, das bei der Auswahl sein Aussehen vollständig ändert.

Beachten Sie aber bitte auch, daß manchmal »Reste« des Häschens übrig bleiben, wenn der Kasten erscheint. Das liegt daran, daß die beiden Piktogramme, die gegeneinander ausgetauscht werden, nicht exakt gleich groß sind. So etwas sollten Sie deshalb bei der Verwendung von *IconMerge* eigentlich vermeiden. Für unser Beispiel spielt es aber keine Rolle.

*IconMerge* kann aber nicht nur zwei Piktogramme zu einem verschmelzen, sondern auch ein zweigesichtiges Piktogramm in zwei einzelne aufspalten. Auch das ist nicht schwierig. Rufen Sie einfach *IconMerge* ein zweites Mal auf und geben Sie auf die Frage »Merge or split [m or s]:« diesmal ein »s« ein (engl. *to split* = aufspalten, aufteilen). Sie werden jetzt wieder nach drei Piktogrammnamen gefragt, die dieses Mal aber eine andere Bedeutung haben.

Der erste Namen muß der eines zweigesichtigen Piktogramms sein. Dessen beide Teile werden dann unter den Namen, die Sie an zweiter und dritter Stelle angeben, auf der Diskette gespeichert. Um das eben zusammengefügte Piktogramm wieder aufzuspalten, können Sie zum Beispiel die folgenden drei Namen eingeben:

```
Icon1: Neues Pikt
Icon2: Pikt2
Icon3: Pikt3
```

Danach sollte es zwei neue Piktogramme im Fenster Tools geben (sobald Sie dieses Fenster einmal geschlossen und dann wieder geöffnet haben). Sie müssen allerdings eventuell die Piktogramme erst ein wenig bewegen, um sie alle sehen zu können. *Pikt2* und *Pikt3* liegen exakt übereinander. Alle drei Piktogramme – auch *Neues Pikt* – sind jetzt wieder »einfache« Piktogramme, die durch Invertierung hervorgehoben werden, wenn man Sie anklickt.

Zum Abschluß Ihrer Experimente mit *IconMerge* sollten Sie die drei neuen Piktogramme am besten möglichst rasch löschen. Sie verwirren nämlich nur, weil es zwar Piktogramme von Programmen sind, aber es sich nicht um Programme handelt, die gestartet werden können. *IconMerge* kann zwar neue Piktogramme erzeugen. Dahinter steckt aber nie ein Programm oder eine andere Datei. Es handelt sich sozusagen um Piktogramme ohne Inhalt.

Für die Piktogrammnamen, die *IconMerge* von Ihnen verlangt, gilt im Prinzip das gleiche wie für die Piktogrammnamen, die Sie in *IconEd* angeben. Um ein Piktogramm zu beschreiben, das in derselben Schublade wie das Programm liegt, reicht es aus, seinen »Kurznamen«, also den Namen der unter dem Piktogramm steht, anzugeben. Befindet sich ein Piktogramm aber in einer anderen Schublade, müssen Sie den »Pfadnamen« angeben. Wenn Sie das vermeiden wollen, müssen Sie entweder *IconMerge* in die Schublade mit den Piktogrammen, die verschmolzen werden sollen, legen oder die Piktogramme in die Schublade *Tools* legen.

### 5.4.4 MEmacs

Ein sehr wichtiges Werkzeug in der Tools-Schublade ist *MEmacs*. Dabei handelt es sich um einen sehr leistungsfähigen Text-Editor, eine Art Super-*Notepad*. Anders als bei *Notepad* haben Sie bei *MEmacs* allerdings nicht die Möglichkeit, verschiedene Schriften, Schriftgrößen, Schriftfarben und so weiter zu verwenden. Statt dessen erhalten Sie aber eine Vielzahl mächtiger Befehle für die Bearbeitung umfangreicher Texte. *MEmacs* ist vor allem für die Arbeit mit dem CLI und der Shell wichtig. Das Programm wird deshalb nicht an dieser Stelle sondern erst in Kapitel 8 ausführlich beschrieben.

### 5.4.5 KeyToy2000

*KeyToy2000* ist ein Programm, mit dem Sie sich jederzeit darüber informieren können, welche Taste oder Tastenkombination welchen Buchstaben ergibt. Das klingt auf den ersten Blick vielleicht etwas merkwürdig, denn schließlich sind ja diese Buchstaben auf den Tasten aufgedruckt. Das sind aber nicht alle Buchstaben, die Sie mit der Tastatur erzeugen können. Mit Hilfe der [Alt]-Tasten können auch eine Vielzahl von Sonderzeichen und Symbolen eingegeben werden, die nicht auf den Tasten zu sehen sind.

Wenn Sie *KeyToy2000* starten, erscheint ein Fenster mit einer zwar schematischen aber recht exakten grafischen Darstellung der Tastatur. Die Tasten in dieser Grafik sind in etwa so beschriftet wie die echten Tasten. Sie können sich nun betrachten, welche Buchstaben die Tasten ergeben würden, wenn Sie gleichzeitig eine der Tasten [Shift], [Alt] und [Ctrl] festhalten, wenn Sie die jeweilige Taste drücken. Klicken Sie dazu eine dieser drei Sondertasten mit der Maus an oder drücken Sie kurz auf die »richtige« Taste auf Ihrer Tastatur. Sie wird dann praktisch arretiert und der Rest der Tasten ist nun mit den Buchstaben

beschriftet, die erscheinen, wenn Sie die arretierte Sondertaste zusammen mit einer anderen Taste drücken. Wenn Sie zum Beispiel die [Shift]-Taste mit einem Mausklick arretieren, ändert diese ihre Farbe und auf den Tasten im mittleren Bereich der Tastatur erscheinen Großbuchstaben (zuvor waren es Kleinbuchstaben).

Sie können auch zwei Sondertasten (zum Beispiel [Alt] und [Shift]) simultan arretieren. Dazu brauchen Sie nur die beiden Tasten nacheinander anzuklicken. Und um eine arretierte Taste wieder zu lösen, müssen Sie sie ein zweites Mal anklicken beziehungsweise ein zweites Mal auf der Tastatur niederdrücken.

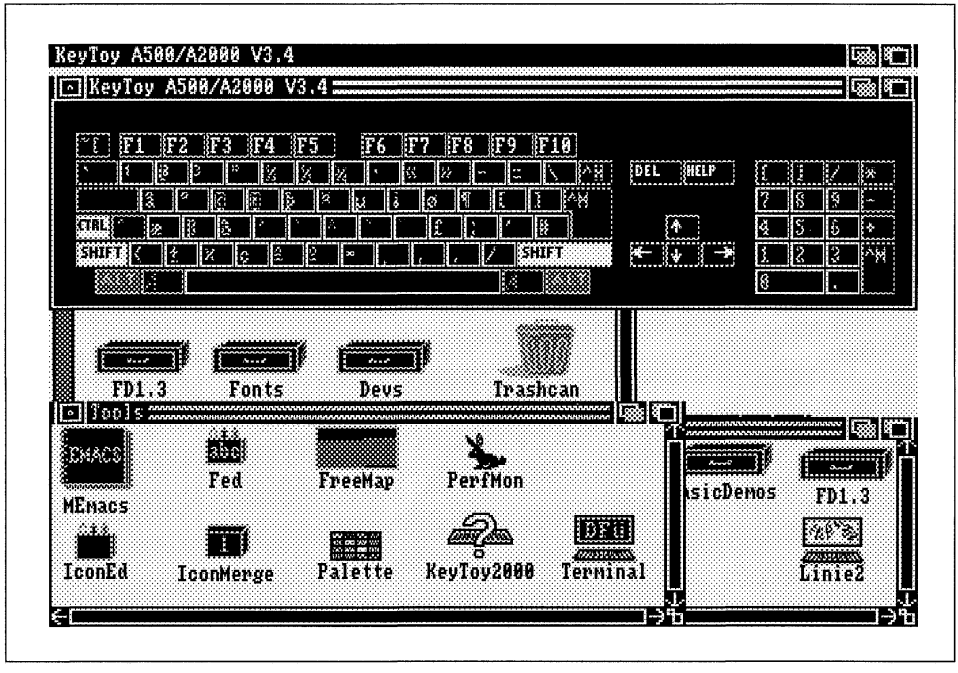



***Bild 5.24:*** *KeyToy2000*

Achten Sie bitte auch darauf, daß einige Buchstaben im mittleren Bereich der Tastatur in einer anderen Farbe (üblicherweise in Orange) erscheinen, wenn Sie die [Alt]-Taste arretieren. Bei diesen Buchstaben handelt es sich um Akzente, wie sie in den romanischen Sprachen verwendet werden. Die entsprechenden Tasten ergeben – zusammen mit der [Alt]-Taste – beim Niederdrücken nicht sofort einen Buchstaben. Sie müssen statt dessen erst eine weitere Taste drücken. Dann erscheint der auf der zweiten Taste angegebene Buchstabe und darüber der zuvor gedrückte Akzent.

Die Tastenfolge ['] und [a] ergibt zum Beispiel das Zeichen á. Akzente können selbstverständlich nicht über beliebige Buchstaben gesetzt werden, sondern nur über solche, bei denen das auch einen Sinn gibt. Die Akzente können nur mit Vokalen, die Tilde (~) nur mit den Buchstaben N und O kombiniert werden.

### 5.4.6 Palette

Das Programm *Palette* dient dazu, die Farben, mit denen die Texte, Grafiken und Piktogramme am Bildschirm gezeichnet werden, zu betrachten und zu verändern – ähnlich wie im Hauptfenster von *Preferences*. Man nennt eine solche Ansammlung von Farben, die einem für die am Bildschirm gezeigten Grafiken zur Verfügung stehen, übrigens auch eine »Palette«; daher der Name des Programms.

Wenn Sie *Palette* starten, taucht ein neues kleines Fenster auf, das drei Schieberegler, drei Tasten und am oberen Rand einige rechteckige Felder zeigt. Diese rechteckigen Felder sind die Palette beziehungsweise enthalten die Farben der aktuellen Palette. Sie können diese Farben ändern, indem Sie das entsprechende Feld anklicken und dann die Regler darunter verschieben. Wenn Sie dann auf die Taste *OK* klicken, verschwindet das Fenster wieder und Ihre Änderungen an den Farben werden wirksam. Mit der Taste *Cancel* können Sie das Fenster verschwinden lassen, ohne die Bildschirmfarben zu ändern. Alle bis dahin durchgeführten Änderungen werden ungültig. Und mit der Taste *Reset* können Sie die ursprünglichen Farben (vor dem Start von *Palette*) wiederherstellen, ohne das Fenster zu schließen.

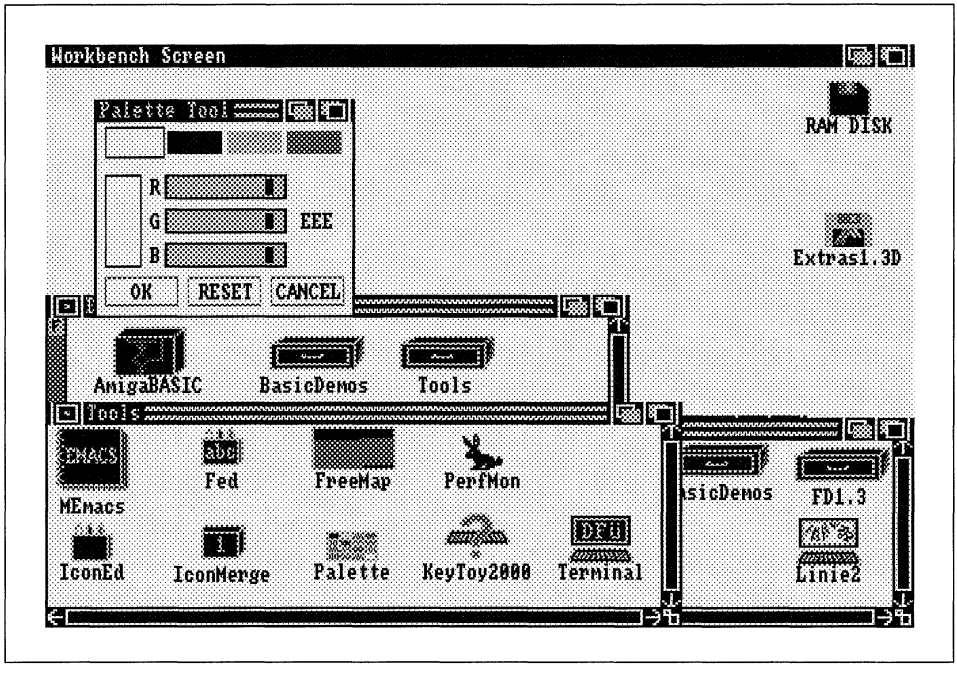



***Bild 5.25:*** *Ändern der Bildschirmfarben mit Palette*

Mit Hilfe von *Palette* können Sie aber nicht nur die Farben der Workbench verändern. Wie Sie in Kapitel 4 erfahren haben, kann es ja auf dem Bildschirm mehrere verschiedene »virtuelle Bildschirme«, »Screens« genannt, geben. Jeden dieser Bildschirme können Sie mit *Palette* manipulieren. Dazu muß er lediglich der vorderste sein. Das *Palette*-Fenster erscheint nämlich immer im vordersten Screen.

Um *Palette*, dessen Piktogramm im Workbench-Screen liegt, starten zu können, wenn ein anderer Screen darüber liegt, müssen Sie wie folgt vorgehen: Ergreifen Sie den vordersten Screen mit der linken Taste der Maus an der Titelleiste und ziehen Sie ihn fast bis nach unten. Jetzt sollten Sie den Workbench-Screen wieder sehen können. Öffnen Sie die Extras-Diskette, dann den *Tools*-Ordner und starten Sie das Programm *Palette*. Ziehen Sie nun den vordersten Screen, dessen Titelleiste am unteren Bildschirmrand noch zu ist, wieder nach oben. Das *Palette*-Fenster sollte jetzt mitten in diesem Screen zu sehen sein und auch dessen Farben enthalten.

### 5.4.7 FreeMap

Einige der Werkzeuge, die im Ordner *Tools* zu finden sind, kann fast jeder Amiga-Besitzer ab und zu einmal gebrauchen. Andere sind nur für Spezialisten von Interesse und verlangen zusätzliche Kenntnisse, um sie sinnvoll anwenden zu können. Zu dieser Kategorie von Werkzeugen gehören auch die beiden Programme *FreeMap* und *PerfMon*.

*FreeMap* öffnet nach dem Start einen Screen (einen eigenen virtuellen Bildschirm), der eine grafische Übersicht der belegten und freien Bereiche im RAM-Speicher des Amiga zeigt. Jeder Punkt in dieser Speicherkarte (engl. *map* = Karte) entspricht einem Speicherblock von 64 Byte Größe. Ist der Punkt schwarz, dann ist der entsprechende Bereich frei und steht Programmen, die zusätzlichen Speicher anfordern zur Verfügung. Ist aber ein Teil der Speicherplätze in dem 64er-Block belegt, leuchtet der entsprechende Punkt auf.

Am unteren Rand der Darstellung werden diese Informationen noch einmal zahlenmäßig zusammengefaßt. Hier erfahren Sie, wieviel Bytes noch insgesamt frei sind, wobei zwischen *Chip Mem* und *Fast Mem* unterschieden wird. (Der Unterschied zwischen diesen Speichertypen wurde in Abschnitt 5.3 kurz beschrieben.

Der von *FreeMap* geöffnete Screen ändert sich während der Arbeit kontinuierlich. Wenn Sie wollen, können Sie die Speicherkarte längere Zeit auf dem Bildschirm stehen lassen und so zum Beispiel recht genau feststellen, welches Programm wieviel Speicher von welcher Sorte benötigt. Falls Sie das nicht interessiert, können Sie *FreeMap* jederzeit beenden (und den Screen damit verschwinden lassen). Wählen Sie dazu einfach den Befehl *Please* aus dem Menü *Quit* von *FreeMap*. Beachten Sie, daß Sie dieses Menü in der Titelleiste des *FreeMap*-Screens finden und nicht in der Titelleiste der Workbench am oberen Bildschirmrand!

### 5.4.8 PerfMon

*FreeMap* liefert Ihnen sozusagen einen Überblick über die Auslastung des Amiga-Hauptspeichers. In ähnlicher Form können Sie mit *PerfMon* die Auslastung der Amiga-CPU (also des zentralen 68000-Chips) und des Hauptspeichers überwachen. Starten Sie dazu *PerfMon* und es erscheint ein neues Fenster am Bildschirm, dessen Inhalt zunächst ein wenig simpel aussieht. Warten Sie am besten einen Moment, dann wird die Bedeutung und der Aufbau dieses Fensters klarer. Bewegen Sie während der Wartezeit am besten ein wenig die Maus und starten Sie auch einmal ein anderes Programm.

Das *PerfMon*-Fenster enthält drei Kurven. In der oberen Bildhälfte wird eine Kurve gezeichnet, die aussagt, wieviel Zeit die Amiga-CPU mit »Nichtstun« beziehungsweise dem Warten auf neue Arbeit verbringt. Sobald Sie die Maus bewegen, Programme starten oder

Programmen Befehle erteilen, geht diese Kurve nach unten. Der Leerlauf der CPU wird also geringer. Wenn Sie ein Fenster bewegen, geht die Kurve sogar bis auf Null zurück. Die CPU ist dann voll ausgelastet. Aber auch ein Programm (wie zum Beispiel das Basic-Programm *Linie2* auf der Extras-Diskette) kann die CPU voll auslasten. Sie können das ausprobieren, indem Sie *Linie2* aufrufen und dann sofort wieder verlassen. Im *PerfMon*-Fenster können Sie dann die tiefen Einschnitte sehen, die dadurch hervorgerufen wurden.

Beim Start eines Programms oder beim Öfnnen eines zusätzlichen Workbench-Fensters ändern sich auch die beiden Kurven in der unteren Fensterhälfte. Diese beiden Kurven stellen dar, wieviel RAM-Speicher noch frei ist. Auch *PerfMon* unterscheidet dabei, wie *FreeMap*, zwischen *Chip Mem* und *Fast Mem* (siehe Abschnitt 5.3).

Die von *PerfMon* dargestellten Kurven werden permanent aufgefrischt. Sie können deshalb während der Arbeit mit anderen Programmen beobachten, wie Speicher verbraucht, wieder freigegeben und wie die CPU belastet wird. *PerfMon* selbst belastet die CPU naturgemäß auch und zwar um so stärker, je häufiger die Kurve am Bildschirm auf den aktuellen Stand gebracht wird. Sie haben deshalb die Möglichkeit, die Abstände zwischen den Zeitpunkten, zu denen *PerfMon* die Kurven aktualisiert, zu verändern. Dazu dient *PerfMons* einziges Menü, in dem Sie zwischen drei Abständen wählen können: eine halbe, eine oder zwei Sekunden. Der vierte Punkt in diesem Menü ist kein echter Befehl. Er gibt lediglich an, welche Aktualisierungsrate zuletzt gewählt wurde. Und der letzte Menübefehl *Info* stellt den Autor des Programms vor.

### 5.4.9 Fed

Das Programm *Fed* dient zum Editieren von Schriften (*Fed* = ***F**ont **Ed**itor*) und zum Erzeugen neuer Schriften oder neuer Schriftgrößen. Es funktioniert ähnlich wie *IconEd*, der Piktogramm-Editor. Zum Erzeugen und Ändern von Schriften gehören aber nicht nur eine Menge Fleiß sondern vor allem auch Kenntnisse der Schrifttheorie, die in dem hier zur Verfügung stehenden Raum unmöglich vermittelt werden können. Ich erspare mir deshalb die Beschreibung des Fed (die recht umfangreich ausfallen müßte). Wer sich für dieses Thema interessiert und sich mit dem Aufbau von Schriften und Buchstabenformen auskennt, wird das Programm sicherlich mit ein wenig Herumprobieren rasch erlernen können.

Für den, der mit dem Gedanken spielt, sich eine eigene, »persönliche« Schrift zu gestalten, aber eine Warnung vorweg: In den Schriften, die Ihnen auf dem Amiga zur Verfügung gestellt werden, steckt eine Menge Arbeit. An professionellen Schriften (in vielen verschiedenen Größen) arbeiten professionelle Designer und Künstler oft ein halbes Jahr oder mehr. Es handelt sich hierbei auf keinen Fall um ein Projekt für ein verregnetes Wochenende!

### 5.4.10 Terminal

Bei dem letzten Programm in der *Tools*-Schublade handelt es sich um ein (sehr einfaches) Telekommunikationsprogramm. Mit dem Programm *Terminal* und einem sogenannten »Modem« können Sie über eine normale Telefonverbindung Kontakt mit einem anderen Computer aufnehmen. Steht der andere Computer in der Nähe Ihres Amiga, genügt für die Verbindung sogar ein einfaches Kabel.

*Terminal* verfügt über insgesamt drei Menüs. Das erste dient dazu, das Programm zu bestimmten Aktionen zu veranlassen (dazu gleich noch mehr), die beiden anderen Menüs legen die Eigenschaften oder »Parameter« der Datenübertragung fest. Das zweite Menü Baudrate dient zur Bestimmung der Datenübertragungsrate. Sie haben hier die Wahl zwischen 300 und 1200. Welche Geschwindigkeit die jeweils richtige ist, können Sie der Betriebsanleitung Ihres Modems entnehmen. Es hängt unter Umständen aber auch von der Arbeitsgeschwindigkeit des anderen Computers ab, mit dem der Amiga verbunden ist.

Ähnliches gilt sinngemäß für die Punkte im dritten Menü *Parameter*. Jeder Befehl in diesem Menü besteht es drei Angaben, die mit Schrägstrichen voneinander getrennt sind. Es handelt sich dabei um die Anzahl der Datenbits, die Eigenschaften der Paritätsprüfung und die Anzahl der Stopbits. Wenn Sie sich schon einmal mit Datenfernübertragung beschäftigt haben, werden Ihnen diese Begriffe wahrscheinlich etwas sagen. Ihre Erklärung würde in diesem Buch etwas zu weit führen, da sie nur für eine Minderheit der Amiga-Besitzer von Interesse ist. (Die meisten öffentlichen Datenbanken erreichen Sie problemlos mit der ersten Einstellung »8/N/1«.)

Das erste Menü *Optionen* dient der Ablaufsteuerung des Kommunikationsprozesses. Der wichtigste Befehl in diesem Menü ist *Online*. Er baut die Verbindung zu dem anderen Computer auf. Wenn Sie danach einen Text eingeben, wird dieser sofort an den anderen Computer gesendet. Im allgemeinen ist es üblich, daß dieser jeden Buchstaben sofort nach dem Empfang einmal zurückschickt, damit Sie auch eine Bestätigung haben, daß er korrekt am anderen Ende angekommen ist. Diese Rücksendung nennt man auch »Echo«. Manchmal erhält man ein solches Echo nicht. Dann muß das eigene Kommunikationsprogramm dafür sorgen, daß eingegebene Buchstaben zur Kontrolle am Bildschirm erscheinen. Sie können diese (»lokale«) Echofunktion mit dem Befehl *Echo* beziehungsweise *Kein Echo* im Menü *Optionen* ein- und ausschalten.

Die ersten beiden Befehle im Menü *Optionen* dienen zur Übertragung kompletter Dateien zwischen zwei Computern. Wenn Sie den Befehl Datei empfangen eingeben, werden Sie nach einem Dateinamen gefragt. Nachdem Sie diesen eingegeben und die [Return]-Taste gedrückt haben, werden alle vom Programm empfangenen Daten in dieser Datei gespeichert. In ähnlicher Form können Sie mit dem Befehl *Datei senden* eine komplette Datei zu dem anderen Computer übertragen. Sie werden dazu wieder nach einem Dateinamen gefragt. Sobald Sie diesen eingegeben und die [Return]-Taste gedrückt haben, wird die komplette Datei abgesendet.

Beachten Sie bitte, daß diese beiden Befehle zum Senden und Empfangen nur für Dateien geeignet sind, die ausschließlich lesbaren Text enthalten (sogenannte Text- oder ASCII-Dateien). Andere Dateien, wie zum Beispiel Programme und Bilder werden bei der Übertragung höchstwahrscheinlich leicht beschädigt und sind danach unbrauchbar. Auch in Textdateien können bei der Übertragung Fehler auftreten. Das ist aber nicht so tragisch, da dabei höchstens einige falsche Buchstaben auftauchen können, die man beim Durchlesen des Textes rasch findet. Bereits ein einziger kleiner Fehler kann hingegen ein Programm völlig unbrauchbar machen!

Die restlichen Befehle im Menü *Optionen* haben lediglich informativen Wert. Auf den Befehl *Autoren-Info* hin stellt sich der stolze Programmierer von *Terminal* vor und der Befehl *Kopier-Info* teilt Ihnen mit, daß Sie das Programm bedenkenlos an andere Benutzer weiterreichen dürfen.

*Terminal* ist wirklich ein sehr simples Telekommunikationsprogramm. Wenn Sie vorhaben, häufiger mit öffentlichen Datenbanken zu arbeiten oder auch einmal Bilder und Programme zu anderen Computern zu senden oder von diesen zu empfangen, werden Ihnen die Grenzen von Terminal bald sehr störend auffallen. Es gibt jedoch eine ganze Reihe kommerzieller Telekommunikationsprogramme für den Amiga, die über den Funktionsumfang von *Terminal* weit hinausgehen und ein wesentlich komfortableres Arbeiten erlauben.

# 6 Einführung in CLI und Shell

Bis jetzt haben Sie in diesem Buch den Amiga sozusagen von seiner »Schokoladenseite« kennengelernt. Ein relativ einfach zu bedienendes (der Computerfachmann sagt auch »benutzerfreundliches«) Programm namens »Workbench« hat Ihnen den Umgang mit dem neuen Computer so einfach wie nur möglich gemacht. Den Inhalt der Computerspeicher, vor allem der Disketten und Festplatten, wird von der Workbench zum Beispiel nicht in langweiligen Listen sondern in grafischer Form verdeutlicht. Alle manipulierbaren Objekte werden in Form kleiner Piktogramme dargestellt und alle wichtigen Operationen lassen sich durch die Auswahl eines Befehls aus einem Menü oder mit ein, zwei Mausklicks erledigen. Diese freundliche, grafische Darstellung der komplexen Vorgänge unterscheidet den Amiga von vielen anderen Computern, bei denen man die Befehle in Form schwer zu behaltender Kommandos über die Tastatur eingibt und die Ergebnisse dann oft als lange Textlisten zu sehen bekommt.

Durch Blättern im Handbuch und den Kontakt mit anderen Amiga-Besitzern erfährt jeder Amiga-Besitzer aber über kurz oder lang auch von der anderen, »dunklen Seite« des Amiga: dem CLI oder (ab der Workbench-Version 1.3) der »Shell«. Mit der Shell kann man die selben Arbeiten erledigen wie mit der Workbench (und einige mehr) – aber auf völlig andere Art und Weise. Sie repräsentiert eine Art, mit dem Amiga umzugehen, die der »üblichen« Bedienungsweise eines Computers, wie man es vielleicht von Großrechnern und IBM-kompatiblen PCs her kennt, recht nahekommt.

Dieses und die folgenden beiden Kapitel werden sich ausführlich mit dem CLI und der Shell des Amiga befassen. Die Informationen zu diesem Thema werden dabei wie folgt aufgeteilt: in diesem Kapitel erhalten Sie zunächst eine Einführung in den Umgang mit der Shell und in die grundlegenden Konzepte und Begriffe, die Sie kennen müssen, um sicher und effizient damit arbeiten zu können. Das nächste Kapitel enthält eine vollständige Beschreibung der Shell-Befehle. Diese Beschreibungen sind mehr zum Nachschlagen als zum systematischen Lesen gedacht. Kapitel Nummer 8 schließlich ist für fortgeschrittene Shell-Anwender gedacht und enthält weitergehende Informationen zu besonders wichtigen Themen, die allerdings einige Vorkenntnisse und etwas Erfahrung beim Umgang mit der Shell voraussetzen.

# 6.1 CLI, Shell und Amiga-DOS

Die Abkürzung »CLI« steht für »Command Line Interpreter« oder »Command Line Interface«. Das könnte man in etwa mit »Kommandozeilen-Ausführer« übersetzen. Das heißt, daß man dem CLI über sogenannte »Kommandozeilen« Befehle erteilt, die es dann ausführt. Die Kommunikation mit dem CLI spielt sich dabei allein über die Tastatur und über ein Bildschirmfenster ab, das nur Text aber keine Grafiken oder Piktogramme enthält. Befehle werden auf der Tastatur eingegeben und die Ergebnisse erscheinen in Form kurzer Meldungen oder langer Listen im Fenster. Die Grafikmöglichkeiten des Amiga liegen völlig brach und die Maus können Sie links (oder rechts) liegen lassen!

»Wozu das alles? Wozu die grafischen Fähigkeiten des Amiga ungenutzt lassen?«, werden Sie jetzt vielleicht fragen. Aber auch diese zweite Art, den Amiga zu bedienen, hat ihre Berechtigung, wie Sie spätestens in den nächsten Folgen dieser Serie noch sehen werden. Viele Aufgaben sind auf der Workbench nur umständlich oder überhaupt nicht zu erledigen, was Ihnen bis jetzt – solange Sie keine anderen Möglichkeiten kennen – wahrscheinlich noch nicht aufgefallen ist. Beim täglichen Gebrauch des Amiga und bei speziellen Anwendungen, wie zum Beispiel der Programmierung, werden diese Einschränkungen aber bald sehr lästig. Deshalb gibt es das CLI. Es mag schwieriger zu erlernen sein als die Workbench, ist aber sehr leistungsfähig. Zudem benötigt man für viele Aufgaben, vor denen auch Sie im Laufe Ihrer »Amiga-Karriere« sicherlich irgendwann einmal stehen werden, einfach ein paar Grundkenntnisse des CLI beziehungsweise der Shell. Die Installation einer Festplatte ist zum Beispiel ohne solche Kenntnisse meist nicht möglich.

## 6.1.1 Was ist Amiga-DOS?

CLI und Shell werden oft mit dem »Amiga-DOS« verwechselt. Amiga-DOS steht für Amiga-Disk-Operating-System und ist ein wichtiger Teil der sogenannten »Systemsoftware« des Amiga, die zum Teil in Form von ROM-Chips fest im Amiga eingebaut ist, zum Teil aber auch auf der Workbench-Diskette gespeichert ist. Der Amiga-DOS genannte Teil dieser Systemsoftware kümmert sich zum Beispiel um die Steuerung und Verwaltung der Diskettenlaufwerke und anderer Geräte, startet Programme und macht es anderen Programmen (und dem Computer-Benutzer) leichter, mit diesen Geräten zu arbeiten.

Das CLI mit Amiga-DOS zu verwechseln, ist aber andererseits auch eine läßliche Sünde, da das CLI einen direkten Zugriff auf nahezu alle Funktionen ermöglicht, die Amiga-DOS anbietet. Mit Amiga-DOS selbst brauchen Sie sich nur dann beschäftigen, wenn Sie ein Programmierer sind oder werden wollen. Programmierer können – auf dem Umweg über eine geeignete Programmiersprache – direkt mit Amiga-DOS »reden«. Alle anderen Anwender müssen das CLI oder die Shell dafür benutzen.

## 6.1.2 Welche Aufgaben haben CLI und Shell?

Das CLI und die Shell sind zwei Programme (fast) wie jedes andere. Computer-Wisssenschaftler würden es eine »Shell« (zu deutsch Schale) nennen. Startet man eines von beiden legt sich dieses Programm wie eine Schale um Amiga-DOS und andere Teile der Systemsoftware des Amiga. (Daher hat die Shell, die neue Version des CLI, die mit der Workbench 1.3 ausgeliefert wird, auch Ihren Namen.)

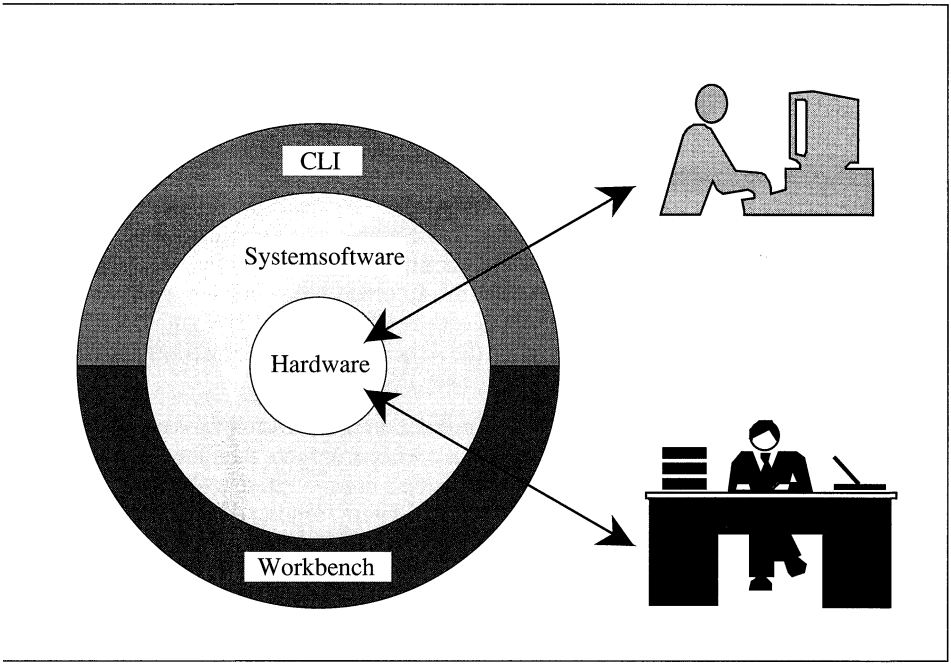



***Bild 6.1:*** *Zwei verschiedene Shells, die den »Kern« des Computers umgeben*

Das CLI beziehungsweise die Shell empfängt dann Ihre Befehle, die Sie in einer menschenverständlichen und leicht erlernbaren Sprache eintippen, und reicht sie in einer Computersprache an Amiga-DOS weiter. Genauso werden die Ergebnisse beziehungsweise Antworten auf die Befehle, die vom Amiga-DOS zurückkommen, zunächst vom CLI oder der Shell empfangen und in eine Sprache übersetzt, die der menschliche Anwender verstehen kann. Die Workbench verhält sich in dieser Hinsicht übrigens ganz ähnlich und erfüllt auch ähnliche Aufgaben. Nur die für die Kommunikation mit dem Anwender von der Workbench, vom CLI und der Shell benutzten Sprachen sind verschieden. Bei der Workbench ist es eine bildliche Sprache, beim CLI und der Shell eine etwas schwierigere, rein verbale Sprache. Aber auch sie ist erlernbar – und zwar leichter als die meisten menschlichen Sprachen.

CLI und Shell sind mehr für den fortgeschrittenen Benutzer gedacht, der schon eine Weile mit dem Amiga gearbeitet hat. Wer gerade erst seinen Amiga gekauft hat und die verschiedenen Möglichkeiten der Workbench schon verwirrend genug findet, der sollte das Studium dieses Buchteils vielleicht auf einen späteren Zeitpunkt verschieben.

### 6.1.3 Unterschiede zwischen CLI und Shell

Für viele Amiga-Anwender ist es nicht immer leicht, zwischen dem CLI und der Shell zu unterscheiden. Deshalb möchte ich schnell noch ein paar Wörter zu den Unterschieden und Gemeinsamkeiten dieser beiden Programme verlieren, bevor die Details dieser Bedienungsweise näher beschrieben werden.

Wichtig ist zunächst einmal folgendes: CLI und Shell können im Prinzip dieselben Aufgaben erfüllen und gehen dabei auch in gleicher Weise vor. Die meisten der Befehle, die Sie der Shell erteilen können, versteht auch das CLI und alle CLI-Befehle können Sie auch an die Shell richten. Wenn Sie also Kenntnisse eines der beiden Programme besitzen, können Sie diese für den Umgang mit dem anderen Programm gebrauchen!

Historisch gesehen ist das CLI das ältere Programm, das schon zusammen mit den allerersten Exemplaren des Amiga 1000 (der heute nicht mehr gebaut wird) ausgeliefert wurde. Die Shell ist aus dem CLI durch eine Reihe von Verbesserungen hervorgegangen. Hauptunterschied zwischen Shell und CLI ist der größere »Komfort« bei der Arbeit mit der Shell. Er macht den Umgang mit der Shell sowohl etwas einfacher wie auch effizienter. Anders formuliert: Sie kommen mit der Shell oft schneller an Ihr Ziel als mit dem CLI. Aus diesem Grund wird in diesem Buch lediglich die Arbeit mit der Shell beschrieben. Hinweise auf das CLI haben nur den Charakter historischer Funktionen!

Damit Sie aber mit der Shell arbeiten können, muß die Workbench-Diskette, mit der Sie Ihren Amiga starten, die Versionsnummer 1.3 oder höher tragen. Auf älteren Workbench-Disketten ist nur das CLI zu finden. Wenn Sie bisher mit einer älteren Workbench-Version gearbeitet haben, sollten Sie sich spätestens jetzt bei Ihrem Händler die neuere Workbench-Diskette besorgen. Gerade für die Arbeit mit der Shell lohnt sich der Aufstieg!

## 6.2 Wie Sie die Shell starten

Kommen wir nach dieser Vorrede nun zur praktischen Arbeit mit der Shell. In diesem Kapitel werden Sie Ihre ersten zaghaften Schritte mit dieser anfänglich vielleicht etwas unfreundlich erscheinenden Art, Ihren Amiga zu bedienen, machen. Aber keine Angst: Die Arbeiten, die Sie bisher auf der Workbench erledigt haben, sind mit der Shell auch nicht soviel schwieriger. Wenn Sie bereits andere Computer kennen – speziell UNIX- und MS-DOS-Systeme – werden Ihnen viele Aspekte beim Umgang mit Shell zudem auch bekannt vorkommen. Sie werden dann vielleicht versucht sein, dieses Kapitel sehr flüchtig oder gar nicht zu lesen. Tun Sie das bitte nicht. Sie werden in den folgenden Abschnitten nämlich einige wichtige Begriffe und Grundlagen der Arbeit mit der Shell kennenlernen, die nicht, oder nicht unbedingt in dieser Form auf anderen Computern gültig sind.

### 6.2.1 Vorarbeiten für die Arbeit mit der Shell

Bevor Sie mit den ersten Versuchen beginnen, sollten Sie zunächst eine Kopie der originalen Workbench-Diskette anfertigen. Wie Sie eine solche Kopie anfertigen (auf der Workbench) wird im zweiten Kapitel dieses Buches ausführlich beschrieben. Wenn die Kopie fertig ist beschriften Sie sie sinnigerweise mit dem Namen »Shell-Diskette«. Geben Sie ihr auch auf der Workbench mit dem *Rename*-Befehl den Namen »Shell-Diskette«). Diese Diskette werden Sie für den Rest dieses Buchteils für alle Experimente verwenden. Die erste Kopie der Workbench-Diskette legen Sie bitte so lange erst einmal beiseite.

Sind Sie mit dem Kopieren fertig, so nehmen Sie bitte beide Disketten aus den Laufwerken und starten den Amiga neu (mit der Tastenkombination [Ctrl]+[*A*]+[*A*]). Schieben Sie dann die neu kopierte Diskette in das interne Laufwerk des Amiga 500. Für den Rest dieses un

den größten Teil der folgenden Kapitel nehme ich an – sofern nicht ausdrücklich etwas anderes erwähnt wird – daß Sie nur mit dieser Diskette arbeiten.

Öffnen Sie nach dem Neustart das Disketten-Fenster Ihrer neuen Shell-Diskette und dann die Schublade. Löschen Sie aus dieser Schublade die Piktogramme *Say*, *ClockPtr*, *Calculator*, *Cmd*, *GraphicDump* und *PrintFiles*. (Das geht am einfachsten, wenn Sie diese Piktogramme nacheinander mit einem Shift-Klick markieren und dann den Befehl *Discard* aufrufen.) Indem Sie diese kleinen Hilfsprogramme, die Sie wahrscheinlich nur selten benötigen, löschen, schaffen Sie etwas Platz auf der Diskette, den Sie für einige der folgenden Übungen benötigen.

### 6.2.2 Erste Schritte in der Shell

Das Piktogramm der Shell finden Sie im Diskettenfenster der Workbench-Diskette. Sollte es sich bei Ihnen dort nicht befinden, so liegt es wahrscheinlich in der Schublade *System*. Legen Sie in diesem Fall das Piktogramm aus der Schublade in das Diskettenfenster, damit Sie es nach dem Start immer sofort sehen, ohne erst eine Schublade öffnen zu müssen.

Starten Sie jetzt die Shell, indem Sie einen Doppelklick auf das entsprechende Piktogramm machen. Daraufhin erscheint ein neues Fenster auf dem Bildschirm. Oben links in diesem Fenster erscheint der Text

```
1.SYS:>
```

Dies ist das sogenannte »Prompt« der Shell, das immer dann erscheint, wenn die Shell bereit ist, einen neuen Befehl von Ihnen entgegenzunehmen und auszuführen.

Spätestens jetzt wird es Zeit für Sie, die Tastatur Ihres Amiga einmal näher zu betrachten. Denn innerhalb des Shell-Fensters zählt nur die Tastatur, die Maus hat keine Funktion mehr. Sie können mit ihr zwar noch wie gewohnt die Position und Größe des Fensters ändern, ansonsten befinden Sie sich nun aber in einer Umgebung, in der die Maus nicht zählt. Benutzer fast aller anderen Computer-Modelle müssen sich allerdings fast immer so ähnlich mit ihren Computern herumschlagen, deswegen sollten Sie sich nicht entmutigen lassen.

Tippen Sie nun hinter dem Prompt Ihren ersten Befehl an die Shell ein:

```
1.SYS:>DIR
```

Während Sie auf der Tastatur die Buchstaben »D«, »I« und »R« tippen, erscheinen diese nacheinander auf dem Bildschirm und die Schreibmarke wandert weiter nach rechts. (Erscheint das »D« nicht sofort im Fenster, nachdem Sie es gedrückt haben, klicken Sie bitte kurz mit der Maus ins Innere des Fensters und versuchen es noch einmal.) Das ganze ist dem Eintippen eines neuen Namens für ein Piktogramm in der Workbench sehr ähnlich, wie Sie es nach dem Befehl *Rename* tun können.

Tippen Sie nun [Return]. Jeder Befehl an die Shell muß mit [Return] abgeschlossen werden! Sie schließen damit die Befehlseingabe ab. Die Shell liest den von Ihnen eingegebenen Befehl und versucht ihn auszuführen. In diesem Fall war es der Befehl DIR, der eine Liste von Datei-und Schubladen-Namen der Diskette im internen Laufwerk ausgeben sollte. Haben Sie den Befehl richtig getippt, so »rollt« diese Liste nun vor Ihnen im Fenster vorbei – manchmal ein, manchmal zwei Namen in einer Zeile. Währenddessen arbeitet das

Diskettenlaufwerk und die grüne Lampe flackert, da die Shell die Namen der Dateien von der Diskette lesen muß. Ist die Liste am unteren Rand des Fensters angekommen, bewegt sich der Fensterinhalt um eine Zeile nach oben (die obere Zeile verschwindet dabei). Die nächsten Namen tauchen dann in der untersten, jetzt leeren, Zeile auf. Diesen Vorgang nennt man »Rollen« oder auch »Scrollen« des Fensters. Sie haben ihn ja schon bei den Workbench-Fenstern kennengelernt. Anders als bei diesen Fenstern ist alles, was über den oberen Fensterrand hinaus verschwunden ist, im Shell-Fenster aber unwiederbringlich verloren. Sie können nicht zurückrollen, wie in der Workbench.

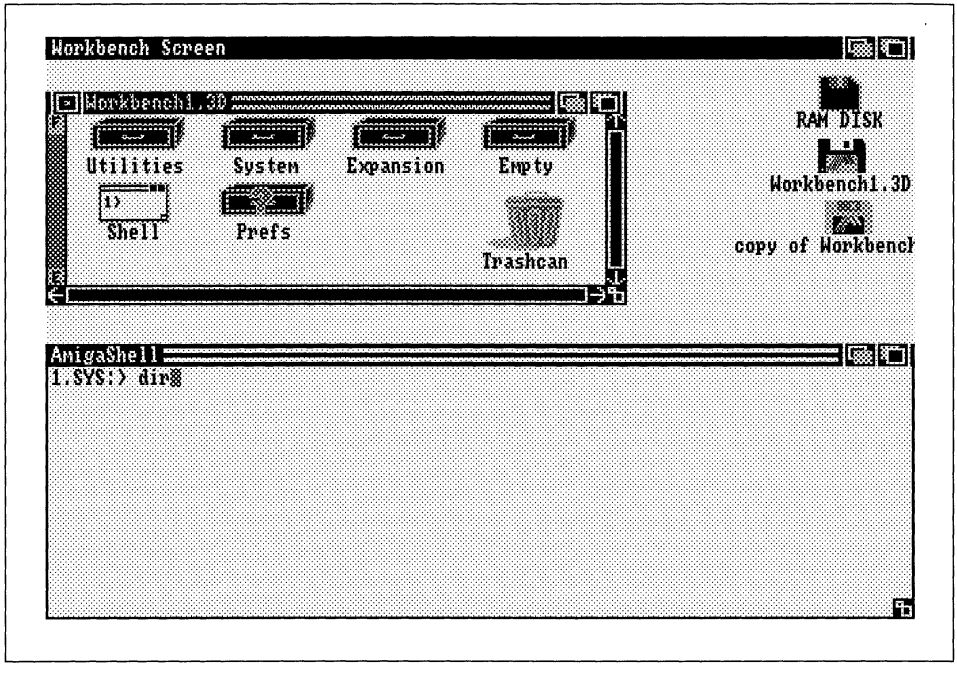



***Bild 6.2:*** *Eingabe eines Befehls im Shell-Fenster*

Ist die Liste komplett, so erscheint wieder das Prompt und Sie können den nächsten Befehl eintippen:

```
1.SYS:>
```

Nun erst ein paar Wörter zur in diesem Buch verwendeten Schreibweise, bevor Sie mit der Arbeit fortfahren: Alles, was im Shell-Fenster erscheint – Ihre Eingaben und die Ausgaben der Shell – wird in diesem Buch mit Schreibmaschinenschrift gedruckt. Um zu unterscheiden, was Sie selbst eingeben müssen, werden in diesem Kapitel die von Ihnen einzugebenden Texte fett hervorgehoben. Wenn Sie also 1.SYS:>**DIR** lesen, so bedeutet das, daß im Shell-Fenster die Buchstaben »1.SYS:>« erscheinen und Sie dahinter »DIR« eintippen müssen. Und nicht vergessen: Wenn nicht ausdrücklich anders erwähnt, müssen Sie alle Eingaben im Shell-Fenster mit [Return] abschließen. Alle Shell-Befehle werden der Auffälligkeit wegen in Großbuchstaben gedruckt. Das heißt aber nicht, daß Sie selbs

Großbuchstaben eingeben müssen. Sie können statt »DIR« auch »Dir« oder »dir« tippen. Der Shell ist die Schreibweise völlig egal.

Die fette Schreibweise der von Ihnen einzugebenden Texte wird übrigens nur in diesem Kapitel verwendet, um Ihnen den Einstieg in die Shell leichter zu machen. Ich gehe davon aus, daß es Ihnen nach diesem Einstieg keine Probleme mehr machen wird, Ihre Eingaben von den Ausgaben der Shell-Befehle zu unterscheiden.

Nun aber zurück zu dem ersten Befehl, den Sie gerade der Shell erteilt haben. Manche der Namen, die in der Liste erscheinen, werden Ihnen bekannt vorkommen. Es sind dieselben, die auch einige Piktogramme im Diskettenfenster auf der Workbench tragen. Viele Namen aber werden neu sein, und das demonstriert schon einen wesentlichen Unterschied zwischen der Workbench und der Shell. Im Shell können Sie alle Dateien und Schubladen (die Shell nennt Schubladen Directories; mehr dazu aber in einem späteren Abschnitt) sehen, die auf der Diskette liegen. In der Workbench sehen Sie hingegen nur die Dateien und Schubladen, von denen die Entwickler des Amiga meinen, daß Sie sie sehen sollten.

## 1.2.3 Wie Sie die Shell wieder verlassen

Wenn Sie zwischendurch auch einmal die Workbench benutzen oder gar die Shell endgültig verlassen wollen, bevor Sie den Rest des Kapitels durcharbeiten, haben Sie mehrere Möglichkeiten: Wenn Sie nur vorübergehend etwas in der Workbench machen, und danach wieder zurück in die Shell wollen, ist es nicht nötig, die Shell zu beenden. Der Amiga besitzt ja ein Multitasking-Betriebssystem, was in diesem Fall bedeutet, daß beim Start der Shell die Workbench nicht beendet wurde. Die Workbench ist nach wie vor – neben oder hinter der Shell – aktiv. Klicken Sie einfach mit dem linken Mausknopf in eines der Fenster oder auf eines der Piktogramme der Workbench. Sind die Fenster der Workbench nicht zu sehen, weil die Shell-Fenster sie alle verdeckt, so können Sie das Shell-Fenster natürlich auch wie jedes andere Fenster mit dem entsprechenden Gadget nach hinten legen.

Sobald Sie »zurück in der Workbench« sind, bemerken Sie, daß sich der Inhalt der Titelzeile des Bildschirms ändert. Wenn Sie den rechten Mausknopf niederdrücken, müssen nun auch wieder die Menüs der Workbench in der Menüzeile erscheinen. Sie können jetzt beliebige Aktionen in der Workbench durchführen und durch einen einfachen Mausklick auf die Shell-Fenster später dann wieder in die Shell zurückkehren.

Eine andere Möglichkeit, die Shell zu verlassen, ist etwas permanenter. Sie beendet das Programm Shell wirklich und schließt auch das entsprechende Fenster. Falls das Shell-Fenster im Moment nicht aktiv ist, machen Sie es aktiv, indem Sie mit der Maus kurz in das Fensterinnere klicken. Tippen Sie dann hinter das Prompt:

```
1.SYS:>ENDSHELL
```

Daraufhin erscheint eine kurze Meldung im Fenster, die aber im allgemeinen zu schnell verschwindet, als daß man sie lesen könnte, und das Shell-Fenster wird geschlossen. Die einzige Möglichkeit, jetzt wieder in die Shell zu gelangen, ist, erneut das Piktogramm *Shell* zu öffnen. Der Befehl ENDSHELL ist also recht endgültig und sollte nur angewendet werden, wenn man sicher ist, die Shell in nächster Zeit nicht mehr zu benötigen. Er kann aber auch dann sehr sinnvoll sein, wenn der vom Shell benötigte Speicherplatz dringend für ein anderes Programm gebraucht wird.

## 6.3 Befehle eingeben und korrigieren

In diesem Abschnitt werden Sie etwas ausführlicher erfahren, was Sie bei der Eingabe eines Befehls an die Shell beachten müssen und welche Möglichkeiten Ihnen dabei zur Verfügung stehen.

### 6.3.1 Eingeben eines Befehls

Wie oben schon am Beispiel DIR zu sehen war, können Shell-Befehle aus einem (meist kurzen) Wort bestehen, das Sie eingeben müssen, damit die Shell eine bestimmte Operation ausführt. Welche Befehle Ihnen normalerweise zur Verfügung stehen, erfahren Sie noch in aller Ausführlichkeit im nächsten Kapitel. Eine Warnung aber vorweg: Es sind deutlich mehr als auf der Workbench – Sie müssen sie aber nicht alle kennen, um mit der Shell arbeiten zu können!

Shell-Befehle bestehen aber nicht immer nur aus dem Befehlsnamen. Manchmal folgen auch ein oder mehrere Wörter, die sogenannten »Parameter«. Als erstes – noch harmloses – Beispiel für einen Befehl, der aus mehreren Wörtern besteht, soll uns eine etwas andere Form des DIR-Befehls dienen. Geben Sie dazu den folgenden Befehl hinter dem Shell-Prompt ein:

```
1.SYS:>DIR System
```

Auch diesmal erscheint wieder eine Liste von Namen am Bildschirm. Wenn Sie sich noch an die Liste erinnern, die erschien, als Sie die ersten Versuche mit dem DIR-Befehl machten, werden Sie aber feststellen, daß es sich um eine andere Liste handelt. Diesmal zeigt Ihnen der DIR-Befehl nämlich nicht den Inhalt der Diskette sondern den Inhalt der Schublade *System* auf der Diskette.

Betrachten wir gleich noch einen anderen Befehl, der aus mehreren Wörtern besteht. Geben Sie dazu den folgenden Befehl ein und drücken Sie die [Return]-Taste:

```
1.SYS:>ECHO "Hallo Du da!"
```

Daraufhin müßte der Text »Hallo Du da!« ein zweites Mal (unter der Befehlszeile) am Bildschirm erscheinen. Der ECHO-Befehl gibt nämlich den hinter dem Wort ECHO folgenden Text (der in Anführungsstriche gesetzt werden muß) in einer neuen Zeile aus. Sie werden später noch erfahren, welchen Sinn ein solcher Befehl haben kann.

Und damit haben Sie auch schon das grundsätzliche Prinzip des Aufbaus eines Shell-Befehls kennengelernt. Jeder Shell-Befehl beginnt immer mit einem »Befehlsnamen« oder »Kommandonamen«. Zwei dieser Kommandonamen haben Sie schon kennengelernt: DIR und ECHO. Der Kommandoname legt fest, **was** ein Shell-Befehl tut.

Hinter dem Kommandonamen können noch ein oder auch mehrere Wörter folgen. Das sind die sogenannten »Parameter« oder »Argumente«. Die Argumente sagen einem Shell-Befehl, **womit** er etwas machen soll. Das Argument des DIR-Befehls legt zum Beispiel fest, der Inhalt welcher Schublade aufgelistet werden soll. (Genauer gesagt, der Inhalt welches »Directorys«; siehe unten). Das Argument des ECHO-Befehls besagt, welcher Text ausgegeben werden soll.

### 6.3.2 Editieren einer Befehlszeile

Das größte Problem der Shell ist – zumindest für »mausverwöhnte« Anfänger – sicherlich, daß man manchmal verhältnismäßig lange Befehle fehlerfrei über die Tastatur eingeben muß. Was ist aber, wenn Sie einen Fehler machen? Die Shell bietet glücklicherweise einige Möglichkeiten zur nachträglichen Korrektur von Fehlern.

Die einfachste und am häufigsten benötigte Korrekturmöglichkeit verbirgt sich hinter der [Backspace]-Taste. Mit ihr können Sie – wie auch auf der Workbench – jeweils den zuletzt eingetippten Buchstaben links von der Schreibmarke wieder löschen. Haben Sie den falschen Buchstaben schon eher – nicht erst als letzten – eingetippt, müssen Sie halt mehrere Buchstaben löschen, bis Sie beim Fehler sind.

Falls Sie den Fehler erst bemerken, wenn Sie schon die [Return]-Taste gedrückt haben, ist es zur Korrektur natürlich zu spät. Die Shell kann den fehlerhaften Befehl dann im allgemeinen nicht ausführen und wird eine Fehlermeldung ausgeben. Probieren Sie es aus. Geben Sie hinter dem Prompt den – nicht erlaubten – Befehl FIER ein.

```
1.SYS:>FIER
```

Darauf antwortet die Shell, nachdem Sie die [Return]-Taste gedrückt haben, mit der Meldung :

```
Unknown command FIER
1.SYS:>
```

Das heißt nichts anderes, als daß der Shell dieser Befehl (engl. *command*) unbekannt (engl. *unknown*) ist. Sie brauchen diesen Befehl aber nicht komplett neu eingeben, wenn Sie eigentlich den DIR-Befehl verwenden wollten. Statt dessen können Sie den alten Befehl »zurückholen« und den oder die fehlerhaften Buchstaben beseitigen. Probieren Sie es aus:

Drücken Sie jetzt einmal auf die nach oben zeigende Cursortaste [↑]. Danach sollte an Ihrem Bildschirm wieder der FIER-Befehl erscheinen:

```
1.SYS:>FIER_
```

Die Schreibmarke (im Text durch Unterstreichung angedeutet) steht hinter dem letzten Buchstaben des Befehls. Drücken Sie jetzt einmal auf die Taste [←]. Die Schreibmarke sollte dann auf dem Buchstaben R stehen. Drücken Sie nun auf die [Backspace]-Taste und der Buchstabe E verschwindet.

```
1.SYS:>FIR
```

Der Befehl heißt jetzt FIR – und das sieht doch schon ganz ähnlich wie DIR aus. Drücken Sie deshalb jetzt noch einmal auf die Taste [←] (die Schreibmarke steht dann auf dem Buchstaben I) und dann auf die [Backspace]-Taste. Jetzt steht nur noch IR in der Befehlszeile und die Schreibmarke steht auf dem Buchstaben I.

```
1.SYS:>IR
```

Drücken Sie jetzt auf die Taste [D]. Der Buchabe D wird dann vor der Schreibmarke in die Befehlszeile eingefügt. Der Befehl DIR steht wieder in voller Pracht und Schönheit vor Ihnen:

```
1.SYS:>DIR
```

Wenn Sie jetzt auf die [Return]-Taste drücken, erscheint wieder die Liste der Dateien und Dateiverzeichnisse auf der Diskette im internen Laufwerk. (Sie brauchen vor der Betätigung der [Return]-Taste die Schreibmarke **nicht** an das Ende der Zeile zu bewegen!)

Ich gebe zu, daß es sich bei diesem kurzen Befehl nicht recht lohnt, ihn zu korrigieren. Es wäre sicherlich schneller gegangen, DIR einfach neu einzutippen. Stellen Sie sich aber vor, es wäre ein Befehl gewesen, der mit seinen Parametern die Zeile fast gefüllt hätte (solche Befehle gibt es), dann werden Sie mir sicherlich zustimmen, daß die eben beschriebene Methode des »Zurückholens und Korrigierens« wesentlich praktischer ist.

### 6.3.3 Anhalten und Fortsetzen der Ausgabe

Wenden wir uns jetzt einem anderen Problem der Shell zu, das Ihnen bestimmt auch schon aufgefallen ist: den langen Listen, die viel zu schnell über den Bildschirm huschen. Wenn Sie zum Beispiel den DIR-Befehl eingeben und die [Return]-Taste drücken, rauscht der größte Teil der Liste schneller vorbei, als man die Namen lesen könnte (schneller zumindest als ich diese Namen lesen kann). Shell-Befehle haben nämlich die unangenehme Eigenschaft, auch lange Texte in einem endlosen Strom auf den Bildschirm zu schreiben. Wird dabei das untere Ende des Shell-Fensters erreicht, rollt der alte Inhalt nach oben, um unten eine neue Zeile freizumachen. Das ist auch ganz sinnvoll. Unangenehm an diesem Rollvorgang ist jedoch, daß Texte, die einmal am oberen Fensterand verschwunden sind, auch verschwunden bleiben und nicht wieder zurückgeholt werden können. Aber auch für dieses Problem gibt es eine Lösung.

Es gibt nämlich eine Möglichkeit, die Ausgabe dieser Liste kurz anzuhalten, sich die Namen in Ruhe anzuschauen, und dann das nächste »Häppchen« auf dem Bildschirm erscheinen zu lassen. Probieren Sie es aus: Geben Sie den DIR-Befehl ein, betätigen Sie die [Return]-Taste und drücken Sie, sobald die ersten paar Zeilen erschienen sind, einmal auf die Leertaste.

```
1.SYS:>DIR
```

Die Listenausgabe hält dann an und auch die Diskettenlampe erlischt. Sie können sich die Liste jetzt in Ruhe betrachten. Drücken Sie dann auf die Taste [Backspace]. Sie löschen damit praktisch das Leerzeichen, das die Listenausgabe aufgehalten hat und prompt »rollen« die restlichen Namen der Liste über den Bildschirm. Auch jetzt können Sie natürlich jederzeit durch Drücken der Leertaste oder einer anderen (beliebigen!) Taste die Ausgabe stoppen und dann mit der Taste [Backspace] wieder fortsetzen.

### 6.3.4 Abbrechen eines Befehls

Was aber ist, wenn Sie einen Befehl korrekt eingegeben haben und dann erst merken, daß Sie das gar nicht wollten oder geistesabwesend den falschen Befehl eingetippt haben? Meist ist das nicht so tragisch, denn selbst der DIR-Befehl, der ja eine recht lange Liste auf dem Bildschirm ausgibt, dauert nur einige Sekunden. Es gibt aber auch Shell-Befehle, die Minuten dauern können!

Deshalb bietet die Shell auch eine Möglichkeit, einen laufenden Befehl »mittendrin« abzubrechen. Probieren Sie es aus. Geben Sie noch einmal den Befehl DIR ein und drücken Sie die [Return]-Taste.

```
1.SYS:>DIR
```

Halten Sie jetzt mit einem Finger der linken Hand die [Ctrl]-Taste (am linken Rand der Tastatur) fest und warten Sie wieder, bis die ersten paar Zeilen der Liste am Bildschirm erschienen sind. Drücken Sie dann mit einem Finger der rechten Hand – während Sie die [Ctrl]-Taste weiter festhalten – die Taste [C]. Die Shell bricht den DIR-Befehl dann mitten in der Ausgabe ab und meldet:

```
*** BREAK
1.SYS:>
```

Das heißt auf gut deutsch nichts anderes, als daß der Befehl unterbrochen wurde. Auf diese Art und Weise können Sie fast jeden Befehl abbrechen, bevor er seine Arbeit vollständig erledigt hat.

## 6.4 Dateien und Directories

Mit dem Grundwissen über die Shell, das Sie in den ersten Abschnitten dieses Kapitels erworben haben, sind Sie schon recht gut gerüstet für die Arbeit mit der Shell. Einige wichtige Konzepte von Amiga-DOS (das der Shell ja zugrunde liegt) müssen aber noch erwähnt werden, bevor Sie mit der Arbeit beginnen können. Die wichtigsten dieser Konzepte sind Dateien, Dateiverzeichnisse und Geräte. Wer schon andere Computer kennt, dem werden diese Begriffe vielleicht schon vertraut sein. Wenn Sie bisher aber nur die Workbench des Amiga kennen, soll Ihnen der folgende Abschnitt einen ungefähren Eindruck von diesen Begriffen aus der Sprache »Computerchinesisch« geben.

### 6.4.1 Was eine Datei ist

In diesem Kapitel (und auch schon in früheren Kapiteln) ist häufig das Wort »Datei« gefallen. Was ist denn nun eine solche Datei? Dateien haben Sie bereits im ersten Teil dieses Buches kennengelernt, der sich mit der Workbench beschäftigte. Programme (Werkzeuge) und Dokumente (Projekte) sind zum Beispiel Dateien. Während eine Bezeichnung wie »Programm« aber schon etwas über die Aufgabe des entsprechenden Objekts sagt, ist »Datei« etwas abstrakter. Eine Datei ist einfach eine Ansammlung von Daten, die zu einem zusammenhängenden Objekt, eben einer Datei, zusammengefaßt wurden. Ein solches Objekt kann mit einem Namen bezeichnet werden, dem »Dateinamen«.

Die Disketten oder Festplatten des Amiga enthalten, wenn man es genau nimmt, keine Dateien, sondern auch eine Folge von Daten (Bytes), die aus technischen Erwägungen zu größeren Blöcken zusammengefaßt werden. Diese Blöcke haben fortlaufende Nummern. Gäbe es keine Dateien, so müßte man jedesmal, wenn man Daten von einer auf eine andere Diskette speichern will, angeben, in welchem Block die Daten liegen und wo sie abgelegt werden sollen. Wenn die Daten länger sind als ein Block, so müssen sie sich auf einen zweiten, dritten, vierten Block und so weiter erstrecken. In welchen Blöcken diese logisch zusammengehörigen Daten liegen, muß man sich natürlich auch merken. Ein Kopierbefehl könnte dann lauten: »Nimm die 2 Byte von dieser Stelle der Diskette, die 374 Byte von dort und die 1235 Byte von hier und lege Sie da und dort auf die andere Diskette.« Das klingt nicht nur sehr umständlich, sondern ist es auch. Deshalb gibt es heute keine Computer mehr, die

Daten auf diese Art und Weise verwalten. Statt dessen übernimmt ein Teil der Systemsoftware – beim Amiga ist es das Amiga-DOS – die Verwaltung der Daten und ihre logische Gruppierung zu Dateien. Wie eine Datei auf die Blöcke einer Diskette verteilt ist, braucht den Anwender und auch den Programmierer deshalb nicht mehr zu interessieren. Er braucht nur Dateinamen zu kennen. Amiga-DOS übernimmt den Verwaltungskram – genauso wie wir auch von einer guten Sekretärin oder einem Sekretär erwarten, daß sie alle einzelnen Blätter, Formulare und so weiter aus den verschiedenen Schränken eines Büros hervorsuchen, wenn wir die »Akte Meier« anfordern.

Welche Art von Daten eine Datei enthält und wie diese Daten strukturiert sind, kümmert das Betriebssystem Amiga-DOS allerdings nicht. Egal, ob es sich bei einer Datei um ein Programm oder um einen Brief an einen Geschäftsfreund handelt, Amiga-DOS sieht nur eine Folge von Bytes, die irgendwo auf der Diskette oder einer Festplatte abgespeichert sind und einen eigenen Namen hat.

### 6.4.2 Was ein Directory ist

Wenn Dateien einzelnen Akten entsprechen, so sind Directories (Dateiverzeichnisse) die Schubladen und Aktenordner unseres elektronischen Büros. Sie haben diese Aktenordner bereits kennengelernt: Was die Workbench als Schubladen zeigt, sind nahezu die gleichen Objekte, die Amiga-DOS und die Shell als »Directory« bezeichnet.

Ein Directory enthält eine Liste von Dateinamen und Informationen, wo diese Dateien auf der Diskette zu finden sind. Weiterhin sind darin auch zusätzliche Informationen über die einzelnen Dateien, wie Größe, Datum der letzten Änderung und zusätzliche Kommentare, verzeichnet. Diese (auf der Diskette abgespeicherte) Liste entspricht ziemlich genau der Liste, die am Bildschirm auftaucht, wenn Sie im Shell den Befehl DIR(ectory) eingeben.

### 6.4.3 Der Dateienbaum

Werden unsere Aktenmengen noch größer, können die Aktenberge auch trotz Schubladen beziehungsweise Directories unübersichtlich werden. Wieso sollte es dann nicht auch möglich sein, die Schubladen wieder zu gliedern? Genau das ist in der Shell genauso wie auf der Workbench möglich. Directories können in anderen Directories liegen und selbst natürlich auch wieder Dateiverzeichnisse enthalten.

Die Strukturierung einer Diskette oder Festplatte mit Hilfe von Directories könnte man mit der Gliederung eines Buchs vergleichen: Wie ein gutes Fachbuch eine vielstufige Gliederung aus Buchteilen, Kapiteln, Abschnitten, Unterabschnitten und so weiter hat, kann auch eine gut aufgeräumte Diskette, und vor allem eine ja noch viel größere Festplatte eine vielstufige Hierarchie von Dateien und Dateiverzeichnissen haben, die es erleichtert, unter den vielen Dateien eine bestimmte zu finden.

Abbildung 6.3 zeigt einen solchen Dateienbaum, wie er auf einer Diskette existieren könnte. Dateiverzeichnisse sind in diesem Baum durch Schubladen, Dateien durch Notepad-ähnliche Piktogramme gekennzeichnet. (Einige Zweige des Baums sind aus Platzgründen nur durch Pünktchen angedeutet.) Es entspricht einer allgemein gültigen Konvention, solche Dateienbäume »auf dem Kopf«, also mit der Wurzel obenliegend darzustellen. Lassen Sie sich dadurch nicht verwirren – es ist einfach so üblich.

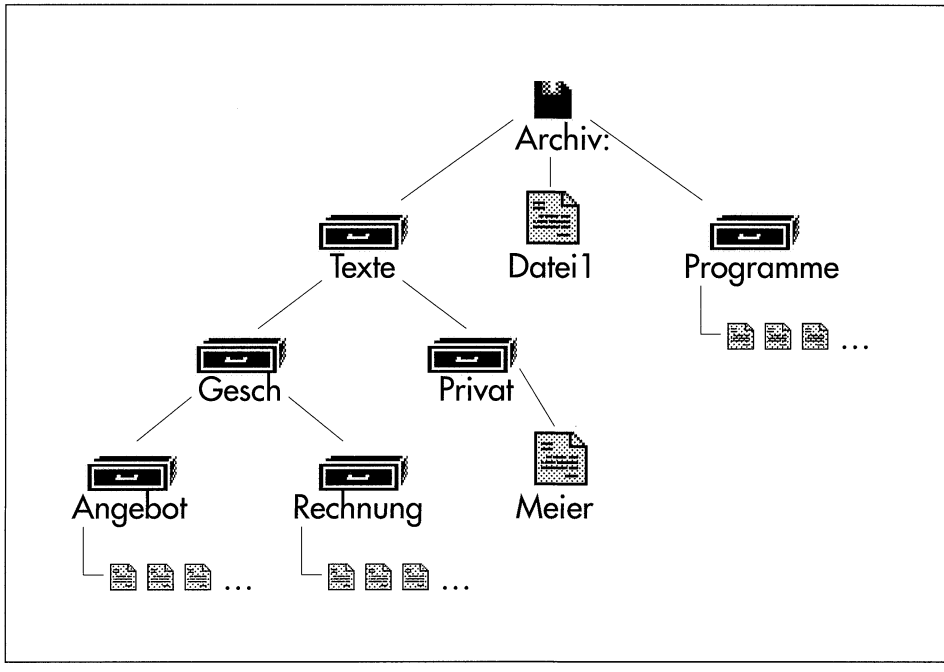



***Bild 6.3:*** *Ein Dateienbaum auf einer Diskette*

### 6.4.4 Pfadnamen – Wege zu Dateien

Die Directory-Hierarchie auf einer Diskette oder Festplatte darf aber auch nicht zu groß und unübersichtlich werden. Es kann sonst extrem schwierig werden, eine Datei genau zu bestimmen. Denn anders als auf der Workbench ist es im Shell-Fenster nicht möglich, auf eine Datei (ein Piktogramm) zu zeigen. Man muß statt dessen, wenn man eine Datei manipulieren (zum Beispiel löschen) will, genau die Folge der Schubladen (Directories) angeben, die man nacheinander öffnen muß, um zu der gewünschten Datei zu gelangen. Diese Aneinanderreihung von Directory- und Dateinamen ist der sogenannte Pfadname einer Datei, weil er beschreibt, auf welchem Pfad man zu der Datei gelangt.

Jede Datei und jedes Directory auf einer Diskette hat einen ganz bestimmten »vollständigen Pfadnamen«. Dieser Pfadname enthält unter anderem den Namen der »Wurzel« des Dateienbaums. Diese Wurzel trägt den Namen der Diskette, wie er auf der Workbench unter dem Disketten-Piktogramm auftaucht, und wird mit einem Doppelpunkt beendet. Heißt eine Diskette zum Beispiel *Archiv*, wäre der Name der Wurzel auf dieser Diskette *Archiv:*. Es spielt dabei keine Rolle, in welchem Laufwerk sich die Diskette befindet; es zählt nur der Diskettenname!

Der vollständige Pfadname einer Datei beginnt immer mit dem Namen der Wurzel (um Dateien auf verschiedenen Disketten auseinanderhalten zu können) gefolgt von einem Doppelpunkt. Dann kommen die Schubladen, die man nacheinander öffnen muß, um zu

dieser Datei zu gelangen, getrennt von Schrägstrichen. Erst als letzter Bestandteil des Pfadnamens kommt der Name, der auf der Workbench unter dem Piktogramm der Datei auftauchen würde.

Die Datei *Meier* in der Schublade *Privat*, die in der Schublade *Texte* auf der Diskette *Archiv* liegt, hätte demnach also den vollständigen Pfadnamen *Archiv:Texte/Privat/Meier* und die Datei *Müller* in derselben Schublade den Pfadnamen *Archiv:Texte/Privat/Müller* (der erste Teil des Pfadnamens bleibt gleich).

### 6.4.5 Das aktuelle Directory

In dem obigen Beispiel mit den nur zweifach verschachtelten Directories *Privat* und Texte *wurde* der Pfadname schon recht lang. Wenn Sie auf Ihrer Diskette die Directories (Schubladen) noch tiefer ineinander geschachtelt oder lange Namen verwendet haben, kann der vollständige Pfadname rasch 30, 40 oder noch mehr Buchstaben lang werden. Das ist nicht nur lästig, sondern erhöht auch die Gefahr von Tippfehlern enorm.

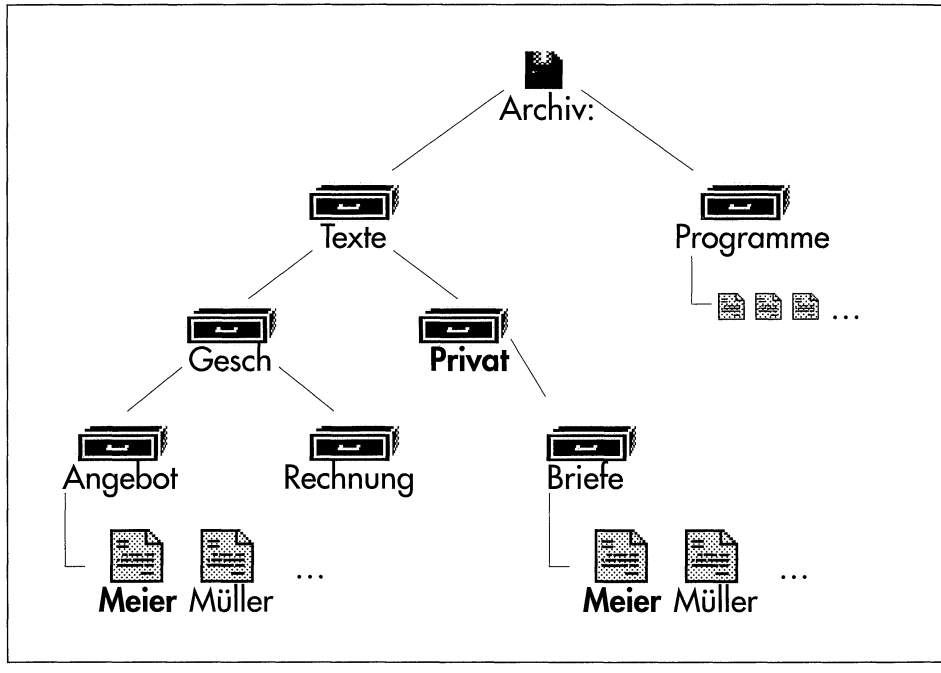



***Bild 6.4:*** *Pfadnamen im Dateienbaum*

Es gibt deshalb in der Shell das Konzept des »aktuellen Directory«. Ein Directory alle Directories auf allen Disketten und Festplatte, die im Moment angeschlossen sind, ist da aktuelle Directory. Um eine Datei im aktuellen Directory zu benennen, braucht man nich den vollständigen Pfadnamen zu verwenden, sondern es reicht aus, den letzten Teil de

Namens, den eigentlichen »Dateinamen« anzugeben. Abbildung 6.4 demonstriert diese praktische Einrichtung. Sie zeigt wieder einen Dateienbaum. In diesem sind das aktuelle Dateiverzeichnis und zwei Dateien etwas hervorgehoben (durch Fettdruck ihrer Namen).

Die Namen, die man für diese beiden Dateien im Shell-Fenster eintippen müßte, würden *Archiv:texte/gesch/angebot/Meier* und *briefe/Meier* lauten (da das aktuelle Dateiverzeichnis *archiv:texte/privat* heißt). Wäre gar *archiv:texte/privat/briefe* das aktuelle Directory, würde es für die rechte der beiden Dateien genügen, einfach *Meier* zu schreiben.

Sie sehen schon an diesen kleinen Beispielen, wieviel Schreibarbeit Sie sich auf diese Weise ersparen können. In einem der folgenden Abschnitte werden Sie deshalb den Befehl CD (*change directory* = ändere Dateiverzeichnis) kennenlernen, mit dem Sie jederzeit ein bestimmtes Directory zum aktuellen Directory »ernennen« können.

### 6.4.6 Die aktuelle Diskette

Ähnlich, wie es immer ein aktuelles Directory gibt, existiert auch immer eine »aktuelle Diskette«. Dies ist die Diskette, auf der sich das aktuelle Directory befindet. Wenn Sie also das aktuelle Directory wechseln (Sie werden den entsprechenden Befehl gleich kennenlernen) und das neue aktuelle Directory sich auf einer anderen Diskette befindet, wird diese dadurch gleichzeitig zur aktuellen Diskette.

Man kann die aktuelle Diskette – ähnlich wie das aktuelle Directory – zur Arbeitserleichterung verwenden. Wenn sich eine Datei, deren Pfadnamen Sie eingeben wollen, nämlich auf der aktuellen Diskette befindet, können Sie den Diskettennamen im Pfadnamen fortfallen lassen (nicht jedoch den Doppelpunkt hinter dem Diskettennamen). Statt *archiv:texte/privat/Meier* könnten Sie deshalb auch *:texte/privat/Meier* schreiben, um die Datei *Meier* in der Schublade *privat* zu bezeichnen.

### 6.4.7 Laufwerksnamen

Wie Sie gesehen haben, enthält der vollständige Pfadname einer Datei immer auch den Namen der Diskette, auf der sich die Datei befindet. Dadurch wird praktisch die Wurzel des Dateienbaums festgelegt, in dem die Datei zu finden ist. Auf jeder Diskette befindet sich nämlich ein eigener Dateienbaum, der mit dem einer anderen Diskette keine Verbindung (auch keine gemeinsame Wurzel) hat.

Als Diskettenname wurde bislang derselbe Name verwendet, der auch auf der Workbench unter dem Piktogramm der Diskette steht. Die Verwendung dieses Namens kann aber manchmal unpraktisch sein (weil er zu lang ist) und manchmal schwierig sein (weil man ihn vergessen hat und ihn auch nicht sieht, wenn das Shell-Fenster das Disketten-Piktogramm verdeckt). In solchen Fällen ist es ganz praktisch, daß es noch eine zweite Möglichkeit gibt, eine Diskette zu bezeichnen. Und zwar ist dies der Name des Diskettenlaufwerks, in dem sich die Diskette befindet.

Ja, Sie haben richtig gelesen; auch Diskettenlaufwerke haben Namen – allerdings merkwürdige. Das interne Laufwerk trägt den Namen DF0:, das erste externe heißt DF1: und falls Sie weitere externe Laufwerke anschließen, haben diese die Namen DF2: und DF3: (mehr als vier Diskettenlaufwerke können Sie nicht verwenden.)

Sie können also die Datei *archiv:texte/privat/Meier* auch mit *DF0:Texte/Privat/Meier* ansprechen, wenn sich die Diskette *archiv* im internen Laufwerk befindet. Beachten Sie aber, daß diese Datei auf einmal *DF1:texte/privat/Meier* heißt, wenn die entsprechende Diskette im externen Laufwerk liegt. Das kann verwirrend sein und ist der Nachteil der Laufwerksnamen. Trotzdem sind Laufwerksnamen ganz praktisch, da man ja meistens weiß, in welchem Laufwerk sich eine Diskette gerade befindet, auch wenn man sich an ihren Namen gerade nicht erinnern kann.

## 6.5 Arbeiten mit der Shell

Nachdem Sie nun – zumindest ungefähr – wissen, was es mit Dateien und Directories auf sich hat, können Sie in diesem Abschnitt Ihre neugewonnenen Kenntnisse ausprobieren. Sie werden dabei eine ganze Reihe der wichtigsten Shell-Befehle kennenlernen, die Sie bei der täglichen Arbeit mit der Shell immer wieder benötigen werden.

### 6.5.1 Directories betrachten

Den Befehl, mit dem Sie ein Directory betrachten können, haben Sie ja bereits kennengelernt: DIR. Wenn Sie im Shell-Fenster DIR eintippen und dann die [Return]-Taste drücken, bekommen Sie den Inhalt des aktuellen Directories angezeigt. Nach dem Aufruf der Shell ist das Wurzelverzeichnis auf der Workbench-Diskette das aktuelle Directory. Sie bekommen deshalb eine Auflistung zu sehen, die in etwa dem Inhalt des Diskettenfensters auf der Workbench entspricht. Wie bereits erwähnt wurde, sind im Shell-Fenster aber mehr Dateien und Directories zu sehen als auf der Workbench. Der DIR-Befehl bringt deshalb im allgemeinen immer auch ein paar neue Namen zum Vorschein.

Probieren Sie den DIR-Befehl am besten gleich noch einmal einmal aus. Machen Sie dazu aber die Shell-Fenster so groß wie möglich.

In der Liste, die der DIR-Befehl ausgibt, kommen zwei verschiedene Arten von Namen vor. Die eine Sorte Namen wird einspaltig aufgelistet und jedem Namen folgt in Klammern das Wort *DIR*. Dies sind andere Directories, die im Wurzelverzeichnis liegen. Einen Teil davon können Sie auch als Schubladen im Diskettenfenster der Workbench-Diskette sehen. Die andere Sorte Namen wird zweispaltig aufgelistet. Dies sind die Namen der Dateien, die sich im Wurzelverzeichnis der Diskette befinden. Auch einige dieser Namen tauchen als Piktogramme auf, andere nicht. Wenn Sie die Namensliste einmal näher betrachten, wird Ihnen auffallen, daß viele Namen in ähnlicher Form zweimal vorkommen. So gibt es zum Beispiel *Preferences* und *Preferences.info*. Betrachten Sie dann die Liste genauer, wird Ihnen auffallen, daß genau die Namen noch einmal mit einem *.info* dahinter auftauchen, die auch als Piktogramme auf der Workbench sichtbar sind. Das kommt nicht von ungefähr, wie Sie gegen Ende dieses Kapitels noch erfahren werden.

Sie können mit dem DIR-Befehl aber nicht nur das aktuelle, sondern auch andere Directories betrachten. Dazu müssen Sie hinter dem Wort DIR nur eine oder mehrere Leerstellen eingeben und dann den Pfadnamen des Directory eintippen, dessen Inhalt Sie sehen wollen, bevor Sie auf [Return] drücken. Geben Sie zum Beispiel einmal die folgende Zeile ein:

```
1.SYS:>DIR c
```

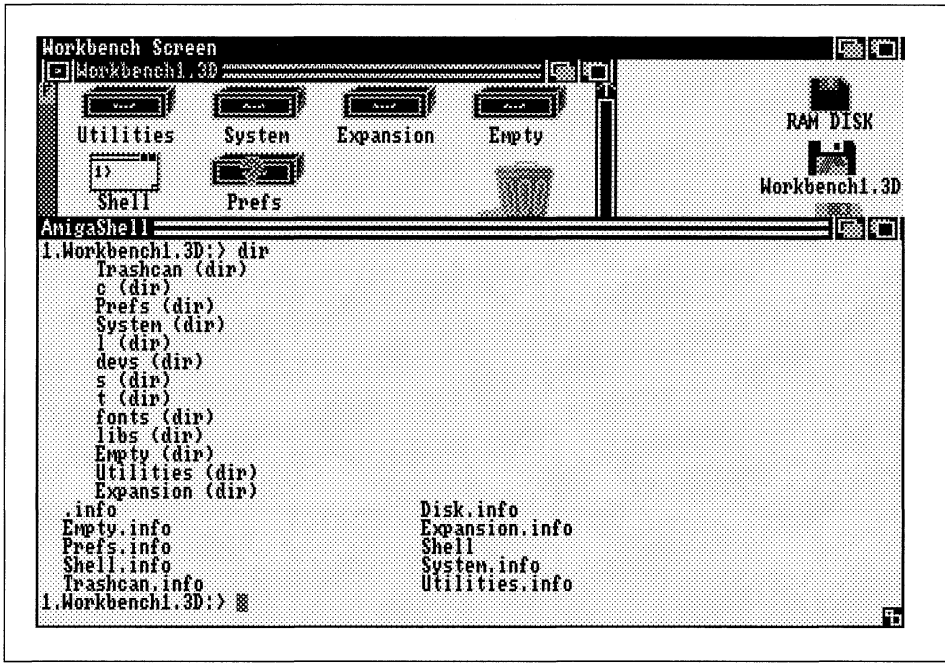



***Bild 6.5:*** *Auflistung eines Directory*

Sie bekommen daraufhin den Inhalt des Directory *c* auf der Workbench gezeigt. Stoppen Sie die Auflistung zwischendurch ruhig einmal mit einem Druck auf die Leertaste (Fortsetzen mit [Backspace]), um sich die Liste in Ruhe anzuschauen. Diese Liste enthält fast alle Befehle (engl. *commands*), die die Shell »versteht«. Im Directory *c* befinden sich nämlich die Programme, die aufgerufen werden, wenn Sie einen Befehl im Shell-Fenster eingeben. Jedesmal, wenn Sie zum Beispiel DIR eintippen, wird daraufhin das Programm DIR im Dateiverzeichnis *c* ausgeführt.

Probieren Sie ruhig ein wenig die Möglichkeiten des DIR-Befehls aus. Schauen Sie auch die Inhalte anderer Dateiverzeichnisse an. Deren Namen sehen Sie ja, wenn Sie den Befehl DIR ganz ohne Parameter eingeben. Es sind die Namen, hinter denen in der Liste »(DIR)« steht.

Der Parameter von DIR muß übrigens unbedingt ein Directory sein. Wenn Sie DIR den Namen einer Datei (zum Beispiel eines Programmes) als Parameter übergeben, erhalten Sie die folgende Fehlermeldung (die soviel heißt wie »Dateiname ist kein Dateiverzeichnis«):

```
1.SYS:>DIR shell
filename is not a directory
```

Der Befehl DIR zeigt aber nur die Namen der Dateien und Directories. Es gibt noch einen anderen Befehl, der Ihnen mehr Informationen über die einzelnen Objekte in einem Directory mitteilt. Dieser Befehl heißt LIST. Geben Sie die folgende Zeile im Shell-Fenster ein, um LIST auszuprobieren.

```
1.SYS:>LIST devs/printers
```

Wenn Sie LIST ohne einen Parameter eingegeben hätten, wäre wie beim DIR-Befehl der Inhalt des aktuellen Directory aufgelistet worden. Mit *LIST devs/printers* erhalten Sie eine Auflistung des Inhalts des Directory *printers* im Directory *devs* auf der Workbench-Diskette (beziehungsweise Shell-Diskette). In jeder Zeile erhalten Sie Informationen über eine Datei oder ein Directory. Nebeneinander werden der Dateiname, die Dateigröße (in Bytes), die »Schutz-Flags« und Uhrzeit und Datum beim Erzeugen dieser Datei angezeigt. Unter Umständen wird in der Zeile unter dem Namen auch noch ein Kommentar angezeigt (die meisten Dateien haben aber keinen solchen Kommentar).

Der Befehl LIST liefert also viel mehr Informationen über jede Datei als DIR. Diese Informationen brauchen aber dafür auch länger, um über den Bildschirm zu laufen, und die Dateinamen sind unsortiert. Es ist deshalb meist einfacher und geht schneller, DIR zu verwenden, wenn man nur »schnell mal eben« eine bestimmte Datei sucht.

### 6.5.2 Programme starten

Genausoleicht wie das Erteilen eines Befehls (zum Beispiel DIR) ist es, ein Programm (Werkzeug) von der Shell aus zu starten. Genaugenommen, haben Sie das ja auch schon getan, denn auch DIR und LIST, die beiden Befehle, die Sie schon kennengelernt haben, sind ja Programme. Um ein Programm zu starten, brauchen Sie nur seinen Namen beziehungsweise Pfadnamen einzugeben und die [Return]-Taste zu drücken. Probieren Sie es aus: Geben Sie die folgende Zeile ein und drücken Sie [Return].

```
1.SYS:>preferences
```

Nach wenigen Augenblicken müßte daraufhin das Hauptfenster von *Preferences* erscheinen. Sie haben auf diese Weise *Preferences* ohne Berührung der Maus vom Shell-Fenster aus gestartet. (Falls Sie statt dessen eine Fehlermeldung erhalten, haben Sie sich vielleicht vertippt. Versuchen Sie es dann noch einmal.) Jetzt brauchen Sie allerdings die Maus, um das Programm wieder verlassen zu können. Klicken Sie auf den Knopf *Cancel* unten rechts im Fenster.

Probieren Sie den Programmstart gleich noch einmal aus. Geben Sie dazu den folgenden Befehl ein (nachdem Sie *Preferences* verlassen haben):

```
1.SYS:>clock
```

Auf diesen Befehl hin erscheint das Fenster mit der Amiga-Uhr. Sobald Sie das Fenster schließen, kehren Sie zur Shell zurück.

Sie können auf diese Weise jedes beliebige Programm starten, wenn Sie seinen Pfadnamen kennen. Zu diesem Thema allerdings noch eine kleine Warnung: Es gibt einige Programme aus der Frühzeit des Amiga 1000, die vom Shell-Fenster aus nicht gestartet werden können. Ein Versuch, sie vom Shell-Fenster aus zu starten, ergibt eine Fehlermeldung oder einen »Absturz« des Amiga (bei dem ein Alert erscheint).

### 6.5.3 Disketten kopieren

Einer der ersten »Handgriffe« die Sie auf der Workbench gelernt haben, war das Kopieren einer Diskette. Dies ist eine sehr wichtige Arbeit, die Sie regelmäßig mit allen wichtigen

Disketten machen sollten. Selbstverständlich kennt auch die Shell einen Befehl zu diesem Zweck, der sinnigerweise DISKCOPY heißt. Er kann sowohl mit einem als auch mit zwei Laufwerken dazu verwendet werden, nahezu identische Kopien kompletter Disketten anzulegen. Legen Sie sich dazu jetzt eine leere, neue Diskette oder eine bereits gebrauchte Diskette, deren Inhalt Sie nicht mehr benötigen, bereit und aktivieren Sie sicherheitshalber den Schreibschutz der Shell-Diskette, mit der Sie gerade arbeiten. Geben Sie dann die folgende Zeile ein:

```
1.SYS:>DISKCOPY FROM df0: TO df0: NAME "Shell 2"
```

Sie werden dann aufgefordert:

```
Place SOURCE disk (FROM disk) in drive DF0:
```

(Legen Sie die Ursprungsdiskette in Laufwerk DF0:) Drücken Sie daraufhin die [Return]-Taste. Der DISKCOPY-Befehl liest nun einen Teil des Inhalts der Shell-Diskette in den RAM-Speicher und meldet dabei:

```
Reading 9, 70 to go
```

Sobald der RAM-Speicher voll ist, werden Sie aufgefordert:

```
Place DESTINATION disk (TO disk) in drive DF0:
```

(Legen Sie die Zieldiskette in Laufwerk DF0:) Nehmen Sie daraufhin die Shell-Diskette aus dem Laufwerk und legen Sie die zweite Diskette ein. Drücken Sie dann wieder auf [Return]. Der DISKCOPY-Befehl schreibt dann die zuvor gespeicherten Daten auf die zweite Diskette und meldet dabei:

```
WRITING 12, 67 to go
```

Nach einiger Zeit erscheint dann wieder die Aufforderung:

```
Place SOURCE disk (FROM disk) in drive DF0:
```

Sie müssen jetzt noch einige Male die Disketten wechseln, während jedesmal ein Häppchen der Daten kopiert wird, dann enthält die zweite Diskette ein exaktes Ebenbild der ersten – allerdings unter dem Namen »Shell 2«. Sobald der Kopiervorgang beendet ist, erscheint die Meldung:

```
Diskcopy Finished
```

Das Diskettenwechseln beim Kopieren ganzer Disketten können Sie sich natürlich sparen, wenn Sie ein zweites Diskettenlaufwerk besitzen. Legen Sie dann einfach die zu kopierende Diskette in das erste Laufwerk, eine leere Diskette in das zweite und geben Sie den folgenden Befehl ein:

```
1>DISKCOPY FROM DF0: TO DF1:
```

Das bedeutet, daß Sie vom internen zum externen Laufwerk DF1: kopieren wollen. Daraufhin erscheint die folgende Meldung im Shell-Fenster:

```
Place SOURCE disk ( FROM disk ) in drive DF0:
Place DESTINATIPON disk ( TO disk ) in drive DF1:
Press RETURN to continue
```

Das heißt soviel wie »Legen Sie die Originaldiskette in das interne Laufwerk, die Diskette, die die Kopie aufnehmen soll, in das externe Laufwerk und drücken Sie die [Return]-Taste um fortzufahren«. Wenn Sie dann noch einmal auf die [Return]-Taste drücken, fertigt der DISKCOPY-Befehl die Kopie in einem Rutsch an. Beachten Sie bei dieser Form des DISKCOPY-Befehls aber bitte, daß kein Name für die Kopie angegeben wurde. In diesem Fall erhält die Kopie denselben Namen wie das Original.

Falls während des Kopiervorgangs ein Fehler gemeldet wird (zum Beispiel *Disk Error* oder eine ähnliche Meldung), ist eine der beiden Disketten defekt. Falls es die Zieldiskette (*Target Disk*) ist, sollten Sie diese in den Müll werfen – oder besser zum Händler zurückbringen. Ist es die Originaldiskette, sollten Sie möglichst schnell alle wichtigen Dateien von dieser Diskette auf eine andere Diskette kopieren. Gehen Sie dazu am besten in die Workbench und legen Sie ein Piktogramm nach dem anderen auf die andere Diskette. Wenn dabei erneut ein Fehler auftaucht, müssen Sie auf das Piktogramm verzichten, bei dem dieser Fehler aufgetaucht ist. Sie können danach die defekte Diskette mit *Initialize* aus dem *Disk*-Menü löschen und initialisieren. Wenn dabei wieder ein Fehler auftaucht, werfen Sie am besten diese Diskette in den Müll oder bringen Sie dem Händler zurück, der sie Ihnen verkauft hat.

### 6.5.4 Eine Datei kopieren

Der DISKCOPY-Befehl dürfte Ihnen schon mal einen ersten Eindruck davon vermittelt haben, wie ein Befehl mit mehreren Parametern aussieht. Die im folgenden noch beschriebenen Befehle sind meist einfacher aufgebaut. Wenn Sie zum Beispiel eine Datei kopieren wollen, so benötigen Sie dazu den – sehr leicht zu bedienenden – COPY-Befehl (engl. to COPY = kopieren).

Das folgende Beispiel geht davon aus, daß Sie mit einer Kopie der Original-Workbench-Diskette arbeiten und auf dieser zumindest noch ein wenig Platz ist. Geben Sie folgendes ein (und achten Sie dabei auf eventuelle Tippfehler):

```
1.SYS:>COPY FROM disk.info TO test1
```

Das Diskettenlaufwerk beginnt nun einen Moment lang zu arbeiten. Der COPY-Befehl legt eine Kopie der Datei *disk.info* in der Datei *test1* an. (Überzeugen Sie sich mit Hilfe des DIR-Befehls davon, daß nach dem Aufruf von COPY eine Datei namen *test1* existiert.)

COPY benötigt zwei Parameter, den Namen der Datei, die kopiert werden soll (hinter dem Schlüsselwort FROM) und den Namen, den die Kopie tragen soll (hinter dem Schlüsselwort TO). Die Schlüsselwörter, die vor den Parametern jeweils angegeben werden, können ihnen helfen, verständlichere, lesbare Befehle zu schreiben. Mit diesen zusätzlichen Wörtern lesen sich manche Shell-Befehle wie englische Sätze. Die Schlüsselwörter bedeuten aber auch erhöhte Tipparbeit. Man kann Sie deshalb auch wegfallen lassen. Probieren Sie einmal den folgenden Befehl aus:

```
1.SYS:>COPY test1 test2
```

Danach gibt es eine neue Datei namens test2 (wovon Sie sich mit Hilfe des DIR-Befehls überzeugen können). Wie Sie sehen, ist das Kopieren einer Datei sehr einfach. Man braucht sich eigentlich nur den Namen des Befehls COPY dafür zu merken!

Bei den beiden Schlüsselwörtern von COPY, FROM und TO, handelt es sich um sogenannte »optionale Schlüsselwörter«. Man kann Sie mit angeben oder auch fortfallen lassen. Das ist aber nicht bei allen Shell-Befehlen und Ihren Schlüsselwörtern so! Gewöhnen Sie sich deshalb bitte nicht an, Schlüsselwörter immer zu ignorieren. Welche Schlüsselwörter es für die einzelnen Shell-Befehle gibt, welche Bedeutung sie haben und welche man wegfallen lassen kann (weil Sie optional sind), erfahren Sie in der Befehlsübersicht im nächsten Kapitel.

### 5.5.5 Eine Datei umbenennen

Genauso einfach wie das Kopieren einer Datei ist es, eine Datei umzubenennen. Der entsprechende Befehl heißt, genau wie auf der Workbench, RENAME. Probieren Sie ihn am besten gleich einmal aus und geben Sie die folgende Zeile im Shell-Fenster ein (Sie müssen dazu aber die Beispiele im vorigen Abschnitt ausprobiert und dabei die drei Dateien *test1*, *test2* und *test3* erzeugt haben):

```
1.SYS:>RENAME test3 TO testNEU
```

Wenn Sie danach den DIR-Befehl aufrufen, werden Sie sehen, daß es danach die Datei *test3* nicht mehr gibt, dafür aber die Datei *testNEU*. Das ist natürlich *test3*; nur unter einem anderen Namen. Statt des Schlüsselworts TO können Sie bei RENAME übrigens auch das Schlüsselwort AS (engl. für als) verwenden oder auch ganz ohne Schlüsselwort arbeiten wie beim COPY-Befehl.

Probieren Sie diese beiden Möglichkeiten am besten gleich aus. Geben Sie dazu die beiden folgenden Zeilen nacheinander ein. Danach sollten Sie wieder drei Dateien namens *test1*, *test2* und *test3* sehen, wenn Sie sich das aktuelle Directory mit dem DIR-Befehl anschauen:

```
1.SYS:>rename testNEU AS xxx
1.SYS:>rename xxx test3
```

Versuchen Sie nun auch einmal, eine Datei umzubenennen und ihr dabei denselben Namen zu geben, den eine schon existierende Datei besitzt:

```
1.SYS:>RENAME test3 TO test2
```

Sie erhalten daraufhin nur die folgende Fehlermeldung:

```
Can't rename test3 as test2
```

Sie besagt nichts anderes, als daß es unmöglich ist, einer Datei mit RENAME den Namen einer schon existierenden Datei zu geben. Die Shell weigert sich, diesen Befehl auszuführen, um Sie vor Flüchtigkeitsfehlern zu schützen. Wenn Sie wirklich einer Datei den Namen einer anderen Datei geben wollen, müssen Sie die zweite Datei zuerst löschen und dann den RENAME-Befehl verwenden.

### 5.5.6 Eine Datei löschen

Nachdem Sie nun gesehen haben, wie man neue Dateien erzeugt (durch Kopieren schon vorhandener Dateien) und dann wieder umbenennt, werden Sie jetzt erfahren, wie Sie diese Dateien auch wieder loswerden. Für solche Aufgaben ist der DELETE-Befehl zuständig

(engl. *to delete* = löschen). Geben Sie die folgende Zeile im Shell-Fenster ein und überzeugen Sie sich dann mit dem DIR-Befehl davon, daß die Datei *test3* nun fehlt:

```
1.SYS:>delete test3
```

Wie Sie sehen, brauchen Sie sich beim DELETE-Befehl nicht einmal ein Schlüsselwort zu merken. Sie können mit diesem Befehl auch mehr als nur eine Datei löschen. Hierzu brauchen Sie nur die Namen der Dateien, die Sie löschen wollen, alle hinter DELETE zu schreiben und [Return] zu drücken. Sie müssen dazu allerdings die Namen in dieser Liste jeweils mit einer Leerstelle voneinander trennen. Probieren Sie es aus: Erzeugen Sie (mit COPY) aus *test2* die beiden neuen Dateien *test3* und *test4* und löschen Sie diese dann mit den folgenden Befehlen

```
1.SYS:>COPY test2 test3
1.SYS:>COPY test3 test4
1.SYS:>DELETE test3 test4
```

Überzeugen Sie sich zwischendurch am besten mit Hilfe des DIR-Befehls davon, daß die beiden neuen Dateien *test3* und *test4* auch wirklich vorhanden sind beziehungsweise von DELETE gelöscht wurden.

### 6.5.7 Den Inhalt einer Datei auf dem Bildschirm betrachten

Wenn Sie Dateien nicht nur hin und her kopieren, umbenennen und wieder löschen wollen, sondern auch gerne wüßten, was diese Datei enthält, kann Ihnen manchmal ein anderer Shell-Befehl helfen: der Befehl TYPE. TYPE benötigt nur einen einzigen Parameter, einen Dateinamen, und gibt den Inhalt dieser Datei auf dem Bildschirm aus. Wichtig ist, daß es sich bei dieser Datei um eine reine »Textdatei« handelt, die nur Buchstaben enthält. Viele Programme erzeugen Dateien, die nicht nur Text, sondern auch Bilder und Formatierungsinformationen enthalten.

Auch Programme selbst sind Dateien, die viel mehr als nur reine Texte enthalten. Wenn Sie versuchen, mit TYPE ein Programm oder ein Bild, das Sie in einem Malprogramm gezeichnet haben, auszugeben, erscheint nur Chaos auf dem Bildschirm, dem Sie nichts entnehmen können. Auf der Workbench-Diskette befindet sich aber glücklicherweise schon eine Textdatei, anhand der Sie TYPE ausprobieren können. Geben Sie dazu den folgenden Befehl ein (und achten Sie sehr genau auf Tippfehler):

```
1.SYS:>TYPE s/startup-sequence
```

Daraufhin läuft ein langer Text über den Bildschirm. Es handelt sich dabei um eine recht komplizierte Folge von Shell-Befehlen, die bei jedem Neustart des Amiga ausgeführt wird und über die Sie im übernächsten Kapitel noch mehr erfahren werden.

Ein abschließender Tip zum TYPE-Befehl: Vielleicht passiert es Ihnen einmal aus Versehen, daß Sie eine Datei mit TYPE auf den Bildschirm ausgeben und dabei merkwürdige Zeichen erscheinen. Es kann dann sogar vorkommen, daß weitere Befehle, die Sie über die Tastatur eingeben, nicht mehr richtig am Bildschirm erscheinen! In diesem Fall hilft unter Umständen die Tastenkombination [Ctrl]+[O] ([Ctrl] festhalten und [O] tippen) oder [Esc] [C] (erst [Esc] drücken und loslassen, dann [C] drücken).

### 6.5.8 Den DIR-Befehl »interaktiv« verwenden

Viele der in den vorangegangenen Abschnitten vorgestellten Shell-Befehle können wesentlich mehr tun, als bisher beschrieben wurde. Der DIR-Befehl kann zum Beispiel nicht nur ein Directory auflisten, sondern auch Dateien löschen und sich am Bildschirm betrachten (sofern es Text-Dateien sind). Dazu dient die »interaktive« Option von DIR. Hierzu müssen Sie am Ende der Zeile mit dem DIR-Befehl (also direkt hinter DIR oder hinter dem Namen eines Directory) noch »INTER« eingeben, bevor Sie [Return] drücken. Probieren Sie es aus:

```
1.SYS:>DIR DF0: INTER
```

Sie bekommen nun Zeile für Zeile dieselbe Liste gezeigt, die sonst in einem Rutsch über den Bildschirm huschen würde. DIR gibt jeweils einen Datei- oder Directory-Namen aus und wartet dann auf Ihre Reaktion. Betätigen Sie daraufhin die [Return]-Taste, erscheint der nächste Name der Liste. Probieren Sie es aus: Drücken Sie zweimal auf die [Return]-Taste.

Wenn der Name eines Directory angezeigt wird, können Sie sich dessen Inhalt anschauen. Sie brauchen dazu nur die Taste [E] zu drücken, wenn dieser Name angezeigt wurde, um in dieses Directory einzusteigen. Bei jeder Betätigung der [Return]-Taste erscheint nun ein neuer Datei-oder Directory-Name aus dem untergeordneten Directory. Probieren Sie es aus: Drücken Sie so lange auf [Return], bis das Directory *fonts* angezeigt wird. Drücken Sie dann auf [E] und [Return]. Sie können nun mit [Return] dieses Directory durchblättern. Wenn Sie auf [B] und [Return] drücken (oder das Directory zu Ende ist), gelangen Sie wieder zum übergeordneten Directory zurück. In diesem Fall ist es das Wurzeldirectory der Diskette.

Wenn Sie das gefunden oder gesehen haben, was Sie sehen wollten oder gesucht haben, können Sie den interaktiven DIR-Befehl jederzeit abbrechen, indem Sie die Tastenkombination [Ctrl]-[C] und dann [Return] drücken. Machen Sie das jetzt und geben Sie dann noch einmal die folgende Zeile ein:

```
1.SYS:>DIR df0: INTER
```

Drücken Sie nun so lange auf die [Return]-Taste, bis das Directory *s* erscheint. Jetzt drücken Sie wieder die Tasten [E] und [Return]. DIR zeigt Ihnen daraufhin die Datei *startup-sequence*. Drücken Sie nun die Tasten [T] und [Return]. Daraufhin erscheint der Inhalt von *startup-sequence* auf dem Bildschirm, als hätten Sie den TYPE-Befehl verwendet. Immer wenn DIR Ihnen eine Textdatei zeigt, können Sie sich mit [T] ihren Inhalt anzeigen lassen. (Sie können einen langen Text dabei wie üblich durch Betätigung der Leertaste stoppen und mit [Backspace] wieder fortsetzen.)

Betätigen Sie nun so lange die [Return]-Taste bis die Datei *test2* erscheint, die Sie vorhin mit dem COPY-Befehl erzeugt haben. Geben Sie die drei Buchstaben »del« ein und drücken Sie auf [Return]. Sie haben dadurch mit drei Tastendrücken die Datei *test2* gelöscht, als hätten Sie den DELETE-Befehl aufgerufen.

Damit sind die wichtigsten Möglichkeiten des interaktiven DIR-Befehls beschrieben. Bevor wir mit dem nächsten Befehl fortfahren, muß aber unbedingt noch eine weitere Option des DIR-Befehls vorgestellt werden. Mit dieser können Sie sich den gesamten Inhalt einer Diskette oder Festplatte anschauen:

```
1.SYS:>DIR ALL
```

Der DIR-Befehl listet nun **alle** Dateien und Dateiverzeichnisse auf der Shell-Diskette in übersichtlicher Form am Bildschirm auf. Dabei sind die Namen um so weiter nach rechts eingerückt, je tiefer die entsprechenden Dateien und Dateiverzeichnisse im Dateienbaum verschachtelt sind.

Wenn Ihnen die Liste zu schnell vor den Augen vorbeiläuft, können Sie sie jederzeit anhalten, indem Sie auf eine beliebige Taste drücken. Zum Fortsetzen der Ausgabe dient dann die [Backspace]-Taste. Falls Sie nicht schnell genug waren, können Sie den DIR-Befehl ja sofort noch einmal eingeben.

Der DIR-Befehl mit dem Schlüsselwort ALL listet übrigens nicht immer den kompletten Inhalt einer Diskette oder Festplatte auf, wie Sie jetzt vielleicht annehmen werden. Statt dessen wird immer ein Dateiverzeichnis samt allen untergeordneten Verzeichnissen ausgegeben. Nur, weil im Moment das aktuelle Verzeichnis gleich der Wurzel des Dateienbaums auf der Shell-Diskette ist, wird deren gesamter Inhalt angezeigt. Wenn Sie einen anderen Teil des Dateienbaums sehen wollen, brauchen Sie lediglich den Namen eines Dateiverzeichnisses vor »ALL« zu setzen. Probieren Sie es aus. Der folgende Befehl zum Beispiel listet lediglich das Verzeichnis DEVS (samt Unterverzeichnissen) auf:

```
1.SYS:>DIR devs ALL
```

Wie Sie am Beispiel des DIR-Befehls gesehen haben, kann sich hinter einem Shell-Befehl beziehungsweise seinen Parametern mehr verbergen als zunächst »sichtbar« ist. Parameter können das Verhalten eines Befehls oft geradezu dramatisch ändern. Und bei den vielen verschiedenen Parametern, die die Shell-Befehle verstehen, ist es leicht, den Überblick zu verlieren.

### 6.5.9 Parameter zu Befehlen erfragen

Deshalb besitzen fast alle Shell-Befehle eine Art Auskunftsfunktion. Um diese zu verwenden, brauchen Sie lediglich hinter dem Befehlsnamen ein Fragezeichen – und sonst keine weiteren Parameter – einzutippen. Geben Sie zum Ausprobieren dieser Auskunftsfunktion zum Beispiel einmal den folgenden Befehl ein:

```
1.SYS:>DIR  ?
```

Dann gibt die Shell darunter die folgende Zeile aus:

```
DIR, OPT/K, ALL/S, DIRS/S, FILES/S, INTER/S:
```

und wartet auf Ihre Eingaben. Diese mehr oder weniger abgekürzten Wörter sind das »Befehlsformular« oder die »Befehlsschablone« (engl. *template*) von DIR und besagen, daß der DIR-Befehl maximal sechs Parameter versteht. Der erste ist ein Dateiverzeichnis (engl. DIRectory) der zweite Parameter beginnt mit OPT. Die Buchstaben »/K« hinter OPT besagen, daß es sich bei OPT um ein unbedingt benötigtes Schlüsselwort (engl. *key* = Schlüssel) handelt, daß vor dem danach noch folgenden Parameter stehen muß. Das Schlüsselwort OPT und der entsprechende Parameter wird in diesem Kapitel aber nicht beschrieben, wenn Sie mehr darüber erfahren wollen, können Sie die vollständige Beschreibung des DIR-Befehls im nächsten Kapitel nachlesen.

Die Schlüsselwörter, hinter denen »/S« steht, sind sogenannte »Schalter«. Diese Schalter stehen alleine in der Befehlszeile. Ihnen folgt kein Parameter. Ein Beispiel für einen solchen

Schalter ist das Schlüsselwort INTER, daß DIR dazu veranlaßt, die Auflistung des Directories interaktiv auszuführen. Mehr zu den unterschiedlichen Arten von Schlüsselwörtern und Parametern erfahren Sie am Anfang des folgenden Kapitels.

Geben Sie jetzt hinter dem Doppelpunkt DEVS ein und drücken Sie auf die [Return]-Taste. Der DIR-Befehl gibt dann den Inhalt des Verzeichnisses DEVS aus, als hätten Sie diesen Parameter sofort hinter DIR eingegeben. Sie können sich also die Befehlsschablone eines beliebigen Shell-Befehls anzeigen lassen, in Ruhe betrachten und dann die gewünschten Parameter eingeben.

### 6.5.10 Mehrere Dateien mit einem Befehl bearbeiten – Muster

Auf der Workbench haben Sie eine Methode kennengelernt, mehrere Piktogramme mit einem Befehl zu bearbeiten. Dazu brauchte man lediglich die [Shift]-Taste festzuhalten, während man nacheinander die gewünschten Piktogramme anklickt.

Einige der Shell-Kommandos (zum Beispiel COPY) bieten ähnliche Möglichkeiten, gleichzeitig mehrere Dateien zu bearbeiten. In Ermangelung der Maus werden für diesen Zweck sogenannte »Muster« für Dateinamen verwendet. Ein solches Muster wird dann statt eines Dateinamens als Parameter eines Shell-Befehls angegeben. Der Befehl geht dann alle Dateinamen auf der Diskette durch. Für jeden Dateinamen wird nun festgestellt, ob er in dieses Muster »paßt« oder nicht. Paßt er in das Muster, wird er in eine Liste aufgenommen und mit dem jeweiligen Befehl bearbeitet. Passen zum Beispiel vier Dateinamen in ein bestimmtes Muster und man verwendet dieses Muster als Parameter eines Shell-Befehls, so wirkt dieser eine Befehl mit Muster als würde man viermal hintereinander denselben Befehl verwenden und dabei das Muster gegen jeweils einen der vier Dateinamen austauschen.

Bei manchen Befehlen ist es möglich, an jeder Stelle ein Muster zu verwenden, an der normalerweise ein Dateiname stehen muß. Manchmal muß das Muster aber auch durch ein spezielles Schlüsselwort eingeleitet werden (zum Beispiel beim Befehl LIST). Der Informatiker nennt solche und ähnliche Muster »reguläre Ausdrücke«. Diese Muster sind für die verschiedensten Zwecke in der Informatik recht weit verbreitet. Es gibt andere Betriebssysteme, zum Beispiel UNIX™, die es dem Benutzer erlauben, bei fast jedem Befehl solche Muster für Dateinamen zu verwenden, beim Amiga gibt es jedoch (leider) nur einige Befehle, die diese verstehen.

Ein kurzer Tip vorweg: Für einen Menschen – und Sie sind ja ein Mensch und kein Computer - ist es nicht immer ganz einfach zu durchschauen, welche Dateien in ein solches Muster passen. Dies gilt besonders in der Anfangszeit, solange Sie noch nicht so gut mit Mustern vertraut sind. Gerade bei Befehlen, die Dateien modifizieren oder löschen können, kann es deshalb zu »peinlichen« Effekten kommen, wenn plötzlich mehr oder andere Dateien durch einen Befehl erfaßt werden, als Sie dachten. Solange Sie nicht ganz sicher sind, welche Dateinamen in ein bestimmtes Muster passen, verwenden Sie niemals Muster in Zusammenhang mit Befehlen, die Dateien modifizieren oder zerstören (löschen) könnten!

Muster unterscheiden sich von den normalen Namen von Dateien und Dateiverzeichnissen durch besondere Zeichen (Buchstaben), die in ihnen auftreten. Diese Sonderzeichen – man nennt Sie manchmal auch »Jokerzeichen« – sind ( ) ? % # und der senkrechte Strich | . Es ist deshalb wenig ratsam, wenn auch nicht unmöglich, diese Zeichen in Dateinamen zu verwenden. Jedes dieser Sonderzeichen hat bei der Bildung von Mustern eine bestimmte

Bedeutung. Einige sind Platzhalter für andere Buchstaben, andere fügen Teile von Mustern zu einem großen Muster zusammen. Muster werden gebildet, indem man »normale« Buchstaben und Sonderzeichen zu einem Muster zusammenfügt.

Der einfachste Platzhalter ist das Fragezeichen »?«. Kommt in einem Muster ein Fragezeichen vor, so passen alle Dateinamen in dieses Muster, bei denen an der Stelle, wo im Muster das Fragezeichen steht, ein beliebiger Buchstabe vorkommt. In das Muster H?ll? passen zum Beispiel die Dateinamen Hallo, Hello und Holla (aber nicht Heillo, da hier zwei Buchstaben an der Stelle des Fragezeichens im Muster stehen). Und in das Muster A?B passen alle dreibuchstabigen Dateinamen, deren erster Buchstabe ein A und deren letzter Buchstabe ein B ist. Die Groß- und Kleinschreibung der Buchstaben in einem Muster oder im Dateinamen spielt dabei, wie im Shell üblich, keine Rolle. Auch der Name aXb paßt deshalb in das Muster A?B.

Der Platzhalter % hingegen paßt nur auf den Leerraum zwischen zwei Buchstaben (nicht zu verwechseln mit der Leerstelle zwischen zwei Wörtern). Er wird Ihnen im Augenblick wahrscheinlich sinnlos erscheinen, erhält seinen Sinn aber zusammen mit anderen Sonderzeichen in einem Muster.

Die restlichen Sonderzeichen außer ? und % dienen zum Kombinieren von größeren Mustern aus anderen kürzeren Mustern. Hierzu brauchen Sie nur 4 Regeln. Diese Regeln sind recht formal und abstrakt, und wenn sie nicht gleich verständlich für Sie sind, machen Sie sich nichts draus! Am Ende dieses Abschnittes folgen eine ganze Reihe von Beispielen auch für komplexere Muster. Diese sollten völlig ausreichen, Ihnen einen Eindruck davon zu geben, wie man Muster aufbaut.

Hier nun die Regeln für den Zusammenbau von Mustern:

Sind <m1> und <m2> zwei Muster, dann ist <m1><m2> ein Muster, in das alle Dateinamen passen, deren erster Teil in das Muster <m1> paßt und deren »restlicher« Teil dann in das Muster <m2> paßt. Aus den beiden Mustern A?B und X??Z kann man zum Beispiel das Muster A?BX??Z bilden, in das unter anderem die Dateinamen AyBXikZ und AlBXpoZ passen. (Ich gebe zu, daß ich mir schönere Beispielnamen hätte einfallen lassen können. Vielleicht hätte man dann aber die Bedeutung der Sonderzeichen nicht mehr so deutlich gesehen.)

Das Doppelkreuz # ist das Wiederholungszeichen. Wenn <m> ein Muster ist, dann ist #<m> ein Muster, in das alle Texte passen, die aus Null, einer oder mehr Wiederholungen von <m> bestehen. In das Muster A#B passen also zum Beispiel die folgenden Dateinamen, die alle mit einem A beginnen, dem eine beliebige Anzahl B folgt: A, AB, ABB, ABB, und so weiter

Der senkrechte Strich | entspricht in etwa dem Wort »oder«. Sind <m1> und <m2> zwei Muster, dann ist <m1> | <m2> ein Muster, in das alle Dateinamen passen, die entweder in das Muster <m1> oder in das Muster <m2> passen. In das Muster »JaINein« passen also die beiden Dateinamen »Ja« und »Nein«.

Die runden Klammern ( ) dienen dazu, größere Muster zu bilden, die die einzelnen oben vorgestellten Möglichkeiten nutzen. Klammern fassen das Muster, das Sie enthalten, zu einer »Einheit« zusammen. So werden größere Muster einerseits lesbarer und andererseits auch eindeutig genug, damit der Amiga sie verstehen kann. Zum Beispiel sind A#AB und A#(AB) völlig unterschiedliche Muster. In das Muster A#AB passen die Dateinamen AB AAB, AAAB, AAAAAAAB und so weiter. In das Muster A#(AB) hingegen passen die

Namen AAB (der auch in das erste paßt), AABAB, AABABABABAB und so weiter; also alle Namen, die mit A beginnen und dann mit einer beliebig langen Folge der beiden Buchstaben AB enden. Die Klammern im Muster A#(AB) ändern also völlig dessen Bedeutung gegenüber demselben Muster ohne Klammern.

Im Zusammenhang mit Klammern und dem | wird auch das % sinnvoll. Das Muster (XY|%)ABC zum Beispiel trifft auf die beiden Dateinamen XYABC und ABC zu. Oder anders formuliert: es trifft auf alle Dateinamen zu, die mit XY oder »Nichts« beginnen und mit ABC enden.

Das letzte Sonderzeichen, das Hochkomma ', dient schließlich dazu, die besondere Bedeutung der anderen Sonderzeichen aufzuheben. Schreiben Sie direkt vor einem Sonderzeichen ein Hochkomma, dann können Sie so auch nach Dateinamen suchen, die ein Sonderzeichen enthalten. In das Muster A#('?) zum Beispiel passen die Dateinamen A, A?, A?????? und so weiter. Das sollte Sie aber nicht dazu animieren, in Dateinamen Sonderzeichen zu verwenden. Beschränken Sie sich am besten auf die Buchstaben des Alphabets und die Ziffern von 0 bis 9 und vermeiden Sie alle Satzzeichen und Sonderzeichen.

Im folgenden werden nun noch ein paar Beispiele für Muster angeführt, die Ihnen hoffentlich eine Vorstellung davon geben, was mit Mustern möglich ist. Achten Sie bitte besonders auf die Bedeutung von Klammern für den Aufbau von Mustern.

| **Muster** | **passende Dateinamen** |
|---|---|
| A?B | AAB, AbB, AcD, A1B, A2B |
| A#BC | AC, ABC, AbBC, ABbBc |
| A#(BC) | A, ABC, ABCbc, AbCBcBC, Abcbcbcbc |
| A#(B\|C)D | AD, ABD, ACD, ABBD, ACCD, ABCD, ACBD |
| A#?B | AB, A9B, ASchubiduB, AXYZ17QRXB |
| A(B\|%)#C | A, AB, ABC, AC, ACC, ABCCCCC |
| #?.TEXT | Brief.TEXT, alpha.TEXT, X1.TEXT |
| brief#? | brief, brief2, briefAnMama, »Brief an Mama« |
| '?#?'# | ?#, ?AB#, ?XYZ# |

***Tabelle 6.1:*** *Beispiel für die Verwendung von Mustern für Dateinamen*

Probieren Sie einfach selbst ein paar Muster aus. Für diesen Zweck eignet sich besonders gut das Directory *s* auf Ihrer Shell-Diskette, das besonders viele einzelne Dateien enthält. (Es handelt sich bei diesen Dateien um die Programme, die die Shell-Befehle ausführen.) Wenn Sie zum Beispiel den folgenden Befehl eingeben, erhalten Sie eine Liste aller Shell-Befehle, deren Name mit einem S beginnen (das »c/« am Anfang besagt, daß das c-Dierctory durchsucht werden soll):

```
1.SYS:>DIR c/s#?
```

Der folgende Befehl bringt hingegen die Befehle zum Vorschein, deren Name an zweiter Stelle ein E enthält:

```
1.SYS:>DIR c/?e#?
```

Und das letzte dieser kleinen Beispiele schließlich zeigt Ihnen eine Liste aller Befehle, deren Namen mit E enden:

```
1.SYS:>DIR c/#?e
```

Wie Sie an allen diesen Beispielen gesehen haben, ist besonders die Kombination #? sehr praktisch. Diese sollten Sie sich deshalb unbedingt einprägen! Sie können Muster aber natürlich nicht nur beim DIR-Befehl verwenden. Sie sind auch für das Löschen und Kopieren ganzer Gruppen von Dateien oft sehr wichtig. Erzeugen Sie nun bitte fünf Dateien mit den Namen test1, test2, test3, test4 und test5 aus der Datei Disk.Info.

```
1.SYS:>COPY Disk.Info test1
1.SYS:>COPY test1 test2
1.SYS:>COPY test1 test3
1.SYS:>COPY test1 testNummer4
1.SYS:>COPY test1 testDatei
```

Sie können alle diese Dateien nun mit einem Schlag in ein anderes Directory kopieren. Geben Sie dazu den folgenden Befehl ein:

```
1.SYS:>COPY test#? Utilities
```

Der COPY-Befehl listet dann die Dateinamen in der Reihenfolge auf, in der sie kopiert werden und meldet hinter jedem Namen *copied* (kopiert), wenn er mit der jeweiligen Datei fertig ist (die Reihenfolge, in der die Namen auftauchen, kann bei Ihnen allerdings auch eine andere sein).

Überzeugen Sie sich am besten erst einmal mit dem folgenden Befehl davon, daß die Dateien auch wirklich kopiert wurden:

```
1.SYS:>DIR Utilities
```

Die soeben kopierten Dateien können Sie nun auch mit einem einzigen Befehl löschen:

```
1.SYS:>DELETE test#?
```

Auch der DELETE-Befehl listet daraufhin die Dateinamen in der Reihenfolge auf, in der er die Dateien löscht und meldet zu jeder Datei Deleted, wenn er damit fertig ist. Beachten Sie bitte, daß dieser Befehl die Originale der Dateien löscht und nicht die Kopien im Directory *Utilities*.

Nun müssen Sie nur noch die Kopien dieser Testdateien löschen, die Sie ja mit COPY in das Directory Utilities gelegt haben. Geben Sie dazu den folgenden Befehl ein:

```
1.SYS:>DELETE #?/test#?
```

Dieser Befehl ist besonders interessant, weil er »directoryübergreifend« wirkt. Er durchsucht alle Directories (mit dem Teil »#?/« des Musters) im aktuellen Directory und löscht in diesen Directories alle Dateien, deren Name mit »test« beginnt und mit einem beliebigen Text endet (das besagt der Teil »test#?« des Musters).

Ein solches Muster ist allerdings nicht nur sehr praktisch, sondern auch ausgesprochen gefährlich. Sie sollten es (und ähnliche directoryübergreifende Muster) am besten erst dann verwenden, wenn Sie mit der Shell schon einige Zeit vertraut sind und die Wirkung von Mustern gut einschätzen können.

Ein weiteres Problem der Muster ist, daß Sie leider nicht von jedem Shell-Befehl in gleicher Weise verstanden werden. Das folgende Kapitel enthält eine Aufstellung aller Shell-Befehle. Bei allen Befehlen, die Muster »verstehen«, wird dies ausdrücklich hervorgehoben und auch beschrieben, bei welchen Parametern und wie Sie Muster einsetzen können. In Kapitel 8 erfahren Sie darüber hinaus auch noch, wie Sie selbst solchen Befehlen das Verständnis für Muster »beibringen« können, die von Hause aus nur mit normalen Dateinamen arbeiten können.

## 6.6 Directories und Geräte

Nach diesem kleinen Exkurs in die wichtigsten Shell-Kommandos werden Sie nun noch einmal mit theoretischen Grundlagen gequält, bevor Sie weitere neue Kommandos kennenlernen. Es geht dabei zum zweiten Mal um das Thema Directories (Dateiverzeichnisse), woraus Sie ersehen können, daß dieses Thema sehr wichtig für den Umgang mit der Shell ist. Besonders wichtig für die Arbeit mit der Shell ist zum Beispiel das aktuelle Directory. Nachdem Sie ja schon wissen, was das ist (siehe Abschnitt 6.4.5), werden Sie unter anderem nun auch erfahren, wie Sie es festlegen beziehungsweise ändern können.

### 6.6.1 Das aktuelle Dateiverzeichnis wechseln

Mit dem Shell-Befehl CD (für *change directory* = setze Dateiverzeichnis) kann ein beliebiges Dateiverzeichnis zum aktuellen Dateiverzeichnis erklärt werden. Geben Sie zunächst einmal den folgenden Befehl ein (nur die erste Zeile; die zweite Zeile zeigt Ihnen, was daraufhin auf Ihrem Bildschirm erscheinen sollte):

```
1.SYS:>CD
Shell-Diskette:
```

Falls die Diskette im internen Laufwerk einen anderen Namen als Shell-Diskette hat, erscheint dieser andere Name. Wenn Sie nämlich den Befehl CD ohne jeden Parameter verwenden, wird der Name des aktuellen Directory ausgegeben. Und das ist im Augenblick das Wurzelverzeichnis der Diskette im internen Laufwerk. Dieses Wurzelverzeichnis trägt denselben Namen wie die Diskette selbst, mit einem Doppelpunkt dahinter.

Sie können so jederzeit feststellen, wo Sie sich »im Dateienbaum« gerade befinden. Wenn Sie hingegen das aktuelle Directory wechseln wollen, müssen Sie den Pfadnamen des Directory, das das neue aktuelle Directory werden soll, hinter CD als Parameter angeben. Probieren Sie einmal den folgenden Befehl aus:

```
1.SYS:>CD DF0:c
1.CLI-Diskette:c>
```

Danach ist das Directory *c* auf der Diskette im internen Laufwerk (DF0:) das aktuelle Directory. Dieses Directory enthält die Shell-Kommandos, wovon Sie sich sofort überzeu-

gen können, indem Sie einfach DIR (und dann [Return]) eingeben. Auch wenn Sie jetzt CD ohne Parameter eingeben, meldet die Shell Ihnen, daß das aktuelle Directory nun *c* heißt. Genaugenommen benötigen Sie dafür aber den CD-Befehl nicht. Wenn Sie nämlich auf das Shell-Prompt achten, werden Sie feststellen, daß sich sein Aussehen gewandelt hat. Es lautet jetzt nicht mehr *1.SYS:>* sondern *1.CLI-Diskette:c>*. Das Shell-Prompt enthält neben der 1 am Anfang (dessen Bedeutung später erläutert wird) auch den kompletten Pfadnamen des aktuellen Directory. Praktisch, nicht wahr? (Wenn Sie der manchmal lange Name des aktuellen Directory stören sollte, können Sie ihn auch abschalten. Sie benötigen dazu den Befehl PROMPT, der im nächsten Kapitel erläutert wird.) Sie können sofort wieder zum Wurzeldirectory zurückkehren, indem Sie die folgende Zeile eintippen:

```
1.CLI-Diskette:c>CD DF0:
1.CLI-Diskette:>
```

Je nachdem, ob und wie Sie den CD-Befehl einsetzen, gibt es ganz verschiedene Wege, »sich einer Datei« zu nähern. Sie können sich mit dem CD-Befehl Schritt für Schritt herantasten oder auch sofort den kompletten Namen verwenden. Die folgenden vier Befehlsfolgen zum Beispiel leisten auf verschiedene Art alle dasselbe. (Überlegen Sie sich am besten einmal in aller Ruhe, warum.)

```
1.SYS:>TYPE df1:texte/privat/Brief1
```

```
1.SYS:>CD df1:texte/privat
1.Disk:texte/privat>TYPE Brief1
```

```
1.SYS:>CD df1:
1.Disk:>CD texte/privat
1.Disk:texte/privat>TYPE Brief1
```

```
1.SYS:>CD df0:texte
1.Disk:texte>CD privat
1.Disk:texte/privat>TYPE Brief1
```

Probieren Sie diese Befehle bitte nicht aus. Sie funktionieren nicht, da es auf Ihrer Shell-Diskette diese Dateien und Directories nicht gibt und Sie vielleicht gar kein zweites Laufwerk namens DF1: besitzen. Diese Befehle sind nur als abstrakte Beispiele für die Verwendung des aktuellen Directory und des CD-Befehls zu verstehen.

### 6.6.2 Auf und ab im Dateienbaum

Sie haben bislang zwei Alternativen kennengelernt, eine Datei auf einer beliebigen Diskette eindeutig zu beschreiben. Die eine Alternative ist der vollständige Pfadname (zum Beispiel *Archiv:Texte/Privat/Meier*), wobei Sie sich den Diskettennamen schenken können, wenn sich die Datei auf der aktuellen Diskette befindet (zum Beispiel *:Texte/Privat/Meier*). Die zweite Alternative ist der Dateiname allein (zum Beispiel *Meier*), der immer dann verwendet werden kann, wenn sich diese Datei im aktuellen Directory befindet. Hierzu müssen Sie gegebenenfalls aber erst einmal das aktuelle Directory ändern, um eine bestimmte Datei mit ihrem kurzen Namen bezeichnen zu können.

Es gibt jedoch noch eine dritte Möglichkeit, Dateien anzusprechen, die sich nicht im aktuellen Directory befinden. Diese Möglichkeit ergibt ebenfalls einen Pfadnamen (jedoch

nicht den »vollständigen Pfadnamen«). Dieser zweite Pfadname einer Datei ist oft kürzer als der vollständige. Er ist allerdings manchmal komplizierter und nicht so leicht zu bilden.

Zunächst einmal gilt, daß der Pfadname für die Dateien abgekürzt werden kann, die sich im aktuellen Directory oder in einem Directory befinden, das im aktuellen Directory (und noch weiter unten im Dateienbaum) liegt. Vor jeden Dateinamen setzt die Shell nämlich immer den Namen des aktuellen Dateiverzeichnisses, um den vollständigen Pfadnamen der Datei zu erhalten, sofern der eingegebene Name nicht mit einem Diskettennamen oder Doppelpunkt beginnt.

Das bedeutet zum Beispiel, daß die Datei *Archiv:Texte/Privat/Meier* auch *Privat/Meier* genannt werden kann, wenn das aktuelle Directory *Archiv:Texte* ist (*Archiv:Texte* + *Privat/Meier* = *Archiv:Texte/Privat/Meier*). Auf diese Weise können Sie alle Dateien und Directories benennen, die hierarchisch »tiefer« liegen. An Dateien, die hierarchisch höher oder in anderen Verzweigungen des Dateienbaums liegen, kommen Sie allerdings nicht heran.

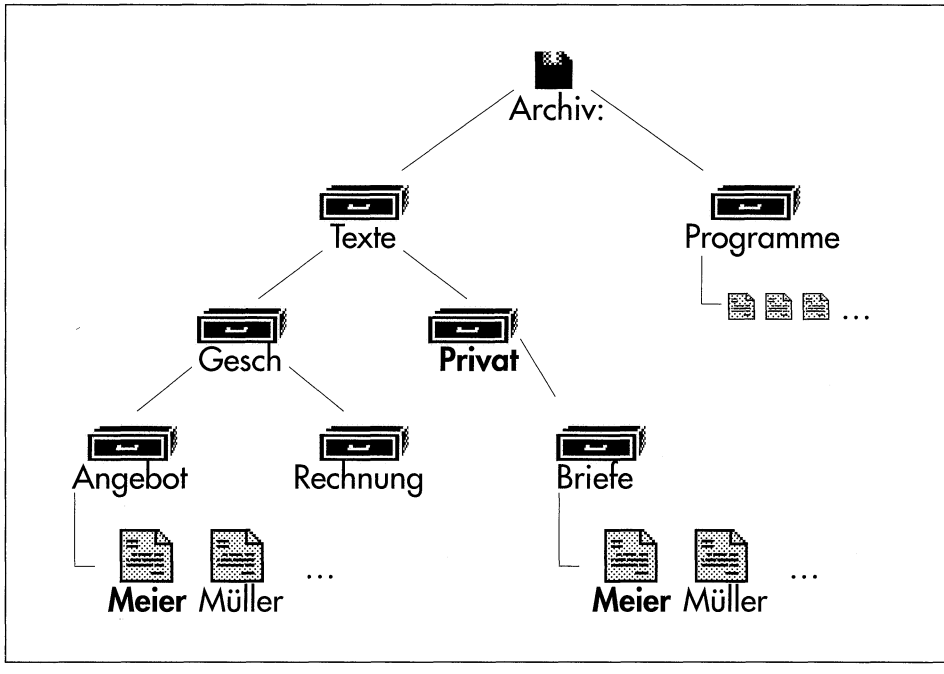



***Bild 6.6:*** *Auf und Ab im Dateienbaum*

In sehr fein und tief verästelten Dateibäumen kann es aber unpraktisch und schwierig sein, solche Dateien mit ihrem kompletten Pfadnamen angeben zu müssen. Wenn der Weg von der Wurzel zur gesuchten Datei sehr weit ist, kann es weniger Tipparbeit bedeuten, den Weg zu einer bestimmten Datei, ausgehend vom aktuellen Dateiverzeichnis, zu beschreiben. Auch dies ist möglich. Ähnlich wie das Zeichen Doppelpunkt (:) zur Wurzel der Dateiverzeichnisse auf einer Diskette führt und man einen Namen dann ausgehend von dieser Wurzel angeben kann, gibt es ein Zeichen, mit dem man im Baum nur »eine Stufe höher«, also in

das über dem aktuellen Directory liegende Directory kommt. Schreibt man zu Beginn eines Dateinamens einen Schrägstrich, so entspricht dieser Schrägstrich dem Dateiverzeichnis, in dem das aktuelle Dateiverzeichnis liegt. Dies kann man verwenden, um »im Baum höherzusteigen« oder auch, um zunächst höherzusteigen und dann wieder in einem Nachbar-Ast »abzusteigen«.

Betrachten Sie dazu noch einmal das Bild 6.6. Wenn das aktuelle Directory *Privat* heißt, kann die Datei Meier im Directory *Angebot* ja ohne Ausnutzung des Schrägstrichs nur erreicht werden, indem man mit *:Texte/Gesch/Angebot/Meier* im Baum ganz nach oben geht und von der Wurzel aus den Namen bildet. Mit der besonderen Funktion des Schrägstrichs könnten Sie aber auch schreiben */Gesch/Angebot/Meier*. Der Schrägstrich führt zunächst in das Directory *Texte* über dem aktuellen Directory. Von diesem aus geht es dann ins Directory *gesch*, dann nach *Angebot* und in diesem schließlich findet sich die Datei *Meier*. Der Schrägstrich kann besonders dann, wenn sich zwischen dem aktuellen Directory und der Wurzel schon sehr viele Directories befinden, sehr viel Tipparbeit sparen.

Indem Sie an den Beginn eines Dateinamens mehrere Schrägstriche schreiben, können Sie übrigens auch mehrere Stufen im Baum »nach oben klettern«. So könnten Sie vom aktuellen Directory *Privat* aus, die Datei *Meier* im Directory *Angebot* auch mit *//Texte/Gesch/Angebot/Meier* bezeichnen, was allerdings gegenüber *:Texte/Gesch/Angebot/Meier* keinen großen Vorteil darstellt – es illustriert nur eine mögliche Verwendung des Schrägstrichs am Anfang eines Dateinamens.

### 6.6.3 Neue Directories erzeugen und löschen

Wie Sie sich durch den Dateienbaum durchhangeln, sollten Sie nun wissen – zumindest so ungefähr. Jetzt brauchen Sie nur noch etwas Übung und Sie verstehen auch die längsten Pfadnamen. In diesem Abschnitt werden Sie noch erfahren, wie Sie neue Directories erzeugen und überflüssig gewordene wieder löschen können. Dann können Sie in der Shell schon alle Arbeiten durchführen, für die Sie bislang die Workbench benötigten.

Auf der Workbench konnten Sie neue Schubladen nur dadurch erzeugen, indem Sie vorhandene duplizierten. Speziell zu diesem Zweck befindet sich ja die Schublade *Empty* auf der Workbench-Diskette. In der Shell sieht das etwas anders aus. Wie Sie vielleicht schon erwartet haben, gibt es auch für das Erzeugen von neuen Directories im Shell einen speziellen Befehl. Er heißt MAKEDIR (engl. *make directory* = erzeuge ein Dateiverzeichnis). Sein einziger Parameter ist der Pfadname des neuen Directory, das Sie einrichten wollen. Wenn Sie den folgenden Befehl eingeben, erzeugen Sie zum Beispiel ein Directory namens *neudir* im Wurzeldirectory (:) der aktuellen Diskette. Probieren Sie es aus:

```
1.SYS:>CD   :
1.Shell-Diskette:>MAKEDIR neudir
```

Die folgenden vier Befehle legen zwei neue Directories im Directory *neudir* an und legen eine Kopie der schon mehrfach für diesen Zweck »mißbrauchten« Datei *Disk.info* unter den Namen *Test* in das neue Directory *neu3*:

```
1.Shell-Diskette:>MAKEDIR neudir
1.Shell-Diskette:>MAKEDIR neudir/neu2
1.Shell-Diskette:>CD neudir
```

```
1.Shell-Diskette:neudir>MAKEDIR neu3
1.Shell-Diskette:neudir>COPY :Disk.Info neu3/Test
```

Beachten Sie bei dieser Befehlsfolge vor allem auch, wie sich zwischendurch das aktuelle Directory ändert und wie mit Hilfe des Doppelpunkts die Datei *Disk.info* im Wurzeldirectory angesprochen wird.

Zum Löschen von Directories können Sie den Ihnen schon bekannten Befehl DELETE verwenden. Mit DELETE können Sie sowohl Dateien, als auch Directories löschen – Directories aber nur, wenn Sie leer sind. Davon können Sie sich sofort überzeugen. Versuchen Sie dazu einmal, das Directory *neu3*, das Sie eben erzeugt haben, wieder zu löschen:

```
1.Shell-Diskette:neudir>DELETE neu3
  Not Deleted - directory not empty
```

Sie erhalten daraufhin eine Fehlermeldung, die nichts anderes besagt, als daß Sie nur leere Directories löschen können. Mit den folgenden beiden Befehlen hingegen können Sie *neu3* wirklich löschen.

```
1.Shell-Diskette:neudir>DELETE neu3/test
1.Shell-Diskette:neudir>DELETE neu3
```

Wenn ein Directory sehr viele Dateien enthält, kann es aber recht lästig werden, diese alle nacheinander zu löschen, bevor man das Directory und seinen Inhalt wirklich los ist. Da scheint der *Discard*-Befehl der Workbench (oder der Mülleimer) schon wesentlich praktischer zu sein. Aber natürlich kennt auch die Shell – oder besser der DELETE-Befehl – eine Abkürzung, mit der Sie auf einen Schlag ein ganzes Directory löschen können. Die folgenden beiden Befehle löschen alle Dateien und Directories, die Sie in diesem Abschnitt erzeugt haben (wovon Sie sich mit dem DIR-Befehl überzeugen können):

```
1.Shell-Diskette:neudir>CD :
1.Shell-Diskette:>DELETE neudir ALL
```

Wenn Sie nämlich am Ende eines DELETE-Befehls das Schlüsselwort ALL verwenden und der einzige Parameter von DELETE der Name eines Directory ist, werden alle Directories, die darin liegen und alle Dateien in allen Directories gelöscht. Der DELETE-Befehl listet während dieses Vorgangs die Namen der gerade gelöschten Dateien auf. Wenn Sie bemerken, daß darunter Namen von Dateien auftauchen, die Sie gar nicht löschen wollten, können Sie mit [Ctrl]-[C] versuchen, »zu retten, was zu retten ist«. Der DELETE-Befehl wird dann sofort abgebrochen. Die bis dahin schon gelöschten Dateien sind allerdings verloren.

### 6.6.4 Geräte – nicht nur Wurzeln der Dateienbäume

Weiter oben haben Sie ja bereits erfahren, daß Sie die Wurzel des Dateiverzeichnisses einer Diskette entweder mit dem Namen der Diskette – wie er auf der Workbench zu sehen ist – bezeichnen können, oder mit dem Namen des Diskettenlaufwerks, in dem sich die Diskette befindet. Sie können einen Pfadnamen zum Beispiel mit DF0: beginnen, wenn sich die Diskette im internen und mit DF1:, wenn sie sich im ersten externen Laufwerk befindet. DF0: und DF1: sind »Gerätenamen«. Nicht nur Diskettenlaufwerke, sondern auch der

Drucker, der Bildschirm, die Tastatur und einige andere Geräte haben solche Namen. Alle diese Geräte muß man vom Shell ja mit einem Namen ansprechen können, wenn man sie für einen Befehl benötigt. (In der Workbench haben Sie es oft leichter – auf eine Diskette kann man zum Beispiel einfach mit der Maus zeigen.)

Amiga-DOS und die Shell behandeln solche Geräte – wenn es sich nicht gerade um Diskettenlaufwerke handelt – wie spezielle Dateien. Sie können deshalb bei vielen Shell-Befehlen statt einem Dateinamen auch einen Gerätenamen als Parameter verwenden. Das Shell »erkennt« Geräte daran, daß hinter ihrem Namen immer ein Doppelpunkt steht. So ist es zum Beispiel möglich, eine Datei auf das Drucker-Gerät (also den Drucker) zu kopieren, worauf sie natürlich ausgedruckt wird. Dies geht genauso wie das Kopieren einer Datei. Dazu müssen Sie nur statt des zweiten Dateinamens im Kopierbefehl den Gerätenamen PRT: (engl. für *printer* = Drucker) angeben. Der Befehl in der folgenden Zeile druckt zum Beispiel die Datei *startup-sequence* im Directory *s* der Shell-Diskette aus. (Wenn Sie keinen Drucker angeschlossen und eingeschaltet haben, sollten Sie diesen Befehl nicht ausprobieren, weil sonst nur ein Requester mit einer Fehlermeldung erscheint, die nichts anderes besagt, als daß der Amiga keinen Drucker finden kann.)

```
1.Shell-Diskette:neudir>COPY :s/startup-sequence to PRT:
```

Geräte sind aber wirklich sehr »spezielle« Dateien, weil man sie meist nicht löschen kann (wie soll man einen Drucker löschen?), sie manchmal nur beschreiben und nicht wieder lesen kann (wie soll man Daten von einem Drucker lesen?) und eine Reihe anderer Operationen, die mit Dateien möglich sind, mit Geräten nicht oder nur eingeschränkt verwendet werden können.

## 6.7 Ein-/Ausgabe-Umleitung

Die Gerätenamen, die Sie im letzten Abschnitt kennengelernt haben, spielen eine bedeutende Rolle im Zusammenhang mit einem weiteren sehr wichtigen Konzept der Shell: der sogenannten »Ein-/Ausgabe-Umleitung« (engl. *I/O redirection*). Viele Befehle der Shell lesen ihre Eingaben – sofern sie überhaupt welche benötigen – von einer Datei und schreiben alle Ausgaben in eine andere. Oft werden die Eingaben dabei von der Tastatur erwartet und die Ausgaben erscheinen im Shell-Fenster. (Auch die Tastatur und das Shell-Fenster sind eine Datei – oder genauer: ein Gerät.)

Den Programmen ist es jedoch in der Praxis meist gleichgültig, von welchen Dateien beziehungsweise Geräten sie ihre Eingaben erhalten und wo sie ihre Ausgaben hinschicken. Deshalb kann man bei den meisten Programmen diese Dateizuordnungen von Ein- und Ausgabe gefahrlos »umlegen«, zum Beispiel auf den Drucker oder in eine Diskettendatei. Sie können auf diese Weise zum Beispiel den Inhalt eines Dateiverzeichnisses mit DIR auf den Drucker schicken oder in einer Datei abspeichern, die Sie nachträglich mit einem Textverarbeitungsprogramm bearbeiten können.

### 6.7.1 Ein- und Ausgaben eines Shell-Kommandos umleiten

Um einem Shell-Kommando eine andere Eingabedatei zuzuweisen als die Tastatur, müssen Sie sofort hinter dem Befehlsnamen ein Kleinerzeichen (<) und danach den Namen der Datei

oder des Gerätes schreiben, von dem (der) die Eingaben kommen sollen. Genauso können Sie alle Ausgaben eines Befehls auf eine Datei oder ein Gerät schicken, indem Sie sofort hinter den Namen des Befehls ein Größerzeichen (>) und dann den Namen der Datei beziehungsweise des Geräts schreiben, in die (das) die Ausgaben gehen sollen.

Hierzu nun ein paar kleine Beispiele: Wenn Sie den folgenden Befehl eingeben, wird das Directorylisting nicht auf dem Bildschirm erscheinen, sondern geht in eine Datei namens *dirliste*.

```
1.Shell-Diskette:>DIR >dirliste
```

Das Diskettenlaufwerk arbeitet dann eine ganze Weile und doch erscheint kein Dateiverzeichnis auf dem Bildschirm. Sie können es sich aber anschauen, indem Sie nun die Datei *dirliste* mit Hilfe von TYPE am Bildschirm ausgeben.

```
1.Shell-Diskette:>TYPE dirliste
```

Alternativ dazu können Sie *dirliste* aber auch im Programm *Notepad* einlesen und dort bearbeiten. Wie man die Ausgaben eines Befehls umlenkt, dürfte damit klar sein. Die Beschreibung der Eingabeumleitung erspare ich mir, da Sie nur selten benötigt wird und von wenigen Befehlen genutzt werden kann. Höchstens als Programmierer und versierter Shell-Benutzer werden Sie die Eingabeumleitung benötigen, während Ausgabeumleitung für nahezu jeden Shell-Benutzer wichtig ist. Den vollen Nutzen der Ein-/Ausgabe-Umleitung können Sie wahrscheinlich sowieso erst abschätzen, wenn Sie täglich mit der Shell umgehen. Solche Umleitungen lassen sich sehr vielseitig einsetzen und sind deshalb eine Möglichkeit, die seit einigen Jahren in keinem neuen Betriebssystem für Computer mehr fehlen darf.

Ab der Version 1.3 der Workbench-Software gibt es neben > noch eine weitere Möglichkeit der Ausgabeumlenkung. Wenn Sie statt eines Größerzeichens zwei verwenden (>>), wird die Ausgabe des Befehls nicht in eine neue Datei geschrieben sondern an eine schon bestehende angehängt. Sie können mit dem folgenden Befehl zum Beispiel die Ausgaben mehrerer DIR-Befehle »hintereinanderhängen«:

```
1.Shell-Diskette:>DIR >>dirliste
```

### 6.7.2 Ein-/Ausgabe-Umleitung auf Geräte und Dateien

Sie sollten bei der Ein-/Ausgabe-Umleitung immer den grundsätzlichen Unterschied zwischen einer Umleitung auf eine Datei und ein Gerät beachten. Beides dient meist völlig unterschiedlichen Zwecken. So wird zum Beispiel die Ausgabe-Umleitung auf eine Datei meist deshalb vorgenommen, um eine »flüchtige« Ausgabe, die schnell über den Bildschirm huschen würde, permanent abzuspeichern, um sie danach in Ruhe betrachten oder mit anderen Programmen weiterverarbeiten zu können. Eine Umleitung auf ein Gerät hingegen erweitert fast immer die Funktionalität eines Befehls. Sie können zum Beispiel ein Directory nicht nur anzeigen, sondern auch ausdrucken lassen, indem Sie es mit dem folgenden Befehl zum Drucker schicken:

```
1.Shell-Diskette:>DIR >PRT:
```

Übrigens hätten sich die Entwickler der Shell wegen der Ein-/Ausgabe-Umleitung eigentlich auch den TYPE-Befehl sparen können. Der TYPE-Befehl macht ja nichts weiter, als den

Inhalt einer Textdatei in das Shell-Fenster zu kopieren. Das kann der COPY-Befehl natürlich auch – wenn man den Gerätenamen des Shell-Fensters kennt. Die beiden folgenden Befehlsfolgen sind deshalb »äquivalent«, das heißt, sie leisten dasselbe:

```
1.Shell-Diskette:>DIR >dirliste
1.Shell-Diskette:>TYPE dirliste
1.Shell-Diskette:>TYPE >PRT: dirliste
```

```
1.Shell-Diskette:>DIR >dirliste
1.Shell-Diskette:>COPY >* dirliste
1.Shell-Diskette:>COPY dirliste TO PRT:
```

Der erste Befehl schreibt jeweils das Dateiverzeichnis des aktuellen Directory in die Datei *dirliste*. Der zweite Befehl listet diese Datei dann auf dem Bildschirm auf und der dritte gibt sie auf dem Drucker aus (diesen letzten Befehl sollten Sie fortfallen lassen, wenn Sie keinen Drucker angeschlossen und eingeschaltet haben). Am interessantesten ist der Befehl *COPY >* dirliste*. In diesem wird die Ausgabe des COPY-Befehls auf das Gerät * (Stern) umgelenkt. Das ist ein ganz besonderes Gerät: das Shell-Fenster, in dem Sie gerade arbeiten. Deshalb erscheint die Datei *dirliste* (das Dateiverzeichnis) auf diesen Befehl hin am Bildschirm.

Ein für Umleitungszwecke besonders interessantes Gerät ist SPEAK:. Dieses Gerät tut genau das, was sein Name andeutet: es spricht (engl. *to speak* = sprechen). Wenn Sie die Ausgabe eines Befehls auf das SPEAK: umleiten, erscheint der entsprechende Text nicht am Bildschirm sondern ertönt aus dem Lautsprecher – sofern Sie einen Lautsprecher an die Tonausgänge an der Rückseite des Amiga angeschlossen haben. Der folgende Befehl spricht zum Beispiel den Inhalt des Directory *c* auf der Shell-Diskette aus:

```
1.Shell-Diskette:>DIR >SPEAK: c
```

Hören Sie einmal genau hin. Nach kurzer Zeit hat man sich an die Aussprache des Amiga gewöhnt und kann die Dateinamen wirklich verstehen. Genauso können Sie sich mit dem COPY-Befehl auch eine Textdatei vorlesen lassen, indem Sie die Datei nach SPEAK: kopieren. Dabei muß man allerdings erwähnen, daß der Amiga gewissen Schwierigkeiten mit nicht-englischen Texten hat.

Amiga-DOS und die Shell kennen über die bisher vorgestellten Gerätenamen hinaus aber noch eine ganze Reihe mehr »echte« Geräte (die »unechten« werden Sie weiter unten noch kennenlernen). Viele dieser Geräte sind für »normalsterbliche« Amiga-Anwender relativ uninteressant, da Sie nur von Programmierern richtig genutzt werden können. Die wichtigsten Geräte für den täglichen Einsatz wurden aber bereits alle beschrieben. Eine vollständige Liste aller Geräte finden Sie in Kapitel 7.

## 6.8 Alternative Wege zu Programmen und Dateien

Wie Sie in den vorangegangenen Abschnitten erfahren haben, ist es nicht immer ganz einfach, im Shell-Fenster eine Datei oder ein Directory zu bezeichnen. Während Sie auf der Workbench einfach nur mit der Maus ein Piktogramm anklicken, müssen Sie oft lange und komplizierte Pfadnamen bilden – oder sich eine Menge Tricks merken, um diese Namen abkürzen zu können. Im Shell-Fenster gibt es deshalb auch noch zwei andere Methoden, mit

denen nach Dateien gesucht wird, deren Namen Sie in seiner kurzen Form und nicht als Pfadnamen eingeben: »logische Geräte« und den »Suchpfad« für Programme.

### 6.8.1 Der Suchpfad für Programme

Wenn Sie einen Befehl im Shell-Fenster eingeben, so verwendet die Shell das erste Wort dieses Befehls als den Namen eines Programms. Um den Befehl auszuführen, wird zunächst dieses Programm gestartet – falls es gefunden wird. Dieses Programm führt dann den Shell-Befehl aus, wobei es gegebenenfalls die zusätzlichen Parameter verwendet, die die Shell ihm mitteilt. Damit das funktionieren kann, muß die Shell natürlich erst einmal das entsprechende Programm finden.

Eine Möglichkeit wäre es, nach der Eingabe jedes Shell-Befehls alle angeschlossenen Disketten und Festplatten komplett zu durchsuchen, bis ein Programm gefunden wird, dessen Name gleich dem ersten Wort in der Befehlszeile ist. Diese Suche würde aber sehr lange dauern. Deshalb sucht die Shell nur in ganz bestimmten Directories nach den Befehlsprogrammen. Diese Directories, die nacheinander überprüft werden, nennt man den Suchpfad (engl. path) für Programme.

Zumindest zwei Directories werden dabei immer durchsucht: das aktuelle und das Directory *c* auf der Workbench-Diskette (also der Diskette, mit der Sie den Amiga gestartet haben). Genauer gesagt, sucht die Shell im aktuellen Directory und im Wurzeldirectory des »logischen Laufwerks« C: nach Programmen. Das logische Laufwerk C: ist anfänglich identisch mit dem Directory *c* der Workbench-Diskette. (Diese Zuordnung kann allerdings geändert werden.) Zu den logischen Laufwerken – es gibt noch mehr als nur C: – erfahren Sie im folgenden Abschnitt noch mehr.

Dieser recht kurze Suchpfad, der nur aus zwei Directories besteht ist aber nicht ausreichend. Wenn Sie sich zum Beispiel einmal kurz die Organisation der Workbench-Diskette vor Augen halten, die Sie zu Ihrem Amiga mitgeliefert bekommen, werden Sie feststellen, daß sich in mindestens drei Directories Programme finden: im Directory *c* (die Shell-Kommandos), in Directory *Utilities* (unter anderem *Notepad*) und im Directory *System* (unter anderem *DiskCopy* und *Format*). Nur eines dieser Directories kann das logische Laufwerk C: sein. Programme in den beiden übrigbleibenden Directories würde die Shell nicht finden, außer unter ihrem vollen Pfadnamen, wenn nicht zufällig eines dieser Directories das aktuelle wäre.

Um das Auffinden von Programmen etwas flexibler zu gestalten und Programme an möglichst vielen Stellen ablegen und sie trotzdem mit kurzen Namen aufrufen zu können, wurde die Möglichkeit eingeführt, den Suchpfad nahezu beliebig zu verlängern. Alle diese Directories werden von der Shell nacheinander durchsucht, um ein Programm zu finden, dessen Namen Sie eingegeben haben. Erst wenn in keinem dieser Directories ein Programm des angegebenen Namens gefunden werden kann, gibt die Shell eine Fehlermeldung aus.

Zur Änderung und zum Betrachten des Suchpfads dient der Befehl PATH (engl. *path* = Weg/Pfad). Geben Sie diesen Befehl einmal ohne alle Parameter ein:

```
1.Shell-Diskette:>PATH
Current Directory
RAM DISK:
Shell-Diskette:c
Shell-Diskette:Utilities
Shell-Diskette:System
Shell-Diskette:s
Shell-Diskette:Prefs
C:
```

Ohne Parameter aufgerufen, listet PATH nämlich die Directories im Suchpfad in der Reihenfolge auf, in der sie die Shell »abklappert«, um ein Programm zu finden. Die zweite Form des PATH-Befehls dient dazu, ein weiteres Directory in den Pfad einzuhängen. Geben Sie beispielsweise die folgende Zeile ein und lassen Sie sich dann noch einmal (mit PATH allein ohne Parameter) den Suchpfad zeigen:

```
1.Shell-Diskette:>PATH DF0:fonts
1.Shell-Diskette:>PATH
Current Directory
RAM DISK:
Shell-Diskette:c
Shell-Diskette:Utilities
Shell-Diskette:System
Shell-Diskette:s
Shell-Diskette:Prefs
Shell-Diskette:fonts
C:
```

Sie haben auf diese Weise das Directory DF0:fonts (also *fonts* auf der Workbench-Diskette) dem Pfad hinzugefügt. Dieses Directory wird nun nach allen anderen Directories aber noch vor C: durchsucht, um Programme zu finden. Das hat allerdings keine Wirkung, da dieses Directory keine Programme enthält. Sie können auch mehrere Directories auf einmal zum Pfad hinzufügen. Schreiben Sie hierzu einfach bis zu zehn Directory-Namen hinter PATH. Alle diese Directories (sie können auch auf anderen Disketten liegen) werden hinten an den Suchpfad angehängt.

Normalerweise wird man möglichst viele Directories in den Pfad hängen. Es kann jedoch auch nötig werden, wieder Directories aus dem Pfad herauszunehmen. Wenn der Suchpfad zum Beispiel Directories auf Disketten enthält, die sich in keinem Laufwerk befinden, so kann es passieren, daß diese Disketten jedesmal angefordert werden, wenn Sie einen Befehl eingeben. In so einem Fall ist es nötig, den Suchpfad »auf Null zu setzen«. Geben Sie dazu den folgenden Befehl ein:

```
1.Shell-Diskette>PATH RESET
```

Mit dem Schlüsselwort RESET werden alle zusätzlich durchsuchten Directories aus dem Suchpfad genommen. Danach wird nur noch das aktuelle Directory und das logische Laufwerk C: durchsucht, um ein Programm zu finden, das einen im Shell-Fenster eingege-

benen Befehl ausführen kann. Alle anderen Directories, die durchsucht werden sollen, müssen Sie einzeln wieder hinzufügen. Es gibt leider keine Möglichkeit, ein einzelnes Directory aus dem Pfad zu entfernen.

Jeder Shell-Benutzer wird im Laufe der Zeit seine eigenen Vorlieben hinsichtlich des Suchpfad-Inhalts entwickeln. Diese Vorlieben hängen natürlich stark davon ab, ob mehrere Diskettenlaufwerke oder gar Festplatten vorhanden sind und wie diese strukturiert (in Schubladen aufgeteilt) sind. Sie werden in Kapitel 8 noch erfahren, wie Sie dafür sorgen können, daß Ihr Lieblingssuchpfad gleich nach dem Einschalten gültig ist.

### 6.8.2 »Logische« Geräte

Eine wichtige Komponente jedes Suchpfads ist das »logische Laufwerk« C:. In vielen Befehlen kann dieses Laufwerk behandelt werden wie ein Diskettenlaufwerk. Die Befehle CD C: oder DIR C: sind zum Beispiel erlaubt. Hinter diesem Laufwerk verbirgt sich aber kein »echtes Diskettenlaufwerk« sondern ein Directory. Nach dem Start ist es das Directory *c* auf der Diskette im internen Laufwerk, theoretisch kann mit C: aber auch ein anderes Laufwerk gemeint sein.

C: ist aber nicht das einzige logische Laufwerk, das die Shell kennt. Genauso gibt es anderes logisches Laufwerk, das dem DEVS-Verzeichnis auf der Workbench-Diskette entspricht, eines, das dem LIBS-Verzeichnis entspricht und so weiter. Sie können sich eine Übersicht sämtlicher logischer Laufwerke – inklusiv der wirklichen, echten Geräte – anzeigen lassen, indem Sie den Befehl ASSIGN ohne alle Parameter eingeben. Probieren Sie es aus! Daraufhin dürfte am Bildschirm eine Liste erscheinen, die so ähnlich aussieht wie die folgende Aufstellung:

```
1.Shell-Diskette:>ASSIGN

Volumes
Platte [Mounted]
CLI-Disk [Mounted]

Directories:
CLIPS        RAM DISK:clipboards
ENV          RAM DISK:env
T            RAM DISK:t
S            Shell-Diskette:s
L            Shell-Diskette:l
C            Shell-Diskette:c
FONTS        Shell-Diskette:fonts
DEVS         Shell-Diskette:devs
LIBS         Shell-Diskette:libs
SYS          Shell-Diskette:

Devices:
PIPE AUX SPEAK NEWCON DH0
DF0  PRT  PAR  SER
RAW  CON  RAM
```

Der – im Moment – interessanteste Teil dieser Liste beginnt hinter der Überschrift *Directories*. Dahinter werden alle logischen Laufwerke und die Dateiverzeichnisse, denen sie zugeordnet sind, aufgelistet. Beachten Sie, daß ein logisches Laufwerk nicht zwangsläufig so heißen muß wie das entsprechende Dateiverzeichnis – selbst wenn das in dieser Liste so aussieht. Sie können beim Erzeugen eines logischen Laufwerks einen nahezu beliebigen Namen verwenden. Geben Sie dazu den folgenden Shell-Befehl ein:

```
1.Shell-Diskette:>ASSIGN K: DF0:SYSTEM
```

Wenn Sie dann noch einmal den ASSIGN-Befehl ohne Parameter aufrufen, werden Sie feststellen, daß es ein neues logisches Laufwerk namens K: gibt und das dieses dem Verzeichnis *Shell-Diskette:System* entspricht. (»Shell-Diskette:« und DF0: haben ja dieselbe Bedeutung, solange die Shell-Diskette im internen Laufwerk liegt.) Sie haben gerade ein logisches Laufwerk erzeugt. Dessen Inhalt können Sie sich sofort mit dem Befehl DIR K: zeigen lassen.

Eine wichtige Aufgabe logischer Laufwerke haben Sie damit schon kennengelernt: Sie erlauben es, praktische Abkürzungen für die Pfadnamen von Dateiverzeichnissen zu formulieren. Wenn Sie das erste Beispiel zu diesem Thema nicht überzeugt, betrachten Sie das folgende:

```
1.Shell-Diskette:>ASSIGN BR: DF0:TEXTE/PRIVAT/VON1987/BRIEFE
```

Dieser ASSIGN-Befehl bewirkt, daß Sie in allen CLI-Befehlen statt des Pfadnamens immer das Kürzel BR: verwenden können. Praktisch nicht wahr?

Das ist aber nicht die einzige oder wichtigste Aufgabe logischer Laufwerke. Logische Laufwerke machen Programmierern und Shell-Benutzern vor allem die Arbeit leichter. Sie können sich zum Beispiel darauf verlassen, das das logische Laufwerk C: immer die Programme enthält, die die CLI-Befehle ausführen. Ob dieses Dateiverzeichnis *c* heißt und auf der Diskette im internen Laufwerk liegt, oder ob Sie das Verzeichnis später einmal *Programme* genannt haben und auf Ihre neue Festplatte gelegt haben, spielt keine Rolle.

Genauso ist es mit den Zeichensätzen, die Ihnen in Textverarbeitungs- und Zeichenprogrammen zur Verfügung stehen. Alle Programme suchen diese Zeichensätze beziehungsweise die Dateien, in denen sie enthalten sind, im logischen Laufwerk FONTS:. Nach dem Start des Amiga entspricht FONTS: zunächst dem Dateiverzeichnis FONTS auf der Workbench-Diskette. Die Auswahl der zur Verfügung stehenden Schriften auf der Workbench-Diskette ist aber nicht sehr groß – nicht zuletzt deshalb, da diese Diskette zuviele andere Dateien aufnehmen muß, um noch Platz für eine Vielzahl von Schriften zu haben.

Sie können sich aber eine Diskette anfertigen, auf der Sie ausschließlich Zeichensätze speichern. Angenommen, diese Diskette heißt »Schriften«. Sie brauchen dann nur diese Diskette in ein beliebiges Laufwerk einlegen und dann den Befehl »ASSIGN FONTS: Schriften:« im CLI eingeben. Dann stehen die Zeichensätze auf der Diskette sofort in allen Programmen zur Verfügung, die die Auswahl eines Zeichensatzes ermöglichen.

Das funktioniert natürlich nur, weil alle Amiga-Programme die Dateien mit den Zeichensätzen nicht in einem bestimmten Dateiverzeichnis sondern immer im logischen Laufwerk FONTS: suchen.

Das »Umlenken« eines logischen Laufwerks können Sie sofort ausprobieren, wenn Sie ein zweites Diskettenlaufwerk besitzen. Auf der Extras-Diskette befinden sich nämlich zusätz-

liche Schriften. Legen Sie deshalb nun in dieses zweite Diskettenlaufwerk die Extras-Diskette ein und geben Sie den folgenden Befehl im Shell-Fenster ein:

```
1.Shell-Diskette:>ASSIGN FONTS: DF1:fonts
1.Shell-Diskette:>notepad
```

Warten Sie dann bis das *Notepad*-Fenster erscheint und betrachten Sie dann das *Font*-Menü. Er sollte jetzt deutlich anders aussehen, als bei allen früheren Experimenten mit *Notepad*. Statt der vielen verschiedenen Schriften, die alle nach Edelsteinen oder Halbedelsteinen benannt sind, werden jetzt lediglich noch *Topaz* und die drei neuen Schriften *Courier*, *Helvetica* und *Times* angeboten. Probieren Sie diese Schriften am besten einmal aus und verlassen Sie *Notepad* dann wieder.

Die Veränderung des *Font*-Menüs ist mit einem einzigen ASSIGN-Befehl bewirkt worden. *Notepad* durchsucht nach dem Start nämlich das logische Laufwerk FONTS: nach den zur Verfügung stehenden Schriften. Dieses Laufwerk entspricht jetzt aber dem Directory *Fonts* auf der Extras-Diskette. Und in diesem Directory befinden sich andere Schriften als im Directory *Fonts* der Workbench-Diskette.

Sinn und Zweck der logischen Laufwerke dürfte mit diesem kleinen Beispiel klar geworden sein. Damit soll die Beschreibung dieses sehr wichtigen Konzepts von Amiga-DOS (beziehungsweise der Shell) aber auch abgeschlossen werden. Wer sich dafür interessiert, welche Bedeutung die anderen logischen Geräte haben, die der ASSIGN-Befehl auflistet, kann eine vollständige Auflistung aller logischen Geräte (mit kurzen Beschreibungen dazu) in Kapitel 7 finden.

## 6.9 Parallel mit mehreren Shells arbeiten

Vielleicht ist es Ihnen noch nicht aufgefallen, aber die Shell hat (scheinbar) einen großen Nachteil gegenüber der Workbench. Der Arbeitsablauf in der Shell sieht so aus, daß man der Shell einen Befehl erteilt beziehungsweise ein Programm startet, dann wartet, bis dieses Programm fertig ist und erst danach den nächsten Befehl erteilen kann. Die Multitasking-Fähigkeiten des Amiga-Betriebssystems werden dabei nicht ausgenutzt. Auf der Workbench hingegen können Sie ein Anwendungsprogramm starten, dann auf ein Disketten-Piktogramm klicken und es öffnen (was einem DIR-Befehl entspricht) und mit einem weiteren Mausklick wieder im zuvor gestarteten Programm zurück sein. Das ist ja schließlich der Vorteil des Multitasking, den der Amiga den meisten anderen PCs voraus hat.

Aber auch in der Shell gibt es die Möglichkeit mit mehreren Programmen gleichzeitig beziehungsweise im schnellen Wechsel zu arbeiten. Dazu muß man allerdings einige spezielle Vorkehrungen treffen. Im Gegensatz zur Workbench startet die Shell ein Programm, wenn Sie dessen Namen am Anfang einer Befehlszeile eingeben, normalerweise nicht als selbstständigen Prozeß. Sie können nicht gleichzeitig mit einem solchen Programm arbeiten und der Shell neue Befehle erteilen, wie es auf der Workbench ja üblich ist. Das liegt daran, daß die Shell immer erst auf die Beendigung eines einmal gestarteten Programms wartet, bevor sie einen neuen Befehl akzeptiert. Das ist aber nicht ganz so schlimm, wie es sich anhört, da die Shell-Befehle fast alle sehr kleine Programme sind, die auch recht schnell mit ihrer Aufgabe fertig sind. Wenn man vom Shell-Fenster aus allerdings ein Programm wie *Notepad* startet, kann dieses Verhalten schon lästig sein.

Sollten Sie es aber sehr eilig haben, können Sie auch schon einen neuen Befehl eingeben, bevor der vorhergehende ganz fertig ist. Während Sie diesen Befehl eintippen, macht der gerade arbeitende Befehl eine Pause und fährt erst dann fort, wenn Sie die [Return]-Taste drücken. Das Shell wartet dann, bis das gerade laufende Programm beendet ist und führt danach den neuen Befehl aus. Das ist natürlich kein Multitasking, da immer nur ein Befehl ausgeführt wird und der nächste so lange warten muß, bis der vorangehende fertig ist. Es gibt aber auch Möglichkeiten für echtes Multitasking im Shell.

### 6.9.1 Mehrere Shell-Fenster öffnen

Die erste dieser Möglichkeiten ist die Verwendung mehrerer Shells. Es hält Sie ja nichts davon ab, nach dem Öffnen des ersten Shell-Fensters zurück in die Workbench zu gehen, und das Shell-Piktogramm ein zweites Mal zu öffnen. Das ergibt ein zweites Shell-Fenster, in dem Befehle parallel zu Befehlen im ersten Shell ablaufen können. Wenn im ersten Shell-Fenster zum Beispiel ein »längliches« Programm läuft (zum Beispiel der Compiler für eine Programmiersprache), können Sie einfach mit der Maus auf das zweite Shell-Fenster klicken und dort andere Befehle eingeben.

Neue Shell-Fenster können aber auch von einem anderen Shell-Fenster aus geöffnet werden. Hierzu dient der Befehl NEWSHELL (neue Shell). Geben Sie diesen Befehl versuchsweise zweimal ein.:

```
1.SYS:>NEWSHELL
```

Daraufhin erscheint ein neues Shell-Fenster und darin die Meldung:

```
New Shell process 2
2.SYS:>
```

Geben Sie den NEWSHELL-Befehl (in dem neuen Fenster gleich noch einmal ein:

```
2.SYS:>NEWSHELL
```

Dann erscheint ein drittes Shell-Fenster. Die drei Shell-Fenster liegen allerdings exakt übereinander. Legen Sie diese Fenster deshalb am besten etwas beiseite und verkleinern Sie sie, damit die Inhalte aller drei Shell-Fenster sichtbar sind.

Vielleicht ist Ihnen beim Erscheinen der beiden neuen Shell-Fenster aufgefallen, daß das Prompt in diesen Fenstern etwas anders aussieht. Statt 1.SYS:> steht dort nun 2.SYS:> oder 3.SYS:>. Normalerweise zeigt jeder Shell nämlich am Beginn des Prompts (vor dem Pfadnamen des aktuellen Directory) seine »laufende Nummer« an. Diese sogenannte »Prozeßnummer« wird aufsteigend von 1 bei der Entstehung eines neuen Shells »vergeben« und ändert sich auch dann nicht, wenn andere Shells mit ENDSHELL beendet werden. (Sie können jedes beliebige Shell-Fenster mit ENDSHELL schließen.) Man kann anhand der Prozeßnummer immer eindeutig feststellen, mit welcher Shell man es »zu tun hat«.

Probieren Sie jetzt einmal aus, wie Sie mit mehreren Shell-Fenstern mehrere Programme gleichzeitig laufen lassen können. Geben Sie dazu in einem der neuen Fenster den folgenden Befehl ein:

```
2.SYS:>DIR DF0: ALL
```

(Klicken Sie das Fenster mit der Maus an, wenn die eingetippten Buchstaben nicht sofort darin erscheinen sollten.) Klicken Sie nun in ein anderes Shell-Fenster und geben Sie dort denselben Befehl (DIR DF0: ALL) ein. Sie können nun betrachten, wie beide DIR-Befehle nahezu gleichzeitig arbeiten. Die Bildschirmausgaben der beiden Programme wechseln sich ab. Immer wenn das eine DIR-Programm auf die Daten von der Diskette warten muß, gibt das andere die Namen sehr schnell aus, bis es selbst wieder Daten von der Diskette lesen muß. Dann wird wieder das erste Programm schneller und so weiter. Multitasking live auf dem Amiga!

Sollten Ihnen drei Shell-Fenster nicht genügen, können Sie jederzeit weitere öffnen. Geben Sie dazu in einem Shell-Fenster – egal in welchem – einfach noch einmal den NEWSHELL-Befehl ein.

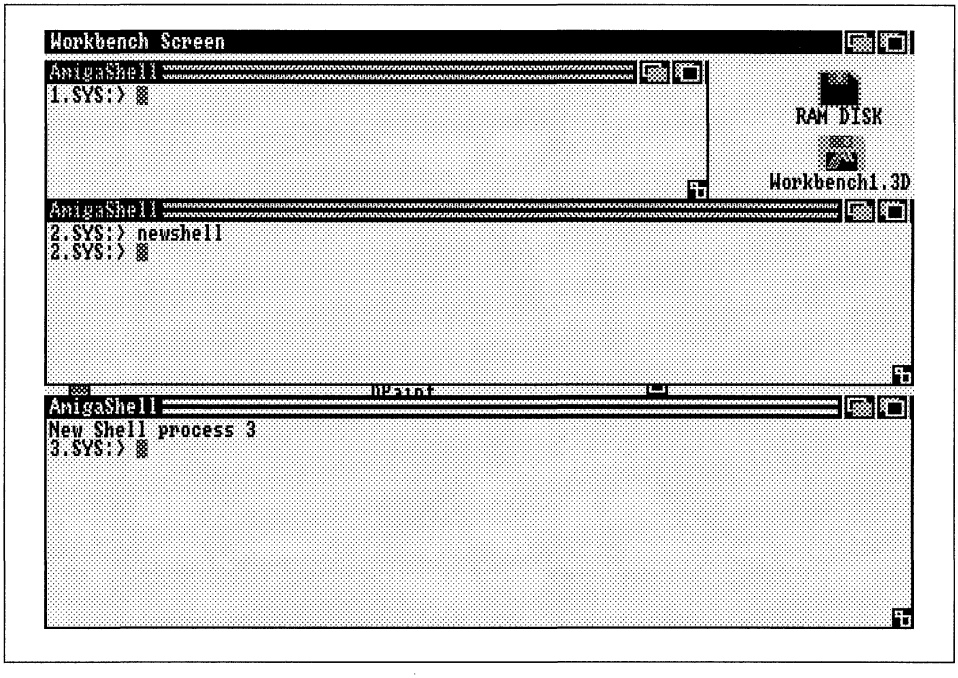



***Bild 6.7:*** *Mehrere Shell-Fenster gleichzeitig am Bildschirm*

Wenn Sie sich schließlich ganz verloren unter den vielen Shell-Fenstern fühlen und gar nicht mehr wissen, wieviele Fenster es gibt und welche Programme darin gerade laufen, gibt es natürlich wieder einen Shell-Befehl, der Ihnen weiterhelfen kann: STATUS. Probieren Sie diesen Befehl gleich aus. Öffnen Sie dazu mit dem NEWSHELL-Befehl mehrere Shell-Fenster (falls Sie das nicht schon getan haben) und geben Sie in einem dieser Fenster den Befehl »DIR ?« ein. Der DIR-Befehl wird dann gestartet, gibt sein Befehlsformular aus (siehe Abschnitt 6.5.9) und wartet auf weitere Eingaben. Wechseln Sie nun sofort zu einem anderen Shell-Fenster – indem Sie es mit der Maus anklicken – und geben Sie dort den

STATUS-Befehl ohne alle Parameter ein. Sie erhalten dann eine Liste, die so ähnlich aussieht wie die folgende Aufstellung:

```
2.SYS:>STATUS
Process 1: No command loaded
Process 2: Loaded as command: status
Process 3: Loaded as command: dir
```

Diese Liste sagt Ihnen, daß es insgesamt 3 Tasks (bei Ihnen vielleicht auch mehr ) gibt, daß der Task Nummer 3 (also das Shell-Fenster, in dem der Prompt mit 3 beginnt) gerade den Befehl DIR ausführt und daß der Task Nummer 2 (das gerade aktive Shell-Fenster) den Befehl STATUS ausführt – was Sie aber wahrscheinlich schon wußten.

### 6.9.2 Hintergrund-Befehle

Man muß aber nicht unbedingt gleich ein neues Shell-Fenster öffnen, wenn man mehrere Programme (Befehle) gleichzeitig verwenden will. Das macht nur Sinn, wenn diese Programme das Shell-Fenster für Ein- oder Ausgaben benutzen (wie zum Beispiel der DIR-Befehl). Wenn Sie ein Programm wie *Notepad* oder *Preferences* aufrufen, öffnet dieses sein eigenes Fenster und benötigt deshalb das Shell-Fenster nicht.

In solchen Fällen ist es praktischer, in der Shell eine andere Art des Multitasking zu nutzen: die »Hintergrund-Befehle«. Wenn Sie einen Befehl aufrufen wollen, von dem Sie genau wissen, daß er zwar lange dauert, aber keine weiteren Tastatureingaben mehr benötigt und nur wenig oder keine Bildschirmausgaben macht, brauchen Sie dazu kein neues Shell-Fenster zu öffnen. (Schließlich benötigt jedes zusätzliche Fenster zusätzlichen Speicherbedarf.) Sie können dieses Programm statt dessen mit dem Befehl RUN »im Hintergrund« laufen lassen.

Den Befehl RUN können Sie an den Anfang jeder Befehlszeile im Shell-Fenster schreiben und er führt den dahinter folgenden Befehl als separaten Prozeß parallel zum Shell aus. Lassen Sie beispielsweise einmal den DIR-Befehl im Hintergrund laufen:

```
1.SYS:>RUN DIR c:
```

Sie können an diesem Beispiel gleich sehen, wie unpraktisch es ist, den RUN-Befehl zusammen mit einem Shell-Befehl wie DIR zu verwenden. Das Shell-Prompt erscheint zwar sofort wieder und Sie könnten auch sofort einen neuen Befehl eingeben. Das »im Hintergrund« laufende DIR-Programm verwendet aber dasselbe Fenster wie die Shell »im Vordergrund«. Deshalb erscheinen laufend neue Textzeilen am Bildschirm und machen die Eingabe neuer Befehle sehr schwierig. Läuft mehr als ein Programm, das Ein- und Ausgaben im Fenster machen will, kann dies sehr verwirrend wirken. Falls Sie einen Eindruck von dem Chaos haben wollen, das dann am Bildschirm entstehen kann, geben Sie einmal die beiden folgenden Befehle ein:

```
1.SYS:>RUN DIR DF0: ALL
1.SYS:>DIR
```

Für die »Hintergrund-Verarbeitung« eignen sich wirklich nur die Befehle, die keine weitere Interaktion mit dem Anwender benötigen oder für diesen Zweck ein eigenes Fenster öffnen. Geben Sie einmal den folgenden Befehl ein:

```
1.SYS:>RUN NOTEPAD
```

Dann erscheint das *Notepad*-Fenster und Sie können jetzt loslegen und einen Text eingeben und mit den Menüs von *Notepad* gestalten. Sie können aber auch jederzeit mit der Maus in das Shell-Fenster klicken und dort einige Befehle eingeben und ausführen lassen, ohne *Notepad* vorher zu beenden.

Würden Sie *Notepad* von einem Shell-Fenster aus aufrufen, ohne RUN an den Anfang der Zeile zu setzen, können Sie zwar auch das Shell-Fenster anklicken und dort Befehle eintippen, ohne *Notepad* zu beenden. Diese Befehle erscheinen aber nur als Text im Shell-Fenster. Sie werden aber erst dann ausgeführt, wenn *Notepad* beendet wird. Gewöhnen Sie sich deshalb am besten an, alle Programme, die eigene Fenster öffnen, immer mit Hilfe von RUN aufzurufen!

## 6.10 Zusammenhänge zwischen Shell und Workbench

Zum Abschluß dieser kleinen Einführung in die Shell möchte ich aber auch noch ein paar Wörter zum Zusammenhang zwischen Dateiverzeichnissen und Workbench-Schubladen beziehungsweise zwischen Dateien und Piktogrammen verlieren.

Man kann deren Beziehung mit wenigen Wörtern zusammenfassen: Jede Workbench-Schublade entspricht immer auch einem Directory, das auch im Shell-Fenster (zum Beispiel mit Hilfe des DIR-Befehls) zu sehen ist. Jedes Piktogramm, das in einem Schubladen-Fenster der Workbench erscheint, ist immer auch eine Datei (wenn es nicht eine Schublade ist). Der Name eines Schubladen-Piktogramms ist dabei auch gleich dem Namen des Dateiverzeichnisses und der Name eines anderen Piktogramms gleich dem Dateinamen in der Shell. Umgekehrt gilt diese Beziehung nicht. Nicht jedes Directory ist als Schublade auf der Workbench zu sehen und in einigen Directories, die als Schubladen sichtbar sind, befinden sich Dateien, die nicht als Piktogramme auf der Workbench zu sehen sind!

Schon auf der Standard-Workbench befinden sich eine ganze Reihe von Dateiverzeichnissen (und natürlich auch Dateien darin), die von der Workbench aus nicht zu sehen sind. Dies sind Dateien, von denen die Entwickler des Amiga meinen, daß sie der »einfache Benutzer« nicht zu sehen braucht, beziehungsweise gar nicht sehen darf, damit er nicht irgendwelchen Unsinn damit anstellt. Diese Einstellung der Amiga-Entwickler ist zwar etwas überheblich aber zumindest nicht völlig falsch. Eine wichtige Datei, die man nicht sieht, kann man nicht versehentlich löschen.

Erst wenn man sich in die Shell begibt, bekommt man alle Dateien und Dateiverzeichnisse über die entsprechenden Shell-Befehle zu sehen. Die Entwickler des Amiga waren offensichtlich der Meinung, daß Benutzer, die sich bis zum Shell vorgearbeitet haben, bereits soviel Wissen, Verantwortungsbewußtsein und Vorsicht erworben haben, daß man ihnen selbst »das Innerste des Amiga ausliefern kann«.

Da stellt sich natürlich die Frage, wodurch eine Datei oder ein Directory als Piktogramm auf der Workbench sichtbar wird. Diese Frage ist sehr einfach zu beantworten: In der Workbench werden Dateien und Dateiverzeichnisse erst dann sichtbar, wenn es im selben Directory eine Datei gibt, die den gleichen Namen mit der zusätzlichen Endung ».info« trägt. Ein Piktogramm namens *Meier* taucht also erst dann in der Workbench auf, wenn es auch eine Datei namens *Meier*.info im selben Directory gibt. Dasselbe gilt sinngemäß für Directories beziehungsweise Schubladen.

Sie können sich davon sehr leicht mit dem DIR-Befehl überzeugen. Zu allen Namen, die aufgelistet werden und die Sie auch auf der Workbench sehen, gibt es ein Pendant mit dem angehängten ».info«. Diese zusätzliche Datei enthält unter anderem eine Beschreibung des Piktogramms der Datei und weitere Verwaltungsinformationen, die nur die Workbench benötigt. Mehr dazu erfahren Sie noch in Kapitel 8. Allen Lesern, die ein klein wenig mitdenken können, sollte jetzt aber schon klar sein, wie Sie ein »unsichtbares« Directory sichtbar machen können. Ein Tip: Sie benötigen den COPY-Befehl dazu.

Mit diesem Tip ist die Einführung in die Amiga-Shell dann aber auch abgeschlossen. »Tiefer« in die Geheimnisse der Shell einzudringen, sollte Ihnen jetzt nicht besonders schwer fallen – es kostet lediglich etwas Zeit für eigene Experimente und Übungen. Einige Informationen zu den noch nicht beschriebenen Shell-Befehlen benötigen Sie allerdings auch. Deshalb finden Sie im nächsten Kapitel zu jedem Befehl, den Sie in der Shell aufrufen können, eine knappe Beschreibung mit Anwendungsbeispielen.

# 7 Die Amiga-Shell Nachschlageteil

Nachdem Sie im letzten Kapitel die Grundzüge der Shell und die wichtigsten Shell-Befehle kennengelernt haben, erhalten Sie auf den folgenden Seiten nun eine (fast) vollständige Liste der CLI-Kommandos. Diese Liste ist insofern unvollständig, da der Befehlsumfang der Shell jederzeit erweitert oder verringert werden kann. Denn jedes Programm, das Sie auf die Workbench-Diskette (oder auch auf eine andere Diskette) legen, kann zu einem Shell-Befehl werden.

## 7.1 Überblick über die Shell-Befehle

Alle Shell-Befehle entsprechen kleinen (und manchmal auch großen Programmen). Anders als bei anderen Computern und deren Betriebssystemen, ist nicht ein einziger Befehl in das Programm Shell »fest eingebaut«. Das Programm Shell stellt lediglich ein Fenster für Ihre Tastatureingaben und für die Ausgaben anderer Programme zur Verfügung und ist imstande, andere Programme zu finden und zu starten. Wenn Sie eine Zeile im Shell-Fenster eingeben und dann die [Return]-Taste drücken, nimmt die Shell das erste Wort dieser Zeile und sucht nach einem Programm gleichen Namens. Findet es dieses Programm, so wird es in den Speicher geladen und gestartet. Der Rest der Zeile, hinter dem Namen des Programms, wird als ein oder mehrere Parameter an das Programm weitergegeben und es bleibt diesem Programm überlassen, was es damit tut.

Die Shell sucht nur an ganz bestimmten Stellen oder genauer gesagt: Directories, nach solchen Programmen. Diese durchsuchten Directories bilden den sogenannten »Suchpfad« (engl. *path*). Am Anfang des Suchpfads steht immer das aktuelle Directory, am Ende das »logische Laufwerk« namens C:. Programme, die sich nicht in den Directories des Suchpfads befinden, müssen unter Angabe des vollen Pfadnamens gestartet werden.

Ein besonders wichtiger Bestandteil des Suchpfads ist das logische Laufwerk C:. Es wird beim Start automatisch dem Directory *<Startdiskette>:c* zugewiesen. <Startdiskette> muß dabei natürlich gegen den Namen der Workbench-Diskette ersetzt werden, mit der Sie den Amiga gestartet haben. Diese Zuordnung kann aber mit dem Befehl ASSIGN, der im folgenden noch ausführlicher erklärt wird, jederzeit geändert werden. Solange sie aber nicht

geändert wird, wird der Amiga Sie jedesmal, wenn Sie einen Shell-Befehl eintippen und die Startdiskette nicht mehr in einem Laufwerk steckt, dazu auffordern, die Startdiskette einzulegen. Ähnliches gilt für die anderen Directories im Suchpfad, den Sie mit dem Befehl PATH, der ebenfalls noch im Detail erläutert wird, ändern können.

Nach diesen Erklärungen dürfte es wohl klar sein, warum die Shell nicht über einen festen Satz von Kommandos verfügt. Löscht man eines der Programme im logischen Laufwerk C:, so verliert man ein mögliches Kommando. Entfernt man ein Directory aus dem Suchpfad, verliert man eine ganze Gruppe möglicher Kommandos. Und schreiben Sie selbst ein Programm und legen es in eines der Directories im Suchpfad, so wird es dadurch automatisch ein Shell-Kommando. Auch, wenn Sie mit unterschiedlichen Disketten starten, kann es sein, daß Ihnen völlig unterschiedliche Befehlsmengen zur Verfügung stehen.

Doch damit genug der verwirrenden Möglichkeiten. Die im folgenden noch beschriebenen Befehle stellen den »Standardsatz« von Befehlen dar, so wie er Ihnen beim Kauf des Amiga auf der Workbench-Diskette mitgeliefert wird. Solange Sie keine Modifikationen an Ihrer Workbench-Diskette und (mit dem Befehl PATH) am Suchpfad, vornehmen, stehen Ihnen alle diese Befehle zur Verfügung. Zusätzlich dazu können auch fast alle Werkzeuge, die in den Kapiteln über die Workbench beschrieben wurden, auch von der Shell aus als Kommandos verwendet werden. Von diesen Programmen (Werkzeuge) werden hier aber nur diejenigen beschrieben, die für Shell-Anwender von besonderem Interesse sind.

## 7.2 Format eines Shell-Befehls

Alle Shell-Befehle verfügen über sich ähnelnde Aufruf-Formate. Am Anfang jeder Zeile, die einen Shell-Befehl enthält, steht der Name des Befehls. Dieser Name besteht immer aus einem kurzen Wort, das im allgemeinen einem (englischen) Verb gleich oder ähnlich ist. Hinter diesem Wort kann einer oder eine Reihe sogenannter Schlüsselworte und Parameter folgen, die dem Befehl genau sagen, was er zu tun hat. Ein Beispiel für einen Befehl mit Parametern und Schlüsselworten ist das Shell-Kommando DIR, das Sie ja bereits kennen. Der wichtigste Parameter von DIR ist der Name des Directory, dessen Inhalt angezeigt werden soll.

Wenn Sie den Namen des Befehls falsch beziehungsweise den Namen eines Befehls eingeben, der im Augenblick im Suchpfad nicht zu finden ist, so erhalten Sie die folgende Fehlermeldung:

```
Unknown command <name>
```

Wenn Sie hingegen den Befehlsnamen richtig eingeben und bei den Parametern einen Fehler machen, erscheint die Fehlermeldung:

```
Bad arguments
```

Sie können anhand dieser beiden Fehlermeldungen recht rasch feststellen, wo der Fehler liegt. Bedenken Sie bitte, daß nur die Befehle, die in den Directories des Suchpfads zu finden sind, unter Angabe ihres (kurzen) Namens aufgerufen werden können. Befehle (beziehungsweise Programme) in anderen Directories müssen unter Angabe des vollständigen Pfadnamens aufgerufen werden.

Bei falschen Parametern kann die Fehlerbehebung schwierig werden. Parameter sagen einem Befehl, mit welchen Objekten (Dateien, Directories, Texten und so weiter) er arbeiten soll und was er genau damit tun soll (falls es mehrere Möglichkeiten gibt). Diese Parameter werden in den im nächsten Abschnitt folgenden Format-Beschreibungen als Platzhalter beschrieben. Ein solcher Platzhalter ist immer ein kurzer Begriff, der in spitze Klammern gesetzt wird (zum Beispiel <dateiname>). Ein Platzhalter darf aber nicht wirklich eingetippt werden, sondern ist – wie der Name schon sagt – ein Platzhalter für den wirklichen Parameter. So muß zum Beispiel <dateiname> gegen einen echten Dateinamen ersetzt werden.

Wie auch im Fall des DIR-Befehls, können Parameter »optional« sein. Das heißt, sie **können**, müssen aber nicht angegeben werden. Wird ein optionaler Parameter nicht angegeben, geht der Befehl von einer Voreinstellung für diesen Parameter aus. Der DIR-Befehl nimmt, wenn er keinen Parameter erhält, zum Beispiel an, daß Sie die Liste der Dateien im aktuellen Directory sehen wollen.

Neben »echten« Parametern – üblicherweise Datei- und Directorynamen – können in Shell-Befehlen oft auch Schlüsselwörter vorkommen. Schlüsselwörter dienen manchmal zur genauen Identifizierung des nachfolgenden Parameters. Oft stehen sie aber auch für sich allein und haben die Funktion eines Schalters, mit dem man ein ganz bestimmtes Verhalten des Befehls aktiviert. (Zur Unterscheidung von Parametern werden Schlüsselwörter in diesem Buch immer groß geschrieben – wie die Namen der Befehle selbst.)

Ein gutes Beispiel für die Verwendung der verschiedenen Arten von Schlüsselwörtern ist der Befehl COPY. Dieser Befehl kennt unter anderem die Schlüsselwörter FROM, TO und ALL. FROM und TO sind »einleitende« Schlüsselwörter, denen ein Datei- oder Directoryname folgen muß. Hinter FROM folgt der Name der »Quelle«, die kopiert werden soll (eine Datei oder ein Directory), hinter TO der Name des »Ziels« (ebenfalls ein Dateiname oder ein Directory). Wenn die Quelle ein Directory ist, können Sie an das Ende des COPY-Befehls noch das Schlüsselwort ALL setzen. ALL ist ein eigenständiges Schlüsselwort, hinter dem kein Parameter folgen darf. Es bedeutet, daß der komplette Inhalt des Directorys, samt allen Unterdirectories und den Dateien in diesen, kopiert werden soll.

Schlüsselwörter vor Parametern sind oft optional. In vielen Fällen genügt es, den Parameter allein anzugeben. Der Befehl schließt dann aus der Position des Parameters in der Befehlszeile auf seine Bedeutung. Statt »COPY FROM Original TO Kopie können Sie deshalb auch einfach »COPY Original Kopie« schreiben. Wenn es jedoch nicht möglich ist, aus der Position eines Parameters auf seine Bedeutung zu schließen, was vor allem bei Befehlen mit vielen optionalen Parametern der Fall ist, muß das Schlüsselwort vorangestellt werden. Es macht den Befehl dann oft auch lesbarer. Wenn Sie ein Schlüsselwort vor einen Parameter schreiben, wird die Position der Parameter völlig irrelevant. Die folgenden Befehle tun deshalb alle exakt das Gleiche:

```
COPY FROM Text1 TO Kopie
COPY FROM Text1 Kopie
COPY Text1 TO Kopie
COPY Text1 Kopie
COPY TO Kopie FROM Text
```

Welche Parameter und Schlüsselwörter optional sind, wird in den Beschreibungen auf den folgenden Seiten immer eindeutig beschrieben. Achten Sie bitte immer darauf, daß es ein

entscheidender Unterschied ist, ob ein Parameter, ein eigenständiges Schlüsselwort oder das Schlüsselwort zu einem Parameter optional ist. Das Schlüsselwort FROM des COPY-Befehls ist zum Beispiel optional, nicht aber der dazugehörige Parameter (die Quelle des Kopiervorgangs).

Zuletzt noch ein Wort zu Parametern selbst. Meist sind diese Parameter, wie bereits erwähnt wurde, Dateinamen, Directorynamen oder andere kurze Texte. Die Shell faßt als Trennsymbol zwischen Namen und den in der Befehlszeile auftauchenden Schlüsselwörtern eine oder mehrere Leerstellen auf. Sie sollten deshalb in Datei- und Directorynamen am besten keine Leerstellen verwenden.

Wenn Sie das nicht vermeiden können (auf der Workbench ist die Versuchung dazu groß), müssen Sie den Namen in Anführungszeichen setzen. Wenn Sie also zum Beispiel die Datei *Test* in die Datei *Neue Datei* kopieren wollen, sollten Sie das wie folgt formulieren:

```
COPY Test TO "Neue Datei"
```

## 7.3 Die wichtigsten Shell-Befehle

Es folgt nun eine Liste, die Erläuterungen zu all den Shell-Befehlen enthält, die Sie »normalerweise« auf der Workbench-Diskette und der Extras-Diskette (jeweils in der Version 1.3) vorfinden werden. Dabei sind auch einige Programme aufgeführt, die Sie auch von der Workbench aus starten können, die sich aber besonders gut für die Benutzung von der Shell aus eignen. Diese Programme befinden sich in den Directories *Utilities* und *System* auf der Workbench-Diskette und im Directory *Tools* auf der Extras-Diskette. Alle anderen Shell-Befehle finden Sie im Directory *c* auf der Workbench-Diskette, das von der Workbench aus nicht sichtbar ist (und dem normalerweise das logische Gerät C: zugeordnet ist).

Aus Gründen der Übersichtlichkeit beginnen die Erläuterungen zu jedem Befehl auf einer neuen Seite. Dies dient nicht dazu – wie böswillige Menschen vielleicht behaupten würden – die Seitenzahl dieses Buches zu erhöhen, sondern wirklich nur, um Ihnen die Arbeit damit möglichst leichtzumachen.

Alle Befehlsbeschreibungen in diesem Abschnitt folgen einem festen Format. Jede Befehlsbeschreibung beginnt mit dem Namen des Befehls in besonders großer Schrift. Dann folgt eine detaillierte Beschreibung des Aufruf-Formats, eine Kurzbeschreibung von Sinn und Zweck des Befehls und schließlich eine etwas ausführlichere Erläuterung zur Anwendung des Befehls.

Die Formatbeschreibung enthält alle wichtigen Angaben, die Sie zum Umgang mit dem Befehl brauchen. Parameter kommen in dieser Beschreibung als Platzhalter (in spitzen Klammern) vor. Schlüsselwörter werden groß geschrieben. Wenn Sie bei einem Parameter auch ein Muster verwenden können, um mehrere Dateien auf einmal zu bearbeiten (wie in Kapitel 6 beschrieben), folgt dem entsprechenden Platzhalter ein Sternchen (*).

Jedem Schlüsselwort folgt ein oder eine Reihe von Buchstaben, die mit Schrägstrichen voneinander getrennt sind. Diese Buchstaben beschreiben exakt und in kompakter Form die Eigenschaften des Schlüsselworts und des zugehörigen Parameters (falls vorhanden).

/A hinter einem Schlüsselwort bedeutet, daß der zugehörige Parameter notwendig ist. Er muß unbedingt angegeben werden muß, damit der Befehl ausgeführt werden kann. (Das heißt aber nicht, daß dem Parameter zwangsläufig auch das Schlüsselwort vorangehen muß.)

/K bedeutet, daß dieses Schlüsselwort dem zugehörigen Parameter vorangestellt werden muß. Parameter und Schlüsselwort müssen nicht zwangsläufig auch notwendig sein. /K besagt nur, daß wenn der Parameter angegeben wird, ihm auch das Schlüsselwort vorangehen muß.

/S bedeutet, daß es sich bei dem Schlüsselwort um einen Schalter handelt, dem kein Parameter folgt.

Zur besonderen Markierung notwendiger Parameter werden diese, wie auch der Befehlsname selbst, fett gedruckt.

Die folgende Beschreibung des COPY-Befehls soll Ihnen als Beispiel dafür dienen, wie eine solche Befehlsbeschreibung aussehen kann:

# COPY

**COPY** FROM <name1>* TO/A **<name2>**
BUF=/K<buffer> ALL/S QUIET/S CLONE/S DATES/S NOPRO/S COM/S

dient zum Kopieren einzelner Dateien und nahezu beliebiger Gruppen von Dateien.

...

Diese Beschreibung sagt aus, daß COPY drei Parameter <name1>, <name2> und <buffer> besitzt. Für <name1> kann auch ein Muster verwendet werden. Dies ist am Sternchen hinter <name1> zu erkennen. Der zweite Parameter <name2> ist notwendig, da hinter seinem Schlüsselwort »/A« steht. Die Schlüsselwörter FROM und TO müssen nicht vor den jeweiligen Parametern angegeben werden (sofern diese beiden Parameter in der hier gezeigten Reihenfolge angegeben werden). Vor dem Parameter <buffer> muß hingegen das Schlüsselwort »BUF=« auftauchen, (sofern dieser optionale Parameter überhaupt angegeben wird). Zusätzlich besitzt COPY noch sechs Schlüsselwörter, erkenntlich am »/S«, die als Schalter wirken. Hinter ihnen und vor dem nächsten Schlüsselwort kann also kein zusätzlicher Parameter folgen.

Dieses Format der Befehlsbeschreibungen soll zwei sehr unterschiedliche Benutzungsweisen des Kapitels erleichtern. Einmal das seitenweise Lesen und Experimentieren, mit dem Sie ja hoffentlich gleich beginnen werden. Sie sollten sich dabei nicht von einzelnen Stellen in den Erläuterungen verwirren lassen, die Sie nicht verstehen. Einige Passagen in den Befehlsbeschreibungen setzen Kenntnisse voraus, die Sie noch nicht haben können. Die meisten der hierfür benötigten Informationen finden Sie im nächsten Kapitel.

Die (vorläufig) unverständlichen Teile der Beschreibungen finden sich deshalb auf den folgenden Seiten, weil Befehlsbeschreibungen in diesem Kapitel auch als Nachschlagewerk für die tägliche Arbeit mit der Shell dienen sollen. Dabei werden Ihnen, wenn Sie langsam

tiefer in die Geheimnisse der Shell eindringen, die ausführlichen Beschreibungen der Shell-Befehle und die zugehörigen Beispiele hoffentlich helfen können. Oft dürfte aber auch die Beschreibung des Aufrufformats eines Kommandos genügen.

Eine vereinfachte Beschreibung dieses Aufrufformats, beziehungsweise der sogenannten »Befehlsschablone«, können Sie übrigens jederzeit bei dem Befehl selbst erfragen. Geben Sie dazu im Shell-Fenster den Namen des Befehls, eine Leerstelle und dahinter ein Fragezeichen ein und drücken Sie auf die [Return]-Taste. Der COPY-Befehl würde dann zum Beispiel die folgende Befehlsschablone ausgeben (die der Beschreibung in diesem Buch stark ähnelt):

```
1.SYS:> COPY ?
FROM, TO/A, ALL/S, BUFFER=BUFFER/K, CLONE/S, DATES/S, NOPRO
/S COM/S:
```

In vielen Fällen dürfte diese Befehlsschablone als Gedächtnisstütze ausreichen, wenn Sie einmal die Parameter und Schlüsselwörter eines Befehls vergessen haben sollten.

# ADDBUFFERS

**ADDBUFFERS** DRIVE/A **<Laufwerk>** BUFFERS/A **<Zahl>**

dient dazu, die Arbeit mit einer Diskette oder Festplatte zu beschleunigen.

Der Befehl ADDBUFFERS reserviert eine Anzahl von Pufferspeichern für die Arbeit mit dem Disketten- oder Festplattenlaufwerk <Laufwerk>. Dadurch werden die meisten Schreib- und Leseoperationen mit diesem Laufwerk beschleunigt. (Dazu kommt die sogenannte Cache-Technik zum Einsatz.) Allerdings geht auch RAM-Speicher verloren, der für andere Zwecke nicht mehr zur Verfügung steht.

<Zahl> gibt an, wieviele Pufferböcke zusätzlich zu den schon vorhandenen für das Laufwerk reserviert werden sollen. Jeder Block ist etwa ein halber Kbyte (500 Byte) groß. Im Allgemeinen genügt es, 20 bis 30 Pufferblöcke pro Laufwerk zu reservieren. Eine weitere Vergrößerung der Pufferzahl bringt nur noch sehr geringe Geschwindigkeitsgewinne.

Mittels ADDBUFFERS geänderte Pufferzuweisungen bleiben nur bis zum Ausschalten oder bis zum nächsten Neustart erhalten. Wenn Sie einem Laufwerk permanent Pufferspeicher zuweisen wollen, müssen Sie den entsprechenden ADDBUFFERS-Befehl von der Startup-Sequence aus aufrufen.

**Beispiele:**

```
ADDBUFFERS DF1: 30
```

Dieser Befehl reserviert 30 Pufferblöcke für die Arbeit mit dem Laufwerk DF1:. Die darauffolgenden Lese- und Schreiboperationen mit der Diskette im Laufwerk DF1: müßten deutlich schneller ablaufen.

# ALIAS

**ALIAS** <Name> <Zeichenfolge>

dient dazu, einen »Alias« beziehungsweise ein »Shell-Makro« zu definieren.

Der Befehl ALIAS ist kein echter Shell-Befehl. Es gibt kein Programm namens ALIAS im Directory *c* oder in einem anderen Directory der Workbench- oder Extras-Diskette. ALIAS ist eine Funktion, die sozusagen in der Shell fest eingebaut ist. Er kann deshalb auch nicht in einem CLI-Fenster verwendet werden, während nahezu alle anderen Shell-Befehle genauso in einem CLI-Fenster funktionieren!

Mit dem Befehl ALIAS kann man einer beliebigen <Zeichenfolge> – üblicherweise einem Shell-Befehl mit oder ohne Parameter – das »Pseudonym« <Name> zuordnen. Findet die Shell dann später am Anfang einer Befehlszeile das Pseudonym, wird dieses gegen die <Zeichenfolge> ersetzt und die so konstruierte Zeile als Shell-Befehl ausgeführt.

Indem man in <Zeichenfolge> den Platzhalter »[]« verwendet, kann man auch eine Art Parameter definieren. Normalerweise hängt die Shell den Rest einer Befehlszeile, an deren

Anfang ein Pseudonym gefunden wird, hinter <Zeichenfolge> und erhält so den Shell-Befehl, der dann schließlich ausgeführt wird. Enthält <Zeichenfolge> jedoch den Platzhalter »[ ]«, wird dieser gegen den restlichen Text, der hinter <Name> folgte, ersetzt.

Um eine ALIAS-Definition wieder zu löschen, muß ALIAS nur mit dem Parameter <Name> aufgerufen werden.

ALIAS ohne alle Parameter listet alle gültigen ALIAS-Definitionen auf.

Alle mit ALIAS definierten Pseudonyme gelten jeweils nur für das Shell-Fenster, in dem die entsprechenden ALIAS-Aufrufe erfolgt sind. Sobald dieses Fenster geschlossen wird (oder auch nach einem Neustart) werden sie »vergessen«. Wenn Sie wollen, daß eine Reihe von ALIAS-Definitionen in jedem neuen Shell-Fenster automatisch zur Verfügung stehen, sollten Sie diese in der Datei *Shell-Startup* im Directory *s* unterbringen. Diese Kommandofolge wird beim Öffnen jedes neuen Shell-Fensters durchgeführt.

**Beispiele:**

```
ALIAS LS  DIR
ALIAS EX  EXECUTE
ALIAS MD  MAKEDIR
ALIAS DEL DELETE
```

Diese vier Befehle legen eine Reihe praktischer Abkürzungen für gebräuchliche Shell-Befehle fest. Danach kann der Befehl EXECUTE unter anderem mit dem Kürzel EX und DELETE mit der Abkürzung DEL aufgerufen werden.

```
ALIAS DIRA DIR [] ALL
```

Nachdem diese ALIAS-Definition durchgeführt wurde, bewirken die folgenden Befehlspaare jeweils das Gleiche:

```
DIRA DF0:                    DIR DF0: ALL
DIRA DEVS:Printers           DIR DEVS:Printers ALL
DIRA DEVS:Printers FILES     DIR DEVS:Printers FILES ALL
```

Der folgende Befehl entfernt die Definition für DIRA aus der ALIAS-Liste:

```
ALIAS DIRA
```

Der Aufruf von DIRA führt danach zu einer Fehlermeldung.

# ASK

**ASK** PROMPT/A **<Text>**

dient dazu, dem Anwender von einer Kommandofolge aus eine Frage zu stellen.

Mit dem Befehl ASK können Sie – ähnlich wie mit ECHO – von einer Kommandofolge aus eine Meldung im Shell-Fenster ausgeben lassen. Diese Meldung muß als einziger Parameter hinter ASK folgen und sollte am besten in Anführungszeichen gesetzt werden. Nach de

Ausgabe der Meldung wartet ASK auf eine Benutzereingabe. Das heißt, der Amiga-Anwender muß einen Text eingeben und dann die [Return]-Taste drücken. Wenn dieser Text »Y« oder »Yes« (engl. *yes* = Ja) lautet, liefert ASK den Returncode 5. Gibt man »N«, »No« oder überhaupt nichts ein und drückt dann die [Return]-Taste ist der Returncode gleich 0. Anhand der Benutzereingabe beziehungsweise des sich daraus ergebenden Returncode kann die Kommandofolge dann eine Fallunterscheidung durchführen.

**Beispiele:**

```
ASK "Weitermachen?"
IF NOT WARN
   QUIT
ENDIF
...
```

Diese Befehle würden eine Kommandofolge beenden, sofern der Benutzer mit »N«oder »No« antwortet. Bei »Y« oder »Yes« fährt sie hinter dem Befehl ENDIF fort.

# ASSIGN

**ASSIGN** NAME <Name> DIR <Dirname> LIST/S EXISTS/S REMOVE/S

dient dazu, einem Directory ein logisches Gerät zuzuordnen.

Mit dem Befehl ASSIGN ist es möglich, die Zuordnung von logischen Geräten (logischen Laufwerken) zu einem Directory zu definieren, die schon festgelegten Zuordnungen zu betrachten und zu ändern.

Rufen Sie ASSIGN ohne Parameter auf, wird unter der Überschrift *Directories* eine Liste aller logischen Geräte und der Directories, denen sie entsprechen, angezeigt. Außerdem werden unter der Überschrift *Devices* (engl. für Geräte) auch noch die wirklichen »physikalischen« Geräte aufgelistet, die Sie verwenden können.

Der »Normalfall« ist aber ein Aufruf von ASSIGN mit zwei Parametern. Der erste ist der Name eines logischen Gerätes, der zweite der Name eines Directory, das mit diesem logischen Gerät verbunden werden soll. Der Name eines logischen Gerätes sollte immer mit einem Doppelpunkt enden! Handelt es sich um ein schon vorhandenes logisches Gerät, so wird die Zuordnung geändert. Ansonsten definiert ASSIGN ein neues logisches Gerät.

Das Schlüsselwort LIST entfernt das logische Laufwerk <Name> aus der Liste der logischen Laufwerke und gibt dann die komplette Liste aller logischen Laufwerke und echten Geräte aus. Wenn Sie die Ausgabe der Liste vermeiden wollen, müssen Sie ASSIGN nur mit dem ersten Parameter <Name> und ohne alle anderen Parameter oder Schlüsselworte aufrufen.

Das Schlüsselwort EXISTS dient zur Überprüfung, ob das logische Laufwerk <Name> vorhanden ist (definiert wurde). Dieses Schlüsselwort ist lediglich für Kommandofolgen interessant. (Siehe auch Kapitel 8).

Das Schlüsselwort REMOVE entfernt das Gerät <Name> aus der Liste der zur Verfügung stehenden Geräte. Diese Option des ASSIGN-Befehls ist nur für Programmentwickler

interessant. Sie sollten sie auf keinen Fall unbedacht ausprobieren! Wenn Sie zum Beispiel das Gerät DF0: entfernen, »kennt« der Amiga sein internes Laufwerk nicht mehr und stürzt wahrscheinlich kurz darauf mit einem Guru-Fehler ab.

Mittels ASSIGN geänderte Zuordnungen zwischen logischen Laufwerken und Directories bleiben nur bis zum Ausschalten oder bis zum nächsten Neustart erhalten. Wenn Sie eine solche Änderung permanent machen wollen, müssen Sie den entsprechenden ASSIGN-Befehl von der Startup-Sequence aus aufrufen.

**Beispiele:**

```
ASSIGN NAME akt: DIR df0:markus/texte/neu
```

Dies ist eine praktische Befehlsfolge, wenn man Dateien aus dem Directory *DF0:markus/texte/neu* häufig braucht, aber dieses Verzeichnis nicht immer zum aktuellen Directory machen kann. Man kann dieses Verzeichnis dann jederzeit mit dem Kurznamen AKT: ansprechen.

```
COPY C: RAM: ALL
ASSIGN C: RAM:
```

Diese beiden Befehle führen dazu, daß zunächst alle Shell-Befehle aus dem Directory C: auf die RAM-Disk kopiert werden und danach das logische Gerät C: auf die RAM-Disk verweist. Shell-Kommandos werden dann auf der RAM-Disk gesucht und arbeiten sehr schnell.

```
ASSIGN CLIPS: LIST
```

Dieser Befehl entfernt das logische Laufwerk CLIPS: und listet dann alle noch vorhandenen Geräte auf. Sie können den Namen CLIPS: danach nicht mehr in Shell-Befehlen verwenden.

```
ASSIGN PICTS: EXISTS
```

Dieser Befehl prüft, ob ein logisches Laufwerk namens PICTS: existiert. Wenn das nicht der Fall ist, liefert ASSIGN einen Returncode von 5, den Sie in Kommandofolgen für eine Verzweigung verwenden können.

# AVAIL

**AVAIL** CHIP/S FAST/S TOTAL/S

dient dazu, den noch freien RAM-Speicher des Amiga festzustellen und anzuzeigen.

Der Befehl liefert, wenn er ohne alle Parameter aufgerufen wird, eine tabellarische Übersicht der aktuellen Inanspruchnahme des Amiga-RAM-Speichers, getrennt nach Chip-RAM, Fast-RAM und Gesamtspeicher, aus. Wird einer der Schalter CHIP, FAST oder TOTAL angegeben, wird lediglich eine Zahl ausgegeben, die besagt, wieviel Speicher des jeweiligen Typs noch frei ist.

**Beispiele:**

```
AVAIL
Type    Available     In-Use     Maximum     Largest
chip       412640     110592      523232      412520
fast       387208     129680      516888      385856
total      799848     240272     1040120      412520
```

Die von AVAIL ausgegebene Tabelle enthält für die Speichertypen Chip-RAM und Fast-RAM sowie für den gesamten RAM-Speicher je eine Zeile. Die Spalte *Available* gibt an, wieviel Byte von der jeweiligen Speichersorte im Augenblick frei sind. Die Spalte *In-Use* gibt an, wieviel Byte der Speichersorte belegt sind. Die Spalte *Maximum* beschreibt die maximal verfügbare Größe des Speichertyps und die Spalte *Largest* gibt die Größe des größten, im Augenblick noch freien, zusammenhängenden Speicherblocks an.

```
AVAIL CHIP
412640
```

Dieser Befehl liefert nur den Umfang des augenblicklich nicht belegten Chip-RAM in Form einer einzigen Zahl.

# BINDDRIVERS

**BINDDRIVERS**

dient zur Aktivierung zusätzlicher Gerätetreiber, die sich im Directory *Expansion* befinden.

Der Befehl BINDDRIVERS besitzt keine Parameter. Er durchsucht bei Aufruf das Directory *Expansion* auf der Workbench-Diskette nach zusätzlichen »Gerätetreibern«, lädt eventuell vorhandende Treiber-Dateien und aktiviert die entsprechenden Treiber. Danach sollte in der Geräte-Liste, die ASSIGN anzeigt, ein neues Gerät auftauchen. Zusätzliche Treiber werden üblicherweise auf Diskette mitgeliefert, wenn Sie ein Zusatzgerät (zum Beispiel eine Festplatte) kaufen, das nicht sofort von der Standard-Systemsoftware erkannt wird. Sie müssen die Treiberdatei dann von der Diskette in die Schublade *Expansion* auf der Workbench-Diskette, mit der Sie den Amiga starten, legen. Beim nächsten Neustart wird der Treiber dann aktiv.

BINDDRIVERS wird typischerweise in der Startup-Sequence verwendet, einer Folge von Shell-Kommandos, die automatisch bei jedem Neustart durchlaufen werden.

# BREAK

**BREAK** TASK/A **<Nummer>** ALL/S C/S D/S E/S F/S

dient dazu, ein Programm, daß in einem anderen Shell-Fenster läuft, abzubrechen.

Mit dem Befehl BREAK können Sie ein Programm, das in einem anderen Shell-Fenster oder als Hintergrund-Task läuft, abbrechen. Hierzu müssen Sie mit dem Befehl STATUS feststellen, welche »Prozeß-Nummer« dieses Programm hat (siehe unten). Mit dieser Nummer können Sie das Programm dann abbrechen, als hätten Sie mit der Maus in das andere Shell-Fenster geklickt und dann [Ctrl]+[C] oder eine andere Tastenkombination zum Abbruch gedrückt.

Wenn Sie keines der vier Schalt-Schlüsselwörter C, D, E und F verwenden, wirkt das wie die Betätigung von [Ctrl]+[C]. Verwenden Sie das Schlüsselwort ALL, werden alle Abbruch-Bedingungen (C, D, E und F) gesetzt.

**Beispiele:**

```
BREAK 2 D
```

Dieser Befehl wirkt, als hätten Sie in dem Fenster, in dem Task 2 läuft (zum Beispiel dem Shell-Fenster mit der Nummer 2) [Ctrl]+[D] gedrückt. Wenn in diesem Fenster gerade eine Kommandofolge läuft, würde diese dadurch abgebrochen, bevor der nächste Befehl dieser Kommandofolge zur Ausführung kommt.

# CD

**CD** DIR <Directory>

dient dazu, das aktuelle Directory zu ändern.

Die Shell (beziehungsweise Amiga-DOS) kennt das Konzept eines aktuellen Directory. Tippt man einen »einfachen« Dateinamen, der nicht mit einer Gerätebezeichnung (zum Beispiel DF0:, RAM: oder C:) beginnt, so hängt die Shell den Namen des aktuellen Directorys davor, um den vollständigen Pfadnamen der Datei zu erhalten. Hat man beispielsweise eine Datei mit dem kompletten Namen *DF1:alpha/text/brief1* und das aktuelle Directory heißt *DF1:alpha/text*, so braucht man im Shell-Fenster nur *brief1* zu tippen, um die Datei genau zu bezeichnen. Der Befehl CD (für ***change current directory*** = ändere das aktuelle Directory) ändert den Namen, den das CLI als aktuelles Directory auffaßt.

Geben Sie CD ohne jeden Parameter ein, so wird der Name des aktuellen Directory ausgegeben.

Geben Sie CD und dahinter den Namen eines Directories ein, so wird dieses das aktuelle Directory. Dabei muß man beachten, daß das CLI auch vor diesen Namen natürlich erst die Bezeichnung des jetzt noch aktuellen Directory setzt. Will man übergeordnete oder gar auf

anderen Disketten liegende Directories »aktuell machen«, muß man den Namen des neuen Directory mit einer Laufwerksbezeichnung beginnen oder andere »Tricks« anwenden, die ausführlich im vorangehenden Kapitel erklärt worden sind.

Beachten Sie bitte, daß CD auch das aktuelle Laufwerk ändern kann beziehungsweise automatisch mitändert. Das aktuelle Laufwerk (das mit : abgekürzt werden kann) ist immer das Laufwerk, auf dem sich das aktuelle Directory befindet.

**Beispiele:**

```
CD
CD Briefe
CD df1:Texte/Briefe
```

Der erste Befehl gibt das aktuelle Directory aus, der zweite macht das Directory *Briefe* im aktuellen Directory zum neuen aktuellen Directory und der dritte Befehl macht das Directory *Briefe* im Directory *Texte* auf der Diskette im externen Laufwerk zum neuen aktuellen Directory und damit eventuell auch das externe Laufwerk zum neuen aktuellen Laufwerk.

# CHANGETASKPRI

**CHANGETASKPRI** PRI/A **<N1>** PROZESS/K <N2>

dient zum Ändern der Priorität eines Task beziehungsweise Prozesses.

Einer der großen Vorteile des Amiga gegenüber anderen preiswerten Personalcomputern ist ja seine Multitaskingfähigkeit. Mehrere Programme (in diesem Zusammenhang meist »Prozesse« oder »Tasks« genannt) können scheinbar gleichzeitig ablaufen und völlig unterschiedliche Dinge erledigen. Jeder dieser Prozesse hat dabei eine Priorität, die bestimmt, welchen »Anteil« der Rechenleistung des Amiga er erhält. Diese Prioritäten – und damit der Parameter <N1> – können zwischen –127 und +128 liegen.

Wenn Sie in einem Shell-Fenster oder in einer Kommandofolge CHANGETASKPRI aufrufen, ändern Sie damit die Priorität des Shell-Prozesses, von dem aus der Befehl aufgerufen wurde, und aller anderen Prozesse, die danach von diesem Shell-Fenster aus gestartet werden. Es kann zum Beispiel sinnvoll sein, die Priorität vor dem Start eines langwierigen Hintergrundtasks (der mit RUN gestartet wird) herab- und danach wieder heraufzusetzen. Der Hintergrundtask läuft dann zwar recht langsam, stört Sie aber nicht beim Weiterarbeiten im Shell-Fenster.

Verwenden Sie für <N1> bitte keine Prioritäten außerhalb des Bereichs von –5 bis +5, außer Sie wissen genau, was Sie tun. Prioritäten außerhalb dieses Bereichs kommen normalerweise nämlich der Arbeit des Betriebssystems Amiga-DOS selbst »ins Gehege«.

CHANGETASKPRI kann auch zur Änderung der Priorität anderer Shell-Prozesse verwendet werden. Dazu muß mit Hilfe des STATUS-Befehls die Nummer des anderen Prozesses festgestellt und diese dann als zweiter Parameter im CHANGETASKPRI-Befehl angegeben werden.

**Beispiele:**

```
CHANGETASKPRI 5
```

Dieser Befehl setzt die Priorität des aktuellen Shell-Prozesses (von dem aus der Befehl aufgerufen wurde) auf 5 hoch. Das ist die höchste »erlaubte« Priorität für Shell-Prozesse. Die normale Priorität eines Shell-Prozesses ist 0.

```
CHANGETASKPRI 5 PROZESS 3
```

Dieser Befehl setzt die Priorität des Shell-Prozesses mit der Prozeßnummer 3 auf 5 hoch.

# CLOCK

**CLOCK** ANALOG/S DIGITAL1/S DIGITAL2/S =<Fenster>/K
12HOUR/S 24HOUR/S SECOND/S DATE/S

Der Befehl CLOCK ruft das Programm *Clock* auf, daß sich in der *Utilities*-Schublade befindet. Dieses Programm wird üblicherweise von der Workbench aus gestartet. Sie können CLOCK aber auch in einem Shell-Fenster aufrufen und alle Angaben, die Sie auf der Workbench mit Menübefehlen machen würden, in Form zusätzlicher Schlüsselworte angeben.

Mit den Schlüsselwörtern ANALOG, DIGITAL1 und DIGITAL2 können Sie die gewünschte Form der Uhr festlegen. Diese Schlüsselwörter entsprechen den gleichnamigen Menübefehlen des Programms *Clock*. Dasselbe gilt für die Schlüsselwörter 12HOURS, 24HOURS, SECONDS und DATE, die bestimmen, ob 12- oder 24-Stundenanzeige verwendet wird (nur bei Digitalanzeige sinnvoll), und ob Sekunden und das aktuelle Datum ebenfalls angezeigt werden.

Besonders interessant ist der Parameter =<Fenster>. Er erlaubt es Ihnen, gleich beim Start die Position des Uhr-Fensters und seine Größe (nur bei Analoganzeige) zu bestimmen. Dieser Parameter hat die Form =*<Links>*,*<Oben>*,*<Breite>*,*<Höhe>* und muß von einem der Schlüsselwörter ANALOG, DIGITAL1 oder DIGITAL2 eingeleitet werden. Das Gleichheitszeichen muß unmittelbar (ohne Leerstelle) hinter dem vorangehenden Schlüsselwort stehen.

**Beispiele:**

```
CLOCK
CLOCK DIGITAL2 24HOURS DATE SECONDS
RUN CLOCK ANALOG=10,10,200,100
```

Der erste Befehl startet das Programm Clock, als hätten Sie einen Doppelklick auf das entsprechenden Workbench-Piktogramm gemacht. Es werden keine Angaben zur Form, zum Aussehen oder zum Anzeigeformat gemacht. (Diese können mit Menübefehlen nachgeholt werden.)

Der zweite Befehl bestimmt gleich beim Start das Aussehen (einzeilige Digitalanzeige) und legt 24-Stundenanzeige, sowie Datums- und Sekundenanzeige fest.

Der dritte Befehl öffnet ein Uhrenfenster in Analoganzeige mit einer ganz bestimmten Position und Größe. Da vor dem CLOCK-Befehl RUN steht, wird das Programm als eigenständiger Prozeß gestartet. Es können sofort danach wieder Shell-Befehle eingegeben werden.

# CLOCKPTR

**CLOCKPTR** 12MMDD/S 12DDMM/S 24MMDD/S 24DDMM/S

Der Befehl CLOCKPTR ruft das Programm *ClockPtr* auf, das sich in der *Utilities*-Schublade befindet. Dieses Programm wird üblicherweise von der Workbench aus gestartet. Sie können es aber auch in einem Shell-Fenster aufrufen, zum Beispiel von der Startup-Sequenz aus. CLOCKPTR funktioniert den Mauszeiger in eine Digitaluhr um, wie es in Kapitel 5 beschrieben wurde.

Mit den Schaltern 12MMDD, 12DDMM, 24MMDD und 24DDMM, die sich gegenseitig ausschließen, können Sie bestimmen, ob die Digitaluhr 12- oder 24-Stundenanzeige verwendet und ob bei der Datumsanzeige erst der Monat kommt (MMDD) oder erst der Tag (DDMM).

Es empfiehlt sich, CLOCKPTR immer über den Befehl RUN als eigenständigen Prozeß zu starten. Ansonsten können in dem Shell-Fenster, in dem der Befehl CLOCKPTR eingegeben wurde, solange keine weiteren Befehle ausgeführt werden, bis das Programm CLOCKPTR beendet ist.

**Beispiele:**

```
CLOCKPTR
RUN CLOCKPTR
```

Der erste Befehl startet CLOCKPTR wie einen normalen Shell-Befehl. Das Shell-Fenster, in dem dieser Befehl eingegeben wurde, kann danach nicht mehr für die Befehlseingabe verwendet werden. Um CLOCKPTR wieder abzubrechen, müssen Sie die Tastenkombination [Ctrl]+[C] drücken. Danach kann das Shell-Fenster wieder verwendet werden.

Der zweite Befehl startet CLOCKPTR als eigenständigen Prozeß. Sie können sofort danach weitere Befehle eingeben. Um CLOCKPTR zu beenden, müssen Sie mit dem Befehl STATUS seine Prozeßnummer feststellen und ihn dann mit dem Befehl BREAK abbrechen:

```
1.SYS:>STATUS
Process 1: Loaded as command: status
Process 2: Loaded as command: ClockPtr
Process 3: Loaded as command: CC
1.SYS:>BREAK 2
```

# CMD

**CMD <Port> <Dateiname>** OPT/K <Option>

Mit dem Befehl CMD (den Sie auch von der Workbench aus aufrufen können), leiten Sie den Datenstrom, der normalerweise an die serielle oder parallele Schnittstelle gehen würde zu einem anderen Gerät oder in eine Datei um. Das wird meist wohl dazu verwendet werden, sogenannte »Druckdateien« zu erzeugen. Eine Druckdatei enthält die Daten, die ein Programm zu einem Drucker schickt, um diesen dazu zu bringen, eine oder mehrere Seiten auszugeben. Eine solche Datei könnte man dann auf einer Diskette abspeichern, damit zu einem anderen Amiga gehen und dort auf einem anderen Drucker ausgeben lassen. So ist es zum Beispiel möglich, auf einem Drucker zu drucken, den man gar nicht an seinen eigenen Amiga angeschlossen hat.

Der Parameter <Port> kann entweder »serial« oder »parallel« heißen. Wenn Sie eine Druckdatei erzeugen wollen und Sie in *Preferences* angegeben haben, daß Ihr Drucker an der seriellen Schnittstelle angeschlossen ist, müssen Sie »serial« angeben, wenn Ihr Drucker an der parallelen Schnittstelle angeschlossen ist »parallel«. Der Parameter <Dateiname> kann ein beliebiger Dateiname (Pfadname) oder auch ein Gerätename sein. Die <Option> hinter dem Schlüsselwort OPT (immer ein einzelner Buchstabe) bestimmt einige Aspekte des Verhaltens von CMD. Es können mehrere Optionen hintereinander angegeben werden.

Die Option I legt fest, daß Druckprozesse, die mit drei Null-Bytes beginnen, nicht umgelenkt werden. Die Option S bewirkt, daß das erste Paket von Druckbefehlen nicht abgespeichert wird (siehe unten). Die Option M erlaubt es, mehrere Druckvorgänge abzuspeichern. Normalerweise wird CMD nach dem Abschluß des ersten Druckvorgangs automatisch beendet. Wenn Sie die Option M angeben, müssen Sie CMD gezielt mit einem BREAK-Befehl abbrechen. Die Option N gibt während der Umleitung kurze Meldungen im Shell-Fenster aus, die Sie über den Fortgang der Dinge informieren. Diese Meldungen erscheinen zum Beispiel beim Start und Ende eines Druckprozesses und vor der Beendigung von CMD selbst.

Besondere Bedeutung kommt der Option S zu. Um sie zu verstehen, muß man wissen, daß viele Programme zu Beginn des Druckprozesses einen sogenannten »Reset-Befehl« zum Drucker schicken, um diesen in eine Art Urzustand zu versetzen. Nach einem solchen Reset-Befehl muß man allerdings einige Sekunden warten, bis der Drucker wieder empfangsbereit ist. Alle Daten, die davor zum Drucker gesandt werden, sind verloren. Ein Reset-Befehl darf nicht in die Druckdatei geschrieben werden, da die Wartezeit später, wenn die Druckdatei an den Drucker gesandt wird, nicht eingehalten werden kann. Wenn Sie zum Beispiel mit dem Programm *GraphicDump* (in der *Utilities*-Schublade) einen Screendump erzeugen lassen, und vorher mit CMD die Druckausgaben umgeleitet haben, sollten Sie deshalb immer die Option S verwenden. *GraphicDump* ist eines dieser Programme, die den Druckprozeß mit einem Reset-Befehl einleiten.

Darüber hinaus ist es empfehlenswert, CMD von der Shell aus über den Befehl RUN zu starten. Ansonsten können in dem Shell-Fenster, in dem der Befehl CMD eingegeben wurde, so lange keine weiteren Befehle ausgeführt werden, bis das Programm CMD beendet ist. Das gilt speziell dann, wenn Sie CMD mit der Option M starten. (Ansonsten wird CMD automatisch nach dem Ende des nächsten Druckprozesses beendet.)

**Beispiele:**

```
CMD parallel RAM:Druckdatei OPT S OPT N
RUN CMD parallel RAM:Druckdatei OPT S OPT N OPT M
```

Der erste Befehl startet CMD und lenkt den Datenstrom zur parallelen Schnittstelle in die Datei *Druckdatei* auf der RAM-Disk um. (Dabei wird angenommen, daß *Preferences* mitgeteilt wurde, daß ein Drucker an der parallelen Schnittstelle angeschlossen ist.) Dieser CMD-Befehl wird automatisch beendet, sobald der nächste Druckvorgang abgeschlossen ist. Erst dann können weitere Shell-Befehle ausgeführt werden.

Der zweite Befehl startet CMD als eigenständigen Prozeß, der alle folgenden Druckvorgänge in die Datei *Druckdatei* auf der RAM-Disk umleitet. Diese Umleitung ist erst dann beendet, wenn CMD gezielt mit einem BREAK-Befehl abgebrochen wird. (Siehe das entsprechende Beispiel bei der Beschreibung des Befehls CLOCKPTR.)

# COPY

**COPY** FROM <Quelle>* TO/A **<Ziel>**
ALL/S QUIET/S BUF=/K <Buffer> CLONE/S DATES/S NOPRO/S COM/S

dient zum Kopieren einzelner Dateien und nahezu beliebiger Gruppen von Dateien.

Die meistverwendete Möglichkeit des COPY-Befehls ist sicherlich, einzelne Dateien oder die Inhalte kompletter Directories zu kopieren. Hierzu muß <Quelle> der Name einer vorhandenen Datei sein und <Ziel> der Pfadname der Datei, die die Kopie aufnehmen soll. Ist <Quelle> ein Dateiname und <Ziel> der Name eines schon existierenden Directory, wird die neue Datei (die Kopie) in dieses Directory gelegt und erhält denselben Namen wie die Datei <Quelle>.

Existiert bereits eine Datei namens <Ziel> oder gibt es im Directory <Ziel> bereits eine Datei mit dem Namen <Quelle> geht deren alter Inhalt verloren!

<Ziel> kann auch der Name eines Gerätes sein. Kopiert man beispielsweise eine Datei zum Gerät PRT: (dem Drucker), so bewirkt dies, daß die entsprechende Datei ausgedruckt wird.

Sind sowohl <Quelle> als auch <Ziel> Namen von Directories, so werden alle Dateien, die sich im Directory <Quelle befinden, in das Directory <Ziel> kopiert. Wenn Sie zusätzlich noch das Schlüsselwort ALL angeben, werden auch alle Unterdirectories in <Quelle> und die darin enthaltenen Dateien nach <Ziel> kopiert. Dazu werden gegebenenfalls im Directory <Ziel> automatisch neue Directories erzeugt. Ohne ALL werden nur Dateien kopiert, die sich in <Quelle> befinden, weitere Directories in diesem bleiben unberücksichtigt.

Wird nur das Directory <Ziel> (mit dem Schlüsselwort) TO davor angegeben, so geht der COPY-Befehl vor, als wäre für <Quelle> das aktuelle Directory angegeben worden.

Für <Quelle> kann auch ein Namensmuster angegeben werden. COPY kopiert dann alle Dateien nach <Ziel> (das in diesem Fall ein Directory sein sollte), deren Namen in das Muster passen.

Wenn der COPY-Befehl mehrere Dateien nacheinander kopiert (zum Beispiel beim Kopieren ganzer Directories), protokolliert er diesen Vorgang im Shell-Fenster. Dabei wird immer der Name der gerade bearbeiteten Datei im Shell-Fenster ausgegeben. Man hat so einen stetigen Überblick darüber, wie weit der Kopiervorgang schon ist. Selbst wenn der Vorgang wegen eines Fehlers (wenn zum Beispiel, weil die Zieldiskette voll ist) abbricht, weiß man genau, bei welcher Datei dies geschehen ist. Möchte man dieses Protokoll nicht sehen, so kann man das Schlüsselwort QUIET (engl. für still oder ruhig) angeben. Dann unterbleiben alle Ausgaben während des Kopiervorgangs.

Die neu erzeugten kopierten Dateien erhalten normalerweise die selben Schutzflags (siehe die Beschreibung des Befehls PROTECT), wie die Originale. Will man das vermeiden, muß man das Schlüsselwort NOPRO verwenden. Dann haben die Kopien keinerlei Schutzflags.

Das Schlüsselwort DATES sorgt dafür, daß die neu erzeugten Dateien dasselbe Modifikationsdatum erhalten, wie die Originale. Wird DATES nicht mit angegeben, erhalten Sie als Modifikationsdatum den Zeitpunkt, in dem ihre Erzeugung durch COPY-Befehl beendet wurde.

Das Schlüsselwort COM sorgt dafür, daß die neu erzeugten Dateien denselben Datei-Kommentar erhalten, wie die Originale (siehe die Beschreibung des Befehls FILENOTE). Wird COM nicht mit angegeben, ist dieser Kommentar bei den neuen Dateien leer.

Das Schlüsselwort CLONE sorgt dafür, daß die neu erzeugten Dateien in jeder Hinsicht exakte Kopien der Originale sind. Die Kopien erhalten dieselben Schutzflags, denselben Kommentar und dasselbe Modifikationsdatum. CLONE ersetzt also DATES, COM und NOPRO.

Der Parameter <Buffer>, vor dem das Schlüsselwort BUF= stehen muß, gibt an, wie groß der Pufferspeicher ist, den COPY beim Kopieren verwendet. Die Zahl <Buffer> ist als Anzahl der einzelnen Pufferblöcke zu verstehen, von denen jeder ein halber Kbyte (gleich 512 Byte) groß ist. Dazu muß man wissen, daß COPY zunächst die zu kopierenden Daten in den RAM-Speicher liest und dann auf die Ziel-Diskette schreibt oder zu einem anderen Gerät schickt. Wenn der Pufferspeicher nicht groß genug ist, läuft der Kopiervorgang in mehreren »Häppchen« ab. Sie können den Kopiervorgang beschleunigen (oder sich Diskettenwechsel ersparen), indem Sie den Puffer größer machen. Normalerweise ist dieser Puffer 100 Kbyte groß. Dies entspricht einer Einstellung von BUF=200.

Wenn Sie allerdings Daten auf die RAM-Disk RAM: oder RAD: kopieren, empfiehlt es sich, weniger Pufferspeicher zu verwenden. Schließlich werden die Daten beim Schreibvorgang nur von einer Stelle im RAM-Speicher zu einer anderen kopiert. Werte von 10 oder 20 für <Buffer> sind deshalb beim Kopieren zur RAM-Disk sinnvoller.

**Beispiele:**

```
COPY Brief TO KopieBrief
COPY Brief "Kopie von Brief"
```

Beide Befehle kopieren eine Datei namens *Brief* (im aktuellen Directory). Die Kopier erhalten die Namen *KopieBrief* und Kopie *von Brief* und liegen ebenfalls im aktueller Directory.

```
COPY DF0:c TO RAM:c
COPY DF0:devs TO RAM:devs ALL BUF=20 CLONE
```

Der erste Befehl kopiert alle Dateien im Directory *c* auf der Diskette im internen Laufwerk in das Directory *c* auf der RAM-Disk RAM:. Gäbe es im Directory *c* Unterdirectories, würden diese aber nicht mitkopiert. Der zweite Befehl kopiert den kompletten Inhalt des Directory *devs* auf der Diskette im internen Laufwerk in das Directory *devs* auf der RAM-Disk RAM:. Dabei werden auch alle Dateien in eventuell vorhandenen Unterdirectories berücksichtigt und gegebenenfalls automatisch neue Directories im Directory *devs* auf der RAM-Disk angelegt. Die Anzahl der Pufferspeicher für diesen Kopiervorgang wird auf 20 verringert und mit dem Schalter CLONE dafür gesorgt, daß die kopierten Dateien in jeder Hinsicht mit den Originalen indentisch sind.

```
COPY TO RAM:
```

Dieser Befehl kopiert alle Dateien im aktuellen Directory auf die RAM-Disk RAM:. Unterdirectories im aktuellen Directory und deren Inhalt werden dabei nicht berücksichtigt.

```
COPY DF0:Utilities/Clock#? TO RAM:
```

Dieser Befehl kopiert alle Dateien im Directory *Utilities* auf der Diskette DF0:, deren Name mit *Clock* beginnt auf die RAM-Disk RAM:. Das sind unter anderem die Dateien *Clock* und *ClockPtr*. Entscheidend ist aber, daß auch die Dateien *Clock.info* und *ClockPtr.info* mitkopiert werden. Deshalb tauchen die Programme *Clock* und *ClockPtr* auch als Piktogramme auf der Workbench auf, wenn man dort das Diskettenfenster der RAM-Disk öffnet. (Ohne die Dateien *Clock.info* und *ClockPtr.info* würden die beiden Programme auf der Workbench unsichtbar bleiben.)

# DATE

**DATE** TIME <Zeit> DATE <Datum> TO/K <datei>

dient zum Anzeigen oder Ändern der vom System gespeicherten Zeit und des Datums.

Der Befehl DATE wird entweder dafür verwendet, die Systemzeit nachzustellen oder anzuzeigen. In der Grundausbaustufe hat der Amiga 500 keine eingebaute, batteriegespeiste Uhr – diese ist erst in der Speichererweiterung Amiga 501 enthalten. Ein Nachstellen ist deshalb eigentlich nach jedem Neustart wieder nötig. Beim Start liest der Amiga zunächst einmal das Datum und die Uhrzeit der letzten Modifikation an der Startdiskette und verwendet dieses als aktuelle Zeit. Wer Wert darauf legt, daß die Datumsangaben, die ihm zum Beispiel bei Anwendung des LIST-Befehls (siehe unten) gezeigt werden, wirklich stimmen, der sollte deshalb nach jedem Start die Uhr nachstellen. Die Verwendung des Shell-Befehls DATE für diesen Zweck ist meist wesentlich schneller als die Einstellung der Zeit mittels *Preferences*, wie es innerhalb der Workbench üblicherweise gemacht wird. Wer also mit der Shell vertraut ist, sollte nicht den Umweg über *Preferences* wählen.

Mittels DATE können sowohl Datum als auch Uhrzeit getrennt oder in einem Zug eingestellt werden. Hierzu dienen die optionalen Parameter <Datum> und <Deit>, die von den beiden Schlüsselwörtern DATE und TIME eingeleitet werden können. Das Datum sollte dazu im Format DD-MMM-JJ (zum Beispiel 07-Jul-87) und die Uhrzeit im Format HH:MM:SS

(zum Beispiel 11:44:00; die Sekundenangabe kann auch fortfallen) angegeben werden. Die Reihenfolge, in der Uhrzeit und Datum eingegeben werden, spielt keine Rolle – auch dann nicht, wenn die Schlüsselwörter DATE und TIME nicht verwendet werden. DATE erkennt die Uhrzeit am Doppelpunkt, das Datum am Bindestrich.

Besonders für die Besitzer eines Amiga 500 ohne Speichererweiterung und Uhr sind einige sehr spezielle »Datumsangaben« interessant, die DATE außer den üblichen Daten auch noch versteht: die (englischen) Namen der Wochentage und *tomorrow* (engl. für morgen) sowie *yesterday* (engl. für gestern). Sie können diese Wörter statt eines Datums wie *01-jan-87* verwenden. Die Angaben *tomorrow* und *yesterday* setzen das Datum jeweils einen Tag vor oder zurück und die Angabe eines Wochentagnamens setzt das Datum auf den jeweils nächsten Wochentag mit diesem Namen.

Wenn Sie DATE ohne Parameter eingeben, werden aktuelle Uhrzeit und aktuelles Datum (oder jedenfalls das, was Ihr Amiga 500 dafür hält) im Shell-Fenster ausgegeben. Wollen Sie diese Ausgabe auf eine Datei oder ein Gerät umleiten, so können Sie dessen Namen hinter dem Schlüsselwort TO angeben.

**Beispiele:**

```
DATE

DATE TO datum
DATE TO PRT:
DATE TIME 11:40
DATE 07-jul-89 11:40:30
```

Der erste Befehl gibt Datum und Uhrzeit auf dem Bildschirm aus. Der zweite und dritte schickt diese Informationen in die Datei *Datum* im aktuellen Directory beziehungsweise auf den Drucker. Der vierte Befehl stellt die Zeit neu ein (auf 11 Uhr 40). Und der fünfte Befehl stellt Datum und Uhrzeit (auf den siebten Juli 1989, 11 Uhr 40 und 30 Sekunden).

Die folgenden Beispiele illustrieren noch einmal die Verwendung von Wochentagsnamen.

```
DATE tomorrow
DATE yesterday
DATE wednesday
```

Der erste Befehl setzt das Datum einen Tag weiter, der zweite einen Tag zurück. Und der dritte Befehl schließlich setzt das Datum auf den nächsten Tag, der ein Mittwoch (engl. *wednesday*) ist.

# DELETE

**DELETE <name1>*** <name2> ... <name10> ALL/S QUIET/S Q/S

dient zum Löschen einzelner Dateien und nahezu beliebiger Gruppen von Dateien.

Das Kommando DELETE dient zwar zum Löschen von Dateien, ist ansonsten dem COPY Befehl vom Format her aber recht ähnlich. Wenn Sie einzelne Dateien löschen wollen, s

können Sie deren Namen (bis zu zehn Stück) direkt hinter DELETE als Parameter <name1> bis <name10> angeben. Alle oder einige dieser Namen können auch Directories sein. Directories müssen allerdings leer sein, um auf diese Weise gelöscht zu werden.

Wenn Sie hingegen nur einen Directorynamen als Parameter angeben und dahinter noch das Schlüsselwort ALL schreiben, werden auch alle Dateien und Directories innerhalb von <name1> und schließlich auch das Directory <name1> selbst gelöscht. Dabei listet DELETE immer den Namen der Datei auf, die gerade gelöscht wird. Wenn Sie diese Auflistung unterdrücken wollen, müssen Sie hinter ALL noch das Schlüsselwort QUIET oder Q schreiben.

Statt für <name1> den Namen eines Directory anzugeben, können Sie an dieser Stelle auch ein Muster angeben. Alle Dateien, deren Namen in dieses Muster passen, werden dann gelöscht. Auch hier können Sie mit dem Schlüsselwort QUIET oder Q die Auflistung der gelöschten Dateien während des Löschvorgangs verhindern. Wie solche Muster genau aussehen können, wurde im Kapitel 6 ausführlich beschrieben.

**Beispiele:**

```
DELETE Brief1 Brief2 Brief3
DELETE :Texte/alteBriefe ALL
DELETE #?.KOPIE
```

Der erste dieser Befehle löscht die drei Dateien *Brief1*, *Brief2* und *Brief3*. Der zweite löscht das Directory *:Texte/alteBriefe* (samt Inhalt) und der letzte DELETE-Befehl schließlich löscht alle Dateien im aktuellen Directory, deren Namen mit ».KOPIE« enden.

# DIR

**DIR** DIR <Name>* OPT/K <Option> ALL/S FILES/S DIRS/S INTER/S

dient zum Anzeigen des Inhalts von Disketten und Directories.

Der Befehl DIR ist wahrscheinlich der am häufigsten benötigte Shell-Befehl. Er ist die wichtigste Möglichkeit, sich einen Überblick über den Inhalt von Directories zu verschaffen. Detailliertere Informationen über einzelne Dateien lassen sich mit dem Befehl LIST gewinnen.

Gibt man den Befehl DIR ohne einen Parameter dahinter ein, so wird der Inhalt des aktuellen Directories aufgelistet. Folgt auf DIR der Name eines Directory, so wird dessen Inhalt aufgelistet. Dabei werden zunächst die in diesem Directory enthaltenen Unter-Directories (ohne Inhalt) und dann die Dateien in alphabetisch sortierter Form gezeigt. Die Liste der Unter-Directories ist einspaltig und hinter jedem Namen steht noch einmal in Klammern der Hinweis »dir«. Die Auflistung der Dateinamen erfolgt zweispaltig.

Ist <Name> ein Muster, so werden alle Dateinamen aufgelistet, die in das Muster passen. Auf diese Weise kann man in einem Directory mit sehr vielen Dateien schnell nach bestimmten Dateien suchen.

Die Schlüsselwörter erlauben eine Modifikation der Arbeitsweise des DIR-Befehls. Das wichtigste Schlüsselwort ist ALL. Es bewirkt, daß DIR nicht nur die Namen von Unter-Directories anzeigt, sondern auch deren Inhalt anzeigt. Man kann sich damit beispielsweise eine komplette Liste aller Dateien auf einer Diskette ausgeben lassen. Die von DIR ausgegebene Liste versucht, die Verschachtelung des Dateienbaums durch Einrückung zu simulieren. Jedesmal, wenn DIR eine Stufe tiefer in den Dateienbaum geht, werden die Namen etwas nach rechts eingerückt.

Das Schlüsselwort FILES bewirkt, daß nur Dateinamen, nicht aber die Namen von Unter-Directories angezeigt werden. Umgekehrt bewirkt DIRS, daß nur Directorynamen und keine Dateien aufgelistet werden.

Das Schlüsselwort INTER (für interaktiv) aktiviert den interaktiven Modus von DIR. Dabei wird Ihnen zunächst jeder Name eines Directory oder einer Datei einzeln gezeigt und dann ein Fragezeichen ausgegeben. Sie können nun durch die Eingabe eines Buchstabens bestimmen, wie weiter vorzugehen ist:

Drücken Sie die [Return]-Taste, so wird Ihnen der nächste Name gezeigt. Tippen Sie [Q] und dann [Return], so wird der DIR-Befehl abgebrochen. Dasselbe können Sie mit [Ctrl]+[C] erreichen. Wird der Name eines Directory angezeigt und Sie tippen [E] (für *enter* = einsteigen) und dann [Return], wird der Inhalt dieses Directory bearbeitet. Der nächste Name, der erscheint, ist der erste Name aus diesem untergeordneten Directory. Befinden Sie sich in einem untergeordneten Directory und drücken Sie die Taste [B] (für *back* = zurück) und dann [Return], so wird im darüber liegenden Directory an der Stelle weitergemacht, wo Sie zuvor den E-Befehl gegeben haben. Mit E und B kann man sich auf diese Weise durch den Baum der Directories »hangeln«.

Wird ein Dateiname angezeigt und Sie tippen hinter dem Fragezeichen »DEL« (nicht die [Del]-Taste, sondern die Buchstaben D, E und L), so wird die Datei gelöscht. Auch Directories können auf diese Weise gelöscht werden, aber nur, wenn sie zuvor vollständig geleert wurden (durch Löschen aller Dateien und Directories in ihnen).

Wird der Name einer Datei angezeigt und man tippt [T] und dann [Return], so wird der Inhalt dieser Datei im Shell-Fenster ausgegeben. (Dies ist natürlich nur dann sinnvoll, wenn es sich beim Inhalt dieser Datei um einen Text handelt. Ansonsten können unvorhersehbare Schwierigkeiten auftauchen.) Haben Sie genug vom Inhalt der Textdatei gesehen und möchten die Ausgabe deshalb abbrechen, brauchen Sie nur mit [Ctrl]+[C] abzubrechen.

Sie können auch jederzeit einen anderen Shell-Befehl ausführen, ohne DIR vorher zu beenden. Drücken Sie dazu ein [C] und [Return]. Dann erscheint die Aufforderung »Command:«. Geben Sie dahinter den gewünschten Befehl ein und drücken Sie wieder auf die [Return]-Taste. Sobald der Befehl beendet ist, können Sie mit dem DIR-Befehl fortfahren.

Sind Sie sich während eines interaktiven DIR-Befehls im unklaren darüber, welche Möglichkeiten Sie gerade haben, tippen Sie hinter dem Fragezeichen einfach noch ein Fragezeichen. Eine Liste der im Augenblick möglichen Eingaben wird dann angezeigt.

Das Schlüsselwort OPT ist nur aus historischen Gründen vorhanden. Es wurde bei früheren Versionen des DIR-Befehls anstatt der Schlüsselworte ALL, DIRS und INTER verwendet. OPT A entspricht ALL, OPT D entspricht DIRS und OPT I entspricht INTER.

**Beispiele:**

```
DIR Briefe
DIR :Texte/alteBriefe
DIR DF0: ALL
DIR DF0: INTER
DIR #?.BAK
```

Der erste dieser Befehle listet den Inhalt des Directory *Briefe* im aktuellen Directory auf, während der zweite das Directory *Texte/alteBriefe* auf der aktuellen Diskette auflistet. Der dritte Befehl zeigt den kompletten Inhalt der Diskette im internen Laufwerk. Der vierte leitet eine interaktive Durchsuchung der Directories und Dateien im internen Laufwerk ein und der letzte DIR-Befehl schließlich zeigt alle Dateien im aktuellen Directory, die mit ».BAK« enden.

# DISKCHANGE

**DISKCHANGE** DEV/A **<Laufwerk>**

dient dazu, einen Diskettenwechsel bei Amiga-DOS anzumelden.

Der Befehl DISKCHANGE teilt Amiga-DOS mit, daß die Diskette im Laufwerk <Laufwerk> (zum Beispiel DF1:) ausgewechselt wurde. Bei den 3,5-Zoll-Laufwerken von Commodore ist dieser Befehl nicht notwendig. Bei diesen merkt Amiga-DOS von selbst, wenn eine Diskette herausgenommen oder eingelegt wird. Bei einigen Laufwerken anderer Hersteller (speziell 5,25-Zoll-Laufwerken) kann dieser Befehl nötig sein und ist dann enorm lästig. Kaufen Sie deshalb am besten kein Laufwerk, das einen Diskettenwechsel nicht von selbst (hardwaremäßig) Amiga-DOS mitteilt.

# DISKCOPY

**DISKCOPY** FROM/A **<diskname1> TO/A/K <diskname2>** NAME/K <name>

dient zum Kopieren kompletter Disketten.

Der Befehl DISKCOPY legt eine komplette Kopie einer Diskette auf einer anderen Diskette an. Der alte Inhalt der zweiten Diskette wird dabei vollständig zerstört. DISKCOPY ist besonders für das regelmäßige Anlegen von Sicherungskopien sehr sinnvoll. Bei Anwendung von DISKCOPY ist ein vorheriges Formatieren der Diskette (zum Beispiel mit dem Shell-Befehl FORMAT) nicht nötig! Für die Kopie kann deshalb sowohl eine schon benutzte als auch eine neue, unbenutzte Diskette verwendet werden.

Genau wie beim Diskettenkopieren in der Workbench kann auch DISKCOPY den Kopiervorgang entweder mit einem oder zwei Diskettenlaufwerken bewältigen. So kopiert »DISKCOPY df0: TO df1:« zum Beispiel den kompletten Inhalt der Diskette, die sich in

diesem Moment im internen Laufwerk befindet, auf die Diskette, die sich in diesem Moment im externen Laufwerk befindet. Dies dürfte die übliche Vorgehensweise sein, wenn Sie stolzer Besitzer eines externen Laufwerks sind. Bevor der Kopiervorgang wirklich beginnt, fordert der Befehl aber noch einmal dazu auf, die korrekten Disketten in beide Laufwerke einzulegen und dann erst <RETURN> zu drücken.

Besitzen Sie nur ein Diskettenlaufwerk (das eingebaute des Amiga 500), so wird die übliche Anwendungsform des Befehls für Sie wohl »DISKCOPY df0: TO df0:« sein. Genau wie beim Diskettenkopieren mit nur einem Laufwerk in der Workbench werden Sie nun immer wieder zu einem Wechsel zwischen Original-Diskette und Kopie aufgefordert. Eine gute Beschriftung beider Disketten ist für diesen Fall sehr wichtig und erleichtert Ihnen die Arbeit.

Egal, ob Sie ein oder zwei Diskettenlaufwerke besitzen, ist es in jedem Fall wichtig, Vorkehrungen zu treffen, damit nicht die Original-Diskette durch einen Bedienungsfehler zerstört wird. Der DISKCOPY-Befehl ist weitgehend gegen solche Bedienungsfehler geschützt, aber sie wissen ja: »Sicher ist sicher !« Deshalb schützen Sie bitte die Original-Diskette vor eventuellem Überschreiben, indem Sie das Schreibschutzplättchen in die Position bringen, bei der ein kleines Loch entsteht. Die Diskette, die die Kopie aufnehmen soll, darf natürlich nicht schreibgeschützt sein.

Die auf einem dieser beiden Wege erzeugte Kopie wird für Sie nicht vom Original zu unterscheiden sein. (Amiga-DOS kann die beiden Disketten allerdings als unterschiedlich erkennen.) Sie enthält alle Daten des Originals und trägt auch denselben Namen. Falls die Original-Diskette irgendwann einmal fehlerhaft wird, können Sie getrost die Kopie verwenden. Dies gilt leider nur in dem Fall, in dem der Hersteller der Programme, die sich eventuell auf dem Original befinden, diese nicht durch besondere Tricks vor dem Kopieren geschützt hat.

Wenn Sie wollen, daß die Kopie einen anderen Namen als die Originaldiskette erhält, so können Sie diesen Namen gleich beim Kopieren verwenden. Dazu dient der Parameter <name> hinter dem Schlüsselwort NAME.

**Beispiele:**

```
DISKCOPY DF1: TO DF0:
DISKCOPY FROM DF0: TO DF0:
DISKCOPY DF0: TO DF1: NAME "Kopie der Diskette 1"
```

Der erste dieser Befehle kopiert die Diskette, die im ersten externen Laufwerk liegt, auf die Diskette im internen Laufwerk. Der zweite kopiert mit nur einem Laufwerk, wobei Sie zwischendurch immer wieder zum Wechsel zwischen Original-Diskette und Kopie aufgefordert werden. Und der letzte Befehl kopiert vom internen zum externen Laufwerk und gibt der Diskette im externen Laufwerk gleichzeitig den Namen *Kopie der Diskette 1*. Beachten Sie, daß die beiden ersten Parameter notwenig sind und daß vor dem zweiten Parameter das Schlüsselwort TO stehen muß.

# DISKDOCTOR

**DISKDOCTOR** DRIVE/A **<Laufwerk>**

dient zum Wiederherstellen defekter Disketten.

Der Befehl DISKDOCTOR kann unter gewissen Umständen Dateien beziehungsweise ganze Disketten retten, die durch Bedienungsfehler oder fehlerhafte Programme zerstört oder beschädigt wurden. Solche Fehler verkündet Ihnen das CLI mit der Fehlermeldung:

```
Disk structure corrupt. Use DiskDoctor to fix it!
```

In solchen Fällen sollten Sie zunächst (am besten mit DISKCOPY; siehe oben) eine Kopie der fehlerhaften Diskette anlegen. Legen Sie dann Ihre Workbench-Diskette in das interne Laufwerk und geben Sie dann den folgenden Befehl ein:

```
DISKDOCTOR DF0:
```

Dann erscheint die Meldung

```
Disk Doctor V1.3.4
Insert disk to be corrected and press RETURN
```

Nehmen Sie dann die Workbench-Diskette aus dem Laufwerk und schieben Sie die Kopie der defekten Diskette ein. Deren Schreibschutz darf dabei nicht aktiviert sein. Drücken Sie nun auf die [Return]-Taste. Diskdoctor versucht dann die Diskette zu »reparieren«, was geraume Zeit dauern kann. Diskdoctor gibt dabei einige Meldungen aus, die besagen, wie weit der Reparaturprozeß ist. Wenn das Programm dabei gelöschte Dateien oder Directories findet und wiederherstellt, erscheint immer die Meldung:

```
Replacing DIR  <Directoryname>
```

beziehungsweise:

```
Replacing file <Dateiname>
```

Wenn die Reparatur beendet ist, erscheint die Meldung:

```
Now COPY files required to a new disk and reformat this disk
```

Kopieren Sie daraufhin alle Dateien, die Sie von der defekten Diskette benötigen, auf eine andere, am besten frisch formatierte Diskette. Dann können Sie versuchen, die defekte Diskette (also das Original) neu zu formatieren (mit *Initialize* auf der Workbench oder mit FORMAT im Shell-Fenster.) Sollten dabei Fehler auftauchen oder gibt DISKDOCTOR während seiner Arbeit die folgende Fehlermeldung aus, sollten Sie diese Diskette lieber in den Müll werfen oder zum Händler zurückbringen:

```
Hard Error Track <nn> Surface <n>
```

Der DISKDOCTOR-Befehl kann nämlich echte (physikalische) Diskettenfehler, die bei der Produktion oder durch falsche Handhabung der Diskette entstanden sind, nicht beheben. Er repariert nur »Software-Fehler« und stellt Dateien wieder her, die Sie vielleicht versehentlich gelöscht haben.

# ECHO

**ECHO** <Text> NOLINE/S FIRST/K <N1> LEN/K <N2>

dient zum Ausgeben eines Textes (einer Meldung) im Shell-Fenster.

Der Text, der hinter ECHO (am besten in Anführungszeichen) als Parameter angegeben ist, wird im Shell-Fenster ausgegeben. Dies ist im allgemeinen nur sinnvoll in Kommandofolgen und bei Befehlen, die im Hintergrund ausgeführt werden. Man kann den Text allerdings auch auf diese Weise zu einem Gerät schicken, indem man die Ausgabe umleitet.

Das Schlüsselwort NOLINE bewirkt, daß nach Ausgabe des Textes nicht automatisch eine neue Zeile begonnen wird. Der nächste ECHO-Befehl würde seinen Text also in derselben Zeile anfügen.

Mit den Schlüsselwörtern FIRST und LEN können Sie bewirken, daß nur ein Teil des Textes am Bildschirm ausgegeben wird; und zwar vom Buchstaben Nummer <N1> bis zum Buchstaben Nummer <N2>. Wird nur <N1> angegeben wird der Anfang des Textes bis zum Buchstaben Nummer <N1> unterdrückt. Wird nur <N2> angegeben, werden lediglich die <N2> ersten Buchstaben ausgegeben. Die Schlüsselwörter FIRST und LEN sind aber nur in raffiniert konstruierten Kommandofolgen im Zusammenhang mit der Ein-/Ausgabeumleitung interessant.

Für alle Shell-Anwender ist die Möglichkeit interessant, Sonderzeichen im <Text> unterzubringen. Die Buchstabenfolge »*N« erzeugt zum Beispiel mitten im Text einen Zeilenvorschub. Sie können deshalb mit einem einzigen ECHO-Befehl mehrere Textzeilen ausgeben. Und die Buchstabenfolge »*N« erzeugt ein Escape-Zeichen, mit dem Sie spezielle Kontrollbefehle an das Shell-Fenster schicken können. Bei diesen Kontrollbefehlen handelt es sich um sogenannte »ANSI-Steuersequenzen«, mit denen vor allem die Darstellung von Texten im Shell-Fenster geändert werden kann. Eine vollständige Liste dieser Steuersequenzen finden Sie im im Anhang dieses Buches.

**Beispiele:**

```
ECHO "Hello world!"
ECHO >PRT: "Hello world!"
```

Der erste dieser beiden Befehle schreibt die Meldung »Hello world!« auf den Bildschirm, der zweite auf den Drucker (falls Sie einen besitzen, angeschlossen und eingeschaltet haben). Die folgenden drei Befehle demonstrieren die Verwendung von Steuerzeichen:

```
ECHO "Hello *Nworld!"
ECHO "Hello *E[1mworld!"
ECHO "Hello *E[1mworld!*E[0m"
```

Der erste dieser Befehle schreibt »Hello« in eine und »world« in die folgende Zeile. Der zweite Befehl schreibt »Hello world« in eine Zeile, wobei aber das zweite Wort fett gedruckt wird (durch die ANSI-Sequenz »*E[1m«). Die Umschaltung auf Fettdruck ist allerdings permanent. Das heißt, auch die folgende Eingabeaufforderung und alle Befehle, die Sie danach eingeben, werden fett gedruckt. Der dritte Befehl demonstriert, wie man den Fettdruck einschaltet und dann wieder ausschaltet (durch die ANSI-Sequenz »*E[0m«).

# ED

**ED** FROM/A **<Dateiname>** SIZE/K <n>

dient zum Anlegen und Ändern von Dateien, die druckbare Texte enthalten.

Gibt man den Befehl ED, so gelangt man in einen einfachen »Editor«. Ich verwende das Wort »Editor« und nicht etwa »Textverarbeitungsprogramm«, um dieses einfache Programm von komfortableren Textverarbeitungsprogrammen abzusetzen. Der Editor ED kennt keine verschiedenen Zeichensätze, kennt keine Möglichkeit zum Unterstreichen oder Fettdrucken oder für die seitenweise Bearbeitung von Texten. Er ist nur für die Bearbeitung von reinen Textdateien – zum Beispiel Programmen und Kommandofolgen – geeignet.

ED ist allerdings ein sehr primitiver Editor, dessen Verwendung nicht empfohlen werden kann – unter anderem, weil auf der Extras-Diskette der wesentlich bessere Editor MEMACS mitgeliefert wird. Auf eine nähere Beschreibung von ED wird deshalb hier verzichtet.

# EDIT

**EDIT** FROM/A <Dateiname TO <Dateiname2>
WITH/K <Kommandodatei> VER/K <Ausgabe> OPT P <n> OPT W <n>

dient zum Anlegen und Modifizieren von Dateien, die druckbare Texte enthalten.

Der Befehl EDIT aktiviert einen sehr altmodischen, aber auch sehr leistungsfähigen Texteditor, mit dem Sie ähnliche Aufgaben erledigen können wie mit ED oder MEMACS. Es handelt sich dabei um einen sogenannten »Zeileneditor«, wie sie früher auf wesentlich primitiveren Computern weit verbreitet waren. Zeileneditoren sind unhandlich zu bedienen und erlauben immer nur die Bearbeitung einer aktuellen Zeile. Der wesentliche Vorteil von EDIT gegenüber moderneren Editoren liegt allerdings darin, daß die Bearbeitung eines Textes mit einer Kommandofolge automatisiert werden kann. Wollen Sie in zehn langen Texten sämtliche Vorkommnisse zweier Wörter durch zwei andere Wörter ersetzen, ist das auch mit einem komfortablen Editor meist recht umständlich.

Sie müssen den Editor aufrufen und dann zehnmal nacheinander eine Datei einlesen, das erste Wort suchen und ersetzen lassen, das zweite Wort suchen und ersetzen lassen und die Datei wieder abspeichern. EDIT hingegen kann die Befehle, mit denen ein Text editiert werden soll, auch von einer Textdatei empfangen. Sie können EDIT auf diese Weise automatisch (ohne Aufsicht) langwierige Operationen mit einer oder mehreren großen Dateien durchführen lassen.

Sie werden EDIT im Normalfall nie benötigen und MEMACS auch als wesentlich angenehmer empfinden. Benötigen Sie jedoch die Möglichkeiten von EDIT, so finden Sie eine ausführliche Beschreibung im Amiga-DOS-Handbuch (erschienen im Markt&Technik-Verlag, Nr. 90465, Haar bei München, 1987).

# ELSE

IF ... **ELSE** ... ENDIF

dient zur bedingten Ausführung von Befehlen in Kommandofolgen.

Das Kommando ELSE steht üblicherweise für sich allein in einer Zeile einer Kommandofolge. (Es ist nicht sinnvoll, ELSE direkt im Shell-Fenster zu verwenden.) Vor ELSE muß in derselben Kommandofolge immer ein IF-Befehl auftauchen. Nach ELSE muß in derselben Kommandofolge noch ein ENDIF-Befehl folgen.

Trifft die Bedingung hinter dem IF-Befehl zu, werden die Befehle zwischen IF und ELSE ausgeführt und die Befehle zwischen ELSE und ENDIF ignoriert (übersprungen). Trifft die Bedingung hinter dem IF-Befehl nicht zu, werden die Kommandos zwischen IF und ELSE übersprungen und die Kommandos zwischen ELSE und ENDIF ausgeführt.

IF-ELSE-ENDIF-Gruppen können auch geschachtelt auftauchen. Das heißt, zwischen ELSE und ENDIF könnte zum Beispiel eine weitere IF-ELSE-ENDIF-Gruppe stehen, deren Anweisungen aber nur dann ausgeführt werden können, wenn die Bedingung des »äußeren« IF-Befehls nicht zutrifft.

Mehr zu den Befehlen IF, ELSE und ENDIF können Sie im Kapitel 8 erfahren.

# ENDCLI

**ENDCLI**

dient zum Beenden eines aktiven Shell-Prozesses.

Der Befehl ENDCLI beendet den Shell-Prozeß, in dem der Befehl erteilt wird. Das zugehörige Shell-Fenster wird geschlossen, nachdem eine entsprechende Meldung ausgegeben wird. (Diese Meldung wird in der Startup-sequence mit >NIL: ins »Nichts« geschickt und ist deshalb nicht zu sehen.)

Mit diesem Befehl sollte man sehr vorsichtig umgehen! Er sollte nur erteilt werden, wenn »im Hintergrund« hinter dem Shell-Fenster noch die Workbench sichtbar ist oder noch ein oder mehrere andere Shell-Fenster offen sind. Ist die Workbench nicht (mit LOADWB) gestartet worden und das geschlossene Fenster war das einzige Shell-Fenster, »hängt« der Amiga nach der Ausführung des ENDCLI-Befehls – er nimmt dann keinerlei Befehle über die Tastatur oder die Maus mehr an.

# ENDIF

**ENDIF**

dient zur bedingten Ausführung von Befehlen in Kommandofolgen.

Das Kommando ENDIF steht meistens für sich allein in einer Zeile einer Kommandofolge. (Es ist nicht sinnvoll, ENDIF direkt im CLI zu verwenden.) Vor ENDIF muß in derselben Kommandofolge noch ein IF-Befehl auftauchen. Zwischen diesem ENDIF und ENDIF **kann** in derselben Kommandofolge ein ELSE-Befehl stehen.

Trifft die Bedingung hinter dem IF-Befehl zu, werden die Befehle zwischen IF und ELSE (fals vorhanden) oder zwischen IF und ENDIF (falls kein ELSE vorhanden ist) ausgeführt und die Befehle zwischen ELSE (falls vorhanden) und ENDIF ignoriert. Trifft die Bedingung hinter dem IF-Befehl nicht zu, werden die Kommandos zwischen IF und ELSE übersprungen und die Kommandos zwischen ELSE und ENDIF ausgeführt.

IF-ELSE-ENDIF-Gruppen können auch geschachtelt auftauchen. Das heißt, zwischen ELSE und ENDIF könnte zum Beispiel eine weitere IF-ELSE-ENDIF-Gruppe stehen, deren Anweisungen aber nur dann ausgeführt werden können, wenn die Bedingung des »äußeren« IF-Befehls nicht zutrifft.

Mehr zu den Befehlen IF, ELSE und ENDIF können Sie in Kapitel 8 erfahren.

# ENDSHELL

**ENDSHELL**

dient zum Beenden eines aktiven Shell-Task.

ENDSHELL ist kein echter Shell-Befehl, sondern ein Aliasname (Pseudonym), der normalerweise in jedem Shell-Fenster gültig ist (das der entsprechende ALIAS-Befehl in der Datei Shell-Startup liegt).

ENDSHELL führt den Befehl ENDCLI aus!

# ENDSKIP

**ENDSKIP**

dient zur bedingten Ausführung von Befehlen in Kommandofolgen.

Das Kommando ENDSKIP steht meistens für sich allein in einer Zeile einer Kommandofolge. (Es ist nicht sinnvoll, ENDSKIP direkt im CLI zu verwenden.) Vor ENDSKIP muß in derselben Kommandofolge noch ein SKIP-Befehl auftauchen.

Wurde dieser SKIP-Befehl ausgeführt (SKIP-Befehle stehen meist in einer IF-ELSE-END-IF-Gruppe), werden alle Befehle zwischen diesem SKIP und ENDSKIP ignoriert (übersprungen).

Mehr zu den Befehlen IF, ELSE, ENDIF, SKIP, LAB und ENDSKIP können Sie im Kapitel 8 erfahren.

# EVAL

**EVAL** VALUE1/A **<Wert1>** OP <Operator> VALUE/2 <Wert2>
TO <Datei> LFORMAT/K <Format>

dient zur Berechnung von arithmetischen Ausdrücken.

Mit dem Befehl EVAL steht Ihnen eine Art Taschenrechner im Shell-Fenster zur Verfügung. EVAL berechnet einen arithmetischen Ausdruck, der in Form mehrerer Parameter übergeben werden muß.

<Wert1> und <Wert2> müssen zwei Zahlen sein. Sie können als Dezimalzahl, Oktalzahl, Hexadezimalzahl oder Buchstabe angegeben werden. Soll eine Zahl als Hexadezimalzahl aufgefaßt werden, müssen Sie ihr die Zeichenfolge »0X« oder »#X« voranstellen. Oktalzahlen muß »0« oder »#« vorangehen. Wenn Sie einen Buchstaben und davor ein einzelnes Anführungszeichen (zum Beispiel »'A«) angeben, so verwendet EVAL den SCII-Code dieses Buchstabens als <Wert1> beziehungsweise <Wert2>.

<Operator> kann ein arithmetischer oder logischer Operator sein. Die folgenden Operatoren werden von EVAL »verstanden« (sie entsprechen in etwa den arithmetischen Operationen der Programmiersprache C):

| | |
|---|---|
| + | ganzzahlige Addition |
| – | ganzzahlige Subtraktion |
| * | ganzzahlige Multiplikation |
| / | ganzzahlige Division (ohne Rundung) |
| mod | ganzzahliger Modulus (Rest) |
| | |
| & | logisches (bitweises) Und |
| \| | logisches (bitweises) Oder |
| xor | logisches (bitweises) Exklusiv-Oder |
| << | bitweise Verschiebung nach links |
| >> | bitweise Verschiebung nach rechts |
| eqv | bitweise Identitätsprüfung |

Das Ergebnis der Berechnung wird normalerweise als eine Dezimalzahl ausgegeben. Mit dem Schlüsselwort LFORMAT ist es aber möglich, das Ergebnis in einen Text einzubetten oder auch in anderen Zahlenformaten ausgeben zu lassen. Der Parameter «Format« hinter dem Schlüsselwort LFORMAT muß zu diesem Zweck aus einem Text bestehen, in dem die Buchstabenfolge »%X«, »%O«, »%N« oder »%C« eingebettet ist. EVAL berechnet dann den Ausdruck aus den ersten drei Parametern und gibt diesen Text aus – die mit dem

Prozentzeichen beginnende Buchstabenfolge wird allerdings gegen das Ergebnis der Berechnung ausgetauscht. Der Buchstabe nach dem Prozentzeichen bestimmt das Ausgabeformat: Bei »%X« wird das Hexadezimalformat verwendet, bei »%O« das Oktalformat, bei »%N« das Dezimalformat und statt »%C« wird ein Buchstabe ausgegeben, desse ASCII-Code gleich dem Ergebnis der Berechnung ist. Bei Verwendung von »%X« und »%O« muß hinter X beziehungsweise 0 noch die Anzahl der gewünschten Stellen der ausgegebenen Zahl angegeben werden. Weiterhin können in dem Text »Format auch dieselben Steuerzeichen verwendet werden wie bei ECHO.

**Beispiele:**

```
1.SYS>EVAL 5 * -3
-15
1.SYS>EVAL 13 / 5
2
1.SYS>EVAL #XFF00 | #X00FF
65535
1.SYS>EVAL #XFF00 | #X00FF LFORMAT "Im Hexformat: %X6!*N"
Im Hexformat: 00FFFF!
1.SYS>EVAL 64 LFORMAT "ASCI-Code 64: %C!*N"
ASCI-Code 64: A!
```

Die Verwendung des Befehls EVAL ist vor allem für Programmierer sowie in Kommandofolgen interessant. Durch geschickte Verwendung von EVAL und von Umgebungsvariablen (siehe auch SETENV und GETENV) können unter anderem Zählschleifen aufgebaut werden. Mehr zum Thema Kommandofolgen können Sie in Kapitel 8 erfahren.

# EXECUTE

**EXECUTE <Textdatei>** <argumentl> <argumentl2> ...

dient zum Ausführen einer Liste von Shell-Befehlen in <Textdatei>.

Der Befehl EXECUTE dient zur »Automatisierung« der Shell. Diese Möglichkeit, längere Aufgabenfolgen ohne manuelles Zutun ablaufen zu lassen, ist einer der wesentlichen Vorteile der Shell gegenüber der Workbench. Hierzu werden die Shell-Befehle, die nacheinander ausgeführt werden sollen, in <Textdatei> gespeichert. Die Ausführung aller dieser Befehle kann mit EXECUTE <Textdatei> »angestoßen« werden.

In der Kommandofolge können Platzhalter für Dateinamen und andere Variablen verwendet werden, die beim Aufruf gegen die Liste der Parameter, die im EXECUTE-Befehl hinter dem Namen der Kommandofolge stehen, ersetzt werden.

Der Befehl EXECUTE ist zur Ausführung einer Kommandofolge im Shell-Fenster nicht mehr unbedingt nötig. Wenn bei der Textdatei, in der die Kommandofolge gespeichert ist, das S-Flag gesetzt ist (siehe auch die Beschreibung des PROTECT-Befehls), genügt der Name dieser Datei allein. Der Dateiname kann dann wie ein Shell-Befehl verwendet werden. Die Möglichkeiten von EXECUTE werden ausführlich im Kapitel 8 erläutert.

# FAILAT

**FAILAT <n>**

dient zum Ändern des Fehlerabbruchlevels.

Der Befehl EXECUTE (siehe oben) führt eine Folge von Shell-Befehlen aus, die in einer Textdatei gespeichert sind. Tritt bei der Ausführung eines dieser Befehle ein Fehler auf, so bricht je nach Schwere des Fehlers eventuell die Ausführung der gesamten Folge sofort ab. Die restlichen Befehle werden nicht mehr aufgerufen.

Die Schwere eines Fehlers meldet jeder Shell-Befehl durch den sogenannten »Returncode« an die Shell zurück. Dieser Returncode ist bei der erfolgreichen Ausführung des Befehls gleich 0. Trat eine besondere Situation ein, die aber nicht zum Versagen des Befehls führte ist der Returncode gleich. Bei echten Fehlern ist der Returncode gleich 10 oder 20. Diese drei gebräuchlichsten Fehlerlevel entsprechen den Bedingungen (Schlüsselwörtern) des IF-Befehls: WARN (Returncode 5), ERROR (10) und FAIL (20).

Bei der Ausführung einer Kommandofolge ist die Höhe des Fehlerabbruchlevels gleich 10. Sobald der Returncode eines Shell-Befehls 10 erreicht oder übersteigt, wird deshalb die Kommandofolge abgebrochen. Ein sofortiger unbedingter Abbruch ist nicht immer wünschenswert. Manchmal möchte man, daß nur bei sehr schwerwiegenden Fehlern abgebrochen wird oder man möchte den Fehler selbst erkennen und innerhalb der Kommandofolge darauf reagieren. In diesem Fall muß man den Fehlerabbruchlevel heraufsetzen.

Hierzu dient der Befehl FAILAT: Durch FAILAT wird Fehlerabbruchlevel auf <n> gesetzt oder der aktuelle Fehlerabbruchlevel ausgegeben, wenn <n> nicht angegeben wird.

**Beispiele:**

```
FAILAT 20
```

Dieser Befehl setzt das Fehlerabbruchlevel auf 20 hoch. Kommandofolgen werden dann nur noch von sehr schwerwiegenden Fehlern abgebrochen.

```
FAILAT 30
```

Dieser Befehl setzt das Fehlerabbruchlevel auf 30 hoch. Kommandofolgen werden dann von Fehlern, die bei der Ausführung von Shell-Befehlen auftreten können, nicht mehr abgebrochen. Lediglich der QUIT-Befehl kann dazu verwendet werden, einen Returncode von über 20 zu erzeugen und Kommandofolgen abzubrechen.

# FASTMEMFIRST

**FASTMEMFIRST**

Der Shell-Befehl FASTMEMFIRST entspricht dem Werkzeug *FastMemFirst* in der Schublade *System*. Wenn Sie diesen Befehl das erste Mal erteilen, werden Programme und Daten bevorzugt in den RAM-Bereich außerhalb des sogenannten Chip-RAM gelegt. Der nächste Aufruf von FASTMEMFIRST hebt diesen Effekt wieder auf.

Solange FASTMEMFIRST aktiv ist, wird der Chip-RAM-Bereich nur dann verwendet, wenn der FastMem-Bereich des RAM-Speichers voll ist oder wenn ein Programm ausdrücklich auf Chip-RAM »besteht«. Das kann dazu führen, daß der Amiga insgesamt etwas schneller wird, da Programme, die im FastMem-Bereich liegen, unter gewissen Umständen etwas schneller laufen.

# FAULT

**FAULT** <n1> <n2> ... <n10>

dient zur Ausgabe einer (englischen) Fehlermeldung im Volltext zu einem bestimmten Fehlercode.

Amiga-DOS beziehungsweise ein Shell-Befehl, der Amiga-DOS benutzt, meldet Fehler oft nur in Form einer Nummer. Wenn diese Nummer in einem Requester oder als Ausgabe eines fehlgeschlagenen Shell-Kommandos auftaucht, können Sie mit FAULT eine knappe (englische) Erläuterung dazu erhalten. Dazu brauchen Sie nur eine Liste der Fehlernummern (maximal 10) hinter FAULT anzugeben. Weitergehende Beschreibungen dieser Fehler, mögliche Fehlerursachen und Behebungsvorschläge finden Sie im Anhang dieses Buches.

**Beispiele:**

```
1.SYS:>FAULT 103 203
FAULT 103: insufficient free store
FAULT 203: object already exists
```

# FF

**FF** <Schrift> -O/S -N/S

dient zur Beschleunigung der Textausgabe.

Text wird vom Amiga wie eine Grafik behandelt. Jeder Buchstabe wird praktisch auf den Bildschirm gemalt. Das ermöglicht es, viele verschiedene Schriften, Schriftgrößen und -Formen zu verwenden, kostet aber auch Zeit. Das merkt man spätestens in einem Textverarbeitungsprogramm, das große Textmengen in verschiedenen Schriften auf den Bildschirm zu bringen hat.

Die Softwarekomponenten im ROM des Amiga, die für das Zeichnen von Texten zuständig sind, erfüllen diese Aufgabe nicht ganz so schnell wie es möglich wäre. Der Befehl FF (engl. fast fonts = schnelle Schriften) aktiviert eine schnellere Software (die übrigens nicht Commodore, sondern der Programmierer Charlie Heath von der Firma Microsmiths entwickelt hat).

Der Schalter –O aktiviert die Fast-Fonts-Software und der Schalter –N deaktiviert sie wieder. Eine Deaktivierung kann deshalb sinnvoll sein, weil die Fast-Fonts-Software RAM-

Speicher benötigt, der eventuell auch für andere Zwecke benötigt wird. Wenn Sie die Beschleunigung der Textausgabe nur auf eine Schrift beschränken wollen (das spart etwas Speicherplatz), müssen Sie deren Namen als ersten Parameter angeben. Dabei müssen Sie hinter den Schriftnamen die Endung ».font« setzen.

**Beispiele:**

```
FF
FF -O
FF -N
FF emerald.font -O
```

Die ersten beiden Befehle aktivieren die Fast-Fonts-Software, der dritte schaltet sie wieder aus. Der letzte Befehl schließlich aktiviert die Fast-Fonts-Software ausschließlich im Zusammenhang mit der Schrift *Emerald*. Die Textausgabe in anderen Schriften wird nicht beschleunigt.

# FILENOTE

**FILENOTE** FILE/A **<Dateiname>** COMMENT/A **<Kommentar>**

dient dazu, eine Datei mit einem Kommentar zu versehen.

Das Betriebssystem des Amiga kann zu jeder Datei einen kurzen Kommentar bis zu einer Länge von maximal 79 Buchstaben verwalten. Dieser Kommentar kann völlig frei gewählt werden. Er wird im allgemeinen zusätzliche Informationen über den Inhalt der Datei enthalten, die über das hinausgehen, was man in einem Dateinamen unterbringen kann. So könnte man zum Beispiel vermerken, daß es sich bei einer bestimmten Datei um die 3. Version handelt und ähnliches. Der Kommentar zu einer Datei kann auf Wunsch mit dem Shell-Befehl LIST (siehe unten) betrachtet, jedoch nicht geändert werden, und mit dem Befehl FILENOTE geändert, aber nicht betrachtet werden. In der Workbench kann derselbe Kommentar mit dem Menübefehl *Info* betrachtet und geändert werden.

Um eine Datei mit einem Kommentar zu versehen, brauchen Sie nur den Namen dieser Datei als ersten Parameter hinter FILENOTE anzugeben und hinter dem Schlüsselwort COMMENT dann den Text des Kommentars. Beide Parameter müssen angegeben werden! Handelt es sich bei diesem Kommentar um mehrere Wörter, die durch Zwischenräume und/oder Sonderzeichen getrennt sind, muß der Kommentar in Anführungszeichen gesetzt werden.

Der Befehl FILENOTE hat übrigens einige merkwürdige Eigenschaften. Eine neue Datei hat zunächst keinen Kommentar. Wird aber – zum Beispiel beim Kopieren – eine alte Datei mit einem neuen Inhalt überschrieben, bleibt der alte Kommentar dieser Datei erhalten (der ja jetzt nicht mehr zutrifft). Es wird also nicht der Kommentar der Datei verwendet, die kopiert wurde, sondern der Kommentar der Datei, die durch den Kopiervorgang zerstört wurde. Wenn Sie dieses unlogische Verhalten vermeiden wollen, müssen Sie beim COPY-Befehl den Schalter COM oder CLONE verwenden.

**Beispiele:**

```
FILENOTE FILE Brief2 COMMENT "Brief an Tante Elli!!!"
FILENOTE Geheim "Brief an Tante Elli!!!"
```

Der erste Befehl versieht die Datei Brief im aktuellen Directory mit einem Kommentar. Der zweite macht genau dasselbe mit der Datei *Geheim* (und verzichtet auf die beiden Schlüsselwörter.

# FORMAT

**FORMAT DRIVE/A/K <Laufwerk> NAME/A/K <Name>**
NOICONS/S QUICK/S FFS/S NOFFS/S

dient zum Formatieren und Initialisieren einer Diskette oder Festplatte.

Bevor eine Diskette oder eine Festplatte verwendet werden kann (bevor Directories und Dateien darauf angelegt werden können), muß sie formatiert werden. Bei diesem Vorgang wird der amorphen Struktur des Speichermediums, das ja nur eine Fläche magnetisierbaren Materials ist, ein Format »aufgezwungen«. Die Diskette besteht danach aus einer Folge von konzentrischen Spuren (engl. *tracks*) und jede dieser Spuren besteht aus einer bestimmten Anzahl von Blöcken oder Sektoren (engl. *sectors*). Nach dieser rein magnetischen Formatierung muß die Diskette noch initialisiert werden. Dabei wird das zunächst leere Directory angelegt, das die Wurzel der Datei-Hierarchie auf dieser Diskette oder Festplatte bildet.

Diese beiden Aufgaben erfüllt der Shell-Befehl FORMAT. Seine Anwendung ist unmißverständlich. Er benötigt zwei Parameter, die beide von einem Schlüsselwort eingeleitet werden. Der erste Parameter <Laufwerk> gibt an, in welchem Disketten- oder Festplattenlaufwerk sich die zu formatierende Diskette oder Festplatte befindet (zum Beispiel DF0: oder HD1:). Der zweite Parameter <Name> gibt den Namen an, den die frisch formatierte Diskette oder Festplatte danach tragen soll. Enthält dieser Name Leerstellen, muß er in Anführungszeichen gesetzt werden.

Wenn Sie die neuformatierte Diskette nur von der Shell verwenden wollen, sollten Sie das Schlüsselwort NOICONS am Ende des FORMAT-Befehls verwenden. Dann erscheint kein Mülleimer auf dieser Diskette und Sie gewinnen ein paar Bytes zusätzlichen Speicherplatz.

Das Schlüsselwort QUICK führt nur die Initialisierung durch, legt also ein leeres Wurzeldirectory an. Diese Form des FORMAT-Befehls ist deshalb nur für Disketten geeignet, die schon einmal formatiert worden sind und kann solche Disketten sehr schnell (engl. *quick*) vollständig löschen.

Das Schlüsselwort FFS ist nur im Zusammenhang mit der Formatierung einer Festplatte sinnvoll. Es gibt an, daß für die Dateiverwaltung auf dieser Festplatte das sogenannte »FastFileSystem« verwendet werden soll. Dieses FastFileSystem ist eine Neuerung der Workbench-Version 1.3 und bietet einen deutlich schnelleren Zugriff auf die Daten einer Festplatte als die in früheren Versionen verwendete Methode.

Wenn Sie auf alle Fälle verhindern wollen, daß das FastFileSystem bei der Formatierung einer Festplatte verwendet wird, müssen Sie das Schlüsselwort NOFFS angeben. Das kann

sinnvoll sein, wenn Sie vorhaben, von dieser Festplatte zu starten (statt bei jedem Start die Workbench-Diskette einzulegen). Eine Festplatte darf nämlich mit dem FastFileSystem formatiert worden sein, wenn man von Ihr starten will.

Das ist nicht so problematisch, wie es jetzt vielleicht den Anschein hat. Sie können eine Festplatte nämlich in mehrere Teile (sogenannte Partitionen) aufteilen, die jede für sich wie eine einzelne Festplatte behandelt werden. Eine dieser Partitionen formatieren Sie mit NOFFS und können die dann als »Start-Diskette« verwenden. Die andere(n) Partition(en) werden mit FFS formatiert, was einen möglichst schnellen Zugriff auf die darauf abgelegten Daten garantiert.

FORMAT ist nur dann nötig, wenn Sie eine leere Diskette benötigen, um auf ihr einzelne neue Dateien anzulegen. Wollen Sie eine Diskette kopieren (mit DISKCOPY), so brauchen Sie die neue Diskette, die die Kopie aufnehmen soll, nicht zu formatieren. Das erledigt DISKCOPY gleich mit!

**Beispiele:**

```
FORMAT DRIVE DF0: NAME Disk1 NOICONS
FORMAT DRIVE HD1: NAME "Schnelle Platte" FFS
```

Der erste Befehl formatiert die Diskette im internen Laufwerk und gibt ihr den Namen *Disk1*. Diese Diskette enthält danach keine Piktogramme. Der zweite Befehl formatiert die Festplatte mit der Laufwerksbezeichnung HD1 und gibt ihr den Namen *Schnelle Platte*. Zur Verwaltung der Daten auf dieser Festplatte wird das schnelle FastFileSystem verwendet. Im Diskettenfenster dieser Diskette erscheint später auf der Workbench ein Mülleimer.

# GETENV

**GETENV** NAME/A **<Name>**

dient zum Abfragen des Inhalts einer Umgebungsvariablen.

Die Amiga-Shell kennt – wie auch viele andere Computer-Betriebssysteme – das Konzept der Umgebungsvariablen. Diese Umgebungsvariablen sind eine Art Zwischenspeicher mit Eigennamen, die beliebige Werte (im allgemeinen einen kurzen Text oder eine Zahl) annehmen können. Mit dem Befehl GETENV kann der Inhalt der Umgebungsvariablen <Name> jederzeit abgefragt werden. In Kommandofolgen kann dieser Inhalt auch in der Form »<$Name>« verwendet werden.

Umgebungsvariablen werden von der augenblicklichen Version der Shell (Workbench 1.3) in Form einzelner Dateien auf der RAM-Disk RAM: abgespeichert. Diese Dateien liegen im Directory *Env* auf der RAM-Disk, dem gleichzeitig das logische Gerät ENV: zugeordnet ist. Der Inhalt einer Umgebungsvariablen kann deshalb auch mit einem TYPE- oder COPY-Befehl festgestellt werden und durch Ausgabeumleitung in eine dieser Dateien geändert werden. (Diese Art der Realisierung des Konzepts der Umgebungsvariablen könnte sich in zukünftigen Versionen der Shell aber eventuell ändern.)

**Beispiele:**

```
1.SYS:>SETENV Compiler "DH0:WB/c/Pascal/Pascal"
1.SYS:>GETENV Compiler
DH0:WB/c/Pascal/Pascal
```

Der erste Befehl definiert eine Umgebungsvariable namens Compiler und belegt Sie mit einem Wert. Der zweite Befehl stellt diesen Wert fest (er wird am Bildschirm ausgegeben). Den selben Effekt könnte man prinzipiell bei der augenblicklichen Realisierung der Umgebungsvariablen auch mit dem folgenden Befehl erzielen:

```
1.SYS:>TYPE ENV:Compiler
DH0:WB/c/Pascal/Pascal
```

# GRAPHICDUMP

**GRAPHICDUMP** TINY/S SMALL/S MEDIUM/S LARGE/S <Größe>/S

dient zum Ausdrucken eines sogenannten Screendumps.

Der Befehl GRAPHICDUMP entspricht dem Programm GraphicDump im der Schublade *Utilities*. Dieser Befehl wartet etwa 10 Sekunden und druckt dann den Inhalt des in diesem Moment vorn liegenden Screens aus (sofern ein Drucker angeschlossen und eingeschaltet ist).

Die Parameter beziehungsweise Schlüsselwörter bestimmen die Größe des Ausdrucks. Das Schlüsselwort TINY (engl. für winzig) sorgt für einen Screendump, dessen Breite etwa ein Viertel der zur Verfügung stehenden Papierbreite beträgt. SMALL (engl. für klein) ergibt einen Screendump, der etwa halb so breit wie das Papier ist. Beim Schlüsselwort MEDIUM (engl. für mittel) erhält man einen Ausdruck, der etwa drei Viertel der Papierbreite umfaßt und bei LARGE (engl. für groß) ist er so breit wie möglich. Die Höhe des Ausdrucks ergibt sich dabei immer aus der Höhe des Bildschirms, so daß es zu keiner nicht-proportionalen Verzerrung kommt.

Damit die Schlüsselwörter TINY, SMALL, MEDIUM und LARGE eine Wirkung zeigen können, muß im Druckeranpassungsfenster von *Preferences* unter der Überschrift *Limits* die Option *Ignore* gewählt worden sein. Ansonsten bestimmt die hier (in *Preferences*) gewählte Option die Größe des Ausdrucks und die Parameter von GRAPHICDUMP haben keine Wirkung.

Wenn Sie wollen, können Sie statt mit dem Parameter <Größe> die Abmessungen des Ausdrucks aber auch sehr exakt angeben. <Größe> muß dazu das Format »Breite:Höhe« besitzen, wobei Breite und Höhe in Bildpunkten gemessen werden.

Es empfiehlt sich, GRAPHICDUMP über den Befehl RUN als eigenständigen Prozeß zu starten, wenn Sie einen Screendump des Shell-Fensters erzeugen wollen. Ansonsten können in dem Shell-Fenster, in dem der Befehl GRAPHICDUMP eingegeben wurde, solange keine weiteren Befehle ausgeführt werden, bis der Screendump fertig ist.

**Beispiele:**

```
GRAPHICDUMP LARGE
RUN GRAPHICDUMP 640:256
```

Der erste Befehl startet GRAPHICDUMP wie einen normalen Shell-Befehl. Gewünscht wird ein Screendump der maximal möglichen Größe. Bei dieser Form des Aufrufs können erst dann weitere Shell-Befehle ausgeführt werden, wenn GRAPHICDUMP beendet ist.

Der zweite Befehl startet GRAPHICDUMP als eigenständigen Prozeß. Sie können sofort danach weitere Befehle eingeben. Beim zweiten Befehl wird ein Ausdruck von 640 Punkten Breite und 256 Punkten Höhe erzwungen.

# ICONX

## ICONX

dient zum Aufruf einer Kommandofolge über ein Workbench-Piktogramm.

ICONX ist nicht dazu gedacht, vom Shell-Fenster aus aufgerufen zu werden. Dieser Befehl kann statt dessen als DEFAULT TOOL eines Projekt-Piktogramms auf der Workbench verwendet werden. Macht man dann einen Doppelklick auf dieses Piktogramm, versucht ICONX, den Inhalt der Datei, die dem Piktogramm entspricht, wie eine Kommandofolge auszuführen. Es sollte sich bei der Datei deshalb auch um eine Textdatei handeln, die zulässige Shell-Befehle enthält.

Vor Ausführung der Kommandofolge macht ICONX die Schublade, in der sich die Kommandofolge befindet, zum aktuellen Directory und öffnet ein Fenster, indem die Kommandofolge auch Ausgaben machen kann.

Wenn in der Kommandofolge auch Eingaben stattfinden sollen (mit Hilfe des ASK-Befehls), muß die erste Zeile der Kommandofolge ».KEY ""« lauten. Über ICONX gestartete Kommandofolgen können aber auch Parameter besitzen, die innerhalb der Kommandofolge auf die übliche Art und Weise (mit KEY-Anweisungen beschrieben) werden. ICONX übergibt der Kommandofolge die Namen aller anderen Piktogramme, die bei ihrem Start ebenfalls markiert waren, als weitere Parameter.

**Beispiele:**

Eine Anwendung von ICONX könnte beispielsweise wie folgt ablaufen:

Sie erzeugen eine Kommandofolge namens *Prog* und speichern diese in einem Directory ab, das auch auf der Workbench in Form einer Schublade zu sehen ist. Wenn Sie diese Kommandofolge mit einem der Shell-Editoren (ED, EDIT oder MEMACS) oder einem ähnlichen Programm erstellt haben, hat sie noch kein Piktogramm. Sie haben zwei Möglichkeiten, ein solches Piktogramm zu erzeugen: Entweder Sie erstellen es mit dem Piktogramm-Editor, der sich auf der Extras-Diskette befindet und speichern es dann im selben Directory unter dem Namen *Prog.info* ab. Oder Sie lesen die Datei einmal unter Angabe des vollen Namens im Programm *Notepad* ein und speichern Sie dann in einem anderen Directory. Dadurch erhält diese Datei das Piktogramm eines *Notepad*-Textes.

Klicken Sie die Kommandofolge, nachdem Sie sie mit einem Piktogramm versehen haben, mit der Maus an und wählen Sie den Befehl *Info* aus dem Menü *Workbench*. Daraufhin erscheint das Info-Fenster. Tragen Sie im Feld DEFAULT TOOL dieses Fensters »C:ICONX« ein. Schließen Sie das Fenster durch einen Mausklick auf SAVE. Sie können diese Kommandofolge jetzt mit einem Doppelklick von der Workbench aus starten. Sie wird dann ausgeführt, als hätten Sie im Shell-Fenster den entsprechenden EXECUTE-Befehl verwendet.

Nach Ablauf der Kommandofolge wartet ICONX einen kleinen Moment, etwa 3 Sekunden, bevor das Fenster geschlossen wird. Sie können sich die letzten Ausgaben der Kommandofolge dann noch anschauen, bevor sie mit dem Fenster verschwinden.

Position und Größe des Fensters und die Wartezeit nach Beendigung der Kommandofolge können Sie jederzeit selbst festlegen. Dazu müssen Sie im Infofenster des Piktogramms lediglich zwei Einträge im Feld TOOL TYPES vornehmen. (Wie das gemacht wird, können Sie in Kapitel 4 nachlesen.) Diese beiden Einträge müssen lauten:

```
DELAY=<Pause>
WINDOW=NEWCON:<PosX>/<PosY>/<Breite>/<Höhe>/<Titel>
```

Dabei ist <Pause> die Dauer der Wartezeit (in Sekunden), die das Fenster nach der Ausführung des letzten Befehls offenbleiben soll. <PosX>, <PosY>, <Breite>, <Höhe>, <Titel> legen die Position des rechten Fensterrands, die Position des linken Fensterrands, die Breite, die Höhe und den Titel des gewünschten Fensters fest. Ein Beispiel für eine solche Fensterbeschreibung ist:

WINDOW=NEWCON:0/20/640/192/EinLangerName

# IF

**IF** NOT/S WARN/S ERROR/S FAIL/S
<Text1> EQ/S GT/S GE/S <Text2> VAL/S EXISTS/K <Name>

dient zum bedingten Ausführen oder zum Überspringen einer Teilfolge von Shell-Befehlen innerhalb einer Kommandofolge.

Den Befehl IF werden Sie nie direkt der Shell erteilen. Er ist nur in Kommandofolgen sinnvoll. IF funktioniert ähnlich wie die IF-Anweisung in einer Programmiersprache (zum Beispiel Basic). Hinter IF folgt immer eine Bedingung, die bei der Ausführung der Kommandofolge überprüft wird. Trifft diese Bedingung nicht zu, werden alle Befehle bis zum nächsten ELSE- oder ENDIF-Befehl ignoriert (je nachdem, welcher Befehl eher gefunden wird. Die danach folgenden Befehle werden wieder ausgeführt.

Trifft die Bedingung zu, so werden die folgenden Befehle in der Kommandofolge bearbeitet, bis der Befehl ELSE oder ENDIF gefunden wird (je nachdem, welcher Befehl eher gefunden wird). Wird der Befehl ELSE zuerst gefunden, werden alle folgenden Befehle bis zu ENDIF ignoriert. Hinter ENDIF fährt die Bearbeitung der Kommandofolge dann auf alle Fälle fort.

Solche IF-ELSE-ENDIF-Gruppen können auch geschachtelt auftauchen. Das heißt, zwischen ELSE und ENDIF könnte zum Beispiel eine weitere IF-ELSE-ENDIF-Gruppe

stehen, deren Anweisungen aber nur dann ausgeführt werden können, wenn die Bedingung des »äußeren« IF-Befehls nicht zutrifft.

Die folgenden Bedingungen können zusammen mit IF verwendet werden:

WARN trifft dann zu, wenn der letzte Befehl einen Returncode ergeben hat, der größer oder gleich 5 ist.

ERROR trifft dann zu, wenn der letzte Befehl einen Returncode ergeben hat, der größer oder gleich 10 ist.

FAIL trifft dann zu, wenn der letzte Befehl einen Returncode ergeben hat, der größer oder gleich 20 ist.

Mit diesen Bedingungen können Kommandofolgen vom Erfolg vorangehender Befehle abhängig gemacht werden. Damit solche Bedingungen sinnvoll sind, muß unbedingt das Fehlerabbruchlevel mit FAILAT hochgesetzt werden. Ist das Fehlerabbruchlevel zum Beispiel kleiner als 10, wird ein IF-Befehl mit der Bedingung ERROR überhaupt nie ausgeführt, weil die Kommandofolge schon vorher abgebrochen wird.

EXISTS <name> trifft dann zu, wenn eine Datei oder ein Directory mit dem Namen <name> existiert (zum Beispiel IF EXISTS Brief1).

<Text1> EQ <Text2> trifft dann zu, wenn <Text1> gleich <Text2> ist (wobei Groß- beziehungsweise Kleinschreibung keine Rolle spielen.

<Text1> GT <Text2> trifft dann zu, wenn <Text1> in alphabetischer Sortierung nach <Text2> kommt.

<Text1> GE <Text2> trifft dann zu, wenn <Text1> in alphabetischer Sortierung nach <Text2> kommt oder gleich <Text2> ist (wobei Groß- beziehungsweise Kleinschreibung keine Rolle spielen.

Bei diesen drei Formen einer IF-Bedingung ist entweder <Text1> oder <Text2> meist ein Parameter der Kommandofolge oder eine Umgebungsvariable. Zwei Texte, deren Inhalt beim Schreiben der Kommandofolge schon feststeht, miteinander zu vergleichen, würde schließlich keinen Sinn ergeben.

Die beiden Bedingungen GT (engl. *greater than* = größer als) und GE (engl. *greater or equal* = größer oder gleich) vergleichen <Text1> und <Text2> als Texte. Die Bedingung »3 GT 12« würde deshalb zutreffen (3 kommt bei alphabetischer Sortierung nach 12). Wenn Sie einen numerischen Vergleich wünschen, müssen Sie zusätzlich das Schlüsselwort VAL angeben. Dann werden <Text1> und <Text2> als Zahlen verglichen und » VAL 3 GT 12« würde nicht mehr zutreffen.

Setzt man schließlich das Schlüsselwort NOT vor eine der bislang beschriebenen Bedingungen (zum Beispiel IF NOT EXISTS Brief1), so wird deren Ergebnis umgekehrt. Obwohl Sie NOT an beliebiger Stelle hinter dem IF-Befehl verwenden können, ließt er sich besser, wenn Sie NOT immer am Anfang der Bedingung, direkt hinter IF verwenden.

Mehr zum IF-Befehl können Sie im nächsten Kapitel erfahren.

# INFO

**INFO** DRIVE <Laufwerk>

dient dazu, Informationen über die angeschlossenen Datenträger auszugeben.

Der Befehl INFO gibt eine Liste aus, in der für jede dem Betriebssystem im Moment bekannte Diskette, Festplatte und so weiter die folgenden Informationen ausgegeben werden:

- das Laufwerk, in dem sich der Datenträger befindet (Unit)
- die Größe des Datenträgers in Kbyte (Size)
- wieviel Datenblöcke zu 512 Byte davon belegt sind (Used)
- wieviel Datenblöcke zu 512 Byte darauf frei sind (Free)
- wie voll der Datenträger in Prozent ist (Full)
- wieviele Diskettenfehler bislang registriert wurden (Errs)
- ob der Schreibschutz aktiviert ist (Read Only) oder nicht (Read/Write)
- den Namen des Datenträgers

**Beispiele:**

```
1.SYS:>INFO
Unit   Size  Used   Free Full  Errs    Status     Name
DF1:       No Disk present
RAM:     2K     4      0 100%   0   Read/Write   RAM DISK
RAD:   242K    21    461   4%   0   Read/Write   RAMB0
DF0:   880K  1676     82  95%   0   Read/Write
Workbench1.3D

Volumes available
Extras1.3D
RAM DISK [Mounted]
RAMB0 [Mounted]
Workbench1.3D [Mounted]

1.SYS:>INFO DF0:
Unit   Size  Used   Free Full  Errs    Status     Name
DF0:   880K  1676     82  95%   0   Read/Write
Workbench1.3D
```

Der erste Befehl listet eine Übersicht aller angeschlossenen Datenträger auf, der zweite gibt nur die Informationen zu einem Datenträger (in diesem Fall der Diskette im Laufwerk DF0:) aus.

# INITPRINTER

**INITPRINTER**

dient dazu, den Drucker zu initialisieren.

Der Befehl INITPRINTER entspricht dem Werkzeug *InitPrinter* in der Schublade *System* der Workbench-Diskette. Er dient zur »Initialisierung« des Druckers, also zur Vorbereitung des Druckens. Es wird nicht unbedingt bei jedem Drucker benötigt. Wenn Sie den von Ihnen verwendeten Drucker aber korrekt mit *Preferences* angemeldet haben und trotzdem nur »Quatsch« aus dem Drucker kommt, sollten Sie INITPRINTER einmal aufrufen und dann noch einmal versuchen, zu drucken.

# INSTALL

**INSTALL** DRIVE/A **<Laufwerk>** NOBOOT/S CHECK/S

dient dazu, eine startfähige Diskette (Workbench-Diskette) zu erstellen.

Der Befehl INSTALL ist nur für fortgeschrittene Anwender gedacht. Mit ihm werden einige der Informationen auf die Diskette im Laufwerk <Laufwerk> (zum Beispiel DF0: oder DF1:) geschrieben, die der Amiga beim Start oder zum Neustart benötigt. Es handelt sich dabei um den sogenannten »Bootblock«.

Eine solche Diskette wird vom Amiga 500 dann akzeptiert, wenn er beim Start eine »Workbench-Diskette« verlangt. Falls nicht noch weitere Vorkehrungen getroffen werden, ist die so entstehende Diskette aber reichlich unbrauchbar. Zu diesen Vorkehrungen gehört das Einrichten einer Reihe von Directories auf dieser Diskette (zum Beispiel *devs*, *fonts* und so weiter) sowie das Füllen dieser Directories mit Dateien. Die einfachste Methode, sich eine Startdiskette zu erzeugen, ist, eine komplette Diskette mit dem Befehl DISKCOPY zu duplizieren und dann die Dateien, die man nicht benötigt, zu löschen und gegebenenfalls andere darauf zu legen.

Wenn Sie allerdings stolzer Besitzer einer Festplatte sind, von der aus der Amiga starten kann, einer sogenannten »autobootfähigen« Platte, müssen Sie die Platte nach dem Formatieren und Initialisieren (siehe FORMAT) mit Hilfe des INSTALL-Befehls startfähig machen.

Normalerweise wird INSTALL nur mit einem Parameter (einem Laufwerksnamen) aufgerufen und erzeugt eine startfähige Diskette. Mit den beiden Schlüsselwörtern NOBOOT und CHECK, die selten gebraucht werden, können Sie das Verhalten des Befehls abändern.

Wenn Sie das Schlüsselwort NOBOOT verwenden, erhalten Sie zwar eine startfähige Diskette. Diese führt aber nach dem Start nicht die Startup-Sequence aus. Es erscheint zwar ein Shell-Fenster (genauer gesagt ein CLI-Fenster) aber die Workbench wird nicht aktiviert. Da auch die logischen Laufwerke und der Suchpfad nicht installiert werden, ist auch ein Arbeiten mit CLI-Befehlen fast nicht möglich.

Das Schlüsselwort CHECK schreibt keinen Bootblock auf die gewünschte Diskette, sondern überprüft statt dessen den Bootblock auf der Diskette. Dabei werden drei Fälle unterschieden und entsprechende Meldungen ausgegeben. Wenn die Diskette einen normalen Bootblock enthält, liefert INSTALL den Returncode 0. Ist kein Bootblock vorhanden, liefert INSTALL den Returncode 5. Ist ein Bootblock vorhanden, der nicht dem Standard entspricht, liefert INSTALL den Returncode 10. Letzteres ist ein Indiz dafür, daß die Diskette entweder defekt ist, oder daß sich ein Computer-Virus darauf eingeschlichen hat.

**Beispiele:**

```
INSTALL DF0:
INSTALL DF0: CHECK
```

Der erste Befehl schreibt einen Bootblock auf die Diskette im internen Laufwerk. Der Amiga akzeptiert diese Diskette dann, wenn er nach dem Einschalten zum Einlegen einer Workbench-Diskette auffordert. Sofern sich auf dieser Diskette aber nicht die auf einer Workbench-Diskette notwendigerweise vorhandenen Dateien befinden, sollten Sie mit dieser Diskette aber besser nicht starten.

Der zweite Befehl überprüft den Bootblock der Diskette im internen Laufwerk. INSTALL meldet, ob sich ein Bootblock auf dieser Diskette befindet und ob dieser »virusverdächtig« ist.

# JOIN

**JOIN** <Name1> <Name2> ... <name15> AS/K <Name>

verbindet bis zu 15 Dateien zu einer neuen Datei.

Der Befehl JOIN (engl. *to join* = verbinden, verknüpfen) dient zum Aneinanderhängen von Dateien. Dies kann aus den verschiedensten Gründen heraus wichtig sein. Eine solche Verknüpfung funktioniert meist nur bei Dateien, die aus einer Folge gleichförmiger Daten bestehen. Textdateien können zum Beispiel ohne weiteres verknüpft werden. Mit JOIN kann man aus mehreren kurzen Textdateien eine lange machen. Bei Dateien mit komplexem Format, wie zum Beispiel Bildern, ist ein JOIN aber meist nicht sinnvoll. Die resultierenden Dateien können nämlich von keinem Programm mehr gelesen werden.

Beachten Sie bitte, daß die neue Datei nicht denselben Namen tragen kann wie eine der darin zu verschmelzenden Dateien!

**Beispiel:**

```
JOIN Brief1 Brief2 "Brief 3" AS Briefe
```

Dieser Befehl erzeugt eine neue Datei namens *Briefe* im aktuellen Directory, in der hintereinander alle Daten aus den Dateien *Brief1*, *Brief2* und *Brief3* stehen. Die Dateien *Brief1*, *Brief2* und *Brief3* werden dabei nicht verändert.

# LAB

**LAB <Name>**

setzt eine Sprungmarke in einer Kommandofolge.

Der Befehl LAB markiert die Zeile einer Kommandofolge, an der er steht, als das mögliche Ziel eines entsprechenden SKIP-Befehls. Stößt die Shell in einer Kommandofolge auf den Befehl SKIP <Name>, so werden alle Befehle übersprungen, bis eine Zeile mit dem Befehl LAB <Name> oder ENDSKIP gefunden wird.

Mehr zum LAB-Befehl können Sie im Kapitel 8 erfahren.

# LIST

**LIST** DIR <Name>*
PAT/K <Muster> P/K <Muster>
KEYS/S DATES/S NODATES/S
TO/K <Datei> SUB/K <suchtext>
SINCE/K <Datum UPTO/K <Datum2>
QUICK/S BLOCK/S NOHEAD/S FILES/S DIRS/S
LFORMAT/K <Ausgabemuster>

gibt eine Vielzahl von Informationen über eine Datei oder eine Gruppe von Dateien aus.

Der Befehl LIST dürfte der Shell-Befehl mit den meisten möglichen Parametern und Schlüsselwörtern sein. Er arbeitet grundsätzlich ähnlich wie der DIR-Befehl, kann aber nicht nur Namen, sondern auch sehr ausführliche Informationen über eine Datei anzeigen. Dazu gehören Name und Größe (in Bytes) der Datei, eventuelle Schutzflags, das Datum und die Uhrzeit der letzten Modifikation und schließlich der Kommentar (siehe auch den Shell-Befehl FILENOTE). Verschiedene Schlüsselwörter steuern, welche Informationen ausgegeben und für welche Dateien sie ausgegeben werden.

Gibt man LIST ohne den ersten Parameter <Name> an, so werden Informationen über alle Dateien im aktuellen Directory aufgelistet (sofern nicht andere Optionen diese Menge einschränken). Ist <Name> ein Dateiname, so werden nur Informationen über diese Datei ausgegeben. Ist <Name> der Name eines Directory, so werden Informationen über alle Dateien dieses Directory aufgelistet (sofern nicht andere Optionen diese Menge einschränken).

Statt <Name> kann auch ein Muster angegeben werden, dann werden nur über die Dateien Informationen ausgegeben, deren Namen in das Muster passen. Dem selben Zweck dient auch das Schlüsselwort PAT (oder P) und das Muster dahinter. Dieses Schlüsselwort hat lediglich historische Bedeutung, da frühere Versionen des LIST-Befehls ein Muster nur dann akzeptierten, wenn dafür das Schlüsselwort PAT kam. Beim heutigen LIST-Befehl können Sie dieses Schlüsselwort fortfallen lassen. Das Schlüsselwort SUB und der Parame-

ter <Suchtext> ist eine vereinfachte Fassung eines Musters. Wenn Sie diesen Parameter (und davor SUB) angeben, werden nur Dateien aufgelistet, in deren Name <Suchtext> vorkommt.

Gibt man das Schlüsselwort KEYS mit an, so erhält man eine technische Information zu jeder Datei, mit der man defekte Dateien eventuell wiederherstellen kann. Das Schlüsselwort NODATES unterdrückt die Ausgabe von Datums- oder Uhrzeit-Informationen zu den Dateien. Das Schlüsselwort QUICK bewirkt, daß nur noch die Namen der Dateien und Directories ausgegeben werden und reduziert deshalb LIST weitgehend auf die Funktion, die auch der DIR-Befehl erfüllt. Möchte man zusätzlich zu den Namen auch noch die Datumsinformationen sehen, kann man zusammen mit QUICK das Schlüsselwort DATES verwenden.

Das Schlüsselwort FILES bewirkt, daß nur Dateinamen ausgegeben werden. Das Schlüsselwort DIRS schränkt die Liste auf Directories ein.

Das Schlüsselwort BLOCK bewirkt, daß die Dateigrößen statt in Kbyte in Blocks (zu 512 Byte) angegeben werden.

LIST beginnt die Ausgaben der Informationen normalerweise mit einer Kopfzeile, die den Namen des aufgelisteten Directorys und das heutige Datum enthält. Am Ende der Liste wird dann noch einmal zusammenfaßt, wieviele Dateien aufgelistet wurden und wieviel Platz diese auf dem Datenträger beanspruchen. Wenn Sie diese beiden Zeilen unterdrücken wollen, müssen Sie das Schlüsselwort NOHEAD angeben.

Mit der Option TO <Datei> können Sie die Liste, die LIST normalerweise auf den Bildschirm ausgeben würde, in eine Datei schicken, um sie dort zu betrachten oder zu bearbeiten.

Mit den Schlüsselwörtern SINCE und UPTO kann die Menge der Dateien aus einem Directory, über die Informationen ausgegeben werden, nach dem Datum der letzten Modifikation eingeschränkt werden. SINCE <Datum1> bewirkt, daß nur Dateien aufgelistet werden, die seit <Datum1> noch modifiziert wurden. UPTO <Datum2> schränkt die Ausgabe auf Dateien ein, deren letzte Modifikation vor <Datum2> stattgefunden hat.

Eine sehr interessante Form von LIST ergibt sich aus der Verwendung des Parameters LFORMAT. Mit LFORMAT können Sie das Aussehen der Liste, die LIST aufgibt selbst bestimmen. Wird LFORMAT <Ausgabemuster> angegeben, gibt LIST für jede Datei und jedes Directory, das aufgrund der anderen Angaben in der Liste berücksichtigt wird, eine Zeile aus. Diese Zeile enthält nur den Text <Ausgabemuster>, wobei allerdings die Buchstabenkombination »%S«, sofern Sie in <Ausgabemuster> vorkommt, durch den jeweiligen Dateinamen ersetzt wird. <Ausgabemuster> darf auch das von ECHO bekannte Steuerzeichen »*N« (Zeilenvorschub) enthalten. Sie können auf diese Weise mehrere Zeilen pro Dateiname erzeugen lassen.

Kommt %S mehrfach in <Ausgabemuster> vor, wird das erste Vorkommen von %S durch den Pfadnamen des Directory, in dem die Datei liegt, ersetzt, das zweite durch den Dateinamen, das dritte wieder gegen den Directorynamen und das vierte wieder gegen den Dateinamen. Das hört sich zunächst vielleicht etwas merkwürdig an. In den Beispielen werden Sie aber gleich erfahren, wie Sie diesen Parameter sinnvoll einsetzen können.

**Beispiele:**

```
LIST SINCE 01-Jan-89 TO DirListe QUICK NOHEAD
LIST C: S ED
```

Der erste Befehl erzeugt eine Datei namens *Dirliste* im aktuellen Directory, die eine Liste aller Dateinamen enthält, die nach dem 1. Januar 1989 noch geändert wurden. Außer den Namen werden keine Informationen abgespeichert. Auch die erste und letzte Zeile, die LIST normalerweise am Bildschirm ausgibt, wird unterdrückt. Der zweite Befehl listet Namen und einige weitere Informationen über alle Dateien im logischen Laufwerk c: auf, die in ihrem Namen den Suchtext »ed« enthalten.

```
1.SYS:>LIST C:I#? LFORMAT "RENAME %S%S TO %S%S.BU"
RENAME C:Install TO C:Install.BU
RENAME C:Info TO C:Info.BU
RENAME C:If TO C:If.BU
RENAME C:IconX TO C:IconX.BU
```

Dieser LIST-Befehl erzeugt durch die geschickte Verwendung des Schlüsselworts LFORMAT eine Reihe von Shell-Befehlen. Für jede Datei, deren Name in das Muster »C:I#?« paßt, wird ein solcher Befehl erzeugt. Dabei wird ausgenutzt, daß die Buchstabenkombination %S%S immer gegen den vollen Pfadnamen einer Datei ersetzt wird.

Diese Anwendung des LIST-Befehls mag auf den ersten Blick nicht als besonders sinnvoll erscheinen. Sie wird es aber spätestens dann, wenn man sich klarmacht, daß man durch Ausgabeumlenkung auf diese Weise automatisch eine Kommandodatei erzeugen kann. Der eben vorgestellte LIST-Befehl könnte zum Beispiel eine Kommandofolge erzeugen, die automatisch alle Dateien umbenennt, deren Name in ein bestimmtes Muster paßt. Der RENAME-Befehl kann aber normalerweise immer nur eine Datei bearbeiten. Die Vorteile der Muster können auf diesem Umweg auch den Shell-Befehlen zugute kommen, die selbst mit Mustern nichts anfangen können.

# LOADWB

### LOADWB

dient zum Aktivieren der Workbench, falls sie noch nicht aktiv ist.

Unter normalen Umständen wird beim Starten des Amiga die Workbench aktiv und bleibt immer aktiv, auch während andere Programme gestartet werden und laufen. Das Starten der Workbench geschieht durch den Befehl LOADWB in der Kommandofolge Startup-Sequence, die bei jedem Neustart automatisch abläuft.

Da der Befehl LOADWB nur einmal aufgerufen werden muß und das üblicherweise schon in der Startup-Sequence passiert, ist ein Aufruf von einem Shell-Fenster aus äußerst selten.

# LOCK

**LOCK** DRIVE/A **<Laufwerk>** ON/S OFF/S PASSKEY <Paßwort>

Aktiviert oder deaktiviert den Schreibschutz einer Festplatte.

Eine Diskette kann man vor versehentlichem Überschreiben schützen, indem man ein kleines Plastikplättchen in einer Ecke des Diskettengehäuses verschiebt. Bei Festplatten gibt es einen solchen mechanischen Schreibschutz nicht.

Festplatten können dafür mit dem Befehl LOCK schreibgeschützt werden. Dieser Schreibschutz funktioniert allerdings nur mit Festplatten, die mit dem neuen FastFileSystem formatiert wurden (siehe FORMAT). Darüber hinaus ist der Schreibschutz auch recht flüchtig. Er geht nämlich bei jedem Neustart und beim Ausschalten des Amiga verloren.

Trotzdem kann LOCK speziell für Softwareentwickler ein sehr wichtiger Befehl sein. Wenn die Festplatte schreibgeschützt wird, bevor man ein neues noch unausgetestetes Programm startet, kann dieses Programm bei einem Fehler keine Dateien mehr löschen (oder gar die Platte neu formatieren).

Mit den Schlüsselwörtern ON und OFF wird der Schreibschutz jeweils ein- beziehungsweise ausgeschaltet. Gibt man LOCK <Laufwerk> ganz ohne weitere Schlüsselwörter an, wird der Schreibschutz eingeschaltet. Der letzte Parameter ist ein maximal vier Buchstaben langes Paßwort. Wird dieses beim Einschalten des Schreibschutzes angegeben, muß beim Abschalten des Schreibschutzes exakt das selbe Paßwort angegeben werden. Sonst weigert sich LOCK, den Schreibschutz abzuschalten.

**Beispiele:**

```
LOCK DH1:
LOCK DH2: ON "ABCD"
```

Der erste Befehl aktiviert den Schreibschutz der Festplatte DH1:. Der zweite aktiviert den Schreibschutz der Festplatte DH2: und vereinbart gleichzeitig ein Paßwort. Dieser Schreibschutz läßt sich nur mit dem folgenden Befehl wieder aufheben:

```
LOCK DH2: OFF "ABCD"
```

LOCK funktioniert nur bei Festplatten, die mit dem neuen FastFileSystem formatiert wurden!

# MAKEDIR

**MAKEDIR** DIR/A **<Name>**

dient zum Anlegen eines neuen Directory.

Der Befehl MAKEDIR legt ein neues Directory namens <Name>. Wenn <Name> ein vollständiger Pfadname ist, kann das neue Directory auch in einem anderen als dem aktuellen Directory liegen. Das Directory, in dem das neue Directory liegen soll, muß bereits

existieren. Es ist unzulässig für <Name> »Dir1/Dir2« anzugeben, wenn es noch kein Directory »Dir1« gibt.

Directories können genau wie Dateien mit dem Befehl DELETE gelöscht werden. Sie müssen dazu aber vollständig leer sein!

Das Anlegen eines neuen Directorys mit MAKEDIR führt im allgemeinen nicht dazu, daß auch eine neue Schublade auf der Workbench sichtbar wird. Directories werden erst durch eine eigene Info-Datei auch auf der Workbench sichtbar.

**Beispiele:**

```
MAKEDIR NeuDir
MAKEDIR :c/selten
```

Der erste Befehl legt ein neues Directory namens *NeuDir* an, das im aktuellen Directory liegt. Der zweite Befehl legt ein neues Directory namens *selten* im Directory *c* auf der aktuellen Diskette an (das Directory *c* muß dazu schon existieren).

# MEMACS

**MEMACS** <Name> GOTO/K <Zeile> OPT/K <Option>

dient zum Anlegen und Modifizieren von Dateien, die druckbare Texte enthalten.

Der Befehl MEMACS entspricht dem Werkzeug *MEmacs* in der Schublade *Tools* der Extras-Diskette. MEMACS ist neben ED und EDIT der dritte und komfortabelste Editor, der dem Amiga beim Kauf beiliegt. Mehr über die Bedienung dieses Text-Editors erfahren Sie im folgenden Kapitel.

Wenn Sie MEMACS ohne Parameter aufrufen, wird das Programm gestartet und erscheint auf dem Bildschirm. Sie können dann Texte einlesen, ändern und wieder abspeichern. Wenn Sie den Parameter <Name> angeben, wird diese Datei gleich nach dem Start geladen und Ihnen am Bildschirm zur Bearbeitung präsentiert. Gibt es <Name> noch nicht, erzeugt MEMACS automatisch eine neue Datei.

Wird der Parameter <Name> angegeben, können Sie dafür sorgen, daß die Schreibmarke gleich nach dem Start in einer bestimmten Zeile des eingelesenen Textes steht. Geben Sie dazu das Schlüsselwort GOTO und dahinter die Nummer der gewünschten Zeile an.

MEMACS öffnet, sofern Sie nichts anderes angeben, einen eigenen Screen zur Bearbeitung des Textes. Dieser füllt den ganzen Bildschirm und ist deshalb ideal zur Bearbeitung großer Texte mit breiten Zeilen. Ein Screen kostet aber relativ viel RAM-Speicher (unabhängig von den darin dargestellten Texten und/oder Grafiken). Wenn Sie deshalb die Benutzung eines Fensters auf dem Workbench-Screen vorziehen, müssen Sie beim Aufruf von MEMACS das Schlüsselwort OPT und dahinter ein W angeben.

**Beispiele:**

```
MEMACS
MEMACS Hello.c GOTO 12
MEMACS Startup.sequence OPT W
```

Der erste Befehl startet MEMACS. Das Programm öffnet daraufhin einen eigenen Screen. In diesem befindet sich kein Text. Wenn Sie einen vorhandenen Text bearbeiten wollen, müssen Sie diesen dann mit einem Programmbefehl einlesen.

Der zweite Befehl startet MEMACS und sorgt dafür, daß gleich nach dem Start die Datei »Hello.c« eingelesen und die Schreibmarke in die vierte Zeile dieses Textes gesetzt wird. Der dritte Befehl sorgt ebenfalls dafür, daß gleich nach dem Start die Datei Hello.c eingelesen wird. Er öffnet aber keinen neuen Screen zur Darstellung und Bearbeitung des Textes, sondern statt dessen ein Fenster auf dem Workbench-Screen. Dieses Fenster ist zunächst bildschirmfüllend, kann aber verkleinert und verschoben werden.

# MERGEMEM

**MERGEMEM**

Der Befehl MERGEMEM entspricht dem Werkzeug *MergeMem* in der Schublade *System* der Workbench-Diskette. MERGEMEM versucht, die in den verschiedenen Speichererweiterungen liegenden RAM-Speicher zu einem großen zusammenhängenden Speicherbereich zusammenzufassen. Danach kann ein Programm einen Speicherbereich benutzen, der so groß ist, daß er sich über mehrere Speichererweiterungen erstreckt. (Ohne Aufruf von MERGEMEM ist das nicht möglich.)

Da aber die wenigsten Amiga-500-Besitzer über mehrere Speicherweiterungen verfügen dürften und darüber hinaus wahrscheinlich auch kaum Programme verwenden werden, die solche riesigen Speicheranforderungen stellen, werden Sie MERGEMEM wahrscheinlich nie benötigen. MERGEMEM ist mehr für den Amiga 2000 oder Amiga 2500 gedacht.

# MORE

**MORE <Dateiname>**

dient zur schrittweisen Ausgabe einer Textdatei am Bildschirm mit der Möglichkeit, vor- oder zurückzublättern.

Der Befehl MORE entspricht dem Werkzeug *More* in der Schublade *Utilities* der Workbench-Diskette. Eine ausführliche Beschreibung dieses Programms finden Sie in Abschnitt 5.2.6 dieses Buches.

Ein wichtiger Befehl von MORE wurde allerdings in Kapitel 5 noch nicht erwähnt – weil er nur mit Grundkenntnissen der Shell zu verstehen ist. Wenn Sie sich eine Datei betrachten, können Sie diese auch gleich ändern, indem Sie ein großes »E« eingeben. More startet dann einen Texteditor und gibt diesem den Befehl, die entsprechende Datei zu öffnen. Nach dem Verlassen des Editors kehren Sie nach MORE zurück.

Damit diese Funktion genutzt werden kann, müssen Sie zuvor die Umgebungsvariable EDITOR definieren (siehe SETENV). SETENV muß den vollständigen Pfadnamen des gewünschten Editors enthalten (zum Beispiel *:Utilities/Notepad* oder *C:ED*).

# MOUNT

**MOUNT** DEVICE/A **<Gerätename>** FROM/K <Dateiname>

dient zum »Anmelden« eines Gerätes beim Amiga-DOS, das durch einen zusätzlichen »Gerätetreiber« angesprochen wird, der nicht zur Standardausstattung der Workbench-Diskette gehört.

Direkt nach dem Start kennt Amiga-DOS (und damit auch jedes andere Amiga-Programm) nur eine kleine Anzahl von Geräten. Dazu gehören unter anderem die Diskettenlaufwerke DF0: bis DF3:, PRT:, SER:, PAR:, CON:, RAW: und RAM:. Jedes zusätzliche Gerät müssen Sie mit dem Befehl MOUNT bei Amiga-DOS »anmelden«. (Einige zusätzliche Geräte wie PIPE: und NEWCON: werden allerdings schon in der Startup-Sequence angemeldet.)

Die Angaben über die Eigenschaften des neuen Geräts entnimmt der Befehl MOUNT der Datei *mountlist* im logischen Laufwerk DEVS: oder – wenn das Schlüsselwort FROM verwendet wird, der Datei <Dateiname>.

Der Inhalt der Datei *mountlist* ist nur für sehr fortgeschrittene Benutzer verständlich. Sie benötigen dazu Informationen, die ich Ihnen im Rahmen dieses Buches nicht vermitteln kann. Der Hersteller eines Gerätes, das mittels MOUNT angemeldet werden muß, wird in der Regel aber eine korrekte *mountlist* und weitere Dateien, die zur Gerätesteuerung eventuell benötigt werden, mitliefern.

MOUNT steht – wie der Befehl BINDDRIVERS, der oft mit MOUNT zusammenarbeitet – üblicherweise in der Startup-Sequence und wird nicht direkt im Shell-Fenster eingegeben.

**Beispiele:**

```
MOUNT RAD: FROM DEVS:Rad2
DIR RAD:
```

Diese beiden Befehle aktivieren die (»resetfeste«) RAM-Disk RAD:, deren Inhalt auch einen Neustart (nicht aber das Ausschalten) überlebt. Die Beschreibung der Eigenschaften von RAD entnimmt der Befehl MOUNT der Datei *Rad2* im logischen Laufwerk DEVS:.

# NEWCLI

**NEWCLI** WINDOW <Fenster> FROM/K <Dateiname>

dient zum Erzeugen eines neuen CLI-Prozesses in einem neuen Fenster.

Der Befehl NEWCLI startet – sofern noch genügend Speicherplatz frei ist – einen neuen Prozeß, in dem auch das CLI läuft. Dieser Prozeß erhält ein eigenes Fenster, in dem sich die Kommunikation mit dem Benutzer abspielt.

Ein solches CLI-Fenster entspricht in etwa einem Shell-Fenster, das mit dem Befehl NEWSHELL geöffnet wird. In Kapitel 6 wurden die Unterschiede zwischen CLI und Shell beschrieben.

NEWCLI verwendet, wenn im Parameter <Fenster> nichts anderes festgelegt wird, das Gerät CON: für Eingaben und Ausgaben in dem neuen Fenster. (NEWSHELL verwendet standardmäßig NEWCON:) Nachdem das neue Fenster geöffnet wurde, wird automatisch die Datei *CLI-Startup* im logischen Laufwerk S: als Kommandofolge ausgeführt. Wenn Sie das nicht wollen, müssen Sie hinter dem Schlüsselwort FROM eine andere Kommandofolge <Dateiname> angeben, die statt dessen ausgeführt werden soll.

Weitere Informationen zu diesem Befehl (unter anderem auch über den Aufbau des Parameters <Fenster>) können Sie der Beschreibung des Befehls NEWSHELL auf der folgenden Seite entnehmen.

# NEWSHELL

**NEWSHELL** WINDOW <Fenster> FROM/K <Dateiname>

dient zum Erzeugen eines neuen Shell-Prozeses in einem neuen Fenster.

Der Befehl NEWSHELL startet – sofern noch genügend Speicherplatz frei ist – einen neuen Prozeß, in dem auch die Shell läuft. Dieser Prozeß erhält ein eigenes Fenster, in das Sie Shell-Befehle eingeben, korrigieren und die Ergebnisse der Befehle betrachten können.

Wenn der Parameter <Fenster> nicht angegeben wird, öffnet NEWSHELL ein Fenster in einer vorbestimmten Größe und verwendet für die Ein- und Ausgaben in diesem Fenster das Gerät NEWCON:. Dazu muß das Gerät NEWCON: allerdings zuvor mit MOUNT angemeldet und mit dem Befehl RESIDENT die NEWCON-Software resident gemacht werden. Beides geschieht üblicherweise in der Startup-Sequence.

Ist eine dieser Bedingungen nicht erfüllt, verwendet NEWSHELL für die Ein- und Ausgaben im neuen Fenster das Gerät CON:, das nicht so komfortable Korrekturmöglichkeiten bietet wie NEWCON:.

Nachdem das neue Fenster geöffnet wurde, wird automatisch die Datei *Shell-Startup* im logischen Laufwerk S: als Kommandofolge ausgeführt. Wenn Sie das nicht wollen, müssen Sie hinter dem Schlüsselwort FROM eine andere Kommandofolge <Dateiname> angeben, die statt dessen ausgeführt werden soll.

Form und Eigenschaften des neuen Fensters können Sie auf Wunsch mit dem Parameter <Fenster> festlegen. <Fenster> ist ein Text in der Form

```
CON:<PosX>/<PosY>/<Breite>/<Höhe>/<Titel>       oder
NEWCON:<PosX>/<PosY>/<Breite>/<Höhe>/<Titel>.
```

Dabei bestimmt der erste Teil des Textes, ob das Gerät CON: oder NEWCON: für Ein- und Ausgaben verwendet werden soll. <PosX>, <PosY>, <Breite>, <Höhe>, <Titel> legen die Position des rechten Fensterrands, die Position des linken Fensterrands, die Breite, die Höhe und den Titel des gewünschten Fensters fest.

Fortgeschrittene Benutzer können aber auch mit NEWSHELL AUX: einen neuen Shell-Prozeß starten, der seine Eingaben über die serielle Schnittstelle erwartet und seine Ausgaben auch an die serielle Schnittstelle schickt. Man kann auf diese Weise zum Beispiel von einem anderen Computer aus, der an die serielle Schnittstelle angeschlossen ist, den Amiga steuern (allerdings nur soweit das mit Shell-Befehlen möglich ist).

**Beispiel:**

```
NEWSHELL "NEWCON:10/30/600/100/Neues Shell-Fenster" FROM MyStart
```

Dieser Befehl öffnet ein neues Shell-Fenster mit festgelegter Größe und dem Namen *Neues Shell-Fenster*. (Bei Namen mit Leerstellen müssen Anführungszeichen verwendet werden.) Für Ein- und Ausgaben in diesem Fenster wird das Gerät NEWCON: verwendet. Gleich nach dem Öffnen des Fensters wird in diesem die Kommandofolge *MyStart* ausgeführt.

# NOFASTMEM

RUN **NOFASTMEM**

dient dazu, allen RAM-Speicher außerhalb des Chip-RAM zu sperren.

Der Befehl NOFASTMEM entspricht dem Programm *NoFastMem* in der Schublade *System* der Workbench-Diskette. Mit NOFASTMEM können Sie einige alte Programme auf Ihrem Amiga 500 benutzen, die zu einer Zeit entwickelt wurden, als es nur den Amiga 1000 gab. Der Amiga 1000 verfügte zunächst nur über 512 Kbyte RAM-Speicher. Später kamen Speichererweiterungen auf den Markt, die ihn oder auch seine Nachfolgermodelle mit wesentlich mehr RAM-Speicher ausstatteten. Dieser zusätzliche Speicher ist aber kein »Chip-RAM« und kann von einigen der Amiga-Spezial-Chips nicht angesprochen werden. Er wird auch »FastMem« (schneller Speicher) genannt. Die meisten Amiga-Modelle verfügen über 512 Kbyte Chip-RAM und verschiedene Mengen FastMem. Die neuesten Modelle des Amiga 500, 2000 und 2500 können auch 1 Mbyte Chip-RAM enthalten.

Clevere Programme kennen die Unterschiede zwischen den beiden Speichersorten und fordern je nach Bedarf einen Speicherblock der korrekten Sorte. Einige sehr alte Programme sind jedoch so programmiert worden, als gäbe es nur Amiga-Modelle mit 512 Kbyte RAM.

Solche Programme laufen oft auf dem Amiga 500 dann nicht mehr, wenn Sie die Speichererweiterung A 501 oder eine vergleichbare Speichererweiterung eines anderen Herstellers

eingebaut haben. Mit NOFASTMEM können Sie allen RAM-Speicher außerhalb des Chip-RAM sperren und dann ein solches »Oldtimer-Programm« benutzen. Der zweite Aufruf von NOFASTMEM gibt den Speicher danach wieder frei.

Es empfiehlt sich NOFASTMEM, über den RUN-Befehl als separaten Prozeß zu starten, da dieser Befehl nicht automatisch beendet wird. Wenn Sie NOFASTMEM einmal mit RUN gestartet haben, können Sie es durch erneuten Aufruf von NOFASTMEM (ohne RUN) oder mit dem BREAK-Befehl beenden.

# PATH

**PATH** SHOW/S ADD <name <name2> ... <name10> RESET/S QUIET/S

ient zum Ändern des Suchpfades.

Die geschickte Verwendung des Befehls PATH ist sehr wichtig, damit Sie im Shell-Fenster oroblemlosen Zugriff auf möglichst viele Programme in verschiedenen Directories haben. PATH verwaltet eine Liste von Directories. Das erste Directory in dieser Liste ist immer das iktuelle Directory, das letzte ist das logische Laufwerk C:. Dazwischen können beliebige indere Directories liegen. Welche Directories der Pfad gerade enthält, können Sie sich inzeigen lassen, indem Sie PATH ohne alle Parameter oder nur mit dem Schlüsselwort SHOW (engl. *to show* = zeigen) verwenden. Mit dem Schlüsselwort ADD oder auch ohne lieses optionale Schlüsselwort können Sie bis zu zehn weitere Directories in den Suchpfad einhängen. Diese werden danach vor allen anderen schon im Pfad liegenden Directories aber nach dem aktuellen) nach Programmen durchsucht.

Etwas problematisch ist die Behandlung von Directories im Suchpfad, die sich auf Disketten befinden, die zur Zeit in keinem Laufwerk eingelegt sind. Wenn Sie sich mit PATH den Suchpfad zeigen lassen, werden diese anderen Disketten angefordet, da PATH prüfen will, ob die entsprechenden Directories noch erreichbar sind. Wenn Sie das vermeiden wollen um nicht Disketten wechseln zu müssen) können Sie die Schlüsselwörter SHOW und QUIET zusammen verwenden. SHOW allein bewirkt lediglich, das PATH nicht auf den Disketten nachprüft, ob die Directories noch vorhanden sind. Das Vorhandensein der Disketten selbst wird auch bei Verwendung von SHOW geprüft (wodurch Sie unter Umständen Disketten wechseln müssen).

Wenn Sie das Schlüsselwort RESET (engl. *to reset* = zurücksetzen) verwenden, wird der Suchpfad in seinen »Urzustand« zurückversetzt. Er enthält dann nur noch das aktuelle Directory und das logische Laufwerk C:.

**Beispiele:**

```
PATH Briefe :Romane ADD
PATH DF0:Utilities DF0:System RESET
PATH
PATH SHOW QUIET
```

Der erste Befehl verlängert den Suchpfad um das Directory *Briefe* im aktuellen Directory und um das Directory *Romane* im Wurzeldirectory der aktuellen Diskette. Der zweite Befehl löscht alle Directories außer C: und dem aktuellen Directory aus dem Suchpfad und fügt dann die beiden Directories *DF0:Utilities* und *DF0:System* in den neuen Suchpfad ein. Der Befehl PATH allein listet den aktuellen Suchpfad auf und versucht, auf jedes der Directories zuzugreifen. Der letzte Befehl hingegen listet den Suchpfad nur auf.

# PREFERENCES

**PREFERENCES** POINTER/S PRINTER/S SERIAL/

dient zur Angabe wichtiger Voreinstellungen für die Arbeit mit dem Amiga.

Der Befehl PREFERENCES entspricht dem Programm *Preferences* in der Schublade *Prefs* der Workbench-Diskette. Dieses Programm wurde bereits ausführlich in Kapitel 5 beschrieben. Mit dem Shell-Befehl PREFERENCES haben Sie die Möglichkeit, nach dem Start erst das Hauptfenster gezeigt zu bekommen oder gleich in drei verschiedene Bereiche des Programms zu gelangen. Dazu dienen die drei Schlüsselwörter POINTER, PRINTER und SERIAL. POINTER führt zu dem Fenster, in dem Sie den Mauszeiger gestalten können, PRINTER zum Druckereinstellungsfenster und SERIAL zu dem Fenster, in dem Sie die Parameter der seriellen Schnittstelle verändern können.

# PRINTFILES

**PRINTFILES** –F/S <Dateiname1> –F/S <Dateiname2> ... –F/S <DateinameN>

Der Befehl PRINTFILES entspricht dem Programm *Printfiles* in der Schublade *Utilities* der Workbench-Diskette. Seine Verwendung von der Workbench aus ist aber relativ umständlich und wurde in Kapitel 5 beschrieben.

PRINTFILES kann mehrere (fast beliebig viele) Textdateien ausdrucken, deren Namen dem Befehl als Parameter folgen müssen. Vor jedem dieser Namen kann das Schlüsselwort –F stehen. Es bewirkt, das nach dem Ausdrucken der jeweiligen Datei ein Formfeed-Befehl an den Drucker gesandt wird (der einen Seitenvorschub zur Folge haben sollte.

**Beispiel:**

```
PRINTFILES -F Text1 Text2 -F Text3
```

Dieser Befehl druckt drei Textdateien aus. Zwischen der ersten und der zweiten sowie nach Beendigung der dritten Datei wird ein Seitenvorschub verursacht.

# PROMPT

**PROMPT** <Text>

legt das Aussehen der Eingabeaufforderung im Shell-Fenster fest.

Der Befehl PROMPT bestimmt, daß die Shell jedesmal dann, wenn sie bereit ist, den nächsten Befehl auszuführen, den Text <Text> ausgibt. <Text> kann die bei ECHO beschriebenen Sonderzeichen zur Gestaltung der Textausgabe enthalten (zum Beispiel *N für einen Zeilenvorschub) und darüber hinaus auch noch die beiden speziellen Buchstabenkombinationen »%N« und »%S«. Diese werden vor der Ausgabe von <Text> gegen die Prozeßnummer des jeweiligen Shell-Prozesses beziehungsweise den Namen des aktuellen Directory ersetzt.

**Beispiele:**

In den folgenden Beispielen werden die Eingaben des hypothetischen Anwenders durch Fettdruck hervorgehoben. Die Beispiele sollten ansonsten selbsterklärend sein.

```
1.WB 1.3:c>PROMPT "Deine Befehle Herr:"
Deine Befehle Herr:PROMPT "(%N)Was is ? "
(1)Was is? PROMPT "Dir %S|"
Dir WB 1.3:c|PROMPT "Prozeß %N*N%S>"
Prozeß 1
WB 1.3:c>
```

# PROTECT

**PROTECT** FILE/A **<Dateiname>** FLAGS +/S –/S <Flags> ADD/S SUB/S

dient zum Ändern der Schutzflags einer Datei.

Das Betriebssystem des Amiga, Amiga-DOS, verwaltet zu jeder Datei acht sogenannte »Schutzflags« (Schalter), die einen Mißbrauch der Datei möglichst schwierig machen. Jedes dieser Schutzflags kann »gesetzt« (An) oder Aus sein und erlaubt oder verhindert bestimmte Operationen mit der Datei. Die derzeit benutzten sieben Flags sind:

r: die Datei kann gelesen werden (r für *read* = lesen).

w: die Datei kann geändert werden (w für *write* = schreiben).

d: die Datei kann gelöscht werden (d für *delete* = löschen).

e : die Datei kann als Programm ausgeführt werden (e für *execute* = ausführen).

s : die Datei ist eine Kommandofolge (s für *script* = Kommandofolge).

p : die Datei ist ein Programm, daß gleichzeitig von mehreren Prozessen ausgeführt werden kann (p für *pure* = reiner multiuserfähiger, reentrant Code).

a : die Datei ist gesichert worden (a für *archive* = Sicherung/Archiv).
Ein Schreibzugriff auf diese Datei setzt dieses Flag wieder auf Aus.

h : die Datei ist unsichtbar (h für *hidden* = verborgen).

Die vier Flags r, w, d und e sind bei einer neuen Datei zunächst einmal alle gesetzt. Jede Datei kann also zu Beginn ihrer Existenz gelesen, geändert, gelöscht und ausgeführt werden. Die vier restlichen Flags sind aus und müssen explizit mit dem PROTECT-Befehl gesetzt werden, wenn ihre Funktion benötigt wird.

Mit PROTECT (engl. *to* protect = schützen) können Sie einzelne oder mehrere Schalter auf einmal setzen oder ausschalten. Dazu müssen die Namen dieser Flags demBefehl als zweiter Parameter folgen. Setzen Sie vor diese Namen ein Pluszeichen oder das Schlüsselwort ADD, werden die entsprechenden Flag gesetzt. Verwenden Sie ein Minuszeichen oder das Schlüsselwort SUB werden die Flag ausgeschaltet.

Wenn Sie als zweiten Parameter nur die Namen einiger Flags angeben und davor kein Plus- oder Minuszeichen und auch die Schlüsselwörter ADD und SUB nicht angeben, werden genau die Schutzflags gesetzt, deren Namen angegeben wurden. Alle anderen Flags werden ausgeschaltet. Die Verwendung des Plus- oder Minuszeichens und der Schlüsselwörter ADD und SUB beschränkt die Wirkung von PROTECT also gezielt auf einige wenige Flags.

Welche Flags bei einer Datei gesetzt sind, können Sie jederzeit mit Hilfe des LIST-Befehls feststellen. Der LIST-Befehl gibt die bei einer Datei gesetzten Flags hinter der Größe der Datei in Form von acht Buchstaben aus. Gesetzte Flags werden mit ihrem Namen ausgegeben, nicht gesetzte erscheinen als Minuszeichen.

Die Schutzflags realisieren in ihrer Gesamtheit ein recht raffiniertes Schutzsystem, mit dem eine fehlerhafte Benutzung einer Datei weitgehend vermieden werden kann – aber leider nur in der Theorie. Lediglich die Flags d, s und p werden momentan vom Amiga-DOS unterstützt. Alle anderen Flags können zwar mit PROTECT geändert werden; das bleibt aber ohne Konsequenzen für den Gebrauch der Datei. Es gibt deshalb keine Möglichkeit, Dateien vor dem Lesen und Überschreiben zu schützen oder sie (für den DIR-Befehl) unsichtbar zu machen.

**Beispiele:**

```
1.SYS:>LIST Datei1
Datei1                    658   ----rwed   Today      18:45:20
1.SYS:>PROTECT Datei1 +sp
1.SYS:>LIST Datei1
Datei1                    658   -sp-rwed   Today      18:45:20
1.SYS:>PROTECT Datei1 e SUB
1.SYS:>LISTDatei1

Datei1                    658   -sp-rw-d   Today      18:45:20
1.SYS:>PROTECT Datei1 spar
1.SYS:>LIST Datei1
Datei1                    658   -spar---   Today      18:45:20
```

# QUIT

**QUIT** RC <Returncode>

Beendet eine Kommandofolge und erzeugt einen Returncode.

Den Befehl QUIT werden Sie nie direkt in einem Shell-Fenster verwenden. Er ist nur in Kommandofolgen sinnvoll und beendet gezielt jede Kommandofolge, in der er aufgerufen wird. QUIT ist besonders für solche Kommandofolgen wichtig, die eventuell von anderen Kommandofolgen aus aufgerufen werden. An diese andere Kommandofolge kann QUIT nämlich einen <Returncode> zurückgeben, der die übergeordnete Kommandofolge über Mißlingen oder Gelingen der durch QUIT beendeten Kommandofolge informiert.

Die Bedeutung des Returncodes und die Reaktion der übergeordneten Kommandofolge darauf richtet sich im Prinzip nach denselben Regeln wie die Reaktion auf den Returncode eines Shell-Befehls.

Mehr zum QUIT-Befehl und zum Thema Kommandofolgen können Sie in Kapitel 8 erfahren.

# RELABEL

**RELABEL** DRIVE/A **<Laufwerk>** NAME/A **<Name>**

dient zum Umbenennen von Disketten und anderen Datenträgern.

Eine neue Diskette oder Festplatte erhält ihren Namen üblicherweise bei der Formatierung oder beim Kopieren einer anderen Diskette. Die dafür »zuständigen« Befehle sind FORMAT und DISKCOPY beziehungsweise ihre Äquivalente *Initialize* und *Duplicate* auf der Workbench. Möchte man nachträglich den Namen einer Diskette ändern, so ist dies im Shell-Fenster nur mit RELABEL möglich.

Der Befehl RELABEL benötigt zwei Parameter. Der erste, <Laufwerk>, ist der Name eines Laufwerks (zum Beispiel DF0: oder HD1:), das einen Datenträger enthält. Alternativ kann aber auch der augenblickliche, alte des Datenträgers verwendet werden. Der zweite Parameter <Name> ist der neue Name, den dieser Datenträger nach Ausführung des RELABEL-Befehls tragen soll. Falls einer der beiden Namen Leerzeichen enthält, muß er in Anführungszeichen gesetzt werden.

Damit die Namensänderung auch auf der Workbench registriert wird, kann es bei einigen Arten von Datenträgern (zum Beispiel bei der RAM-Disk RAD:) nötig sein, nach dem RELABEL-Befehl auch noch den DISKCHANGE-Befehl zu verwenden.

**Beispiele:**

```
RELABEL DF0: WBDisk

RELABEL SYS: WBDisk
RELABEL Workbench1.3D: WBDisk
```

Alle diese drei Befehle tun dasselbe (wenn im internen Laufwerk die Startdiskette liegt und den Namen Workbench1.3D trägt). Sie benennen die Diskette im internen Laufwerk um in *WBDisk*.

Dateien und Directories müssen mit dem Befehl RENAME umbenannt werden.

# REMRAD

**REMRAD**

dient dazu, die RAM-Disk RAD: zu entfernen.

Die RAM-Disk RAD: hat die meist angenehme, manchmal aber auch unangenehme Eigenschaft, einen Neustart zu überleben. Das kann deshalb unangenehm sein, weil diese RAM-Disk permanent einen bestimmten Teil des RAM-Speichers belegt, auch wenn Sie keine Dateien enthält.

Damit Sie den Amiga nicht ausschalten müssen, um diese RAM-Disk zu entfernen, gibt es den Befehl REMRAD (*remove RAD* = entferne RAD). Er löscht alle Daten auf RAD: und verringert dann den von ihr benötigten Speicherplatz auf ein Minimum. Aus technischen Gründen kann REMRAD diese RAM-Disk aber nicht ganz entfernen. Sie verschwindet aber nach dem nächsten Neustart völlig und wird dann auch nicht mehr vom INFO- oder ASSIGN-Befehl angezeigt.

# RENAME

**RENAME** FROM/A **<alter Name>** TO/A AS/A **<neuer Name>**

dient zum Umbenennen von Dateien und Directories.

Der Befehl RENAME gibt einer Datei oder einem Directory einen neuen Namen. Er hat zwei Parameter: <alter Name>, den alten Namen der Datei/des Directory und <neuer Name>, den die Datei/das Directory erhalten soll. Falls einer der beiden Namen Leerzeichen enthält, muß er in Anführungszeichen gesetzt werden. Weiterhin darf noch keine Datei beziehungsweise kein Directory namens <Neuer name> existieren, sonst liefert RENAME nur eine Fehlermeldung.

Mit RENAME kann man Dateien aber nicht nur im strengen Sinne umbenennen, sondern sie auch von einem Directory in ein anderes legen, sofern sich beide Directories auf derselben Diskette oder Festplatte befinden. Hierzu muß sich nur der erste Teil des vollständigen Pfadnamens in <alter Name> und <neuer Name> unterscheiden. Das wirkt dann, als würde <alter Name> in ein anderes Directory kopiert und das Original dann gelöscht. (Ähnlich wie der *mv*-Befehl in UNIX.) Das zweite Directory muß allerdings vor Ausführung des Befehls schon existieren.

**Beispiele:**

```
RENAME Text1 TO Text2
RENAME Text1 AS Text2
RENAME Text1 Text2
RENAME :Texte/Text1 AS :Texte/Briefe/Text2
```

Die ersten drei Befehle tun dasselbe. Sie benennen die Datei *Text1* im aktuellen Directory um in *Text2*. Der letzte Befehl aber legt die Datei *Text1* aus dem Directory *:Texte* unter dem Namen *Text2* in das Directory *:Texte/Briefe* auf der aktuellen Diskette. Dazu muß das Directory *:Texte/Briefe* aber schon existieren. RENAME kann keine neuen Directories anlegen.

Datenträger (Disketten, Festplatten und RAM-Disks) müssen mit dem Befehl RELABEL umbenannt werden.

# RESIDENT

**RESIDENT** NAME <Name> FILE <Datei>
REMOVE/S ADD/S REPLACE/S PURE/S SYSTEM/S

dient dazu, bestimmte Programme oder Programmteile speicherresident zu machen.

RESIDENT ist einer der wenigen Befehle, der nicht in einem CLI-Fenster sondern nur in einem Shell-Fenster aufgerufen werden kann. Der Befehl sorgt dafür, daß ein anderer Shell-Befehl, nachdem er das erste Mal geladen und ausgeführt wurde, im RAM-Speicher bleibt. Er braucht beim nächsten Aufruf nicht mehr von der Diskette geladen sondern nur erneut gestartet werden (was sehr schnell geht). Solche Befehle nennt man »speicherresidente« oder einfach »residente« Befehle. Residente Befehle haben aber nicht nur den Vorteil, daß Sie schneller gestartet werden können. Sie können darüber hinaus auch von mehreren unabhängigen Prozessen aus genutzt werden, wobei immer nur eine Kopie des Programms im Speicher liegt. (Der Fachmann spricht in diesem Fall von »reentrant Code«.

Nicht jeder Befehl ist aber dafür geeignet, in einer einzigen Version für mehrere Prozesse tätig zu werden. Befehle, bei denen das möglich ist, werden deshalb extra gekennzeichnet, indem das Schutzflag P (*pure*) gesetzt wird (siehe PROTECT).

Wenn Sie RESIDENT ohne Parameter aufrufen, gibt der Befehl eine Liste der schon residenten Befehle aus. Das sind üblicherweise Befehle, die schon in der Startup-Sequence resident gemacht wurden. Dabei werden aber nur Shell-Befehle aufgelistet. Wenn Sie auch andere residente Programme sehen wollen, müssen Sie das Schlüsselwort SYSTEM angeben. RESIDENT zeigt dann auch die sogenannten System-Befehle, einige spezielle residente Programme, die Amiga-DOS intern benötigt.

Wenn Sie einen Shell-Befehl resident machen wollen (weil Sie ihn häufig benötigen), müssen Sie zwei Parameter angeben. <Name> muß der Befehlsname sein, wie Sie ihn im Shell-Fenster eingeben würden und <Datei> der vollständige Pfadname des Programms, das diesen Befehl ausführt. Zu dem Befehl DIR würde beispielsweise der Pfadname C:DIR gehören. Das Schlüsselwort ADD dient dazu, einen neuen Befehl resident zu machen. Sie

können es aber auch wegfallen lassen. Wenn Sie zusätzlich noch das Schlüsselwort SYSTEM angeben, wird der residente Befehl ein System-Befehl. Er kann dann unter anderem nicht mehr aus der Liste der residenten Befehle entfernt werden. Sie sollten normale Shell-Befehle (wie zum Beispiel DIR) niemals zu System-Befehlen erklären. Sie können diese Befehle nämlich dann nicht mehr einfach mit ihrem Namen aufrufen. Das SYSTEM-Flag hat andere Zwecke und wird nur für ganz spezielle Arten von Programmen benötigt.

Wenn Sie nur den Parameter <Name> und das Schlüsselwort REMOVE angeben, wird der Befehl <Name> aus der Liste der residenten Befehle entfernt (engl. *to remove* = entfernen/löschen). Wenn Sie den Befehl <Name> das nächste Mal aufrufen, muß er wieder von der Diskette geladen werden.

Das Schlüsselwort REPLACE dient dazu, einen Befehl in der Liste der residenten Befehle zu ersetzen (engl. *to replace* = ersetzen/austauschen).

Das Schlüsselwort PURE dient dazu, einen Befehl resident zu machen, bei dem das Schutzflag P nicht gesetzt ist. RESIDENT weigert sich normalerweise – mit gutem Grund – solche Befehle resident zu machen. Diese Option sollten Sie möglichst vorsichtig verwenden. Wenn Sie ein Programm resident machen, das Code enthält, der nicht reentrant ist (siehe oben), kann das zu schwerwiegenden Fehlern und unter anderem auch zu Datenverlusten führen.

**Beispiele:**

```
RESIDENT MEMACS DF0:Tools/MEmacs
RESIDENT DIR C:Dir
RESIDENT SYSTEM
RESIDENT MEMACS DF0:Tools/MEmacs REMOVE
```

Die ersten beiden Aufrufe von RESIDENT machen die Befehle MEMACS und DIR resident. Der dritte Befehl zeigt dann die vollständige Liste der residenten Befehle und die Anzahl der Prozesse, die die Befehle gerade nutzen, an. Der vierte Befehl schließlich entfernt MEMACS wieder aus der Liste der residenten Befehle.

# RUN

**RUN <beliebiger Shell-Befehl >** +[Return]
<weitere Shell-Befehle>

dient zum Ausführen eines Shell-Befehles als eigenständigen »Hintergrund-Prozeß«.

Wenn Sie den Befehl RUN an den Anfang einer Befehlszeile und dahinter einen weiteren kompletten Shell-Befehl setzen, so wird der letztere als separater Prozeß abgearbeitet, der vom aktuellen Shell-Task unabhängig ist. (Ein solcher Prozeß kann auch mehrere Befehle nacheinander ausführen; siehe unten.) Das heißt, daß der Shell-Prompt sofort wieder erscheint und sie weitere Shell-Befehle eingeben und ausführen können, bevor der mit RUN gestartete Prozeß beendet wurde.

Man nennt dies »Ausführen eines Befehls im Hintergrund«. Die Bezeichnung »im Hintergrund« kommt daher, daß dieser neue Task nicht mit dem Benutzer kommunizieren kann, da er kein eigenes Fenster besitzt. Er bleibt also im Hintergrund Ihrer Aufmerksamkeit, während Sie sich vordringlich (»im Vordergrund«) mit anderen Dingen beschäftigen.

Der über RUN gestartete Prozeß verwendet für Ein- und Ausgaben allerdings dasselbe Shell-Fenster, von dem aus er gestartet wurde. Das kann dann verwirrend sein, wenn zwei oder mehr Prozesse gleichzeitig ins Fenster schreiben. So können zum Beispiel während der Ausführung eines DIR-Befehls zwischen den aufgelisteten Dateinamen Meldungen eines Hintergrund-Task auftauchen. Hat man gar mehrere Prozesse im Hintergrund laufen, die Ausgaben ins Shell-Fenster schicken, ist das Chaos geradezu vorprogrammiert. Üblicherweise wird man deshalb zusammen mit RUN wohl einen Shell-Befehl verwenden, der längere Zeit zu seiner Ausführung benötigt und kein weiteres Eingreifen von seiten des Benutzers nötig hat und auch keine Meldungen ausgibt.

Wenn Sie die Meldungen, die der Hintergrundprozeß ausgibt, nicht interessieren (was im Fall von eventuell möglichen Fehlermeldungen etwas riskant ist), können Sie verhindern, daß der mit RUN gestartete Prozeß Ihre Arbeit stört. Lenken Sie seine Ausgaben einfach mittels »> NIL:« (direkt hinter RUN) in die »Leere«. Diese Umleitung hat noch einen weiteren Vorteil. Wenn der Prozeß das Shell-Fenster nicht benötigt, kann er unabhängig von diesem und dem zugehörigen Shell-Prozeß existieren. Dann kann dieses Shell-Fenster geschlossen werden, bevor der Prozeß beendet ist. Das wäre sonst nicht möglich.

Um einen mit RUN gestarteten Prozeß abzubrechen, müssen Sie die Befehle STATUS und BREAK verwenden. Stellen Sie erst mit STATUS fest, welche Prozeßnummer der gesuchte Prozeß hat. Sie können ihn dann mit dem Befehl BREAK beenden (und das gerade laufende Programm abbrechen).

Der Befehl RUN kennt keine Parameter im eigentlichen Sinn des Wortes. Der Rest der Zeile hinter dem Wort RUN muß ein gültiger Shell-Befehl sein, der dann von RUN im Hintergrund ausgeführt wird. Dieser Befehl kann sich sogar über mehrere Zeilen ausdehnen, indem man die Möglichkeit der Zeilenfortführung nutzt. Hierzu gibt man am Zeilenende ein Pluszeichen (+) ein und drückt dann auf [Return]. Nun kann man einen zweiten Befehl eingeben und wenn man möchte, nach einem weiteren Pluszeichen auch noch weitere. Erst nach dem ersten [Return] ohne ein Pluszeichen davor beginnt die Abarbeitung der so erstellten Liste von Befehlen.

Obwohl die Verkettung beziehungsweise »Verlängerung« von Befehlszeilen mit dem Plus-Zeichen in Zusammenhang mit dem RUN-Befehl eine sehr praktische Sache ist, sollten Sie sie sehr vorsichtig anwenden – zumindest dann, wenn Befehle dabei sind, die direkt oder indirekt destruktiv wirken können! Die Shell überprüft nicht, ob Fehler bei vorangehenden Befehlen aufgetreten sind, sondern führt den neuen Befehl »blindgläubig« aus.

**Beispiele:**

```
RUN LIST
```

Dieser Befehl ist ein schlechtes Beispiel für die Anwendung von RUN. Er läßt den LIST-Befehl mit vielen Bildschirmausgaben im Hintergrund laufen.

Der folgende Befehl ist hingegen schon sinnvoller. Er schickt den Inhalt einer Datei auf den Drucker, nachdem er sie vorher kopiert hat, löscht die Kopie nach Beendigung des Druckvorgangs und meldet dann, daß er fertig ist:

```
RUN COPY BriefRobert druckvorlage +
COPY druckvorlage PRT: +
DELETE druckvorlage +
ECHO "Druckvorgang beendet!"
```

Achten Sie bei der Verkettung mehrerer Zeilen darauf, daß unmittelbar hinter dem Fortführungssymbol + sofort [Return] getippt werden muß, ansonsten gilt die folgende Zeile als neue Zeile und nicht als Fortführung der vorangehenden.

# SAY

**SAY** –M/S –F/S –R/S –N/S <text> –S/F <Tempo> –P/F <Tonhöhe> –X/F <Textdatei>

dient zur Ausgabe eines (meist kurzen) Textes als synthetische Sprache.

Der Shell-Befehl SAY entspricht dem Werkzeug *Say*, das in Kapitel 5 beschrieben wurde. Der Befehl SAY ist das akustische Gegenstück zum Befehl ECHO. Er nutzt die Fähigkeit des Amiga zur Sprachsynthese und gibt den Text, der als Parameter übergeben wurde, über einen angeschlossenen Lautsprecher aus. Die Aussprache ist nicht gerade natürlich zu nennen, aber doch verständlich. Wenn keine Lautsprecher an die Stereo-Ausgänge hinten am Gehäuse des Amiga 500 angeschlossen sind, kann man solche Meldungen allerdings nicht hören.

Das Schlüsselwort –M bewirkt eine männliche Aussprache, das Schlüsselwort –F eine weibliche. Mit –N erhalten Sie eine natürlich auf- und abgehende Betonung der Sätze, –R erzwingt eine tonlose, roboterhafte Aussprache. Die Schlüsselwörter –S und –P legen die Sprechgeschwindigkeit und die Tonhöhe fest. Mehr zur Bedeutung der Parameter <Temp> und <Tonhöhe> können Sie in Abschnitt 5.2.3 bei der Beschreibung des Werkzeugs Say erfahren.

Das Schlüsselwort –X spricht die gesamte <Textdatei> aus, als hätten Sie diesen Text als Parameter eingegeben.

Eine Verwendung von SAY bietet sich vor allem innerhalb von Kommandofolgen an. So könnte die Beendigung einer Kommandofolge zum Beispiel statt mit einer Bildschirmmeldung, die man vielleicht übersehen würde, durch ein lautes »Fertig!« kundgetan werden.

Bedenken Sie bitte immer, daß Ihr Amiga ein amerikanischer Computer ist. Er spricht deshalb mit deutlichem Slang. Sie können ihn aber überlisten, indem Sie <Text> falsch schreiben; und zwar so, daß ein Amerikaner, der diesen Text ausprechen würde, als wären es englische Worte, einigermaßen korrekte deutsche Worte ausspricht.

**Beispiele:**

```
SAY "Hullow Doo"
SAY -x s:Startup-Sequence
```

Der erste Befehl spricht nur die beiden Worte »Hallo Du«, der zweite den gesamten Inhalt der Textdatei *Startup-Sequence* im logischen Laufwerk S: aus.

# SEARCH

**SEARCH** FROM <Name>* SEARCH/A **<Suchtext>**
ALL/S NONUM/S QUIET/S QUICK/S FILE/S

dient zur Suche nach einem bestimmten Text in einer beliebigen Gruppe von Dateien.

Falls Sie einmal in die Verlegenheit kommen, nicht mehr genau zu wissen, in welchen Dateien Sie sich auf bestimmte Begriffe oder Namen bezogen haben, so hilft Ihnen der SEARCH-Befehl. Er erlaubt es, eine Gruppe von Dateien nach dem Text zu durchsuchen, der hinter dem Schlüsselwort SEARCH steht. (Falls dieser Text Leerzeichen enthält, muß er in Anführungszeichen gesetzt werden.)

Die zu durchsuchenden Dateien gibt der Parameter <Name> an. Ist dies ein Directory, werden alle Dateien in diesem Directory durchsucht. Wenn Sie auch noch das Schlüsselwort ALL folgen lassen, erstreckt sich die Suche auch auf die Dateien in den Unter-Directories <Name>. Sie können für <Name> aber auch ein Muster angeben. Nur die Dateien, deren Namen in das Muster passen, werden dann durchsucht. Wird überhaupt kein Muster oder kein Directory-Name angegeben, so werden die Dateien im aktuellen Directory durchsucht.

Zeile in jeder durchsuchten Datei, die diesen Text enthält, wird ausgegeben. (Diese Liste kann mittels > auch in eine Datei statt auf den Bildschirm geschickt werden.) Beim Vergleich des Suchtextes mit dem Inhalt der Datei wird Groß- oder Kleinschreibung nicht berücksichtigt. (Mit Umlauten hat diese Suchmethode allerdings Schwierigkeiten: ein großes Ö ist so zum Beispiel verschieden von einem kleinen ö.)

Beim Durchsuchen der verschiedenen Dateien haben Sie die Möglichkeit, die Suche in der aktuellen Datei oder auch komplett abbrechen zu lassen. Halten Sie die [Ctrl]-Taste fest und tippen Sie gleichzeitig [D], so wird die Durchsuchung der gerade bearbeiteten Datei abgebrochen und gleich mit der nächsten fortgefahren. Halten Sie hingegen die [Ctrl]-Taste fest und tippen gleichzeitig [C], so wird die ganze Suche abgebrochen.

Das Schlüsselwort NONUM unterdrückt die Ausgabe von Zeilennummern in der ausgegebenen Liste. Das Schlüsselwort QUIET unterdrückt sämtliche Ausgaben. Es ist deshalb wohl nur in Zusammenhang mit Kommandofolgen interessant. Zu diesem Zweck liefert SEARCH den Returncode 5, wenn der gesuchte Text nicht gefunden wurde und 0, wenn er in einer beliebigen der durchsuchten Dateien gefunden wurde.

Das Schlüsselwort FILES bewirkt, daß SEARCH keine Dateien nach einem bestimmten Text durchsucht, sondern statt dessen nach einer Datei sucht, deren Namen <Suchtext> enthält.

Der SEARCH-Befehl ist vor allem für Programmierer gedacht, die in ihren Programmen nach bestimmten Dingen (Prozeduren, Variablen, und so weiter) suchen. Er kann unter Umständen auch für den »normalen« Anwender von Nutzen sein.

**Beispiele:**

```
SEARCH :Texte SEARCH "Sehr geehrt" ALL
SEARCH DF0:s "ASSIGN" NONUM
```

Der erste Befehl sucht den Text »Sehr geehrt« in allen Dateien im Directory *texte* und deren Unter-Directories. Der zweite Befehl durchsucht alle Dateien im logischen Laufwerk S: nach dem Befehl ASSIGN. Die Fundstellen werden ohne Angaben von Zeilennummern ausgegeben.

# SETCLOCK

**SETCLOCK** LOAD/S SAVE/

dient zum Stellen und Abfragen der batteriegepufferten Uhr.

Wenn Sie Ihrem Amiga 500 eine Speichererweiterung Amiga 501 oder eine vergleichbare Speichererweiterung mit eingebauter Uhr spendiert haben, so behält diese Uhr die aktuelle Zeit auch nach dem Ausschalten des Amiga. Diese Hardware-Uhr wird von der Amiga-Software aber nicht andauernd abgefragt, wenn ein Programm die aktuelle Zeit wissen will. Statt dessen verwaltet die Betriebssystemsoftware eine »Software-Uhr«, die nur solange funktioniert wie der Amiga an ist. Beim Start wird diese Software-Uhr (die sogenannte Systemzeit) einmal anhand der Uhrzeit in der Hardware-Uhr initialisiert (sofern eine Hardwareuhr vorhanden ist). Das geschieht mit Hilfe des SETCLOCK-Befehls in der Startup-Sequence. Der SETCLOCK-Befehl kann aber auch dazu verwendet werden, die Uhrzeit in der Hardware-Uhr zu ändern.

Das Schlüsselwort LOAD bewirkt, daß SETCLOCK die Uhrzeit aus der Hardware-Uhr liest und die Systemzeit (Software-Uhr) entsprechend ändert. Das Schlüsselwort SAVE justiert umgekehrt die Hardware-Uhr entsprechend der aktuellen Systemzeit. Das Schlüsselwort RESET, setzt die Hardware-Uhr in einen definierten Nullzustand zurück, wenn ein fehlerhaftes Programm die Uhr ausgeschaltet oder verstellt hat. (Danach muß die Uhrzeit natürlich neu eingestellt werden.)

Ab der Workbench-Version 1.3 wird SETCLOCK außerhalb der Startup-Sequence kaum noch gebraucht. Wenn Sie die Uhrzeit mittels *Preferences* ändern, so wird dadurch automatisch auch die Hardware-Uhr verstellt. Der Befehl SETCLOCK (mit dem Schlüsselwort SAVE) wird dazu nicht benötigt. Lediglich, wenn Sie die Systemzeit mit einem anderen Programm verstellen, kann es nötig sein, sie mit Hilfe von SETCLOCK in die Hardware-Uhr zu übertragen.

**Beispiel:**

```
SETCLOCK LOAD
```

Dieser Befehl bewirkt, daß Datum und Uhrzeit aus der batteriegepufferten Uhr gelesen und Amiga-DOS mitgeteilt werden. Amiga-DOS weiß danach also, »wie spät es ist« und stellt die Systemzeit entsprechend.) Dieser SETCLOCK-Befehl ist nach jedem Neustart nötig, damit Sie Datum und Uhrzeit nicht selbst eingeben müssen und findet sich deshalb normalerweise auch in der Startup-Sequence.

# SETDATE

**SETDATE** FILE/A **<name>** DATE <datum> TIME <zeit>

dient zur Änderung des Datums der letzten Modifikation an einer Datei.

Sie werden den Befehl SETDATE sehr selten benötigen. Mit ihm können Sie nämlich das Datum der letzten Schreiboperation (Modifikation) bei einer Datei beliebig ändern. Wenn dieses Datum stimmt, also die letzte Änderung wirklich zu diesem Datum und dieser Uhrzeit vorgenommen wurde, gibt es keinen Grund, es nachträglich zu ändern.

Für die Datums- und Uhrzeitangaben <datum> und <zeit>, die Sie zusammen mit setdate verwenden, gelten dieselben Einschränkungen und Möglichkeiten wie beim DATE-Befehl. Zum Beispiel sind auch die englischen Namen von Wochentagen und *tomorrow* sowie *yesterday* korrekte Datumsangaben. Optional kann auch der Name eines Wochentags zusätzlich vor Angabe eines Datums kommen. Die Ausgabe des Befehls DATE könnte deshalb als Eingabe (Parameter) für SETDATE verwendet werden, was aber wohl nur für Kommandofolgen von Interesse ist.

**Beispiele:**

```
SETDATE Brief 10-jul-89 15:00:00
SETDATE Brief tomorrow
```

Der erste Befehl bewirkt, daß danach der 10. Juli 1989 15 Uhr als Zeitpunkt der letzten Änderung an der Datei *Brief* gilt. Nach dem zweiten Befehl gilt die Uhrzeit *0 Uhr* am Tag nach dem, an dem der Befehl eingegeben wurde, als Zeitpunkt der letzten Änderung an Briefe. Sie könnten sich mit Hilfe des Befehls *LIST Brief* davon überzeugen.

# SETENV

**SETENV** NAME/A **<Name>** STRING <Text>

dient zum Festlegen des Inhalts einer Umgebungsvariablen.

Die Amiga-Shell kennt – wie auch viele andere Computer-Betriebssysteme – das Konzept der Umgebungsvariablen. Diese Umgebungsvariablen sind eine Art Zwischenspeicher mit Eigennamen, die beliebige Werte (im allgemeinen einen kurzen Text oder eine Zahl) annehmen können. Mit dem Befehl SETENV kann der Inhalt einer Umgebungsvariablen geändert oder eine neue Umgebungsvariable erzeugt werden. Der erste Parameter <Name> ist die Bezeichnung der Umgebungsvariablen. <Text> ist der (neue) Inhalt der Umgebungsvariablen. Dieser Inhalt kann mit dem Befehl GETENV jederzeit abgefragt werden. In Kommandofolgen kann der Wert einer Umgebungsvariablen auch in der Form »$Name« anstatt eines Parameters in der Bedingung von IF-Befehlen verwendet werden.

Umgebungsvariablen werden von der augenblicklichen Version der Shell (Workbench 1.3) in Form einzelner Dateien auf der RAM-Disk RAM: abgespeichert. Diese Dateien liegen im Directory Env auf der RAM-Disk, dem gleichzeitig das logische Gerät ENV: zugeordnet ist.

Der Inhalt einer Umgebungsvariablen kann deshalb auch mit einem TYPE- oder COPY-Befehl festgestellt werden und durch Ausgabeumleitung in eine dieser Dateien geändert werden. (Diese Art der Realisierung des Konzepts der Umgebungsvariablen könnte sich in zukünftigen Versionen der Shell aber eventuell ändern.)

**Beispiele:**

```
1.SYS:>SETENV Compiler "DH0:WB/c/Pascal/Pascal"
1.SYS:>GETENV Compiler
DH0:WB/c/Pascal/Pascal
```

Der erste Befehl definiert eine Umgebungsvariable namens Compiler und belegt Sie mit einem Wert. Der zweite Befehl stellt diesen Wert fest (er wird am Bildschirm ausgegeben). Den selben Effekt könnte man prinzipiell bei der augenblicklichen Realisierung der Umgebungsvariablen auch mit dem folgenden Befehl erzielen:

```
1.SYS:>TYPE ENV:Compiler
DH0:WB/c/Pascal/Pascal
```

# SETPATCH

**SETPATCH** R/S

dient zur Korrektur von Fehlern im Amiga-Kickstart-ROM.

Der sogenannte »Kickstart-ROM« im Amiga enthält einen großen und wichtigen Teil der Amiga-Betriebssystemsoftware. Wie bei allen großen Programmen ist auch die hier gespeicherte Software nicht völlig fehlerfrei. Der Befehl SETPATCH korrigiert die bekannten Fehler, indem er Aufrufe von fehlerhaften Routinen in korrigierte Routinen »umleitet«, die er selbst im RAM-Speicher ablegt. Die fehlerhaften Routinen werden danach von allen Programmen, die sich an die festgelegten Aufruf-Regeln für Routinen der Betriebssystem-Software halten (das sind die meisten), nicht mehr verwendet. SETPATCH ist üblicherweise einer der ersten Befehle in der Startup-Sequence.

Der Schalter R wird nur bei den neuen Amiga-Modellen benötigt, die über mehr als 512 Kbyte Chip-RAM verfügen. Die Verwendung der resetfesten RAM-Disk RAD: kann bei diesen Modellen zu Fehlern führen. Wenn Sie RAD: auf einem Amiga mit 1 Mbyte Chip-RAM verwenden wollen, müssen Sie diesen Schalter deshalb beim Aufruf angeben. Dazu sollten Sie Startup-Sequence entsprechend abändern, damit SETPATCH sofort nach dem Start in der korrekten Form ausgeführt wird.

# SKIP

**SKIP** LABEL <Marke> BACK/S

dient zum »Überspringen« von Shell-Befehlen innerhalb einer Kommandofolge.

Den Befehl SKIP (engl. *to skip* = überspringen, übergehen) werden Sie nie direkt im Shell-Fenster benötigen. Er ist nur in Kommandofolgen sinnvoll. Stößt die Shell beim Abarbeiten einer Kommandofolge auf den Befehl SKIP <Marke>, so werden alle Befehle übersprungen, bis eine Zeile mit dem Befehl LAB <Marke>, dem Befehl ENDSKIP oder das Ende der Kommandofolge auftaucht (je nachdem, was eher der Fall ist). Danach wird die Abarbeitung der Kommandofolge fortgesetzt.Wird <Marke> nicht angegeben, springt SKIP bis zum nächsten ENDSKIP oder zum Dateiende.

SKIP springt normalerweise nur nach vorne, also zum Dateiende. Wird jedoch das Schlüsselwort BACK angegeben, so beginnt die Suche nach dem entsprechenden LAB- oder ENDSKIP-Befehl am Anfang der jeweiligen Kommandofolge. Ein solcher Rücksprung ist aber nur bis maximal zum letzten, dem SKIP-Befehl vorangehenden EXECUTE-Befehl möglich. Nur dann, wenn die Kommandofolge vor dem SKIP keinen EXECUTE-Befehl enthält, ist ein Rücksprung bis zum Anfang möglich. Mit dem Schlüsselwort BACK ist der Aufbau von Programmschleifen möglich.

Mehr zum SKIP-Befehl erfahren Sie in Kapitel 8.

# SORT

**SORT** FROM/A **<Dateiname1>** TO/ **<Dateiname2>** COLSTART/K <n>

dient zum Sortieren einer Textdatei.

Mit dem Befehl SORT (engl. *to sort* = sortieren) können Sie die Textdatei <Dateiname1> zeilenweise sortieren. Eine solche Textdatei zeichnet sich dadurch aus, daß Sie nur druckbare Zeichen enthält und in Zeilen aufgeteilt ist, die durch ein besonderes Zeichen (den unsichtbaren Zeilenvorschub) getrennt sind.

Die Datei mit dem Namen <Dateiname2> enthält nach Ausführung des Befehls den sortierten Text. Die Sortierung geschieht alphabetisch, beginnend mit der ersten Spalte jeder Zeile, und ignoriert Unterschiede zwischen Groß- und Kleinschreibung. (Mit Umlauten hat diese Sortiermethode allerdings Schwierigkeiten: ein großes Ö wird zum Beispiel an anderer Stelle als ein kleines ö und auch nicht hinter O angeordnet.) Wollen Sie, daß bei der Sortierung der Text in jeder Zeile bis zu einer bestimmten Spalte ignoriert wird, so können Sie die Nummer dieser Spalte hinter dem Schlüsselwort COLSTART angeben. Enthält <Dateiname1> eine Art Tabelle, können Sie auf diese Art und Weise nach einer bestimmten Spalte der Tabelle sortieren lassen.

Der Befehl SORT ist ein sehr primitives Sortierprogramm, das nur für die wenigsten Anwendungsfälle geeignet sein dürfte. Normalerweise kann es nur Dateien sortieren, die maximal 200 Zeilen enthalten. Bei größeren Dateien müssen Sie den Speicher, der dem

Programm für diese Zwecke zur Verfügung steht, mit dem Befehl STACK (siehe unten) erhöhen. Wie groß der Speicher für Dateien bestimmter Größe sein muß, ist dabei sehr stark Erfahrungssache. SORT ist deshalb eigentlich nur für erfahrene Benutzer und kleine Datenmengen geeignet!

**Beispiele:**

```
SORT liste TO "sortierte Liste"
SORT Adressen TO nachPLZ COLSTART 20
```

Der erste Befehl bewirkt, daß die Zeilen der Textdatei *Liste* sortiert werden und die sortierte Folge von Zeilen in der Datei *sortierte Liste* abgelegt wird. Der zweite Befehl sortiert die Datei *Adressen* und legt die sortierten Zeilen in *nachPLZ* ab. Dabei werden für die Sortierung nur die Buchstaben in jeder Zeile berücksichtigt, die ab der zwanzigsten Spalte kommen. Enthält die Datei *Adressen* eine Adressenliste, in der jede Adresse eine Zeile einnimmt und die Postleitzahl in der zwanzigsten Spalte beginnt, haben Sie auf diese Weise die Adressen nach PLZ sortiert.

# STACK

**STACK** SIZE <n>

dient zum Anzeigen und Ändern der Stack-Größe.

Mit dem Befehl STACK kann die Größe eines bestimmten Speicherbereichs (des sogenannten Stacks oder Stapelspeichers) geändert werden, die die Shell einem Programm (Befehl) zuteilt, bevor es gestartet wird. Dieser Speicherbereich dient im wesentlichen für den sich laufend ändernden (dynamischen) Speicherbedarf eines Programms.

Ohne tiefergehende Kenntnisse der Programmierung dürfte der STACK-Befehl für Sie wenig brauchbar sein. Falls aber bei Anwendung des SORT-Befehls einmal ein Fehler auftauchen sollte, so liegt dies wahrscheinlich an mangelndem Speicherplatz. Auch andere Programme können in solche Schwierigkeiten kommen. Verdoppeln Sie dann versuchsweise diesen Speicherplatz mittels STACK.

Tippen Sie STACK ohne Parameter ein, so wird Ihnen die aktuelle Stackgröße angezeigt. Sie beträgt normalerweise 4000 Byte. Tippen Sie hinter STACK noch eine Zahl <n> ein, so wird der für den Stack reservierte Platz auf <n> Byte festgelegt. Seien Sie aber vorsichtig mit zu kleinen Werten für <n>. Viele Programme benötigen mindestens 4000 Byte.

# STATUS

**STATUS** PROCESS <n> FULL/S TCB/S CLI/S ALL/S COMMAND=<Name>/K

dient zum Anzeigen von Informationen über die verschiedenen laufenden Shell-Tasks.

Mit dem Befehl STATUS können Sie jederzeit erfahren, was gerade in den verschiedenen Shell- oder CLI-Fenstern geschieht, die Sie zuvor geöffnet haben. Wenn Sie einfach STATUS ohne Parameter angeben, erhalten Sie eine Liste der aktiven Shell-Prozesse (das sind entweder Shell-Fenster oder Prozesse, die mit RUN gestartet wurden) und der Kommandos, die im jeweiligen Fenster gerade laufen.

Wenn Sie den Schalter TCB verwenden, wird zu jedem Task der »Task-Control-Block« ausgegeben. Dieser enthält neben der Prozeßnummer Informationen über die Stackgröße (siehe STACK) und die Priorität (siehe CHANGETASKPRI). Der Schalter FULL bewirkt, daß sowohl diese Informationen gezeigt werden als auch die gerade in dem entsprechenden Shell-Pozeß bearbeiteten Programme genannt werden.

Die Schalter CLI und ALL geben unter Umständen zu jedem Prozeß noch weitere Informationen aus, die sogenannten »Sektionsnamen«. Diese sind aber nur für fortgeschrittene Programmierer von Interesse. Bei Shell-Befehlen gibt es zudem meist keine Sektionsnamen, da diese Programme nur aus einer Sektion bestehen.

Der Parameter <Name> (mit dem Schlüsselwort NAME= davor) bewirkt, daß der STATUS-Befehl die Nummern der Prozesse ausgibt, in denen gerade das Programm <Name> ausgeführt wird. Das kann (mit Hilfe der Ein-/Ausgabeumleitung) in Kommandofolgen dazu verwendet werden, die Nummer eines bestimmten Prozesses festzustellen und diesen dann mit BREAK gezielt »abzuschießen«.

Wenn Sie nur zu einem bestimmten Prozeß Informationen wünschen, müssen Sie dessen Prozeßnummer als ersten Parameter angeben. Diese Prozeßnummer können Sie jederzeit mit STATUS selbst festellen.

**Beispiele:**

```
STATUS
STATUS FULL
STATUS COMMMAND=Notepad
```

Der erste Befehl listet Nummern und Programmnamen aller Shell-Prozesse auf, während der zweite umfangreichere Informationen über diese Prozesse liefert. Der dritte Befehl gibt die Nummer des Prozesses aus, in dem das Programm *Notepad* läuft.

# TYPE

**TYPE** <Dateiname> TO <Ausgdatei> OPT/K <opt> HEX/S NUMBER/S

dient zum Ausgeben einer Textdatei (auf den Bildschirm)

Der Befehl TYPE entspricht weitgehend dem Befehl COPY, wenn dessen Ausgabe »auf den Bildschirm geschickt wird« (mit >* ). Er ist allerdings etwas einfacher zu handhaben, wenn man die Datei auf dem Bildschirm ausgeben will (dann kann man sich den TO-Parameter sparen) und bietet gegenüber COPY zwei zusätzliche Möglichkeiten. Das Schlüsselwort NUMBER sorgt dafür, daß vor jeder neuen Zeile die aktuelle Zeilen-Nummer ausgegeben wird. Das Schlüsselwort HEX bewirkt, daß die Datei nicht in Form von Text, sondern als Folge von sogenannten Hexadezimal-Zahlen ausgegeben wird. Dabei werden zwei Buchstaben pro Byte an Daten ausgegeben. Sie können dann auch Dateien, die nicht nur Text enthalten, auf dem Bildschirm betrachten, was aber wohl nur für Programmierer oder sehr erfahrene Benutzer von Interesse ist.

Das Schlüsselwort OPT ist nur noch aus historischen Gründen vorhanden. Zusammen mit dem Buchstaben N dahinter (als Parameter <opt>) wirkt OPT wie NUMBER, zusammen mit H wie HEX.

**Beispiele:**

```
TYPE S:StartupII
TYPE S:StartupII NUMBER
TYPE S:StartupII TO PRT: NUMBER
TYPE C:DIR HEX
```

Der erste Befehl gibt den Inhalt der Datei StartupII auf dem logischen Laufwerk S: auf dem Bildschirm aus. Der zweite listet dieselbe Datei mit Zeilennummern auf. Der dritte schickt diese Datei (mit Zeilennummern) zum Drucker. Der letzte Befehl gibt den Inhalt der Programmdatei DIR auf dem logischen Laufwerk C als Folge von Hex-Bytes aus.

# WAIT

**WAIT** <n> SECS/S MINS/S UNTIL/K <Uhrzeit>

dient zum Verzögern eines Befehls, entweder um eine gewisse Zeit oder bis zu einem bestimmten Zeitpunkt.

Der Befehl WAIT (engl. *to wait* = warten) dürfte vor allem innerhalb von Kommandofolgen oder bei RUN-Befehlen sinnvoll sein. WAIT verzögert die Ausführung eines nachfolgenden Befehls und ist deshalb für sich alleingenommen reichlich sinnlos. Er wird erst dann beendet, wenn eine Anzahl von Sekunden oder Minuten verstrichen ist, oder wenn eine bestimmte Uhrzeit erreicht wurde. Die letzte Möglichkeit ist natürlich nur dann sinnvoll einsetzbar, wenn die Amiga-DOS bekannte Uhrzeit (Systemzeit) korrekt ist.

WAIT kann auch ohne Parameter aufgerufen werden. Der Befehl wartet dann eine Sekunde. Wird nur <n> angegeben und weder das Schlüsselwort SECS noch das Schlüsselwort MINS verwendet, wartet wait <n> Sekunden. Statt der Schlüsselwörter SECS und MINS können Sie übrigens auch SEC und MIN angeben. (Der Befehl WAIT 1 MIN liest sich einfach besser als WAIT 1 MINS.)

Wird das Schlüsselwort UNTIL verwendet, wartet WAIT bis <Uhrzeit>.

**Beispiele:**

```
WAIT
WAIT 10 secs
WAIT 5 mins
WAIT UNTIL 11:11
```

Der erste Befehl (WAIT ohne jeden Parameter) wartet genau eine Sekunde. Der zweite wartet 10 Sekunden und der dritte fünf Minuten. Besonders exakt sind diese Verzögerungszeiten aber nicht, da das CLI auch eine gewisse Zeit benötigt, um das Programm WAIT zu finden, zu laden und zu starten. Der dritte Befehl hingegen ist relativ exakt, er wartet, bis die von Amiga-DOS registrierte Uhrzeit (die nicht mit der in der batteriegepufferten Uhr übereinstimmen muß) gleich 11 Uhr 11 ist.

# WHICH

**WHICH** FILE/A **<Name>** NORES/S RES/

dient dazu, festzustellen, wo das Programm zu einem Shell-Befehl gespeichert wird.

Wenn Sie einen Shell-Befehl eingeben, sucht die Shell ein gleichnamiges Programm in einem der Directories im aktuellen Suchpfad und führt dieses aus. Das ist praktisch, aber kann auch verwirrend sein, da man nie so recht weiß, wo sich das Programm befindet, das man aufruft. Das muß man zwar nur selten wissen; für diese seltenen Fälle gibt es aber einen Befehl, der einem bei der Suche hilft: WHICH.

WHICH sucht im aktuellen Suchpfad nach einem Programm namens <Name> und gibt dann den Pfadnamen des Directory aus, in dem das Programm gefunden wurde. Falls es sich um einen residenten Befehl handelt (siehe RESIDENT), wird auch das angegeben.

Mit den Schlüsselwörtern NORES und RES können Sie für residente Befehle eine Sonderbehandlung festlegen. Geben Sie NORES an, wird die Liste der residenten Befehle nicht durchsucht. WHICH gibt dann immer nur den Pfadnamen des Programm-Directories aus (sofern das Programm gefunden wird). Wenn Sie hingegen das Schlüsselwort RES angeben, sucht WHICH nur bei den residenten Befehlen.

WHICH kann auch in Kommandofolgen verwendet werden, um festzustellen, ob ein bestimmter Befehl verwendet werden kann. WHICH liefert zu diesem Zweck den Fehlercode 5, falls das gesuchte Programm nicht gefunden wurde und ansonsten Null. Im Zusammenhang mit Kommandofolgen sind auch die Schlüsselwörter RES und NORES interessant.

**Beispiele:**

```
WHICH DIR
WHICH MAKEDIR NORES
```

Der erste Befehl sucht das Programm DIR im aktuellen Suchpfad und gibt dann den Pfadnamen des Directorys aus. Ist DIR resident, wird statt des Pfadnamens nur diese Tatsache gemeldet. Wird DIR nicht gefunden, erfolgt eine Fehlermeldung und der Returncode 5 wird zurückgegeben. Der zweite Befehl sucht das Programm MAKEDIR und ignoriert dabei die Liste der residenten Befehle.

# WHY

### WHY

dient zum Erläutern der Gründe für den Fehlschlag des vorangehenden Shell-Befehls.

Wenn ein Shell-Befehl – aus welchem Grund auch immer – mißlang, so wird üblicherweise eine Fehlermeldung ausgegeben. Diese Fehlermeldung kann ausreichen, manchmal ist sie aber recht knapp oder besteht nur aus einer Zahl (eine Unsitte, deren Gründe weit zurück in der Urgeschichte der Computer, also vor 1980, liegen).

Tippt man sofort nach dem fehlgeschlagenen Befehl WHY, so bekommt man eine knappe Erläuterung der Fehlerursache in (englischem) »Klartext«. Dieser Klartext ist aber oft mit so vielen Fachausdrücken geladen, daß er Ihnen auch nicht immer helfen kann. In diesem Fall hilft Ihnen aber vielleicht die Fehlerliste im Anhang dieses Buches.

Vergleichen Sie zu diesem Thema auch die Beschreibung des Shell-Befehls FAULT, der jederzeit – nicht nur unmittelbar nach einem Fehler – aufgerufen werden kann.

## 7.4 Geräte

Die folgende Tabelle enthält eine Aufstellung der Gerätenamen, die unter Amiga-DOS »normalerweise« bekannt sind und jeweils ein Stichwort zu ihrer Bedeutung. Weitere Erläuterungen finden Sie im Anschluß an die Tabelle. Eine ähnliche Übersicht mit den logischen Laufwerken, deren Namen denselben Aufbau wie Gerätenamen haben, finden Sie in Abschnitt 7.5.

| Gerätename | Bedeutung |
|---|---|
| DF0: | Internes Diskettenlaufwerk |
| DF1: | 1. externes Diskettenlaufwerk |
| DF2: | 2. externes Diskettenlaufwerk |
| DF3: | 3. externes Diskettenlaufwerk |
| DH0: oder HD0: | 1. Festplattenlaufwerk |
| DH1: oder HD1: | 2. Festplattenlaufwerk |
| ... | |
| RAM: | Pseudo-Diskettenlaufwerk im RAM-Speicher (RAM-Disk)<br>Inhalt geht beim Ausschalten oder Neustart verloren |
| RAD: | Pseudo-Diskettenlaufwerk im RAM-Speicher<br>Inhalt überlebt Neustart |
| NIL: | Leeres Pseudogerät |
| SER: | Serielle Schnittstelle |
| PAR: | Parallele Schnittstelle |
| PRT: | Drucker |
| PIPE: | Gerät zum Aufbau von Pufferspeicher-Dateien |
| AUX: | Verbindung zu einem anderen Computer |
| SPEAK: | Sprachausgabe |
| CON: | Konsole (Bildschirm und Tastatur) |
| NEWCON: | Ähnlich CON: mit zusätzlichen Möglichkeiten |
| RAW: | Ähnlich CON: aber nur für Programmierer nutzbar |
| * | Shell-Fenster |

***Tabelle 7.1:*** *Physikalische Geräte des Amiga*

Die Namen von Disketten-, Festplatten- und RAM-Disk-Laufwerken werden vor allem für die Formulierung von Dateinamen benötigt: Der vollständige Pfadname einer Datei beginnt

immer mit dem Namen des Datenträgers oder mit dem Namen des Laufwerks, in dem sich dieser Datenträger befindet. Im Betriebssystem des Amiga sind insgesamt vier Diskettenlaufwerke (DF0: bis DF3:) vorgesehen. Die Verwendung von mehr Diskettenlaufwerken dürfte auch kaum sinnvoll sein. Die Anzahl der Festplattenlaufwerke ist hingegen nicht begrenzt. Der Gerätename einer Festplatte kann vom Hersteller der Platte beliebig festgelegt werden. (Es ist darüber hinaus für versierte Shell-Anwender leicht, diese Namen zu ändern.) Es hat sich aber die Gewohnheit eingebürgert, die ersten beiden Festplattenlaufwerke DH0: und DH1: oder HD0: und HD1: zu nennen. Denken Sie bitte bei allen diesen Laufwerksnamen daran, daß Sie sie nur dann in einem Pfadnamen verwenden können, wenn das entsprechende Gerät auch vorhanden und an den Amiga angeschlossen ist. Ein Amiga ohne externes Diskettenlaufwerk »kennt« den Gerätenamen DF1: nicht.

Zwei ganz besondere Laufwerke sind RAM: und RAD:. Das Gerät RAM: haben Sie bereits in den ersten Kapiteln dieses Buches kennengelernt. Es handelt sich dabei um eine sogenannte »RAM-Disk«, eine Diskette, die im RAM-Speicher des Amiga simuliert wird. Auf der Workbench taucht diese Diskette als Piktogramm namens *RAM Disk* auf. Eine solche RAM-Disk ist wesentlich schneller als jede Diskette oder Festplatte – aber auch wesentlich flüchtiger. Ihr Inhalt geht beim Ausschalten oder auch schon bei einem Neustart verloren. Die RAM-Disk RAD: ist etwas robuster: Sie überlebt den Neustart, nicht aber das Ausschalten des Amiga. Sie muß, bevor Sie verwendet werden kann, allerdings erst »eingerichtet« werden. Geben Sie dazu die folgenden Befehle in einem Shell-Fenster ein:

```
MOUNT RAD:
DIR RAD:
```

Erst dann existiert ein Gerät namens RAD: und auch auf der Workbench taucht das Piktogramm einer weiteren Diskette mit dem Namen RAMB0 auf. Der Inhalt dieses RAM-Disk geht auch bei einem Neustart nicht verloren (lediglich beim Abschalten des Amiga). Sie hat aber auch einen Nachteil: Anders als die RAM-Disk RAM:, die immer nur soviel RAM-Speicher verbraucht, wie gerade für die darauf gespeicherten Dateien benötigt wird, schlägt RAD: gleich unbarmherzig zu. Sobald Sie den MOUNT-Befehl aufrufen, werden mehr als 240 Kbyte RAM-Speicher reserviert und sind für andere Zwecke verloren (auch, wenn nur 20 Kbyte Daten auf RAD: liegen). Das ist gleichzeitig auch die maximale Größe der RAD-RAM-Disk. (Diese maximale Größe kann allerdings von erfahrenen Shell-Anwendern geändert werden.)

Wenn Sie den von RAD: benötigten Speicher wieder freigeben wollen, ohne den Amiga auszuschalten, müssen Sie den Befehl REMRAD verwenden. Dabei wird der Inhalt der RAM-Disk allerdings auch gelöscht.

NIL: ist ebenfalls ein ganz besonderes »Gerät«. Manchmal kann es nämlich auch sinnvoll sein, zum Beispiel für Testzwecke Daten »ins Nichts« zu schicken. Dazu dient NIL:. Kopiert man zum Beispiel eine Datei zum Gerät NIL:, so verschwinden die Daten auf Nimmerwiedersehen in diesem Gerät. NIL: wird wahrscheinlich nur von fortgeschrittenen Benutzern und auch von diesen nur sehr selten benötigt. Auch der »Noch-nicht-Profi« kann NIL: aber sinnvoll einsetzen. Sie können mit dem folgenden Befehl nämlich erreichen, daß eine Datei komplett gelesen wird. Sie können so feststellen, ob die Datei physikalisch noch in Ordnung ist oder ob sie einen Diskettenfehler enthält, der ein Lesen verhindert (in diesem Fall erscheint eine entsprechende Fehlermeldung am Bildschirm).

```
COPY dateiX to nil:
```

Etwas »bodenständiger« als NIL: sind die Geräte SER:, PAR: und PRT:. SER: bezeichnet die serielle Schnittstelle (Anschlußbuchse) an der Rückseite des Amiga, PAR: die parallele Schnittstelle. Sie können verschiedene Geräte an diese Stecker anschließen, die sowohl Daten empfangen als auch senden können. Dies können zum Beispiel Drucker und Geräte zur Telekommunikation (Modems und ähnliche) sein. Die Software des Amiga sorgt dafür, daß sich solche Geräte wie Dateien verhalten, um etwas einfacher damit umgehen zu können. Je nachdem, welches Gerät Sie an welcher Schnittstelle angeschlossen haben, können Sie sowohl Ein- als auch Ausgaben nach SER: und PAR: umleiten. Von Ausnahmen abgesehen, werden Sie diese beiden Gerätenamen aber trotzdem sehr selten benötigen, da mit den Geräten, die Sie dort anschließen, meist auch Programme kommen, die sich um solche Details selbst kümmern. Ein Modem an der seriellen Schnittstelle werden Sie zum Beispiel nie benutzen, indem Sie eine Datei nach SER: kopieren. Sie werden statt dessen ein sogenanntes Terminal- oder DFÜ-Programm verwenden, für dessen Benutzung Sie gar nicht wissen müssen, daß das Modem etwas mit dem Gerät SER: zu tun hat.

Das Gerät PRT: hingegen werden Sie vielleicht häufiger verwenden. PRT: steht für *printer* und bezeichnet immer den Drucker, unabhängig davon, an welche Schnittstelle Sie ihn angeschlossen haben. Kopieren Sie eine (Text-) Datei nach PRT:, so wird diese im allgemeinen ausgedruckt. PRT: kümmert sich dabei automatisch um solche Feinheiten wie den Typ des angeschlossenen Druckers, wie schnell man mit ihm kommunizieren kann, ob er farbig drucken kann, und so weiter. Verwenden Sie stets PRT:, wenn Sie einen Text ausdrucken wollen und nicht SER: oder PAR:, selbst wenn Sie genau wissen, daß Ihr Drucker an der seriellen Schnittstelle angeschlossen wurde! Ansonsten müssen Sie natürlich auch noch beachten, daß PRT: nur Daten empfangen und keine abgeben kann – im Gegensatz zu SER: und PAR:, die, wenn die entsprechenden Geräte angeschlossen sind (keine Drucker), auch als Eingabe»dateien« verwendet werden können.

Wenn Sie vom Shell-Fenster aus eine Datei nach PRT: schicken, sollten Sie unbedingt darauf achten, daß es wirklich eine sogenannte »reine Textdatei« ist, die nur druckbare Buchstaben und Zeilenvorschübe enthält. Der Text-Editor *MEmacs*, der im folgenden Kapitel beschrieben wird, erzeugt zum Beispiel solche Dateien. Erstellen Sie mit einem speziellen Textverarbeitungsprogramm einen Text, in dem Sie verschiedene Farben und/oder Zeichensätze und Größen verwenden, so müssen Sie diesen im allgemeinen auch innerhalb des Textverarbeitungsprogramms ausdrucken. Das gilt zum Beispiel auch für das Programm *Notepad*. Wenn Sie damit einen Text erstellen, der unterschiedliche Schriften, Schriftgrößen und -Farben verwendet, handelt es sich nicht mehr um eine reine Textdatei. PRT: versteht solche Dateien nicht und ignoriert deshalb entweder die Angaben zur Textgestaltung oder liefert sogar völligen Unsinn auf dem Papier.

Das Gerät PIPE: kann dazu verwendet werden, Daten zwischen zwei Programmen, die gleichzeitig arbeiten (zum Beispiel in verschiedenen Shell-Fenstern), auszutauschen. Zu diesem Zweck wird eine sogenannte *Pipeline* aufgebaut, in die das eine Programm Daten schreibt, die das andere Programm nahezu gleichzeitig lesen kann. Ohne eine Pipeline müßten diese Daten erst auf der Diskette gespeichert werden und würden dort Platz wegnehmen (zudem wäre die Datenübertragung langsamer). PIPE: ist aber nur für fortgeschrittene Shell-Anwender interessant und dürfte auch von diesen nur selten benötigt werden. Lediglich Programmierer, die eine Hochsprache wie C oder Pascal verwenden, könnten PIPE: auf Anhieb bei der Programmübersetzung verwenden, da dabei normalerweise viele Zwischendateien auf der Diskette erzeugt werden, die man mit PIPE einsparen kann.

Auch das Gerät AUX: ist nur für sehr wenige fortgeschrittene Anwender interessant. Es erlaubt es, von einem anderen Computer (oder einem sogenannten Terminal) aus mit Ihrem Amiga zu arbeiten. Dazu muß dieser andere Computer über die serielle Schnittstelle mit dem Amiga verbunden werden. Das kann mit einem kurzen Kabel geschehen; die Verbindung kann aber auch über eine Telefonleitung und damit über nahezu beliebig große Strecken erfolgen. Der Benutzer des anderen Computers kann allerdings lediglich Shell-Befehle benutzen. Er kann von seinem Arbeitsplatz aus nicht die Workbench verwenden.

Um ein »Shell-Fenster« für den anderen Computer zu öffnen, müssen Sie AUX: in einem NEWSHELL oder NEWCLI-Befehl verwenden:

```
NEWSHELL AUX:
```

Dieses »Shell-Fenster« taucht aber nicht am Bildschirm des Amiga auf, sondern entspricht dem Bildschirm des anderen Computers. Genaugenommen wird überhaupt kein Fenster geöffnet, sondern lediglich ein neuer Shell-Prozeß gestartet, der seine Befehle vom Gerät AUX: (also von der seriellen Schnittstelle) empfängt und alle Ausgaben, die normalerweise in das Shell-Fenster gehen würden, an das Gerät AUX: schickt.

Während PIPE: und AUX: wirklich nur etwas für Amiga- und Shell-Profis sind, kann SPEAK: von jedem Shell-Anwender benutzt werden. (Meist wird sich diese Benutzung aber wohl auf ein oder zwei spielerische Versuche beschränken.) Das Gerät SPEAK: tut genau das, was sein Name andeutet: es spricht (engl. *to speak* = sprechen). Wenn Sie die Ausgabe eines Befehls auf das Gerät Speak umleiten, erscheint der entsprechende Text nicht am Bildschirm, sondern ertönt aus dem Lautsprecher – sofern Sie einen Lautsprecher an die Tonausgänge an der Rückseite des Amiga angeschlossen haben. Der folgende Befehl spricht zum Beispiel den Inhalt des Directory *c* auf der Shell-Diskette aus:

```
DIR > SPEAK: c
```

Hören Sie einmal genau hin. Nach kurzer Zeit hat man sich an die Aussprache des Amiga gewöhnt und kann die Dateinamen wirklich verstehen. Genauso können Sie sich mit dem COPY-Befehl auch eine Textdatei vorlesen lassen, indem Sie die Datei nach SPEAK: kopieren. Dabei muß man allerdings erwähnen, daß der Amiga gewisse Schwierigkeiten mit nicht-englischen Texten hat und mit einem stark amerikanischen Akzent spricht.

Sie können zur Beeinflussung der Sprachausgabe mit SPEAK: dieselben Optionen verwenden, die in Abschnitt 5.2.3 zusammen mit dem Programm *Say* beschrieben wurden. Sie müssen diese Optionen dazu direkt hinter dem Doppelpunkt von SPEAK: angeben und das Schlüsselwort OPT voranstellen. Statt die Optionen mit einem Minuszeichen einzuleiten, müssen Sie aber in diesem Fall mit einem Schrägstrich beginnen und müssen weiterhin direkt aneinandergesetzt werden (ohne Leerstellen). Der folgende Befehl gibt beispielsweise die Datei StartupII auf dem logischen Laufwerk S: mit einer weiblichen, recht hoch und schnell sprechenden Stimme aus:

```
COPY S:StartupII SPEAK:OPT/F/S300/P300
```

Die Geräte CON:, NEWCON: und RAW: entsprechen den »Geräten« des Amiga, mit denen Sie wohl am meisten zusammenarbeiten: der Tastatur und dem Bildschirm. Zusammengefaßt nennt man Tastatur und Bildschirm in der Computerbranche oft Konsole (engl. *console*). Alle drei Geräte können sowohl als Ein- als auch Ausgabedatei verwendet werden – sie vereinen Tastatur und Bildschirm in einem Gerät. Sie werden diese Geräte aber

trotzdem recht selten verwenden. Sie können eine Textdatei zum Beispiel nicht auf den Bildschirm schicken, indem Sie sie nach CON: oder RAW: kopieren.

Dafür ist das besondere Gerät * (Sternchen) gedacht, das das aktuelle Shell-Fenster bezeichnet. Es wird normalerweise nur als Ausgabegerät benutzt. Wenn Sie die Ausgabe eines Befehls nach * schicken oder eine Datei nach * kopieren, so erscheint der entsprechende Textstrom im Shell-Fenster.

CON:, NEWCON: und RAW: werden hauptsächlich beim Erzeugen zusätzlicher Shell-Fenster benötigt. Dazu muß der entsprechende Gerätename und direkt dahinter eine Fensterspezifikation als Parameter des NEWSHELL- oder NEWCLI-Befehls verwendet werden. Auf diese Weise ist es zum Beispiel möglich, die Größe eines Shell-Fensters gleich mit dem NEWSHELL-Befehl festzulegen. Wie eine solche Fensterspezifikation aussieht, wird bei den Erläuterungen zum Befehl NEWSHELL beschrieben.

Der Unterschied zwischen einem CON: und einem NEWCON:-Fenster ist im wesentlichen auf die zusätzlichen Editiermöglichkeiten beschränkt, die Ihnen in einem Shell-Fenster zur Verfügung stehen. Verwenden Sie NEWCON:, können Sie innerhalb der Zeile Korrekturen mit Hilfe der Cursortasten und durch Einfügen von Buchstaben vornehmen. In einem mit CON: geöffneten Fenster können Sie nur jeweils den Buchstaben links von der Schreibmarke löschen (und die Schreibmarke nicht mit den Cursortasten bewegen). Das Gerät RAW: sollten Sie nicht benutzen, um ein neues Fenster zu öffnen. Es ist wirklich nur für Programmentwickler gedacht und auch diese werden es nur sehr selten benötigen.

## 7.5 Logische Laufwerke

Die folgende Tabelle gibt einen Überblick über die logischen Geräte, die standardmäßig vom Amiga-DOS benutzt werden, sowie darüber, welchen Dateiverzeichnissen sie beim Start zugeordnet werden und wozu sie dienen. <Startdiskette> bezeichnet in dieser Tabelle die Workbench-Diskette, mit der der Amiga gestartet wurde.

| Gerätename | Dateiverzeichnis | Speicherort für |
|---|---|---|
| SYS: | <Startdiskette>: | Workbench-Diskette |
| C: | SYS:c | Shell-Kommandos |
| CLIPS: | RAM-Disk:clipboards | Zwischenspeicher |
| DEVS: | SYS:devs | Gerätetreiber |
| ENV: | RAM-Disk:env | Umgebungsvariable |
| FONTS: | SYS:fonts | Zeichensätze |
| L: | SYS:l | System-Libraries |
| LIBS: | SYS:libs | Libraries |
| S: | SYS:s | Kommandofolgen |
| T: | RAM-Disk:t | Temporäre Dateien |

***Tabelle 7.2:*** *Logische Geräte des Amiga*

Diese Liste oder auch eine noch längere können Sie sich – ohne die Erläuterungen in der dritten Spalte natürlich – anzeigen lassen, indem Sie im Shell-Fenster ASSIGN ohne Parameter eingeben und [Return] drücken. Sie erfahren dann, welche logischen Laufwerke im Augenblick existieren und welchen Directories auf welchen Disketten sie zugeordnet sind.

Einige der Bezeichnungen in der dritten Spalte dürften Ihnen (noch) nicht viel sagen. Hierzu müßten Sie noch mehr über Konzepte wissen, die erst im letzten Teil dieses Buches erläutert werden. Diese logischen Laufwerke werden der Vollständigkeit halber aber hier trotzdem schon mit aufgeführt.

Eine zentrale Bedeutung bei fast allen logischen Geräten nimmt SYS: ein. Dies ist der logische Laufwerkname für die Diskette, mit der der Amiga gestartet wurde (die »Workbench-Diskette«). Benötigt man irgendwelche Dateien oder Dateiverzeichnisse auf der Startdiskette, so kann man immer den Namen SYS: für das Wurzel-Dateiverzeichnis verwenden, egal in welchem Laufwerk die Diskette liegt, oder ob sie überhaupt in einem Laufwerk ist.

C: ist das logische Laufwerk, in dem die Programme gesucht werden, die die Shell-Befehle (commands) ausführen. Angenommen, Sie haben auf einer anderen Diskette als der Startdiskette einen vollständigeren Satz von Shell-Befehlen, der sich dort ebenfalls im Dateiverzeichnis C befindet. Der folgende Befehl sorgt dann dafür, daß die Shell immer, wenn Sie einen Befehl eintippen, dort nach dem entsprechenden Programm sucht (und überhaupt nicht mehr auf der Startdiskette):

```
ASSIGN C: AndereDisk:c
```

Das logische Laufwerk C: wird allerdings erst dann nach Programmen abgesucht, wenn in den anderen Directories im Suchpfad kein Programm gefunden werden konnte.

Die Geräte L: und LIBS: enthalten Programmbibliotheken (engl. *Libraries*), die vom System beziehungsweise von anderen Programmen genutzt werden können. Diese Bibliotheken werden erst dann für Sie interessant, wenn Sie beginnen, den Amiga 500 selbst zu programmieren. Es ist selten nötig, die Zuordnung von L: oder LIBS: zu einem Dateiverzeichnis zu ändern.

Im logischen Laufwerk DEVS: (für *devices* oder zu deutsch Geräte) befinden sich die Gerätetreiber. Deren Namen enden alle mit ».device«, wovon Sie sich mit Hilfe des DIR-Befehls jederzeit überzeugen können. Gerätetreiber enthalten, ähnlich wie Bibliotheken, kleine Programmteile, die von anderen Programmen benötigt werden. In diesem Fall helfen diese Programmteile, die verschiedenen Geräte, die man an die Schnittstellen an der Rückseite Ihres Amiga 500 anschließen kann, zu bedienen. Dies können ja sehr verschiedene Geräte sein, die man beim Schreiben der Betriebssystem-Software nicht schon alle vorhersehen konnte. Deshalb ist es sinnvoller, deren Steuerung auf solche Geräte-Treiber zu verlagern, die man problemlos austauschen kann, wenn Sie einmal ein neues Gerät bekommen.

Auch »Pseudo-Geräte« wie zum Beispiel RAM:, RAD: und SPEAK: werden durch Gerätetreiber simuliert, die sich im Laufwerk DEVS: befinden. Weiterhin liegt im logischen Laufwerk DEVS: auch das Directory *printers*, das die verschiedenen Druckertreiber enthält, die Ihnen im Programm *Preferences* zur Auswahl angeboten werden.

Das logische Laufwerk FONTS: enthält die Dateien, die die verschiedenen Schriftarten beschreiben, die Sie zum Beispiel in *Notepad* verwenden können. Sie wurden oben bereits erläutert.

Auf dem Laufwerk S: schließlich befinden sich die »Kommandofolgen« oder »Kommandosequenzen«. Kommandofolgen bieten die Möglichkeit, eine Folge von Shell-Befehlen zu einem neuen Befehl zusammenzufassen. Wenn Sie versuchen, eine solche Kommandofolge aufzurufen und die Shell findet sie nicht im aktuellen Directory, so sucht es im Laufwerk S: danach. Mehr zu Kommandofolgen erfahren Sie noch in einem der folgenden Kapitel.

Viele der üblichen Zuordnungen von logischen Laufwerken zu Dateiverzeichnissen werden Sie kaum jemals ändern müssen. Es kann jedoch in einem Fall nötig werden, sie alle zu ändern. Falls Sie sehr viel RAM-Speicher in Ihrem Amiga oder gar eine Festplatte besitzen, so kann es sehr sinnvoll sein, die RAM-Disk RAD: oder die Festplatte als Startdiskette zu verwenden. Dies ist vor allem deshalb »sehr sinnvoll«, weil dann alle Vorgänge wesentlich schneller ablaufen. Festplatten, und erst recht das RAM-Laufwerk, sind Disketten von der Geschwindigkeit her weit überlegen. Mehr zu diesem Thema können Sie in Kapitel 8 ertfahren.

## 7.6 Überblick

Zum Abschluß dieses Kapitels folgt nun eine Liste mit allen Shell-Befehlen. Diese Liste dürfte meist für die tägliche Arbeit mit der Shell ausreichen. Falls Sie die genauen Parameter, die ein bestimmter Befehl hat, vergessen haben, so können Sie diese meist mit der sogenannten »Help-Funktion« erfahren. Tippen Sie dazu einfach den Namen des Befehls, dessen Parameter Sie vergessen haben, und ein Fragezeichen ein und drücken Sie auf [Return]. Die Namen der möglichen Parameter und die Eigenschaften (optional, nichtoptional, mit Schlüsselwort und so weiter) werden Ihnen dann gleich mitgeteilt. Die Codes /A, /K und /S, die hinter den Parameternamen auftauchen können, werden am Anfang dieses Kapitels erläutert.

| | |
|---|---|
| ADDBUFFERS | reserviert Pufferspeicher zur Beschleunigung des Zugriffs auf Disketten und Festplatten. |
| ASK | erbittet (in Kommandofolgen) eine Ja-Nein-Entscheidung vom Anwender. |
| ASSIGN | ändert Zuordnung eines Directory zu einem logischen Laufwerk (kann für Abkürzungen mißbraucht werden). |
| ALIAS | legt einen Alias-Namen für einen Shell-Befehl fest. |
| AVAIL | ändert Zuordnung eines Directory zu einem logischen Laufwerk (kann für Abkürzungen mißbraucht werden). |
| BINDDRIVERS | aktiviert die Treiberdateien in der Expansion-Schublade. |
| BREAK | bricht ein Programm in einem anderen Shell-Prozeß ab. |
| CD | ändert das aktuelle Directory. |

| | |
|---|---|
| CHANGETASKPRI | ändert die Priorität eines Shell-Prozesses gegenüber anderen gleichzeitig aktiven Prozessen (Programmen). |
| CLOCK | ruft die Amiga-Uhr auf. |
| CLOCKPTR | macht aus dem Mauszeiger eine Digitaluhr. |
| COPY | kopiert eine Datei oder eine Gruppe von Dateien. |
| CMD | lenkt Druckausgaben in eine Datei um. |
| DATE | zeigt oder ändert das intern gespeicherte Systemdatum und die Systemzeit. |
| DELETE | löscht eine Datei oder eine Gruppe von Dateien. |
| DIR | listet den Inhalt eines Directory auf. |
| DISKCHANGE | meldet einen Diskettenwechsel bei einem Laufwerk, das einen solchen Vorgang nicht automatisch dem Amiga-DOS mitteilt. |
| DISKCOPY | kopiert komplette (nicht kopiergeschützte) Disketten. |
| DISKDOCTOR | repariert defekte Datenträger und stellt versehentlich gelöschte Dateien wieder her. |
| ECHO | gibt einen Text am Bildschirm aus. |
| ED | ruft den Fullscreen-Texteditor auf. |
| EDIT | ruft den Zeilen-Texteditor auf. |
| ELSE | leitet die zweite Hälfte einer IF-ELSE-ENDIF-Gruppierung zur bedingten Ausführung einer Teilfolge von Befehlen ein. |
| ENDCLI | beendet die Shell und schließt deren Fenster. |
| ENDIF | beendet eine IF-ELSE-ENDIF-Gruppierung zur bedingten Ausführung einer Teilfolge von Befehlen. |
| ENDSHELL | beendet die Shell und schließt deren Fenster. |
| ENDSKIP | markiert das Sprungziel eines SKIP-Befehls ohne Angabe eines Namens. |
| EVAL | berechnet einen arithmetischen Ausdruck. |
| EXECUTE | führt eine Kommandofolge aus. |
| FAILAT | legt fest, ob und bei welchen Fehlern Kommandofolgen fortgesetzt werden dürfen, wenn während der Ausführung einer Kommandofolge Fehler auftreten. |
| FASTMEMFIRST | sorgt dafür, daß bei Speicheranforderungen zuerst der RAM außerhalb des Chip-RAM verwendet wird. |
| FAULT | zeigt den Fehlertext zu einer Fehlernummer an. |
| FF | aktiviert die FastFonts-Routinen zur Beschleunigung der Textausgabe. |
| FILENOTE | versieht eine Datei mit einem Kommentar. |

| | |
|---|---|
| FORMAT | formatiert und initialisiert eine Diskette oder Festplatte. |
| GETENV | gibt den Wert einer Umgebungsvariablen aus. |
| GRAPHICDUMP | druckt den Bildschirm aus. |
| ICONX | führt eine,von der Workbench aus mit einem Doppelklick gestartete Kommandofolge aus. |
| IF | leitet eine IF-ELSE-ENDIF-Gruppierung zur bedingten Ausführung einer Teilfolge von Befehlen ein. |
| INFO | listet eine Reihe von Informationen über die zur Verfügung stehenden Laufwerke und die darin befindlichen Disketten auf (zum Beispiel den noch freien Platz). |
| INITPRINTER | initialisiert den Drucker. |
| INSTALL | macht aus einer formatierten Diskette eine bootfähige (startfähige) Diskette. |
| JOIN | verbindet bis zu 15 einzelne Dateien zu einer neuen Datei. |
| LAB | versieht eine Stelle in einer Kommandofolge mit einer Sprungmarke. |
| LOCK | aktiviert den Schreibschutz einer Festplatte. |
| LIST | listet detaillierte Informationen über ein Directory oder eine Gruppe von Dateien auf. |
| LOADWB | startet die Workbench-Software. |
| MAKEDIR | erzeugt ein neues Directory. |
| MEMACS | startet den mausgesteuerten Texteditor MEMACS. |
| MERGEMEM | fügt Speicherblöcke in verschiedenen Speichererweiterungen zusammen. |
| MORE | zeigt eine Textdatei komfortabler an als TYPE. |
| MOUNT | meldet ein zusätzliches Peripheriegerät mit seinem Namen beim Amiga-DOS an. |
| NEWCLI | erzeugt einen neuen CLI-Prozeß in einem CON:-Fenster. |
| NEWSHELL | erzeugt einen neuen Shell-Prozeß in einem NEWCON:-Fenster. |
| NOFASTMEM | Sperrt den RAM-Bereich, der außerhalb des Chip-RAM liegt. |
| PATH | ändert den aktuellen Suchpfad für Programme oder zeigt ihn an. |
| PREFERENCES | ruft das Programm *Preferences* auf. |
| PRINTFILES | druckt eine oder mehrere Textdateien aus. |
| PROMPT | ändert die Meldung, mit der die Shell meldet, daß es zum Empfang des nächsten Befehls bereit ist. |

| | |
|---|---|
| PROTECT | setzt die Schutzflags einer Datei (wodurch diese zum Beispiel vor unbeabsichtigtem Löschen geschützt wird). |
| QUIT | verläßt eine Kommandofolge »mittendrin« und gibt eventuell einen Fehlercode zurück. |
| RELABEL | ändert den Namen einer Diskette. |
| REMRAD | entfernt die resetfeste RAM-Disk RAD:. |
| RENAME | benennt eine Datei um und/oder legt sie in ein anderes Directory. |
| RESIDENT | macht einen Befehl resident, sorgt also dafür, daß er im Speicher gehalten wird, nachdem er das erste Mal aufgerufen wurde. |
| RUN | läßt einen oder mehrere Shell-Befehle im Hintergrund ablaufen. |
| SAY | spricht einen Text aus. |
| SEARCH | durchsucht alle Dateien eines Directory nach einem bestimmten Text. |
| SETCLOCK | stellt die batteriegepufferte Uhr in der Speichererweiterung Amiga 501 oder übergibt deren Uhrzeit an Amiga-DOS. |
| SETDATE | ändert Datum und Uhrzeit des Zeitpunkts, an dem eine Datei (angeblich) das letzte Mal modifiziert wurde. |
| SETENV | ändert den Wert einer Umgebungsvariablen oder definiert eine neue Umgebungsvariable. |
| SETPATCH | beseitigt Fehler in der Systemsoftware des Amiga. |
| SKIP | springt zu einer mit LAB definierten Marke. |
| SORT | sortiert Textdateien alphabetisch nach der ersten oder einer beliebigen Spalte in jeder Zeile. |
| STACK | ändert die Größe des Stack-Speichers, der beim Start eines anderen Programms von der Shell reserviert wird. |
| STATUS | zeigt verschiedene Informationen über andere Shell-Prozesse, die im Moment aktiv sind. |
| TYPE | gibt eine Datei auf dem Bildschirm aus (auf Wunsch hexadezimal oder mit Zeilennummern). |
| WAIT | wartet eine bestimmte Zeitspanne oder bis zu einem bestimmten Zeitpunkt mit der Ausführung des nächsten Befehls. |
| WHICH | gibt den Pfadnamen des Programms aus, das einen Shell-Befehl ausführen würde. |
| WHY | zeigt (als Text) die Ursache des letzten Fehlers, der sich bei der Ausführung eines Programms ereignet hat. |

# 8 Die Shell für Fortgeschrittene

Obwohl der Amiga ein stark grafisch orientierter Computer mit einer entsprechend grafisch gestalteten Benutzeroberfläche ist, werden Profis immer wieder zur Shell greifen. Viele Aufgaben lassen sich nur in einem Shell-Fenster erledigen, andere gehen hier einfach schneller. Im Kapitel 6 dieses Buches haben Sie eine Einführung in die Arbeit mit Shell gefunden. Zusammen mit den Beschreibungen der Shell-Befehle in Kapitel 7 und etwas Experimentierfreude sollte es möglich sein, anhand dieser Einführung mit der Shell vertraut zu werden und nach kurzer Zeit genauso schnell und sicher zu arbeiten, wie auf der Workbench. Für den, der noch etwas tiefer in die Geheimnisse der Shell einsteigen möchte, ist dieses Kapitel gedacht. Dazu möchte ich aber vorher erwähnen, daß die Erläuterungen in den folgenden Abschnitten von der Annahme ausgehen, daß Sie Kapitel 6 sorgfältig durchgearbeitet haben, sich zumindestens einen Überblick über die Befehlsbeschreibungen in Kapitel 7 verschafft haben und darüber hinaus auch schon etwas mit der Shell gearbeitet haben.

Weiterhin gilt auch für dieses Kapitel dieselbe Warnung, die Sie auch schon in Kapitel 6 erhalten haben: Benutzen Sie bei allen Experimenten immer eine Kopie der Workbench-Diskette. Sobald sich auf dieser Kopie wichtige Dateien befinden (die auf dem Original nicht vorhanden sind), sollten Sie auch diese Dateien in Sicherheit bringen – auf einer anderen Diskette. Im Shell-Fenster ist es relativ leicht, Dateien, oder auch eine ganze Diskette versehentlich zu löschen. Diese Gefahr muß Ihre Arbeit aber nicht weiter beeinträchtigen – wenn Sie die eben genannten Vorsichtsmaßnahmen ergreifen.

## 8.1 Arbeiten im Shell-Fenster

Bei der Arbeit im Shell-Fenster werden Sie einen großen Teil Ihrer Zeit damit verbringen, Befehle einzutippen und sich die Ergebnisse dieser Befehle zu betrachten. Lästig ist dabei sicherlich oft, daß man sich bei den of recht langen Shell-Befehlen leicht vertippen kann. Glücklicherweise gibt es aber eine Reihe von »Tricks«, die ihnen das Leben mit den langen Shell-Befehlen im allgemeinen und Tippfehlern im speziellen wesentlich leichter machen können. Diese Tricks funktionieren allerdings nur in Shell-Fenstern, die vom Amiga-DOS-Gerät NEWCON: kontrolliert werden, nicht in Fenstern, die vom Gerät CON: gesteuert

werden. (Vergleichen Sie die Beschreibung von NEWCON: und des Befehls NEWSHELL in Kapitel 7.) Benutzern der älteren Workbench-Versionen bis 1.2 stehen Sie deshalb nicht zur Verfügung, da es in diesen Workbench-Versionen das Gerät NEWCON: nicht gibt.

### 8.1.1 Das Gedächtnis der Shell

Das wichtigste »Werkzeug« bei der Fehlerkorrektur ist sicherlich die Cursor-Taste [↑], mit der Sie die jeweils zuletzt eingetippte Befehlszeile zurückholen, korrigieren und dann erneut abschicken können. Dieser wichtige Handgriff wurde ja bereits in Kapitel 6 beschrieben. Die Cursor-Taste [↑] kann aber noch mehr. Wenn Sie sie mehrmals hintereinander betätigen erscheinen Schritt für Schritt alle zuvor eingegebenen Befehlszeilen. Mit jedem erneuten Druck auf die Taste gehen Sie einen Befehl zurück. Wenn Sie dabei »über das Ziel hinausschießen«, können Sie mit [↓] auch wieder zurück – also zu später erteilten Befehlen – gehen. Das »Gedächtnis« der Shell, das Sie mit diesen beiden Tasten durchblättern können, reicht weit zurück. Sie können damit auch Befehle zurückholen, die Sie vielleicht Stunden zuvor eingegeben haben – wenn Sie das Fenster nicht inzwischen geschlossen haben. Jedes Shell-Fenster hat nämlich ein eigenes Gedächtnis.

Damit bleibt eigentlich nur noch die Frage offen, wozu dieses Gedächtnis gut ist. Daß es praktisch sein könnte, den letzten fehlerhaften Befehl zurückzuholen, um ihn zu korrigieren, ist leicht einsehbar. Welchen Sinn aber könnte es haben, einen Befehl »hervorzukramen«, den man vor einer halben Stunde eingetippt hat? Die Antwort auf diese Frage ist simpel: es ist eine enorme Arbeitsersparnis! Gerade, wenn Sie oft Befehle mit vielen und langen Parametern verwenden, müssen Sie bei der Eingabe sehr sorgfältig vorgehen. Befehl und Parameter müssen ja schließlich fehlerfrei eingetippt werden. Und genau hier liegt der Nutzen des Shell-Gedächtnisses. Denn selbst, wenn Sie stundenlang mit der Shell arbeiten, werden Sie dabei meistens mit einer Handvoll von Befehlen auskommen (zum Beispiel DIR, LIST, CD, EXECUTE und so weiter). Das einzige, was sich von Befehl zu Befehl unterscheidet, werden oft einzelne Parameter sein. In vielen Fällen wird es sogar vorkommen, daß Sie exakt den selben Befehl immer wieder verwenden. Und da ist es wesentlich praktischer, diesen Befehl mit Hilfe der Taste [↑] zurückzuholen, als ihn noch einmal korrekt einzutippen. Aber auch bei einer Vielzahl ähnlicher Befehle, die sich nur bei einem Parameter unterscheiden, kann man sich mit [↑] die Arbeit deutlich leichter machen. Man holt sich mit [↑] den Befehl zurück, »korrigiert« ihn – ändert also den Parameter – und schickt ihn dann mit einem Druck auf [Return] wieder ab.

Um diese Arbeitsweise noch weiter zu erleichtern, gibt es einen weiteren Trick, der besonders dann hilfreich ist, wenn der gesuchte Befehl schon eine Weile weiter zurückliegt. Geben Sie dann einfach die ersten paar Buchstaben des gesuchten Befehls ein und halten Sie die [Shift]-Taste fest, bevor Sie die Taste [↑] drücken. Die Shell holt dann den jeweils letzten Befehl hervor, der mit der vorher eingegebenen Buchstabenfolge beginnt. Um einen EXECUTE-Befehl mit vielen komplizierten Parametern zu finden, würde es beispielsweise genügen, »EX« einzugeben und dann die Tastenkombination [Shift]+[↑] zu drücken. (Die Tastenkombination [Shift]+[↓] bringt Sie übrigens immer an das Ende des Gedächtnisses also in die leere Befehlszeile, mit der Sie begonnen haben.)

Versuchen Sie bei Ihrer Arbeit mit der Shell, deren Gedächtnis so intensiv zu nutzen wie möglich! Wenn Sie sich erst einmal an diese Arbeitsweise gewöhnt haben, werden Sie sicherlich feststellen, wie effizient diese Methode ist und um wieviel schneller und

problemloser Sie nun arbeiten können. Erfahrene Shell-Benutzer dürften nur noch sehr selten vollständige Befehle neu eingeben. Im Allgemeinen ist nach kurzer Zeit das Gedächtnis der Shell mit den gebräuchlichsten Befehlen gefüllt, die man dann nur noch abrufen und ändern muß.

### 8.1.2 Korrigieren einer Befehlszeile

Um diese Arbeitsweise noch weiter zu vereinfachen, gibt es in einem Shell-Fenster (das vom Gerät NEWCON: kontrolliert wird) recht komfortable Möglichkeiten zur Durchführung von Korrekturen innerhalb einer Befehlszeile, die in Kapitel 6 noch nicht alle vorgestellt wurden. Damit kann man sehr schnell früher eingegebene Befehle beziehungsweise deren Parameter ändern.

Am wichtigsten beim Korrigieren einer Befehlszeile sind die Cursor-Tasten [→] und [←]. Damit können Sie die Schreibmarke innerhalb der aktuellen Zeile nach rechts und links bewegen. Wenn Sie dann neue Buchstaben eingeben, erscheinen diese immer bei der Schreibmarke und verschieben die rechts von der Schreibmarke stehenden Buchstaben weiter nach rechts.

Wenn Sie keine Buchstaben einfügen, sondern welche löschen wollen, stehen Ihnen mehrere Möglichkeiten zur Wahl. Die Tasten [Backspace] und [Del] löschen jeweils nur einen Buchstaben links und rechts von der Schreibmarke. Die Tastenkombination [Ctrl]+[U] löscht alle Buchstaben links von der Schreibmarke bis zum Anfang der Zeile und [Ctrl]+[K] löscht die Buchstaben rechts von der Schreibmarke bis zum Ende der Zeile. [Ctrl]+[X] löscht die gesamte Zeile. Das können Sie aber auch mit [Shift]+[↓] erreichen.

| **Taste(nkombination)** | **Wirkung** |
|---|---|
| [↑] | vorangehende Zeile aus dem Shell-Gedächtnis holen. |
| [↓] | folgende Zeile aus dem Shell-Gedächtnis holen. |
| [Shift]+[↑] | einen Befehl suchen. |
| [Shift]+[↓] | zur letzten, leeren Zeile des Shell-Gedächtnisses gehen. |
| [→] | Schreibmarke nach rechts. |
| [←] | Schreibmarke nach links. |
| [Shift]+[→] | Schreibmarke an das Ende der Zeile. |
| [Shift]+[←] | Schreibmarke an den Anfang der Zeile. |
| [Ctrl]+[W] | Schreibmarke zur nächsten Tabulatorposition. |
| [Backspace] | Buchstabe links von der Schreibmarke löschen. |
| [Del] | Buchstabe rechts von der Schreibmarke löschen. |
| [Ctrl]+[U] | alle Buchstaben links von der Schreibmarke löschen. |
| [Ctrl]+[K] | alle Buchstaben links von der Schreibmarke löschen. |
| [Ctrl]+[X] | die ganze Zeile löschen. |

***Tabelle 8.1:*** *Wichtige Tastenkombinationen in einem Shell-Fenster*

In langen Befehlszeilen ist es oft auch lästig, sehr oft auf die Cursor-Tasten [→] und [←] drücken zu müssen, um zu einer bestimmten Stelle in der Zeile zu kommen. Für solche Fälle hält die Shell (beziehungsweise NEWCON:) drei weitere Tastenkombinationen bereit: [Shift]+[←] setzt die Schreibmarke an den Anfang der Zeile und [Shift]+[→] an ihr Ende. Und mit der Tastenkombination [Ctrl]+[W] springen Sie jeweils zur nächsten »Tabulatorposition« rechts von der aktuellen Position der Schreibmarke. Diese Tabulatorpositionen liegen jeweils acht Buchstaben auseinander und verhalten sich zusammen mit [Ctrl]+[W] ähnlich wie Tabstopps an einer Schreibmaschine. Sie können mit [Ctrl]+[W] in größeren Schritten durch eine lange Zeile springen. (Die [Tab]-Taste kann für diesen Zweck leider nicht genutzt werden!)

## 8.2 Textbearbeitung in der Shell

Eine Aufgabe, die – nicht nur bei der Arbeit mit der Shell – immer wieder anfällt, ist das Erzeugen und Ändern von Textdateien. Ob dies einfache Notizen, komplexe Programme oder Kommandofolgen mit Shell-Befehlen sind, immer wird ein Werkzeug benötigt, das diese Aufgabe so gut wie möglich unterstützt. Ein solches Werkzeug wird üblicherweise »Texteditor« genannt. Für den Amiga 500 gibt es bereits viele derartige Texteditoren. Einige davon erhalten Sie beim Kauf des Amiga 500 schon dazu: ED, EDIT, MEMACS und das von der Workbench bekannte *Notepad*. Daneben gibt es eine ganze Reihe weiterer Editoren von Drittherstellern, die hier nicht näher beschrieben werden sollen. Für die meisten Arbeiten reichen die Programme, die Sie kostenlos zu Ihrem Amiga dazubekommen, auch völlig aus.

### 8.2.1 Überblick

ED, EDIT, MEMACS sind Editoren »in Reinkultur«, mit denen reine Textdateien behandelt werden können. Es handelt sich hierbei nicht um eine komplette Textverarbeitung, die man zum Beispiel für Bürozwecke einsetzen könnte, sondern nur um ein Hilfsmittel zum Bearbeiten von unstrukturierten Texten. (Im angloamerikanischen Bereich wird diese Unterscheidung auch im verbalen Bereich etwas deutlicher. Bei einem komfortablen Textverarbeitungssystem spricht man meist von einem *word processor* und bei einem einfachen Texteditor eben auch von einem *editor* oder *text editor*.) Ein Texteditor erkennt keine Seiten, Kapitel, Absätze und vielleicht nicht einmal Worte. Er erkennt nur einen Strom von Buchstaben, der in Zeilen aufgeteilt ist. In einem Texteditor können Texte nicht fettgedruckt oder unterstrichen werden. Es ist ein sehr simples Werkzeug, das selbst für Briefe etwas spartanisch ist, aber doch wichtig für die Aufbereitung von Texten, die als Eingaben für andere Programme dienen sollen, oder für die Betrachtung von Texten, die andere Programme erzeugen.

So bestehen zum Beispiel Kommandofolgen, über die Sie später in diesem Kapitel noch Näheres erfahren werden, nur aus Buchstaben, die zu Zeilen gruppiert sind. Auch die Dateien, die entstehen, wenn man die Ausgabe eines Shell-Befehls mittels > vom Shell-Fenster auf eine Datei umleitet, sind meist einfache Textdateien.

Der Texteditor ED ist schon ein recht komfortabler Vertreter seiner Gattung. Es ist ein sogenannter Bildschirm-Editor (englisch *screen editor*), der einen Ausschnitt (ein Fenster)

aus dem bearbeiteten Text zeigt. In diesem Ausschnitt kann man sich mit den Cursor-Tasten hin- und herbewegen und an beliebigen Stellen Texte überschreiben, löschen oder ergänzen. Andere Texteditoren – die Vertreter der Gattung Zeilen-Editor (englisch *line editor*) zum Beispiel – machen die Bearbeitung eines Textes wesentlich umständlicher. Sie erlauben unter anderem nur die Bearbeitung einer aktuellen Zeile, und jeder Wechsel zu einer anderen Zeile ist mit speziellen Befehlen verbunden. Obwohl ED in mancher Hinsicht einiges zu wünschen übrig läßt, kann man mit ihm doch schon recht gut auskommen. Der ebenfalls jedem Amiga beiliegende Zeilen-Editor EDIT hat hingegen lediglich historische Bedeutung und dürfte heute nur noch von sehr wenigen Amiga-Besitzern genutzt werden.

Auch der deutlich einfacher zu bedienende Editor ED entspricht aber nicht ganz dem, was man sich auf dem Amiga unter einem komfortablen Text-Editor vorstellt. Sein Nachteil – zumindest für einen Amiga-Benutzer – ist, daß er nur über die Tastatur gesteuert wird und ihm Maus- und Menübefehle unbekannt sind. Dies ist bei MEMACS (dessen vollständiger Namen übrigens Micro-Emacs lautet) anders. Dieser Texteditor ist eine leicht vereinfachte Version des extrem leistungsfähigen Editors *emacs*, der sich in der Computerwelt – speziell auf Computern mit dem Betriebssystem UNIX – seit langem großer Beliebtheit erfreut. MEMACS **kann** allein über die Tastatur gesteuert werden, akzeptiert aber auch Menübefehle und Mausklicks. Zu fast jedem Kommando, das über die Tastatur eingegeben wird, existiert zum Beispiel auch ein Menübefehl. Und zur Bewegung der Schreibmarke im Text kann die Maus eingesetzt werden. Für den, der sehr viel Texte editieren muß – zum Beispiel Programmierer – bietet MEMACS eine Fülle von Fähigkeiten, die weit über das von ED gebotene hinausgehen. Man muß allerdings zugeben, daß es inzwischen einige andere Editoren auf dem Markt gibt, die noch leistungsfähiger sind und die Möglichkeiten der Maus und der Menübedienung besser nutzen. Diese Editoren müssen dafür aber auch extra bezahlt werden.

Neben ED und MEMACS gibt es noch eine Reihe weiterer Werkzeuge, die Sie zur Textverarbeitung benutzen können. Hierzu gehört der schon erwähnte, umständliche Zeileneditor EDIT, das Werkzeug *Notepad* oder auch das List-Fenster von Amiga-Basic. *Notepad* kann recht gut zum Editieren kleinerer Textdateien eingesetzt werden, während das List-Fenster von Amiga-Basic nur zur Bearbeitung von Basic-Programmen wirklich gut geeignet ist.

Ich werde mich darauf beschränken, nur die beiden wichtigsten Werkzeuge für die Textbearbeitung zu beschreiben: *Notepad*, für den, der möglichst einfach und ohne ein neues Programm zu erlernen, kleine Text bearbeiten will und MEMACS, für den, der auch umfangreichere Texte komfortabel handhaben möchte. Der Umgang mit MEMACS wird ausführlich in Abschnitt 8.3 beschrieben. *Notepad* sollten Sie bereits aus Kapitel 5 kennen. Lediglich die Benutzung von *Notepad* aus einem Shell-Fenster heraus wird deshalb im folgenden Abschnitt kurz beschrieben.

### 8.2.2 Wie Sie das Werkzeug Notepad als Editor »mißbrauchen«

Wenn Ihnen die leistungsfähigen Texteditoren, die speziell für die Shell gedacht sind, zu umständlich oder schwierig zu erlernen sind, bietet sich das Programm *Notepad* als ideale Notlösung an. Sie können *Notepad* dazu – wie jeden anderen Editor auch – von einem Shell-Fenster aus aufrufen mit:

```
NOTEPAD
```

Wenn Sie vermeiden wollen, daß *Notepad* das *Font*-Menü aufbaut und dazu lange die Diskette nach Schriftarten durchsucht, geben Sie dahinter einfach noch das Schlüsselwort –q an.

```
NOTEPAD -q
```

Und wenn Sie gleich eine Datei laden wollen, um Sie zu betrachten oder zu ändern, können Sie deren (Pfad-)Namen auch gleich beim Start angeben.

```
NOTEPAD -q S:Startup-sequence
```

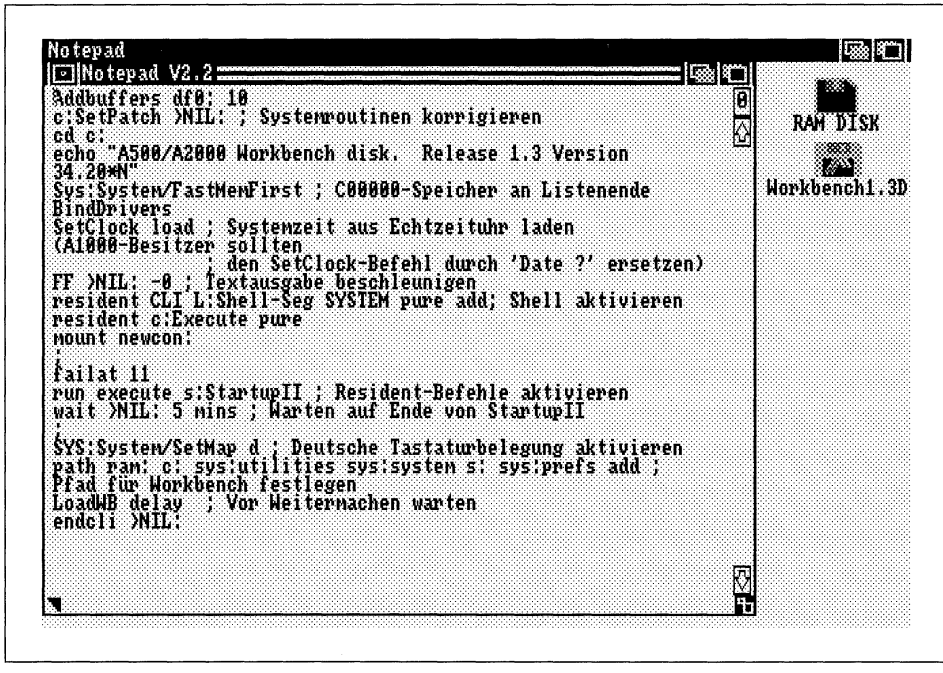



***Bild 8.1:*** *Die CLI-Textdatei Startup-Sequence im Notepad*

Sie können Textdateien aber natürlich auch nachträglich einlesen, indem Sie den Menü-Befehl *Open* aufrufen. Achten Sie aber bitte darauf, daß Sie in dem daraufhin erscheinenden Requester den vollen Pfadnamen der zu ladenden Datei angeben müssen, wenn diese nicht im aktuellen Directory liegt.

Ansonsten können Sie mit dem *Notepad* Textdateien bearbeiten wie mit jedem anderen Editor auch. Verwenden Sie aber bitte keine verschiedenen Schriftarten und Schriftstile in einer solchen Datei und beschränken Sie sich auf ein »Blatt«. Textdateien mit verschiedenen Schriftarten, -Größen und -Farben können unter anderem nicht mehr als Kommandofolgen verwendet werden. Wenn Sie in einem längeren Text nicht mehr sicher sind, ob Sie denn nun Schrift- oder Stilwechsel vorgenommen haben oder nicht, können Sie sichergehen, indem

Sie die beiden Befehle *Remove Fonts* und *Remove Styles* im Format-Menü auswählen, bevor Sie den Text wieder abspeichern.

## 8.3 Der mausgesteuerte Editor MEMACS

Während ED ein eher simpler Texteditor ist, wie man ihn oft kostenlos zu einem Computer dazu bekommt, ist MEMACS ein recht leistungsfähiges Programm, das es durchaus mit vielen kommerziellen Produkten aufnehmen kann. MEMACS kann sowohl ausschließlich über die Tastatur, als auch (nahezu) ausschließlich über die Maus und Menübefehle gesteuert werden. Er stellt damit einen fast idealen Kompromiß für ein Programm dar, das für den Anfänger leicht zu erlernen ist und dem geübten Profi trotzdem schnelles Arbeiten ermöglicht. Obwohl MEMACS eigentlich als Programmeditor »gedacht« ist, zeigt er seine Vorteile gegenüber dem simplen ED oder dem eigentlich für andere Zwecke gedachten *Notepad* auch schon bei relativ kurzen simplen Texten – wie zum Beispiel Kommandofolgen.

Das Programm ist andererseits aber so vielseitig und die Befehlsliste so lang, daß ich es im Rahmen dieses Buches nicht vollständig beschreiben kann. In den folgenden Abschnitten werden deshalb nur die wichtigsten Befehle erläutert. (Im letzten Abschnitt zum Thema MEMACS finden Sie allerdings einen kompletten Überblick über alle Befehle mit knappen Erläuterungen.) Mit etwas Experimentierfreude werden Sie aber bald auch die restlichen Befehle kennenlernen – bei einem menügesteuerten Programm ist das meist recht einfach.

### 8.3.1 Die Geschichte von MEMACS

MEMACS auf dem Amiga hat eine interessante Vorgeschichte, die ich Ihnen nicht vorenthalten möchte. Falls Sie an solchen »Histörchen« nicht interessiert sind, können Sie diesen Abschnitt natürlich übergehen und gleich mit dem nächsten fortfahren.

MEMACS (der eigentlich Micro-Emacs heißt) ist aus dem berühmten Editor »emacs« (die Kleinschreibung ist absichtlich) hervorgegangen, der zunächst nur auf Großrechnern zu finden war und sich bis heute auf UNIX-Systemen großer Beliebtheit erfreut. Dieser Editor war speziell für die Programmentwicklung geschrieben worden und wird auch meist für das Editieren von Programmtexten eingesetzt. Insbesondere die Spracheigenschaften von C und Lisp wurden und werden von *emacs* besonders gut unterstützt, ähnlich wie der Editor im List-Fenster von Amiga-Basic besonders gut mit Basic-Programmen fertig wird.

Als die ersten Mikrocomputer auftauchten und bald weite Verbreitung fanden, waren viele der ersten Benutzer ehemalige Programmierer von Großrechnern. Diese aber waren – besonders an den amerikanischen Universitäten – oft eingeschworene *emacs*-Fans und trauerten ihrem geliebten Editor nach. Dies nicht zuletzt deshalb, weil die ersten Editoren auf Mikrocomputern sehr primitiv waren. Aus dieser unglücklichen Situation und dem Bemühen, sie zu beseitigen, ging Micro-Emacs hervor. Eine »abgespeckte« Version des großen *emacs*, die vollständig in der Programmiersprache C geschrieben war und fast alle vertrauten Eigenschaften von *emacs* beinhaltet. Nur die aufwendigsten Fähigkeiten des großen *emacs* wurden aus Platz- und Zeitgründen nicht übernommen.

Da man bei der Programmierung von Micro-Emacs von vornherein darauf geachtet hatte, möglichst keine Eigenheiten eines speziellen Computers zu benutzen, tauchten in der Folgezeit für fast alle leistungsfähigen Personalcomputer Versionen des Micro-Emacs auf. Diese wurden meist mit viel Liebe von den Fans des jeweiligen Computers an ihren Rechner angepaßt und umsonst an die Kollegen Programmierer verteilt. Diese Art umsonst verteilter Software wird üblicherweise »Public-Domain-Software« genannt. Auch heute noch kann man den Programmtext von Micro-Emacs umsonst erhalten und sich mit kleinen Änderungen daran ganz persönliche Versionen dieses Editors für seinen eigenen Computer schaffen.

Auch auf dem Amiga waren die Programmierer von Anfang an mit den zur Verfügung stehenden Werkzeugen (zum Beispiel ED) sehr unzufrieden und begannen Micro-Emacs an den Amiga anzupassen. Die ersten Ergebnisse dieser Bemühungen waren noch recht anfällig und nutzten die Fähigkeiten des Amiga so gut wie überhaupt nicht. Mit der Zeit wurden dann aber Pull-down-Menüs für alle wichtigen Befehle eingeführt und auch die Maus zur Cursorsteuerung eingesetzt. So entstand die heutige Version von MEMACS. Man kann sie immer noch (inklusive Programmtext) umsonst erhalten und dann damit machen, was man will.

Da man auch bei Commodore mit ED als Programmeditor recht unzufrieden war, entschloß man sich, die damals verfügbare MEMACS-Version noch etwas aufzupolieren, Fehler zu entfernen und sie schließlich auf der sogenannten »Amiga-Toolkit-Diskette« allen interessierten Amiga-Besitzern zur Verfügung zu stellen. Ab 1988 wird MEMACS jedem Amiga auf der Extras-Diskette beigelegt. Und so kommen Sie als Anwender des Amiga 500 heute – dank der Bemühungen der internationalen Programmierer-Gemeinschaft – zu einem Texteditor, der den guten alten ED um Längen schlägt und auch für die professionelle Programmentwicklung durchaus geeignet ist!

Nach dieser etwas ausschweifenden Vorrede werden Sie nun aber auch wirklich erfahren, was es mit MEMACS auf sich hat.

### 8.3.2 MEMACS aufrufen und verlassen

Wie bei jedem anderen Programm auch, sind die wichtigsten Informationen zu MEMACS, wie man hinein- und wieder hinauskommt. Oder anders: wie man es startet und beendet. Bei MEMACS haben Sie hierzu mehrere Möglichkeiten. Welche Sie nutzen, hängt davon ab, ob Sie sich gerade auf der Workbench oder im CLI befinden und ob Sie gerne mit der Maus oder lieber mit Tastaturbefehlen arbeiten.

MEMACS befindet sich in der Schublade Tools (der augenblicklichen Version) der Extras-Diskette. Sie müssen diese Diskette nur in ein Laufwerk Ihres Amiga 500 einlegen, das Disketten-Piktogramm öffnen, das Schubladen-Piktogramm Tools öffnen und können MEMACS dann mit einem Doppelklick auf sein Piktogramm starten.

Im CLI können Sie mit dem PATH-Befehl MEMACS »auffindbar machen« und dann starten, indem Sie den Programmnamen eingeben:

```
PATH Extras1.3D:Tools
MEMACS
```

(Wenn Ihre Extras-Diskette einen anderen Namen als »Extras1.3D:« hat, müssen Sie natürlich diesen im PATH-Befehl verwenden.) Wenn Sie nur ein Laufwerk besitzen,

müssen Sie beim Aufruf von MEMACS unter Umständen zwei Diskettenwechsel durchführen. Denn für den PATH-Befehl werden nämlich die Workbench- und die Extras-Diskette und für den Aufruf von MEMACS die Extras-Diskette benötigt.

Wenn Sie vorhaben, MEMACS häufiger zu verwenden, ist es wahrscheinlich sinnvoll, ihn auf die Workbench-Diskette zu legen (und ED sowie EDIT zu löschen). Legen Sie ihn am besten (auf der Workbench) in die Schublade Utilities, in der auch *Notepad* liegt. Dieses Directory (Schublade) liegt normalerweise schon im Suchpfad. Sie können MEMACS dann vom Shell-Fenster und von der Workbench aus relativ leicht starten und können sich den soeben beschriebenen PATH-Befehl sparen.

Egal, auf welchen (Um-)Wegen Sie MEMACS starten, wenn es Ihnen gelungen ist, erscheint kurz darauf der Bildschirm von MEMACS. Das Programm öffnet einen eigenen Screen ohne Fenster darin. Dieser Screen ist blau wie der Workbench-Screen und die Schrift darin erscheint in Weiß – wenn Sie die Workbench-Farben nicht mit *Preferences* geändert haben. Er enthält eine Titelleiste am oberen Rand (die auch diverse Menüs enthält) und die üblichen Front- und Back-Gadgets, mit denen Sie den Screen hinter den Workbench-Screen legen und wieder hervorholen können.

Wenn Sie wollen, daß MEMACS keinen eigenen Screen, sondern statt dessen ein ganz normales Fenster auf dem Workbench-Screen öffnet (was etwas Speicher spart), müssen Sie beim Aufruf die Schlüsselworte OPT W angeben. Genauso können Sie auch gleich beim Start eine bestimmte Datei einlesen lassen, indem Sie deren Namen als ersten Parameter angeben. Und wenn Sie wollen, daß nach dem Start des Programms die Schreibmarke gleich in einer bestimmten Zeile des Textes steht, müssen Sie die Nummer dieser Zeile hinter dem Schlüsselwort GOTO angeben. Der folgende Befehl ruft zum Beispiel MEMACS auf, liest die Datei StartupII ein und setzt die Schreibmarke in die vierte Zeile dieses Textes:

```
MEMACS S:StartupII GOTO 4 OPT W
```

### 8.3.3 Der Bildschirmaufbau

Der Screen oder das (bildschirmfüllende) Fenster von MEMACS wird durch eine Linie geteilt, die am unteren Rand einige Zeilen optisch abtrennt. Diese Linie ist gestrichelt und enthält am linken Rand einige Informationen (zum Beispiel den Namen der gerade bearbeiteten Datei – sobald Sie eine eingelesen haben). Oberhalb der Linie steht der jeweils editierte Text. Unterhalb der Linie erscheinen Nachrichten und Fehlermeldungen. Weiterhin müssen Sie hier auch Texte eingeben, wenn dies für die Ausführung eines Befehls nötig ist. Dieser Bereich des Bildschirms beziehungsweise des Fensters hat ähnliche Aufgaben wie ein Requester und wird »Kommandozeile« oder »Befehlszeile« genannt.

Betrachten Sie sich nun einmal die Menüs von MEMACS, indem Sie bei gedrückter Menütaste mit dem Mauszeiger in der Titelleiste hin- und herfahren – aber bitte erst einmal, ohne einen Befehl daraus auszuwählen. Es sind sieben Menüs, die die gesamte Titelleiste einnehmen. Nahezu jeder dieser Menübefehle hat ein Tastaturäquivalent, das rechts neben dem Namen des Befehls angezeigt wird. Es sind aber keine üblichen Tastaturäquivalente, für die Sie die Amiga-Tasten festhalten müssen. Statt dessen werden alle Tastaturäquivalente durch Drücken der [Ctrl]-Taste, die in den Menüs durch den Dachakzent »^« symbolisiert wird, oder der [Esc]-Taste, die in den Menüs durch ESC bezeichnet ist, eingeleitet. Weiterhin bestehen diese Tastaturäquivalente manchmal nicht nur aus einer Tastenkombi-

nation, sondern aus mehreren Tasten, die Sie nacheinander drücken müssen. Das Tastaturäquivalent ESC^V bedeutet zum Beispiel, daß Sie zunächst die [Esc]-Taste drücken müssen und dann die Tastenkombination [Ctrl]+[V]. Und ^X^S bedeutet, daß Sie nacheinander die Tastenkombinationen [Ctrl]+[X] und dann [Ctrl]+[S] drücken müssen.

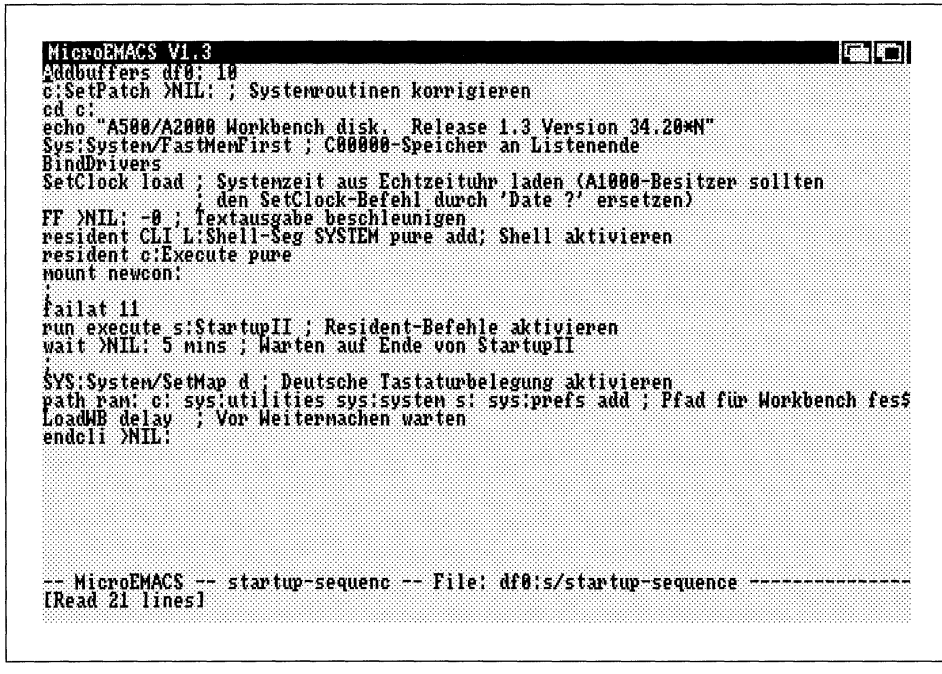



***Bild 8.2:*** *MEMACS mit der Datei* Startup-Sequence *in Bearbeitung*

Auch der Befehl zum Beenden von MEMACS hat ein solches Tastaturäquivalent. Wenn Sie MEMACS verlassen wollen, müssen Sie entweder mit der Maus den Befehl *Quit* aus dem Menü *Project* wählen oder die Tastenkombination [Ctrl]+[C] (im Menü als »^C« angezeigt) drücken, die auch bei anderen Shell-Befehlen dazu dient, Programme abzubrechen. MEMACS wird aber nicht unbedingt sofort beendet, wenn Sie [Ctrl]+[C] drücken. Wenn Sie einen Text eingegeben oder geändert haben und wollen das Programm verlassen, ohne den Text vorher gesichert zu haben, erscheint am unteren Rand des Bildschirms die Frage *Modified buffers exist, do you really want to exit [y/n]?*. Nur wenn Sie daraufhin mit »Y« oder »y« antworten und die [Return]-Taste drücken, wird das Programm wirklich verlassen. Mit der Eingabe »N« oder »n« bleiben Sie im Programm und können die Änderungen erst sichern, bevor das MEMACS beendet wird.

### 8.3.4 Texte abspeichern und einlesen

Das Einlesen und Abspeichern eines Textes läuft mit MEMACS im wesentlichen genau so ab, wie im Programm *Notepad*. Wählen Sie zum Einlesen eines Textes von Diskette den Befehl *Read-file* aus dem Menü *Project*. In der Befehlszeile am unteren Bildschirmrand

erscheint daraufhin die Meldung *Read file* und dahinter ein Doppelpunkt. Geben Sie hinter diesem Doppelpunkt den Pfadnamen einer Textdatei ein und drücken Sie auf [Return]. MEMACS liest diese Datei dann in seinen »Puffer« (englisch *buffer*) ein und präsentiert sie Ihnen im oberen Bereich des Bildschirms. In diesem Puffer (im RAM-Speicher) werden zunächst alle Änderungen durchgeführt. Gibt es noch keine Datei dieses Namens, wird eine neue angelegt. Zunächst existiert diese Datei aber erst im Puffer. Erst, wenn Sie diesen Puffer sichern, wird auch auf der Diskette oder Festplatte eine neue Datei erzeugt.

Noch einfacher als das Einlesen ist das Abspeichern eines neuen oder geänderten Textes. Wählen Sie dazu einfach den Befehl *Save* aus dem Menü *Project*. MEMACS speichert dann den aktuellen Inhalt des Puffers in die Datei, deren Name in der Trennlinie zwischen oberem und unterem Bildschirmbereich zu lesen ist. Sie können den Inhalt eines Puffers auch unter anderem Namen speichern. Wählen Sie hierzu den Befehl *Save-as-file* aus dem Menü *Project* und MEMACS fragt Sie nach dem Namen der Datei, in die der Puffer geschrieben werden soll. Wenn Sie diesen eingeben und die [Return]-Taste drücken, wird der Inhalt des Puffers in einer Datei dieses Namens abgespeichert. Wenn diese bereits existiert, wird ihr alter Inhalt überschrieben. Und schließlich hat der *Save*-Befehl auch noch eine dritte Variante. Wählen Sie *Save*-exit aus dem Menü *Project*, wird MEMACS wie bei Aufruf von *Quit* verlassen. Vorher wird aber noch der aktuelle Pufferinhalt, wie mit dem Befehl *Save-file*, gesichert.

### 8.3.5 Im Text bewegen

Genau wie im *Notepad* und allen anderen Editoren, gibt es auch in MEMACS eine Schreibmarke, auch »Cursor« oder »dot« (englisch für Punkt) genannt, die die Stelle markiert, an der neu eingegebener Text am Bildschirm erscheint. Diese Schreibmarke können Sie auf viele verschiedene Arten bewegen; einige davon kennen Sie bereits aus anderen Programmen oder auch aus dem Shell-Fenster.

Erzeugen Sie am besten eine neue Textdatei, damit Sie mit den Möglichkeiten von MEMACS etwas herumexperimentieren können. Starten Sie dazu MEMACS, geben Sie ein paar Zeilen ein (zum Beispiel einige Absätze aus diesem Buch) und speichern Sie diesen Text dann unter einem beliebigen Namen ab. Sie können ihn dann in Zukunft jederzeit für andere Experimente verwenden. Nun aber zurück zu den Befehlen und Funktionen von MEMACS:

Die Cursortasten bewegen die Schreibmarke zum Beispiel in die auf der jeweiligen Taste angegebene Richtung. Halten Sie die Shift-Taste fest, bevor Sie eine Cursor-Taste drücken, springt die Schreibmarke bis zum »Anschlag« in der jeweiligen Richtung, also bis zum oberen, unteren, rechten oder linken Rand des Bildschirms. Mit der Maus können Sie die Schreibmarke ebenfalls bewegen. Sie brauchen – wie im *Notepad* – einen Buchstaben nur anzuklicken und schon springt die Schreibmarke an diese Stelle.

Darüber hinaus kennt MEMACS aber noch eine Vielzahl anderer Möglichkeiten zur Bewegung der Schreibmarke und zur Verschiebung des am Bildschirm sichtbaren Textausschnitts. Zu jeder dieser Bewegungsmöglichkeiten gibt es einen Menübefehl im Menü *Move* (englisch *to move* = bewegen) – obwohl gerade für diese Befehle die Tastaturäquivalente oft wohl »handlicher« sind. Ich werde der besseren Lesbarkeit wegen in den folgenden Beschreibungen aber die Befehlsnamen verwenden.

Wenn Sie schnell zum Anfang oder Ende des gesamten Textes wollen, so können Sie das mit den beiden Befehlen *Top-of-buffer* beziehungsweise *End-of-buffer*. Die Befehle *Top-of-window* und *End-of-window* setzen die Schreibmarke in entsprechender Weise an den oberen beziehungsweise unteren Rand des Bildschirms beziehungsweise Fensters. Mit *Goto-Line* können Sie die Schreibmarke in eine bestimmte Zeile des Textes setzen, deren laufende Nummer Sie dazu allerdings kennen müssen. Diese Nummer wird in der Befehlszeile abgefragt (wie der Dateiname beim Einlesen), wenn Sie diesen Befehl aufrufen.

Besonders interessant ist der Befehl *Swap-dot&mark*, mit dem Sie jederzeit zu einer bestimmten Stelle im Text springen können, die Sie einmal markiert haben. Zum Markieren dieser Stelle dient der Befehl *Set-Mark* im *Edit*-Menü, der im übernächsten Abschnitt erläutert wird. *Swap-dot&mark* tauscht die Schreibmarke und diese Markierung aus. Das heißt, nach dem Befehl ist die Schreibmarke dort, wo die Markierung war und als markiert gilt der Buchstabe, auf dem zuvor die Schreibmarke stand.

Die restlichen Befehle im *Move*-Menü sind dann wieder leichter verständlich. *Next-page* und *Prev-page* bringen die nächste beziehungsweise vorangehende Seite des Textes auf den Bildschirm. Als »Seite« gilt dabei die Anzahl der Zeilen im oberen Bereich des Bildschirms oder Fensters. Mit den beiden Befehlen *Next-Word* und *Previous-Word* kann man sich innerhalb einer Zeile rasch bewegen. Diese Befehle setzen die Schreibmarke auf den Buchstaben vor dem nächsten Wort beziehungsweise den ersten Buchstaben des vorangehenden Wortes (ausgehend von der augenblicklichen Position der Schreibmarke). Und die beiden Befehle *Scroll-Up* und *Scroll-Down* schließlich verschieben nur den am Bildschirm sichtbaren Ausschnitt des Textes eine Zeile nach oben beziehungsweise nach unten – wobei die Schreibmarke aber an derselben Stelle im Text stehenbleibt.

Eine Verschiebung des sichtbaren Bildausschnitts passiert aber oft auch ohne ausdrücklichen Befehl, immer dann, wenn Sie die Schreibmarke über den Rand des sichtbaren Bereichs hinausbewegen. Wenn Sie zum Beispiel in der letzten sichtbaren Textzeile auf die Taste [↓] drücken, geht die Schreibmarke eine Zeile tiefer. Gleichzeitig verschiebt sich der gesamte Text aber um eine Zeile nach oben, da ja sonst die Schreibmarke nicht mehr sichtbar wäre.

### 8.3.6 Text eingeben und bearbeiten

Wie schon erwähnt wurde, erscheint Text, den Sie (über die Tastatur) neu eingeben, wie in allen anderen Amiga-Programmen vor der Schreibmarke und wird in den Text eingefügt (überschreibt also nicht vorhandenen Text). Insofern dürfte Ihnen MEMACS schon vertraut erscheinen. Aber auch die meisten »besonderen« Tasten wirken genauso wie im *Notepad*.

[Return] beginnt zum Beispiel eine neue Zeile oder bricht eine alte Zeile in zwei Zeilen auf, wenn die Schreibmarke zuvor mitten in einer Zeile gestanden hat.

Mit [Tab] fügen Sie vor der Schreibmarke einen Tabulator in den Text ein. Dieser Tabulator ist im MEMACS ein besonderes Zeichen, das »so aussieht wie« mehrere Leerstellen, aber nur ein Zeichen ist. Zu wievielen Leerstellen MEMACS ein Tabulatorzeichen ausdehnt, können Sie mit einem speziellen Befehl einstellen – nach dem Start sind es acht Leerstellen.

Mit [Del] und [Backspace] können Sie wie üblich kleine Mengen Text löschen. [Del] löscht den Buchstaben »an« der Schreibmarke, [Backspace] den links daneben. Zum Löschen

größerer Textmengen kennt MEMACS ähnlich wie *Notepad* aber das Konzept des »aktuellen Textblocks«, mit dem solche Arbeiten wesentlich schneller erledigt werden können.

### 8.3.7 Mit Textblöcken arbeiten

In MEMACS werden Anfang und Ende des aktuellen Textblocks durch zwei Positionen bestimmt, die in den Menüs *dot* (englisch für Punkt) und *mark* (englisch für Markierung/Marke) genannt werden. Der *dot* entspricht dabei der aktuellen Position der Schreibmarke, während die Marke *mark* einmal festgelegt wird und dann stehenbleibt, auch wenn sich die Schreibmarke bewegt. Um die Marke festzusetzen, müssen Sie zunächst die Schreibmarke an die gewünschte Position bringen und dann den Befehl *Set Mark* aus dem Menü *Edit* auswählen. Der in diesem Moment hervorgehobene Buchstabe ist dann die *mark*. Der Bereich zwischen *dot* und *mark* wird in den Menüs *region* und in diesem Buch »markierter Bereich« genannt.

Mit dem Befehl *Kill-region* können Sie den von der Marke und der aktuellen Position der Schreibmarke begrenzten Bereich (*region*) dann löschen. Alle Buchstaben zwischen der Marke und der Schreibmarke (dem *dot*), einschließlich der beiden Buchstaben, an denen sich Schreibmarke und Marke befinden, verschwinden dann aus dem Text. Sie landen im sogenannten *kill buffer* (Lösch-Puffer), einer Art Zwischenablage, aus der Sie den gelöschten Text jederzeit wieder herausholen können.

Wählen Sie dazu den Befehl *Yank* aus dem Menü *Edit*, und der Inhalt des Lösch-Puffers wird vor der Schreibmarke in den Text eingefügt. Wenn Sie also kurz hintereinander *Kill region* und dann *Yank* aufrufen, so wirkt das, als hätten Sie überhaupt nichts getan – die beiden Befehle heben sich auf. Mit *Kill-region* und *Yank* können Sie natürlich auch leicht große Textpassagen innerhalb eines Textes verschieben. Dazu löschen Sie die zu verschiebende Textpassage, gehen dann mit der Schreibmarke an die gewünschte neue Position und geben den Befehl *Yank* ein. *Yank* können Sie natürlich auch mehrfach hintereinander aufrufen und so einen Text vervielfältigen. Statt *Yank* sollten Sie zum Duplizieren einer Textpassage aber besser den Befehl *Copy-region* im Edit-Menü verwenden. Dieser legt eine Kopie des markierten Bereichs in den Lösch-Puffer, aber ohne die Textpassage vorher aus dem Text zu entfernen!

Die Befehle im Menü *Edit* bearbeiten immer größere Textblöcke, die von *dot* und *mark* eingerahmt werden. Wenn Sie nur eine einzelne Zeile oder ein einzelnes Wort kopieren, löschen oder duplizieren wollen, finden Sie spezielle Befehl für diesen Zweck in den Menüs *Line* und *Word*.

### 8.3.8 Texte suchen und ersetzen

Wie jeder gute Editor, besitzt MEMACS natürlich auch eine leistungsfähige Suche&Ersetze-Funktion. Mit *Search-forward* und *Search-backward* können Sie eine kurze Textpassage, ausgehend von der aktuellen Position der Schreibmarke vorwärts oder rückwärts suchen lassen. Wenn Sie einen dieser beiden Befehle auswählen, erscheint in der Befehlszeile die Meldung *Search* und dahinter ein Doppelpunkt. Sobald Sie dann hinter dem Doppelpunkt den Suchbegriff eingeben und mit [Return] abschließen, beginnt die Suche. Findet MEMACS den Suchbegriff, wird er am Bildschirm gezeigt und die Schreibmarke befindet sich

auf dem ersten Buchstaben hinter dem Suchbegriff. Wenn Sie *Search-forward* oder *Search-backward* später ein zweites Mal auswählen, erscheint der Suchbegriff, den Sie beim ersten Mal gesucht haben, in eckigen Klammern hinter der Meldung Search (also zum Beispiel *Search [Markus]:*, wenn Sie beim ersten Aufruf nach »Markus« gesucht haben). Dann brauchen Sie nur auf [Return] zu drücken, um noch einmal diesen Begiff zu suchen. Wenn Sie aber einen anderen Suchbegriff eingeben und dann [Return] drücken, wird der neue Suchbegriff gesucht.

Wie auch bei allen anderen Gelegenheiten in anderen Programmen reagieren die Suchfunktionen des MEMACS auf Unterschiede in der Groß-/Kleinschreibung nicht. Wenn Sie also den Text »Markus« suchen, würde auch »markus« und »MARKUS« gefunden. Auch hier gilt (wie zum Beispiel auch beim Shell-Befehl SEARCH), daß die Groß-/Kleinschreibung doch eine Rolle spielt, wenn es sich Umlaute handelt. Wenn Sie nach »Müller« suchen, würde deshalb »MÜLLER« nicht gefunden (weil »Ü« und »ü« für die Suchfunktion zwei verschiedene Buchstaben sind).

Ähnlich wie die Suchfunktion funktioniert die eigentliche Suche&Ersetze-Funktion. Wollen Sie eine kurze Textpassage, die häufiger in einem Text vorkommt, gegen eine andere ersetzen, so müssen Sie diese zunächst einmal suchen. Wählen Sie dazu den Befehl *Search-replace* aus dem Menü *Search*. Daraufhin erscheint wieder die Aufforderung, einen Suchbegriff einzugeben (*Search:*) in der Befehlszeile. Geben Sie dann das Wort oder die Textpassage ein, die Sie ersetzen wollen und drücken Sie [Return]. Sobald MEMACS den Suchbegriff das erste Mal gefunden hat, erscheint die Frage *Replace:* in der Befehlszeile. Tragen Sie dann hinter dem Doppelpunkt den Text ein, der den Suchbegriff ersetzen soll und drücken Sie wieder [Return]. MEMACS ersetzt dann alle Vorkommnisse des Suchbegriffs hinter der Position der Schreibmarke gegen den hinter *Replace:* eingetragenen Text. Wenn dieser Vorgang abgeschlossen ist (er kann bei großen Textdateien eine Minute oder länger dauern), meldet MEMACS, wieviele Vorkommnisse des Suchbegriffs ersetzt wurden: Replaced <nn> occurrences, wobei statt <nn> natürlich die Anzahl der durchgeführten Ersetzungen auftaucht.

Wenn Sie nicht alle, sondern nur einige Vorkommnisse eines Suchbegriffs durch einen anderen Text ersetzen wollen, können Sie den Befehl *Query-s&r* im Search-Menü verwenden. Er verhält sich exakt so wie der *Search-replace*-Befehl – nur wird vor jedem Ersetzen des Suchbegriffs die Fundstelle in der oberen Bildschirmhälfte gezeigt und dann gefragt *Change string [y/n/c/^G]?*. Auf diese Frage müssen Sie mit einem Tastendruck antworten ([Return] ist nicht nötig!). Antworten Sie mit [Y], wird der Suchbegriff ersetzt und die Suche fortgesetzt. Antworten Sie mit [N], wird der oben angezeigte Suchbegriff nicht ersetzt, die Suche aber fortgesetzt. Und wenn Sie mit [C] antworten, werden alle restlichen Vorkommnisse des Suchbegriffs bis zum Ende der Datei ohne weitere Rückfragen ersetzt. Wenn Sie statt dessen aber die Suche abbrechen wollen, können Sie das mit der Tastenkombination [Ctrl]+[G].

Der letzte Befehl im Search-Menü, *Fence-match*, wird weiter unten noch erläutert.

### 8.3.9 Mit mehreren Puffern und Fenstern arbeiten

Für den intensiven Anwender – speziell den Programmierer – bietet MEMACS noch einen besonderen Leckerbissen. Dieser verbirgt sich hinter dem häufigen Auftreten des Wortes

*buffer* oder *buffers* in den Menüs und Meldungen des Programms. Dieses Wort bezieht sich ja auf den Puffer im RAM-Speicher, in dem jeder Text zunächst bearbeitet wird, bevor er eventuell (!) auf Diskette gespeichert wird. Von diesem Puffer kann es nicht nur ein Exemplar, sondern gleich mehrere geben. Sie können also auch zwei oder drei Texte (mehr oder weniger) gleichzeitig bearbeiten.

Hierzu müssen Sie den zweiten (und alle folgenden) Text(e) nur mit dem Befehl *Visit-file* im Menü *Project* und nicht mit dem Befehl *Read-file* einlesen. Mit *Visit-file* wird der alte Pufferinhalt (also die zuerst eingelesene Datei) nicht zerstört und durch die neue Datei ersetzt, sondern statt dessen die neue Datei in einen weiteren, neuen Puffer eingelesen. Nach *Visit-file* ist dann zunächst die neue Datei am Bildschirm zu sehen und die alte verschwindet. Sie können aber jederzeit zu einem anderen Puffer wechseln. Rufen Sie dazu den Befehl *Select-buffer* im Menü *Edit* auf und geben Sie dann den Namen der Datei ein, den Sie sehen wollen. Eine Liste mit den Namen und Größen aller schon geladenen Dateien können Sie sich mit *List-buffers* im Menü *Edit* anzeigen lassen. In dieser Liste wird auch (durch einen Stern in der ersten Spalte) markiert, welche Puffer modifiziert (und noch nicht abgespeichert) wurden und welche nicht.

Auf diese Art Zugriff auf mehrere Texte in verschiedenen Dateien zu haben, ist besonders dann praktisch, wenn man einzelne Textpassagen aus einer in eine andere Datei kopieren möchte. Legen Sie die zu kopierende Textpassage einfach mit *Kill-region* oder *Copy-region* in den Lösch-Puffer. Eine Kopie des zuvor gelöschten Textes liegt dann im Lösch-Puffer. Nun wählen Sie mit *Select-buffer* die zweite Datei (in die Sie den Text einfügen wollen), setzen die Schreibmarke an die gewünschte Stelle und können nun mit dem *Yank-Befehl* eine Kopie der Textpassage aus der anderen Datei einfügen.

Die bislang beschriebene Handhabung mehrerer Puffer ist aber noch recht umständlich. Bei jedem Wechsel, von einem Puffer zum anderen, muß man zum Beispiel den Namen der jeweiligen Datei angeben. Praktisch wäre es auch, wenn man mehrere Puffer auf einmal sehen könnte. Genau das ist auch möglich. Mit dem Befehl *Split-window* im Menü *Window* können Sie den oberen Teil des Bildschirms beziehungsweise Fensters, der den bearbeiteten Text zeigt, in zwei Hälften teilen. Die Dokumentation von MEMACS nennt diese Hälften *Windows*; sie haben aber recht wenig mit den komfortablen Fenstern von Intuition zu tun, die Sie von der Workbench und aus anderen Programmen her kennen. Die *Windows* von MEMACS können zum Beispiel nicht verschoben werden, haben eine feste Breite und können sich nicht überlappen. Trotzdem bieten auch diese simplen Fenster aber die wertvolle Möglichkeit, zwei oder mehr Texte gleichzeitig am Bildschirm betrachten zu können.

Nach dem Teilen des MEMACS-Fensters mit dem Befehl *Split Window* enthalten beide Fenster zunächst einmal denselben Text. Sie können mit den Befehlen *Next-Window* und *Prev-Window* zwischen diesen beiden Fenstern hin- und herspringen und – indem Sie zum Beispiel die entsprechenden Befehle im *Move*-Menü benutzen – sich gleichzeitig Anfang und Ende dieses einen Textes zeigen lassen. Das kann manchmal ganz praktisch sein.

Immer eines der Fenster (es können auch mehr als zwei sein, wenn Sie wiederholt *Split-window* aufrufen) ist das aktuelle. Sie erkennen es daran, daß sich in ihm die Schreibmarke befindet. Wenn Sie nun die Befehle *Read-file*, *Visit-file* oder *Select-buffer* aufrufen, so taucht die dadurch auf den Bildschirm kommende Datei im aktuellen Fenster auf. Wenn Sie also zwei verschiedene Texte gleichzeitig sehen wollen, müssen Sie zunächst das eine

Fenster teilen (mit *Split-window*) und dann mit *Open* den zweiten Text einlesen oder ihn mit *Select-buffer* auswählen, wenn Sie ihn schon zuvor eingelesen hatten. Dann werden zwei verschiedene Texte am Bildschirm gezeigt.

Mit drei anderen Befehlen im Window-Menü können Sie die Fensteraufteilung des Bildschirms weiter beeinflussen. *Shrink window* und *Expand window* machen zum Beispiel das aktuelle Fenster um eine Zeile kleiner oder größer. Und mit dem Befehl *One window* können Sie erreichen, daß das aktuelle Fenster wieder den ganzen Bildschirm füllt und die anderen Fenster verschwinden (nicht jedoch der Inhalt der Puffer, der in ihnen angezeigt wurde).

### 8.3.10 Befehlsüberblick

Nach diesem »Mini-Schnell-Kurs« zu MEMACS und einigen eigenen Experimenten sollten Sie diesen hervorragenden Editor zumindest so gut kennen, um damit eigene Texte erstellen und ändern zu können. Wahrscheinlich sind aber jetzt noch mehr Fragen offen als geklärt wurden. Auf MEMACS noch intensiver einzugehen, wäre aber unfair gegenüber den anderen Themen, die in diesem Buch auch noch behandelt werden müssen. Nicht jeder Shell-Benutzer will schließlich mit MEMACS arbeiten. Vielleicht kann Ihnen aber die folgende Befehlsübersicht helfen, zumindest einige der verbleibenden Fragen zu klären. Zu jedem Menübefehl von MEMACS finden Sie darin eine knappe Erläuterung, die manchmal auch auf Aspekte des Programms eingeht, die bislang nicht besprochen wurden.

#### Das Project-Menü

Im *Project*-Menü finden Sie – wie auf dem Amiga allgemein üblich – Befehle, die ganze Dokumente oder das Programm selbst betreffen. Zum Beispiel die Befehle zum Öffnen neuer Textdateien oder zum Verlassen des Programms sind hier zu finden. Einen Befehl zum Drucken eines Textes werden Sie vergeblich suchen. Dazu müssen Sie die Shell-Befehle TYPE oder COPY (mit Ausgabeumlenkung auf >PRT:) oder PRINTFILES verwenden.

*Rename* dient zum Umbenennen eines Puffers. Dies ist zunächst eine interne Aktion, die nichts weiter bewirkt, als daß dieser Puffer unter einem anderen Namen aufgelistet wird und zum Beispiel beim *Select-buffer* unter dem neuen Namen angesprochen werden muß.

*Read-File* ersetzt den Inhalt des Puffers, der im aktuellen Fenster angezeigt wird, durch den Inhalt einer anderen Textdatei. Dazu müssen Sie den Namen dieser Datei (eventuell den vollen Pfadnamen) nach dem Aufruf des Befehls in der Befehlszeile eingeben und mit [Return] abschließen. War der alte Pufferinhalt modifiziert und noch nicht gesichert worden, erhalten Sie eine Gelegenheit, den Befehl abzubrechen. (Antworten Sie dazu auf die Frage *Discard changes* mit [N] und [Return].)

*Visit-file* arbeitet im wesentlichen exakt so wie *Read-file*. Die neu eingelesene Datei ersetzt allerdings nicht den Inhalt eines alten Puffers, sondern wird in einen neuen Puffer eingelesen.

*Insert-file* fügt in den Inhalt einer anderen Textdatei eine Zeile vor der Zeile, in der sich die Schreibmarke befindet in den Puffer ein, der im aktuellen Fenster gezeigt wird. Dazu müssen Sie den Namen dieser Datei (eventuell den vollen Pfadnamen) nach dem Aufruf des Befehls in der Befehlszeile in der Befehlszeile eingeben und mit [Return] abschließen.

*Save-file* sichert (schreibt) den Inhalt des Puffers, der im aktuellen Fenster gezeigt wird, auf Diskette beziehungsweise Festplatte.

*Save-as-file* sichert wie *Save-file* den Inhalt des Puffers, der im aktuellen Fenster gezeigt wird, auf Diskette beziehungsweise Festplatte. Sie können mit *Save-as-file* einen Text jedoch unter einem anderen Namen sichern als dem, unter dem er eingelesen wurde. Dazu müssen Sie den neuen Namen (eventuell den vollen Pfadnamen) nach dem Aufruf des Befehls in der Befehlszeile und mit [Return] abschließen.

*Save-mod* arbeitet ebenfalls ähnlich wie *Save-file*, sichert jedoch alle Puffer, die in irgendeiner Form modifiziert wurden (nicht nur den aktuellen), nacheinander auf der Diskette beziehungsweise Festplatte.

*Save-exit* ist eine Kombination von *Save-mod* und *Quit* (siehe unten).

*New Cli* erzeugt – ähnlich wie der Shell-Befehl NEWCLI – ein neues CLI-Fenster. Es erscheint auf dem Workbench-Screen und Sie können in diesem Fenster ganz normal CLI-(Shell-)Befehle verwenden, wie in jedem anderen CLI-Fenster auch. Wenn Sie dieses Fenster mit dem Kommando ENDCLI schließen, kehren Sie zu MEMACS zurück. Das von MEMACS geöffnete Fenster ist leider ein Fenster, daß von dem Gerät CON: kontrolliert wird und nicht von NEWCON: wie die von NEWSHELL geöffneten Fenster. In diesem Fenster können Sie deshalb nicht die Schreibmarke mit den Cursor-Tasten verschieben oder vergangene Befehle wieder »zurückholen«, wie es in Abschnitt 8.1 beschrieben wurde.

*CLI-Command* dient dazu, einen einzelnen Shell- beziehungsweise CLI-Befehl aufzurufen. Wenn Sie *CLI-Command* aufrufen, können Sie in der Befehlszeile am unteren Bildschirmrand einen Befehl erteilen (zum Beispiel DIR DF0:). Die Ausgabe dieses Befehls wird dann auf dem Bildschirm aufgelistet und dahinter erscheint die Meldung »[End]«. Sie können sich nun diese Ausgabe in Ruhe betrachten. Wenn Sie mit dem MEMACS weiterarbeiten wollen, brauchen Sie nur die [Return]-Taste betätigen und der Screen vom MEMACS mit dem bearbeiteten Text darin erscheint unverändert wieder am Bildschirm.

*Quit* beendet MEMACS. Falls modifizierte Puffer existieren, deren Inhalt noch nicht gesichert wurde, erhalten Sie aber die Gelegenheit, diesen Befehl abzubrechen.

*About ...* zeigt einen kleinen Requester mit Informationen über den Ursprung des Programms (unter anderem erfahren Sie den Namens des Autors) und den augenblicklichen Stand dieser Version von MEMACS. Hier wird auch gemeldet, daß Commodore Copyright auf diese Version beansprucht. Sie können den Requester mit einem Klick auf eine der beiden OK-Tasten wieder schließen.

### Das Edit-Menü

Das *Edit*-Menü dient – wie der Name schon sagt – zum Editieren, also Bearbeiten des Textes in einem Puffer. Sie finden in diesem Menü unter anderem eine Reihe von Befehlen zum Löschen, Kopieren und Duplizieren großer Textblöcke.

*Kill-region* entspricht dem Befehl *Cut* des Werkzeugs *Notepad*. Dieser Befehl löscht den Bereich zwischen *dot* (Schreibmarke) und *mark* (einer Position, die Sie mit dem Befehl *Set Mark* festlegen) aus dem bearbeiteten Text (Puffer) und legt ihn in den Lösch-Puffer. Sie können den so gelöschten Bereich wieder in den Text einfügen, eventuell auch an anderer Stelle, indem Sie *Yank* (siehe unten) aufrufen.

*Yank* entspricht dem Befehl *Paste* des Werkzeugs *Notepad*. Er setzt den Inhalt des Lösch-Puffers vor der Schreibmarke in den gerade bearbeiteten Text ein.

*Set-mark* markiert das eine Ende des Bereichs, der später zum Beispiel mit *Kill-region* aus dem Text gelöscht werden kann.

*Copy-region* entspricht dem Befehl *Copy* des Werkzeugs *Notepad*. Er legt den Bereich zwischen *dot* (Schreibmarke) und *mark* (einer Position, die Sie mit dem Befehl *Set-Mark* festlegen) in den Lösch-Puffer. Der alte Inhalt des Lösch-Puffers wird dadurch ersetzt.

*Upper-region* ändert die Groß-/Kleinschreibung des Bereichs zwischen Schreibmarke (*dot*) und Markierung (*mark*). Alle Buchstaben darin sind nach Aufruf des Befehls groß geschrieben. (Das funktioniert allerdings nicht mit deutschen Umlauten!)

*Lower-region* arbeitet wie Upper-region; nur daß nach Aufruf alle Buchstaben des markierten Bereichs klein geschrieben sind.

*List-buffers* öffnet ein neues Fenster, in dem alle schon geladenen Dateien (beziehungsweise ihre Puffer) aufgelistet werden.

*Select-buffer* erlaubt die Auswahl eines anderen Puffers, der im aktuellen Fenster gezeigt werden soll. Sie müssen den Namen dieses anderen Puffers dazu in der Befehlszeile eingeben und mit [Return] abschließen.

*Insert-buffer* fügt – ähnlich *Insert-file* – den Inhalt eines anderen Puffers in den Puffer ein, der im aktuellen Fenster gezeigt wird. Sie müssen den Namen dieses anderen Puffers dazu in der Befehlszeile eingeben und mit [Return] abschließen.

*Kill-buffer* entfernt einen Puffer aus dem Speicher. Sie müssen den Namen dieses Puffers dazu in der Befehlszeile eingeben und mit [Return] abschließen. Dieser Puffer darf nicht (in einem oder mehreren Fenstern) sichtbar sein. Wurde er modifiziert und sind die Änderungen noch nicht gesichert, erhalten Sie die Gelegenheit, den Befehl abzubrechen. (Antworten Sie dazu auf die Frage *Discard changes* mit [N] und [Return].)

*Justify-buffer* löscht in allen Zeilen des Puffers, der im aktuellen Fenster gezeigt wird, sämtliche Leerstellen und Tabulatoren, die sich am linken Rand befinden. Danach ist der Text in diesem Puffer also »linksbündig«.

*Redisplay* ist ein Befehl, der nur nach einem Programmfehler nötig sein sollte. Er zeichnet den gesamten Bildschirminhalt einmal neu (frischt ihn auf).

*Quote-Char* unterdrückt die »spezielle Bedeutung« des unmittelbar darauffolgenden Tastendrucks. Sie können damit Tasten und Tastenkombinationen in den Text einfügen, die normalerweise einen Befehl aufrufen würden (zum Beispiel ein [Ctrl]+[C]). Rufen Sie dazu erst *Quote-Char* auf und drücken Sie dann die gewünschte Taste oder Tastenkombination. Solche »Control-Zeichen« oder »Escape-Sequenzen« innerhalb eines Textes werden manchmal zum Beispiel zur Druckersteuerung benötigt. Sie sind auch für die Eingabe von Makros wichtig (siehe unten).

*Indent* erzeugt eine neue Zeile hinter der aktuellen Zeile (in der sich die Schreibmarke befindet) und setzt die Schreibmarke in diese Zeile – allerdings nicht an den Anfang der Zeile sondern genauso weit nach rechts eingerückt, wie auch die vorangehende Zeile. Wenn Sie also am Anfang einer Zeile erst fünf Leerstellen eingeben und dann den Text beginnen, können Sie mit Indent eine weitere Zeile erzeugen, die automatisch auch um fünf Leerstellen

nach rechts eingerückt ist. Das ist besonders beim Schreiben von Programmen und Kommandofolgen sehr praktisch, weil in solchen Texten oft die Programmstruktur durch Einrückung verdeutlicht wird.

*Transpose* vertauscht die beiden Buchstaben an der Position der Schreibmarke und links daneben. Das ist eine häufig benötigte Korrekturmöglichkeit. Besonders beim schnellen Schreiben vertippt man sich oft so, daß zwei aufeinanderfolgende Buchstaben in der falschen Reihenfolge eingegeben werden. Mit *Transpose* können Sie diesen Fehler sehr leicht rückgängig machen.

*Cancel* bricht die Eingabe in der Befehlszeile und den damit verbundenen Befehl ab. Wenn Sie beispielsweise versehentlich den Befehl *Save-file-as* aufrufen und dann nach einem Dateinamen gefragt werden, können Sie den Befehl mit *Cancel* (oder seinem Tastaturäquivalent [Ctrl]+[G] abbrechen).

### Das Window-Menü

Das *Window*-Menü dient dazu, den Bildschirm oder das Fenster, in dem MEMACS arbeitet, in eine Reihe verschiedener Bereiche (Fenster) zu unterteilen und diese Unterteilung zu ändern. In jedem dieser Fenster, die wenig mit den Intuition-Fenstern zu tun haben, wird der Inhalt eines Puffers angezeigt.

*One-window* dehnt das aktuelle Fenster (in dem sich die Schreibmarke befindet) aus, bis es (fast) den ganzen Bildschirm füllt. Alle anderen Fenster verschwinden. Die Puffer, die in den anderen Fenstern angezeigt wurden, bleiben aber erhalten.

*Split-window* teilt das aktuelle Fenster in zwei Hälften, in denen zunächst derselbe Puffer gezeigt wird.

*Next-window* setzt die Schreibmarke in das nächste Fenster unterhalb des aktuellen Fensters. Das können Sie aber auch mit einem Mausklick machen.

*Prev-window* setzt die Schreibmarke in das vorangehende Fenster oberhalb des aktuellen Fensters.

*Expand-window* vergrößert das aktuelle Fenster auf Kosten eines benachbarten Fensters um eine Zeile.

*Shrink-window* verkleinert das aktuelle Fenster um eine Zeile.

*Next-w-page* blättert im Fenster unterhalb des aktuellen Fensters eine Seite weiter.

*Prev-w-page* blättert im Fenster oberhalb des aktuellen Fensters eine Seite weiter.

Gibt es nur ein MEMACS-Fenster, so blättern *Next-w-page* und *Next-w-page* in diesem.

### Das Move-Menü

Das *Move*-Menü dient zur schnellen Verschiebung der Schreibmarke (schneller als mit den Cursortasten) zu gewissen »markanten Punkten« innerhalb eines Textes.

*Top-of-buffer* setzt die Schreibmarke in die erste Spalte der ersten Zeile des Puffers im aktuellen Fenster.

*End-of-buffer* setzt die Schreibmarke hinter den letzten Buchstaben der letzten Zeile des Puffers im aktuellen Fenster.

*Top-of-window* setzt die Schreibmarke in die erste Spalte der ersten Zeile des im aktuellen Fenster gezeigten Textes.

*End-of-window* setzt die Schreibmarke in die letzte Zeile des im aktuellen Fenster gezeigten Textes.

*Go-to-line* setzt die Schreibmarke in eine bestimmte Zeile. Dazu müssen Sie die Nummer dieser Zeile (wobei die erste Zeile im Puffer mit 1 numeriert wird) in der Befehlszeile eingeben und mit [Return] abschließen.

*Swap-dot&mark* setzt die Schreibmarke an die Stelle der Marke, die zuvor mit Set Mark oder einem Doppelklick markiert wurde. Die Marke steht danach dort, wo zuvor die Schreibmarke gestanden hat.

*Next-page* blättert das aktuelle Fenster eine Seite (Fensterhöhe) weiter.

*Prev-page* blättert das aktuelle Fenster eine Seite (Fensterhöhe) zurück.

*Next-word* setzt die Schreibmarke an den Anfang des Leerraums vor dem nächsten Wort im Text (auch wenn dieses sich erst in der folgenden Zeile befindet).

*Previous-word* setzt die Schreibmarke an den Anfang des nächsten vorhergehenden Wortes. Befindet sich die Schreibmarke vor Aufruf dieses Befehls in einem Wort, steht sie danach auf dem Anfang dieses Wortes.

*Scroll-up* verschiebt den im aktuellen Fenster angezeigten Text um eine Zeile nach oben. Die oberste Zeile verschwindet und am unteren Fensterrand wird Platz frei für eine der dort folgenden Zeilen.

*Scroll-down* verschiebt den im aktuellen Fenster angezeigten Text um eine Zeile nach unten. Die unterste Zeile verschwindet und am oberen Fensterrand wird Platz frei für eine der vorangehenden Zeilen.

### Das Line-Menü

Die Befehle im *Line*-Menü dienen unter anderem dazu, die Schreibmarke innerhalb einer Zeile zu bewegen, zeilenweise zu löschen, Zeilen aufzubrechen und die Schreibmarke auf und ab zu bewegen.

*Open-line* spaltet die Zeile in der sich die Schreibmarke befindet, in zwei Zeilen auf. Der Teil ab der Position der Schreibmarke kommt in eine neue Zeile. Insoweit kommt *Open-line* einer Betätigung der [Return]-Taste gleich. Nach dem Aufteilen mit *Open-line* steht die Schreibmarke aber am Ende der ersten Zeile, während sie nach [Return] am Anfang der neuen Zeile stehen würde. Das kann in Makros wichtig sein.

*Kill-line* löscht die komplette Zeile, in der sich die Schreibmarke gerade befindet, aus dem entsprechenden Puffer und legt sie in den Lösch-Puffer. (Erfolgen mehrere Löschoperationen hintereinander, ohne ein Yank oder Copy region dazwischen, wird der gelöschte Text hinter den alten Inhalt des Lösch-Puffers gehängt.)

*Kill-to-eol* löscht den Rest der Zeile rechts von der Schreibmarkenposition (einschließlich des Buchstabens an der Schreibmarkenposition), und legt ihn in den Lösch-Puffer.

*Start-of-line* setzt die Schreibmarke an den Anfang der aktuellen Zeile.

*End-of-line* setzt die Schreibmarke an das Ende der aktuellen Zeile.

*Next-line* setzt die Schreibmarke eine Zeile tiefer.

*Previous-line* setzt die Schreibmarke eine Zeile höher.

*Line-to-top* verschiebt den im Fenster sichtbaren Bereich des Textes so, daß die aktuelle Zeile (mit der Schreibmarke) am oberen Fensterrand liegt.

*Delete-blanks* löscht hinter der aktuellen Zeile alle Leerzeilen, so lange bis wieder eine Zeile auftaucht, die Buchstaben enthält. Beachten Sie, daß eine scheinbar leere Zeile nicht wirklich leer sein muß. Wenn Sie am Anfang einer Zeile ein paar Lerstellen eingeben oder auf die [Tab]-Taste drücken, sieht die Zeile zwar leer aus, ist es aber nicht (zumindest nicht für den Befehl *Delete-blanks*).

### Das Word-Menü

Das *Word*-Menü enthält Befehle, die Operationen an Worten (durch Leerstellen getrennte Buchstabenfolgen) entsprechen. Sie ähneln den entsprechenden Befehlen im *Line*-Menü. Beachten Sie, daß sich auch im *Move*-Menü einige Befehle finden, die wortweise arbeiten.

*Delete-forw* löscht den »Rest« (rechts von der Schreibmarke) des Wortes, in dem sich die Schreibmarke befindet. Als Ende eines Wortes gelten dabei Leerstellen, Tabulatoren, Satzzeichen wie Punkt und Komma, Control-Zeichen und das Zeilenende. Wenn sich die Schreibmarke in dem Leerraum vor einem Wort befindet, wird dieser Leerraum und das ganze folgende Wort gelöscht.

*Delete-back* arbeitet wie *Delete-forw*. Nur wird nicht nach rechts zum Wortende, sondern in umgekehrter Richtung, zum Wortanfang hin, gelöscht. Steht die Schreibmarke vor Aufruf des Befehls auf der Leerstelle hinter einem Wort, so wird das ganze vorangehende Wort gelöscht – nicht aber die Leerstelle dahinter.

*Upper-word* wandelt den Rest des Wortes (rechts von der Schreibmarke), in dem sich die Schreibmarke befindet, in Großbuchstaben um. (Diese Operation wirkt nicht auf deutsche Umlaute und internationale Sonderzeichen.)

*Lower-word* arbeitet wie *Upper word*. Nur werden die restlichen Buchstaben des Wortes in Kleinbuchstaben umgewandelt.

*Cap-word* wandelt den Buchstaben, auf dem sich die Schreibmarke befindet, in einen Großbuchstaben um und den Rest des Wortes (rechts von der Schreibmarke) in Kleinbuchstaben. Steht die Schreibmarke am Anfang eines Wortes, kann man so dafür sorgen, daß das Wort mit einem Großbuchstaben beginnt. (Diese Operation wirkt nicht auf deutsche Umlaute und internationale Sonderzeichen.)

*Switch-case* wechselt die Groß-/Kleinschreibung aller Buchstaben des aktuellen Wortes, beginnend bei der Schreibmarke bis zum nächsten Wortende. Aus allen Kleinbuchstaben werden Großbuchstaben und umgekehrt. (Diese Operation wirkt nicht auf deutsche Umlaute und internationale Sonderzeichen.)

### Das Search-Menü

Das *Search*-Menü enthält Operationen zum schnellen Aufsuchen einer Textpassage in einem längeren Text und zum Austauschen einer häufig vorkommenden Textpassage gegen eine andere. Dieser Austausch kann entweder vollautomatisch oder erst nach vorheriger Bestätigung erfolgen.

*Search-forward* sucht eine Textpassage, ausgehend von der aktuellen Position der Schreibmarke, nach vorne, also zum Ende des Textes hin. Dazu muß die gesuchte Textpassage nach Aufruf des Befehls in der Befehlszeile eingegeben und mit [Return] beendet werden. Ist die Suche erfolgreich, steht danach die Schreibmarke auf dem ersten Buchstaben hinter dem gefundenen Suchbegriff. Ist die Suche erfolglos, erscheint in der Befehlszeile die Meldung *Not found* (nicht gefunden).

*Search-backward* arbeitet genau wie *Search-forward*; nur in umgekehrter Richtung, zum Anfang des Textes hin.

*Search-replace* dient zum Austauschen einer Textpassage gegen eine andere im gesamten Text. Der Befehl verhält sich zunächst genau wie *Search forward.* Sobald der Suchbegriff das erste Mal gefunden worden ist, wird in der Meldungszeile (hinter dem Prompt *Replace:*) nach der Textpassage gefragt, die den Suchbegriff ersetzen soll. Geben Sie ihn ein und drücken Sie auf [Return]. Danach werden alle Vorkommnisse des Suchbegriffs, die hinter der aktuellen Position der Schreibmarke liegen, gegen den neu eingegebenen Text ersetzt.

*Query-s-r* verhält sich im Prinzip wie *Search-replace,* ersetzt also einen wiederholt vorkommenden Suchbegriff gegen einen anderen Text. Sie erhalten aber die Gelegenheit, sich bei jeder Fundstelle einzeln zu entscheiden, ob die Ersetzung durchgeführt werden soll oder nicht. Jedesmal, bevor der Suchbegriff gefunden wurde, erscheint dazu in der Befehlszeile jedoch die Frage *Change string [y/n/c/^G]?*. Antworten Sie auf diese Frage mit der Taste [Y], wird der Suchbegriff ersetzt. Antworten Sie hingegen mit [N], wird der Suchbegriff nicht ersetzt, die Suche aber fortgesetzt. Antworten Sie mit [C], werden alle folgenden Vorkommnisse des Suchbegriffs ohne weitere Rückfragen ersetzt. (*Query-s-r* verhält sich dann wie *Search-replace.*) Wenn Sie die Suche abbrechen wollen, können Sie das jederzeit tun, indem Sie auf die Frage hin die Tastenkombination [Ctrl]+[G] drücken oder den Befehl *Cancel* aus dem *Edit*-Menü auswählen.

*Fence-match* setzt die Schreibmarke auf den nächsten Buchstaben, der zu dem Buchstaben »paßt«, der sich bei der Schreibmarke befindet. Markiert die Schreibmarke gerade eine Klammer (rund, eckig oder geschweift), so springt die Schreibmarke auf diesen Befehl hin zur dazugehörigen anderen Klammer. Das funktioniert vorwärts und rückwärts. *Fence-match* springt von »(« zu »)« nach rechts aber von »)« zu »(« nach links. Dasselbe gilt entsprechend für eckige und geschweifte Klammern. *Fence-match* ist bei der Suche nach der passenden Klammer ausgesprochen clever und kann sogar geschachtelte Klammern korrekt bearbeiten. Steht die Schreibmarke zum Beispiel auf der letzten Klammer von »(((2+3)+4)*(1+5))«, steht sie nach dem Aufruf von Fence-match auf der ersten Klammer.

Steht die Schreibmarke nicht auf einer Klammer, wenn *Fence-match* aufgerufen wird, setzt der Befehl die Schreibmarke auf das nächste Vorkommnis desselben Buchstabens. Sie können so zum Beispiel von ».« zu ».« oder von »;«« zu »;« springen. Dieser Befehl dürfte besonders für Programmierer interessant sein.

### Das Extras-Menü

Das *Extras*-Menü enthält einige Befehle, die die Programmentwickler scheinbar in keinem anderen Menü sinnvoll unterbringen konnten. Es ist deshalb eine recht bunte Mischung von Befehlen.

*Set-Arg* erlaubt es, eine Zahl einzugeben, die von einem unmittelbar darauffolgenden Befehl als »Argument« verwendet werden kann. Meist wird dieses Argument zur Multiplikation

des folgenden Befehls also zu seiner mehrfachen Ausführung verwendet. Wenn Sie zum Beispiel *Set Arg* aufrufen, (hinter dem Prompt *Arg:)* die Zahl 12 eingeben und dann die Taste [↑] drücken, setzt das die Schreibmarke 12 Zeilen höher (falls möglich). Der Befehl wirkt also genauso, als hätten Sie zwölfmal auf [↑] gedrückt. Damit alle Befehle auf diese Art und Weise multipliziert werden können, dürfen Sie nicht auf [Return] drücken, wenn Sie die Zahl eingegeben haben (auf diese Art und Weise würden Sie nur mehrere Leerzeilen erzeugen). Versuchen Sie auch nicht eine eventuell schon dort stehende Zahl mit [Backspace] zu löschen. Geben Sie einfach die gewünschte Zahl mit den Zifferntasten am oberen Rand der Tastatur ein, und wählen Sie dann den Befehl, dem sie als Argument dienen soll. Verwenden Sie zur Zahleingabe nicht die numerische Tastatur sondern nur die Zifferntasten im Hauptfeld!

*Set-Arg* kann auch für andere Befehle verwendet werden, die eine Zahlangabe benötigen. Diese Befehle werden gleich noch beschrieben. Wenn Sie einen solchen Befehl verwenden wollen, müssen Sie erst *Set-Arg* aufrufen, die gewünschte Zahl eintippen und dann sofort den anderen Befehl (zum Beispiel *Set*) aus dem Menü auswählen.

*Set* dient dazu, einige wichtige Parameter von MEMACS zu setzen, die das Verhalten des Programms entscheidend beeinflussen. Nach dem Aufruf von *Set* werden Sie zunächst in der Befehlszeile *Set what:* gefragt. Daraufhin müssen Sie den Namen eines der MEMACS-Parameter eingeben und die [Return]-Taste drücken. Bei manchen Parametern genügt dieser Namen – es handelt sich dann um eine Art Schalter, der durch Nennung des Namens quasi umgelegt wird. Bei anderen Parametern erscheint nach der Nennung des Namens eine weitere Frage, die Sie erst beantworten müssen, bevor der *Set*-Befehl beendet ist.

Es folgt nun eine Liste der vom Set-Befehl erkannten Parameternamen:

*Screen* ist ein Schalter, der festlegt, ob MEMACS einen eigenen Screen oder ein Fenster auf dem Workbench-Screen zur Darstellung und Bearbeitung des Textes verwendet. Wenn Sie auf die Frage *Set what: Screen* antworten, wird zwischen diesen beiden Betriebsarten hin- und hergewechselt. Ob MEMACS nach dem Start einen eigenen Screen oder nur ein Fenster öffnet, hängt davon ab, ob Sie beim Aufruf vom Shell-Fenster aus das Schlüsselwort OPT W verwendet haben oder nicht.

*Interlace* ist ein Schalter, der den Interlace-Grafikmodus des Amiga an- und ausschaltet. Wenn Sie auf die Frage *Set what:* das erste Mal *Interlace* antworten, wird der Interlace-Modus angeschaltet. MEMACS kann dann statt 25 bis zu 50 Textzeilen am Bildschirm darstellen, wobei das Bild aber relativ stark flimmert. Durch erneuten Aufruf von *Set* und Eingabe von *Interlace* können Sie diesen Modus dann wieder abstellen. Der Interlace-Modus kann nur dann aktiviert werden, wenn MEMACS einen eigenen Screen und nicht ein Fenster auf dem Workbench-Screen verwendet. Welche der beiden Möglichkeiten das Programm nutzt, können Sie beim Start oder mit dem *Set*-Parameter *Screen* bestimmen.

*Mode* ist ein Parameter, der eine weitere Eingabe fordert. Sie müssen daraufhin eines der Schlüsselworte *cmode* oder *wrap* angeben. Wenn Sie *cmode* angeben und auf die [Return]-Taste drücken, geht MEMACS in einen Modus, der die Bearbeitung von Programmen in der C-Sprache erleichtert. Die Eingabe von *wrap* aktiviert den Word-wrap-Modus, in dem MEMACS automatisch eine neue Zeile beginnt, wenn eine bestimmte vorgegebene Zeilenbreite erreicht ist. MEMACS bricht dabei die alte Zeile nicht mitten im Wort auf, sondern legt das letzte Wort der Zeile, das nicht mehr in die

Breite paßt, in die neue Zeile. Die Modi *cmode* und *wrap* können wieder abgeschaltet werden, indem man erneut *Set Mode* aufruft und dann *–cmode* oder *–wrap* eingibt.

*Wrap* ist ein Parameter, mit dem Sie die Zeilenbreite, bei dem der automatische Zeilenumbruch erfolgen soll (siehe Mode), einstellen können. Geben Sie dazu nach dem Aufruf von Set den Parameternamen *Wrap* an. Sie werden dann hinter dem Prompt *Column:* nach einer Zahl gefragt. Geben Sie jetzt die gewünschte maximale Breite der Zeile ein und drücken Sie die [Return]-Taste.

*Left* dient dazu, einen linken Rand für einen Text festzulegen. Wenn Sie die Breite dieses Randes nicht zuvor mit *Set Arg* angegeben haben, werden Sie nach der Eingabe *Left* danach gefragt. Beim Eingeben neuer Zeilen wird dieser Rand automatisch eingehalten. Das heißt, am Anfang jeder neuen Zeile werden die entsprechenden Leerstellen und Tabulatoren eingefügt. Die Breite des linken Randes darf zwischen 0 und 80 Leerstellen liegen (60, wenn Sie die Workbench im 60-Zeichen-Modus betreiben).

*Right* dient dazu, den rechten Rand und damit die maximale Breite (in Buchstaben) für den Text festzulegen. Dieser kann zwischen 1 und 80 liegen. Sie brauchen diese Zahl nicht vorher mit *Set Arg* anzugeben, sondern können sie auch nachträglich (nach Aufruf von *Right mg*) in der Befehlszeile eingeben. MEMACS stellt Zeichen, die rechts vom rechten Rand liegen, nicht mehr am Bildschirm dar. Ist eine Zeile länger, so wird als ihr letzter Buchstabe ein Dollarzeichen ($) gezeigt. Sie können daran sehen, daß diese Zeile länger ist als der von Ihnen bestimmte rechte Rand es erlaubt.

Beachten Sie, daß *Right* und *Wrap* sich gegenseitig beeinflussen, sobald der Wordwrap-Modus eingeschaltet ist. Es ist nicht sinnvoll, zu kleine Werte für *Wrap* zu verwenden oder Werte, die größer sind als die mit *Right* festgelegte Zeilenbreite.

*Tab* ist ein Parameter, mit dem Sie die Abstände zwischen den Tabstopp-Positionen festlegen können. Die gewünschte Zahl wird nach der Eingabe von *Tab* erfragt. Wenn Sie dann beispielsweise eine 4 eingeben, setzt MEMACS alle 4 Spalten einen Tabstopp. Eine Betätigung der [Tab]-Taste bringt sie dann jeweils zum nächsten Tabstopp, also in Spalte vier, acht, zwölf und so weiter.

*Case* ist ein Schalter (also ein Parameter, der keine weiteren Angaben erfragt). Er legt fest, ob bei Such-Operationen Groß- und Kleinschreibung eine Rolle spielt oder nicht. Dieser Schalter ist nach dem Start zunächst einmal aus (Groß- und Kleinschreibung spielt keine Rolle).

*Backup* ist ein Schalter mit drei Stellungen. Nach der Eingabe von Backup wird man gleich zu einer weiteren Eingabe aufgefordert. Drei Eingaben sind erlaubt: ON, SAFE und OFF. Wenn Sie ON angeben, legt MEMACS von jeder Datei, die Sie sichern, einen »Backup«, eine Sicherungskopie an. Dabei wird das unveränderte Original der Datei, bevor es beim Sichern überschrieben wird, im logischen Laufwerk T: gesichert. Wenn Sie die Option SAFE angeben, ist es nicht erlaubt, eine Datei abzuspeichern, die schon existiert (und dabei den alten Inhalt der Datei zu überschreiben). Sie erhalten eine Fehlermeldung, wenn Sie das versuchen. Die Option OFF schaltet alle diese Sicherungsmechanismen aus.

Der Befehl *Start-macro* im Menü *Extras* beginnt die Aufzeichnung eines sogenannten »Makros«. Jede Taste, die Sie betätigen, nachdem Sie *Start-macro* aufrufen, wird von

MEMACS aufgezeichnet, bis Sie den Befehl *Stop-macro* aufrufen (siehe unten). Diese Tastenfolge ist ein Makro und kann später mit einem einzigen Befehl wiederholt werden.

*Stop-macro* beendet die Aufzeichnung eines Makros, die mit *Start-macro* begonnen wurde (siehe oben).

*Execute-macro* »führt das zuletzt aufgezeichnete Makro aus«. Das bedeutet, daß sich MEMACS dann verhält, als würden Sie noch einmal die Tasten drücken, die Sie zwischen *Start-macro* und *Stop-macro* gedrückt haben. Das gilt natürlich auch für Tastenkombinationen, die Befehlen entsprechen!

*Set-key* dient dazu, eine beliebige Taste auf der Tastatur umzudefinieren (ähnlich, wie es das Werkzeug *SetMap* tut). Wählen Sie dazu *Set-key* aus dem Menü *Extras* und drücken Sie die Taste, deren Bedeutung Sie ändern wollen. Sie können dann in der Befehlszeile eine beliebige Folge von Tasten eingeben (auch solche Tastenkombinationen, die Befehlen entsprechen). Schließen Sie die Tastenfolge mit [Return] ab. Wenn Sie später auf diese umdefinierte Taste drücken, werden die zuvor aufgezeichneten Tastendrücke »abgespielt«, als würden Sie nacheinander auf diese Tasten drücken.

Bei der Eingabe von Befehlstastenkombinationen (die meist die [Ctrl]-Taste beinhalten) ist Vorsicht geboten. Diese dürfen nicht direkt in der Befehlszeile eingegeben werden, wenn Sie mit Set-key eine Taste umdefinieren. Statt dessen müssen Sie vor jeder solchen Tastenkombination den Befehl *Quote-Char* ([Ctrl]+[Q]) aufrufen. Um die Tastenkombination [Ctrl]+[X] einzugeben, müßten Sie also zuerst [Ctrl]+[Q] und dann [Ctrl]+[X] drücken.

Die Umdefinition »gewöhnlicher« Tasten, wie [A], [,], [#] und so weiter, die verwirrend sein könnte, ist übrigens mit *Set-key* nicht möglich. Umdefiniert werden können nur die Funktionstasten [F1] bis [F10] am oberen Rand der Tastatur und die Tasten des numerischen Blocks am rechten Rand.

| Taste | Wirkung |
|---|---|
| [F1] | Aktuelle Zeile duplizieren. |
| [F2] | Aktuelle Zeile löschen. |
| [F3] | Makro ausführen. |
| [F4] | Eine Seite nach unten blättern. |
| [F5] | Eine Seite nach oben blättern. |
| [F6] | Aktuelles Fenster teilen (Split window). |
| [F7] | Aktuelles Fenster bildschirmfüllend machen (One window). |
| [F8] | Aktuelles Fenster eine Zeile nach oben schieben. |
| [F9] | Aktuelles Fenster eine Zeile nach unten schieben. |
| [F10] | Alle modifizierten Puffer sichern und Programm beenden. |

***Tabelle 8.2:*** *Belegung der Funktionstasten im MEMACS*

Die Funktionstasten sind übrigens gleich nach dem Start schon mit solchen Tastenkombinationen belegt. Diese Belegung kann aber bei Ihnen unter Umständen von der in Tabelle 8.2 gezeigten etwas abweichen, da sich Commodore Änderungen am Programm vorbehält.

*Reset-keys* macht alle mit *Set-key* definierten Tastenbelegungen wieder rückgängig und stellt den in Tabelle 8.2 beschriebenen Urzustand wieder her.

*Execute-file* liest eine Textdatei ein, nach deren Namen Sie gefragt werden und führt diese als Folge von MEMACS-Befehlen aus. Sie müssen in dieser Datei die MEMACS-Befehle genau so schreiben, wie Sie auch in den Menüs auftauchen.

Wenn Sie eine solche Datei *emacs_pro* nennen und in das Directory *S:Emacs* legen (dieses Directory müssen Sie allerdings erst mit MAKEDIR erzeugen), wird die Befehlsdatei gleich nach dem Start von MEMACS automatisch ausgeführt. Die folgenden Befehle würden, wenn Sie in dieser Datei enthalten wären, beispielsweise bei jedem Start den Interlace-Modus aktivieren und die Datei *S:StartupII* einlesen (wobei Letzteres wenig Sinn ergibt):

```
set interlace
read-file S:StartupII
```

Um die korrekte Schreibweise der Befehle auszuprobieren, können Sie den Befehl *Execute-line* verwenden. Wenn Sie diesen Befehl aufrufen, haben Sie die Gelegenheit, einen MEMACS-Befehl (so wie er im Menü geschrieben wird) einzugeben. Wenn Sie dann die [Return]-Taste drücken, wird der Befehl ausgeführt.

## 8.4 Kommandofolgen

Im vorangehenden Abschnitt 8.3 dieses Kapitels haben Sie den Editor MEMACS kennengelernt. Eine der wichtigsten Aufgaben, für die ein Shell-Benutzer einen solchen Editor üblicherweise einsetzt, ist das Erstellen und Ändern von Kommandofolgen. Mit diesen Kommandofolgen – einer Art Programmiersprache, deren »Wortschatz« sich aus den Shell-Befehlen zusammensetzt – beschäftigt sich dieser und die folgenden Abschnitte dieses Kapitels. Für alle, die intensiver mit der Shell arbeiten wollen, ist das sicherlich einer der wichtigsten Teile dieses Buches. Kommandofolgen sind eine sehr wesentliche Komponente der Shell-Software.

### 8.4.1 Wozu Kommandofolgen?

Ein großer Vorteil, den die Shell gegenüber der Workbench bietet, ist die Möglichkeit, langwierige, häufig anfallende Aufgaben zu »automatisieren«. Das geschieht, indem eine Folge von Shell-Befehlen zu einer »Kommandofolge« zusammengefaßt wird. Eine solche Kommandofolge wird immer in einer Textdatei abgespeichert und kann danach unter Angabe des Namens dieses Datei durch den Befehl EXECUTE gestartet werden. Sie läuft dann selbständig und normalerweise ohne weiteres Zutun des Benutzers ab. Mit Hilfe des Befehls *run* kann man Kommandofolgen sogar als separaten Prozeß »im Hintergrund« ablaufen lassen und sich unterdessen um andere Dinge kümmern.

Kommandofolgen sind einfachen Programmen sehr ähnlich. Die dabei verwendete »Programmiersprache« ist simpel, kennt aber schon bedingte Anweisungen, Sprunganweisun-

gen, Schleifen, Parameter und Variablen. Und durch die Möglichkeit, in Kommandofolgen andere Kommandofolgen – und natürlich normale Kommandos – aufzurufen, können auch größere Aufgaben angegangen werden, indem man ein großes Problem in kleine Teilaufgaben zerlegt. Aber keine Angst: die Programmiersprache für Kommandofolgen ist sehr einfach und vor allem auch einfach zu erlernen! Sie brauchen kein Programmierer zu werden, um Kommandofolgen nutzen zu können.

Der Sinn von Kommandofolgen wird Ihnen als bislang vor allem mit der Workbench vertrautem Amiga-Benutzer vielleicht nicht gleich klar sein – speziell dann, wenn Sie bisher nur gelegentlich mit dem Amiga gearbeitet haben. Man benötigt Kommandofolgen üblicherweise erst dann, wenn man intensiv mit dem Computer arbeitet. Sie werden dann rasch merken, daß viele Dinge, die Sie täglich tun, sich immer wiederholen oder sich zumindest ähneln.

Nehmen wir einmal an, Sie würden häufiger eine Datei erst sortieren, dann ausdrucken und dazu immer die drei folgenden Shell-Befehle hintereinander ausführen:

```
SORT   kundenliste TO druckdatei
TYPE   druckdatei  TO PRT:
DELETE druckdatei
```

In diesem Fall ist es doch eigentlich recht lästig, daß Sie jedesmal alle drei Befehle fehlerfrei eintippen müssen, nach jedem Befehl darauf warten müssen, daß er fertig wird und alles in allem eine stupide mechanische Tätigkeit ausführen müssen, für die ein Computer viel besser geeignet wäre. Schöner wäre es sicherlich, diesen Vorgang mit einem Befehl starten zu können und sich dann anderen Aufgaben zuwenden zu können.

Genau das ermöglichen Kommandofolgen! Wie Sie im folgenden sehen werden, können Sie eine Kommandofolge schreiben, die exakt dasselbe leistet wie die drei oben gezeigten Zeilen und nur mit einem einzigen Befehl aufgerufen werden kann. Falls die Datei, die sortiert und gedruckt werden soll, nicht immer denselben Namen hat, kann die Kommandofolge auch daran flexibel angepaßt werden. Die wichtigsten Vorzüge von Kommandofolgen sind damit schon demonstriert: Kommandofolgen sparen Tipparbeit, machen Fehler unwahrscheinlicher und nehmen einem stupide Arbeit ab!

### 8.4.2 Eine einfache Kommandofolge

Das Format von Kommandofolgen ist denkbar simpel. Man tippt einfach die Shell-Befehle in genau derselben Form, wie man sie auch im Shell-Fenster eingeben würde, in eine Textdatei. Idealerweise macht man dies mit einem Texteditor. Im letzten Kapitel wurden ja der Texteditor MEMACS etwas ausführlicher vorgestellt, der sich für diese Aufgabe ideal eignet. Sie können aber natürlich auch *Notepad* verwenden, wenn Sie damit vertrauter sind.

Für die folgenden Versuche öffnen Sie nun bitte ein Shell-Fenster – falls noch keines vorhanden ist. Richten Sie ein neues Dateiverzeichnis namens *Test* ein und machen Sie es zum aktuellen Directory:

```
MAKEDIR DF0:Test
CD DF0:Test
```

Dieses Dateiverzeichnis wird die Dateien aufnehmen, die für die folgenden Versuche benötigt werden. Der Befehl EXECUTE, mit dem Sie Kommandofolgen ja starten müssen,

»findet« Kommandofolgen nämlich nur im aktuellen Dateiverzeichnis und im logischen Laufwerk S:. Kommandofolgen in anderen Directories müssen Sie unter Angabe des vollen Pfadnamen aufrufen. Wenn Sie später »ernsthafte« Kommandofolgen schreiben, sollten Sie diese deshalb immer in das logische Laufwerk S: legen. Für Ihre ersten Versuche in dieser Hinsicht ist ein separates Directory aber besser, da Sie es später leicht mit einem DELETE-Befehl wieder löschen können.

Sie werden nun Ihre erste Kommandofolge erstellen. Verwenden Sie dazu den Texteditor Ihrer Wahl. Falls Sie beim Eintippen der folgenden Zeilen Fehler machen sollten, benutzen Sie bitte die Korrekturmöglichkeiten des jeweiligen Editors. Starten Sie jetzt den Editor mit einem der folgenden Befehle:

```
MEMACS ZeigeText
notepad
```

Tippen Sie dann die folgenden Zeilen ein:

```
ECHO "Start der Kommandofolge"
COPY text TO kopie
TYPE kopie NUMBER
DELETE kopie
ECHO "Kommandofolge beendet!"
```

Sichern Sie dann Ihr Werk (wenn Sie *Notepad* verwenden, müssen Sie dazu den Namen »ZeigeText« angeben) und verlassen Sie den Editor. Wenn Sie zurück im Shell-Fenster sind, können Sie sich mit CD davon überzeugen, daß Sie Ihre erste Kommandofolge unter dem Namen *ZeigeText* erzeugt haben. Starten Sie sie nun mit dem folgenden Befehl

```
EXECUTE ZeigeText
```

Sie wird aber schon in der zweiten Zeile abgebrochen, da der COPY-Befehl nicht funktionieren kann. Es gibt ja keine Datei namens *Text* im aktuellen Dateiverzeichnis, die kopiert werden könnte. Diese müssen Sie erst einmal erzeugen.

Starten Sie dazu noch einmal den Editor Ihrer Wahl und geben Sie ein paar kurze Zeilen ein, die Sie unter dem Namen »Text« abspeichern. Wenn Sie sich Arbeit ersparen wollen, können Sie auch eine vorhandene Textdatei (zum Beispiel S:Startup-Sequence) unter dem Namen »Text« in das aktuelle Directory kopieren. Starten Sie dann noch einmal Ihre Kommandofolge mit *execute ZeigeText*. Es sollten nun die folgenden Zeilen auf Ihrem Bildschirm erscheinen:

```
1.SYS:> EXECUTE ZeigeText
Start der Kommandofolge
    1 Hier erscheinen die von Ihnen einge-"
    2 gebenen Zeilen der Datei text mit
    3 vorangestellten Zeilennummern .....
...
Kommandofolge beendet!
1.SYS:>
```

Die Kommandofolge *ZeigeText* hat also soeben selbständig und ohne Ihr Zutun fünf Shell-Befehle ausgeführt. Nicht schlecht für den Anfang! Nur ist die gezeigte Kommandofolge noch sehr unflexibel. Sie kann nur eine einzige Datei, die den Namen *text* hat, ausgeben.

Wollen Sie einmal eine andere Datei ausgeben, so müßten Sie dafür eine neue Kommandodatei schreiben, die sich kaum von der ersten unterscheiden würde. Nur der Name *text* müßte bei jeder dieser neuen Kommandodateien durch einen anderen ersetzt werden. Eine sehr lästige Angelegenheit.

Glücklicherweise können Sie diese Ersetzung auch durch den EXECUTE-Befehl ausführen lassen. Sie werden gleich eine Kommandofolge schreiben, die genau diese Möglichkeit nutzt und eine beliebige Datei ausgeben kann, deren Namen Sie der Kommandofolge beim Aufruf als *Parameter* übergeben müssen.

### 8.4.3 Kommandofolgen ohne EXECUTE aufrufen

Vorher möchte ich Ihnen aber schnell noch einen Trick vorstellen, mit dem Sie sich ersparen können, den relativ langen Befehlsnamen EXECUTE beim Aufruf einer Kommandofolge zu verwenden. Genaugenommen handelt es sich dabei nicht einmal um einen Trick, sondern um eine von den Amiga-Entwicklern genau für diesen Zweck vorgesehene Möglichkeit. Ab der Version 1.3 der Workbench-Version kann eine Datei mit einem »Schutzflag« versehen werden, das hauptsächlich von der Shell verwendet wird und nichts anderes besagt, als daß die entsprechende Datei eine Kommandofolge enthält. Es handelt sich dabei um das S-Flag, das mit dem Befehl PROTECT gesetzt werden kann. (Eine ausführliche Beschreibung dieses Befehls finden Sie in Kapitel 7.) Probieren Sie es am besten gleich einmal aus:

```
PROTECT ZeigeText +s
```

Nachdem dieser Befehl ausgeführt wurde, brauchen Sie den EXECUTE-Befehl nicht mehr, um die Kommandofolge *ZeigeText* zu starten. Geben Sie statt dessen einfach den Namen der Kommandofolge ein, als würde es sich um einen Shell-Befehl handeln:

```
ZeigeText
```

Die Shell startet dann sofort die Kommandofolge, da Sie an dem S-Flag sieht, daß es sich um eine Kommandofolge und nicht um ein Befehlsprogramm handelt. Ein zusätzlicher Nutzen des S-Flags ist übrigens auch, daß die Shell Kommandofolgen, die mit diesem Flag versehen sind, nicht nur im aktuellen Directory und im logischen Laufwerk S: findet, sondern alle Directories im Suchpfad danach absucht. Kommandofolgen können also wirklich genauso verwendet werden, als würde es sich um »echte« Programme handeln.

### 8.4.4 Kommandofolgen mit Parametern

Um Parameter in Ihre Kommandofolge »einzubauen«, müssen Sie zwei Dinge tun. Sie müssen am Anfang der Kommandofolge dem EXECUTE-Befehl mitteilen, welche Parameter mit welchen Namen die Kommandofolge benötigt. In der Kommandofolge selbst muß dann noch der Name *text* gegen einen »Platzhalter« ersetzt werden. Diesen Platzhalter ersetzt EXECUTE dann, bevor die Kommandofolge wirklich ausgeführt wird, gegen den Namen, den Sie beim Aufruf der Kommandofolge angeben. Die Kommandofolge *ZeigeText* sieht dann aus wie folgt:

```
.KEY datei/A
ECHO "Start der Kommandofolge"
COPY <datei> TO kopie
TYPE <datei> NUMBER
DELETE kopie
ECHO "Kommandofolge beendet!"
```

Wie Sie sehen, taucht in dieser Kommandofolge überall, wo vorher *text* stand, nun der Platzhalter *<datei>* auf. Ferner ist eine zusätzliche erste Zeile eingefügt worden, die angibt, welche Parameter die Kommandofolge benötigt. Führen Sie nun bitte die nötigen Änderungen an der Datei *ZeigeText* mit Hilfe ihres Lieblings-Editors durch und speichern Sie die Datei wieder unter dem Namen *ZeigeText* ab.

Die neue erste Zeile von *ZeigeText* sieht deutlich anders aus als die restlichen Zeilen. Sie beginnt zum Beispiel mit einem Punkt, was sie als eine »besondere« Zeile kennzeichnet, die nicht als Shell-Befehl ausgeführt werden darf, sondern spezielle Informationen für EXECUTE enthält. Eine solche Zeile nennt man auch »Punktbefehl«. EXECUTE sucht vor der Ausführung jeder Kommandofolge nach solchen Zeilen und wertet sie aus. Das Wort KEY nach dem Punkt bedeutet, daß der Rest der Zeile die Beschreibung eines Parameters enthält. In diesem Fall lautet sie *.KEY datei/A*, was bedeutet, daß *ZeigeText* einen Parameter namens *datei* hat und dieser Parameter notwendig ist, also beim Aufruf nicht fortfallen darf. Letzteres regelt der Zusatz /A, der nicht zum eigentlichen Namen des Parameters gehört. Stände hinter *datei* kein /A, so wäre *datei* ein optionaler Parameter, was unter Umständen auch recht sinnvoll sein kann. Mehr darüber aber weiter unten. (Wenn Sie sich noch an die Erläuterungen am Anfang von Kapitel 7 erinnern, wird ihnen sicherlich klar sein, daß hinter dem Schrägstrich noch andere Buchstaben folgen können und das diese dieselbe Bedeutung haben wie die Buchstaben hinter den Schlüsselwörtern im Aufrufformat eines Shell-Befehls.)

Probieren Sie nun einmal das neue *ZeigeText* aus. Wenn Sie einfach nur den Befehl *EXECUTE ZeigeText* eingeben, wie bisher, erhalten Sie sofort eine Fehlermeldung, die besagt, daß die Parameter, die Sie angegeben haben (keine) nicht geeignet (... *not suitable* ...) sind. Tippen Sie aber *EXECUTE ZeigeText Text*, so erhalten Sie am Bildschirm dasselbe Ergebnis wie früher mit *EXECUTE ZeigeText*. Probieren Sie auch einmal den Befehl *EXECUTE ZeigeText ZeigeText* aus. Dann wird Ihnen am Bildschirm der Inhalt der Kommandofolge selbst gezeigt.

Kommen wir nun zum trockeneren Teil dieses Abschnitts. Bisher haben Sie ja nur am Beispiel gesehen, wie man Parameter verwendet. Nun sollen Ihnen auch alle Details offenbart werden. Wie schon oben zu sehen, gehört zu jeder Kommandofolge mit Parametern eine KEY-Zeile. Eine solche Zeile teilt EXECUTE mit, welche Parameter benötigt werden, wie sie heißen, wie sie behandelt werden müssen und ob sie optional oder notwendig sind. In einer KEY-Zeile können gleichzeitig mehrere Parameter spezifiziert werden. Die einzelnen Parameter werden dann durch Kommata getrennt. Die folgende Zeile spezifiziert zum Beispiel drei (optionale) Parameter:

```
.KEY datei,disk,name
```

Zwischen den Namen der Parameter und den Kommata dürfen auf keinen Fall Leerstellen stehen! Schreiben Sie alle Namen direkt hintereinander.

Hinter dem Namen jedes Parameters können, von einem Schrägstrich getrennt, weitere Buchstaben stehen, die angeben, wie der Parameter zu behandeln ist. Es handelt sich dabei um dieselben Buchstaben, die auch in den Befehlsbeschreibungen im letzten Kapitel (im Aufrufformat) verwendet wurden. Auch ihre Bedeutung ist dieselbe. Für die Parameter einer Kommandofolge sind allerdings nur die Buchstaben A und K sinnvoll zu verwenden. (S kann verwendet werden, macht aber nicht viel Sinn.) Der Zusatz /A bedeutet, daß der Parameter nötig ist, also beim Aufruf unbedingt angegeben werden muß (mehr zu optionalen Parametern weiter unten). Wird er beim Aufruf vergessen, ergibt das eine Fehlermeldung und die Kommandofolge wird abgebrochen.

Der Zusatz /K bedeutet, daß der Parameter immer von seinem Namen eingeleitet werden muß. Würde die erste Zeile einer Kommandofolge *.KEY datei/A/K* lauten, so müßte beim Aufruf das Schlüsselwort DATEI immer verwendet werden (die Großschreibung von DATEI dient nur der besseren Lesbarkeit); zum Beispiel:

```
ZeigeText DATEI Text
```

Der Sinn dieser auf den ersten Blick etwas umständlichen Einrichtung mit einem notwendigen Schlüsselwort vor dem Parameter liegt in der besseren Lesbarkeit eines Aufrufs – besonders, wenn optionale Parameter ins Spiel kommen. Hat eine Kommandofolge drei optionale Parameter und Sie übergeben nur zwei, hat es EXECUTE nämlich schwer, »sich zu entscheiden«. Haben Sie den ersten und zweiten Parameter übergeben und den dritten freigelassen? Oder handelt es sich um den ersten und dritten und der zweite soll freibleiben? Oder sind es gar der zweite und dritte Parameter und der erste bleibt unbelegt? Schreiben Sie aber vor die beiden Parameter auch noch ihre Namen, wird die Sache eindeutig. Sie können dann sogar die Reihenfolge austauschen und erst den zweiten, dann den ersten Parameter angeben, und so weiter. Die Namen machen die ganze Sache eindeutig.

Nachdem Sie einen oder mehrere Parameter in der ersten Zeile spezifiziert haben, können diese Parameter in der Kommandofolge innerhalb jedes Shell-Befehls verwendet werden. An den Stellen, an denen der »Wert« des Parameters erscheinen soll, müssen Sie den Namen des Parameters in spitzen Klammern eingeben; zum Beispiel *<datei>* für den Parameter *datei*. Der Wert eines Parameters ist der Text, der beim Aufruf anstelle des Parameters eingegeben wurde. Vor der Ausführung der Kommandofolge wird dann jedes Vorkommen des Platzhalters <datei> gegen den Wert des beim Aufruf angegebenen Parameters ersetzt. Zu diesem Zweck erzeugt EXECUTE wirklich eine zweite Version der Kommandofolge (im logischen Laufwerk T:), in der alle Platzhalter durch die entsprechenden Parameter ersetzt werden. Diese modifizierte Fassung der Kommandofolge wird dann ausgeführt – nicht das Original mit den Platzhaltern.

Problematisch wird diese Ersetzung bei optionalen Parametern, für die beim Aufruf der Kommandofolge kein Wert angegeben wurde. Gegen welchen Text sollen die entsprechenden Platzhalter ersetzt werden? Wenn Sie nichts »Besonderes« unternehmen, ersetzt EXECUTE solche Platzhalter gegen den leeren Text "". Dies dürfte manchmal sinnvoll sein, oft aber auch nicht. Wenn der Befehl TYPE oder COPY beispielsweise "" an einer Stelle als Parameter erhält, wo eigentlich ein Dateiname erwartet wird, gibt das entweder einer Fehlermeldung oder es passieren unerwartetet Dinge. COPY kopiert beispielsweise den kompletten Inhalt des aktuellen Directory, wenn der erste Parameter "" lautet.

Für die Parameter von Kommandofolgen, die niemals »leer« sein dürfen, gibt es deshalb eine besondere Einrichtung, die sogenannten »Defaultwerte«. Ein Defaultwert wird immer

dann für einen Platzhalter eingesetzt, wenn dem entsprechenden Parameter beim Aufruf der Kommandofolge kein Wert zugeteilt wurde. Es gibt zwei Möglichkeiten, einem Parameter einen Defaultwert zuzuteilen.

Die erste Möglichkeit besteht daraus, den Punkt-Befehl DEF am Anfang der Kommandofolge zu verwenden. Dieser Punkt-Befehl sieht ähnlich wie der KEY-Befehl aus. Das allgemeine Format eines DEF-Befehls lautet:

```
.DEF Parameter "Defaultwert"
```

Jedes Vorkommen des entsprechenden Platzhalters <*Parameter*> wird dann, natürlich nur, falls kein anderer Wert für *Parameter* beim Aufruf übergeben wurde, gegen den Defaultwert ersetzt. Dieser Punktbefehl ist also global und beeinflußt alle Platzhalter des jeweiligen Parameters.

Flexibler ist die Möglichkeit, direkt beim Platzhalter einen Defaultwert anzugeben. Der Platzhalter eines Parameters kann so an unterschiedlichen Stellen einen unterschiedlichen Defaultwert haben. Hierzu schreibt man hinter dem Namen des Parameters im Platzhalter den Defaultwert. Name und Defaultwert werden getrennt durch ein Dollarzeichen. Der Platzhalter <*Datei$Text*> hat also den Wert des Parameters *Datei*, falls dieser Parameter beim Aufruf angegeben wird und den Wert »Text«, falls kein Wert für *Datei* beim Aufruf angegeben wird. Den selben Effekt könnten Sie für alle Vorkommnisse des Platzhalters <Datei> erzielen, wenn Sie am Anfang der Datei den folgenden Punkt-Befehl verwenden würden:

```
.DEF Datei "Text"
```

Diese Möglichkeit soll jetzt sofort einmal für die erste Beispiel-Kommandofolge eingesetzt werden. Hierzu ändern Sie bitte die Kommandofolge *ZeigeText*, bis sie das folgende Aussehen hat:

```
.KEY Datei
.DEF Datei "Text"
ECHO "Start der Kommandofolge"
COPY <Datei> TO kopie
TYPE <Datei> OPT N
DELETE kopie
ECHO "Kommandofolge beendet!"
```

Es muß also in der ersten Zeile der alten Version der Datei das */A* gelöscht werden und die zweite Zeile so eingefügt werden, wie sie hier gezeigt wurde. Falls Sie nicht mehr wissen, wie die Änderungen auszuführen sind, lesen Sie die dafür nötigen Befehle und Handgriffe einfach in der Beschreibung des Texteditors nach.

Nun sollten Sie Ihre neue und verbesserte Kommandofolge aber auch ausprobieren. Sichern Sie dazu die geänderte Fassung von *ZeigeText*, verlassen Sie den Texteditor und geben Sie dann den folgenden Befehl ein:

```
ZeigeText
```

Wie in der allerersten Version der Kommandofolge *ZeigeText* wird Ihnen nun der Inhalt der Datei *Text* gezeigt – sofern kein Parameter angegeben wurde. Probieren Sie jetzt auch gleich noch die Verwendung eines Parameters aus:

```
ZeigeText S:startup-sequence
```

Nun wird der Inhalt der Datei *Startup-sequence* angezeigt. Defaultwerte sind ganz schön praktisch, nicht wahr?

### 8.4.5 Spezielle Punktbefehle

Damit wissen Sie nun schon fast alles, was es zum Thema Parameter zu sagen gibt. Für die Verwendung von Parametern sind Punktbefehle besonders wichtig. Und zu diesem Thema gibt es noch einiges mehr zu sagen.

Am wichtigsten ist, das die erste Zeile jeder Kommandofolge, in der Parameter (beziehungsweise deren Platzhalter) oder Punktbefehle vorkommen, ein Punktbefehl sein muß! Ist die erste Zeile kein Punktbefehl (egal welcher) ersetzt EXECUTE keine Platzhalter durch die aktuellen Parameter.

Außer den zwei Punktbefehlen *.KEY* und *.DEF* gibt es noch eine Reihe anderer Punktbefehle, die zwar nicht häufig benötigt werden, der Vollständigkeit halber nun aber auch noch aufgelistet werden. Wie *.KEY* und *.DEF* müssen auch sie alle am Beginn einer neuen Zeile stehen (der Punkt muß also der erste Buchstabe der Zeile sein), sonst erkennt EXECUTE sie nicht als Punktbefehle.

*. "beliebiger Text"* ist ein Kommentar, also eine Zeile, die nicht als Befehl ausgeführt wird. Zwischen dem Punkt am Anfang der Zeile und dem nachfolgenden Kommentartext muß mindestens eine Leerstelle stehen.

Der eben beschriebene Punkt-Befehl kann nur dann verwendet werden, wenn eine komplette Zeile als Kommentar gelten soll. Wenn Sie hinter einen Shell-Befehl einen Kommentar (zur Erläuterung) setzen wollen, können Sie dafür das Semikolon (Strichpunkt ;) verwenden. Setzen Sie diesen einfach hinter den Shell-Befehl und schreiben Sie dahinter einen beliebigen Text. Text hinter einem Semikolon wird bei der Ausführung der Kommandofolge ignoriert.

*.K* ist eine Abkürzung für .KEY.

*.DOT* und dahinter ein einzelner Buchstabe führt für den Rest der Kommandofolge dazu, daß nicht mehr der Punkt, sondern der hier angegebene Buchstabe zur Einleitung von Punktbefehlen dient. Nach *.DOT !* wird also *.DEF* nicht mehr als Punktbefehl erkannt. Man muß statt dessen *!DEF* schreiben.

*.BRA* und dahinter ein einzelner Buchstabe, führt für den Rest der Kommandofolge dazu, daß nicht mehr das Kleiner-Zeichen < zur Einleitung von Platzhaltern dient. Statt dessen müssen Platzhalter für Variablen jetzt mit dem angegebenen Buchstaben beginnen. Nach *.BRA {* wird also *<datei>* nicht mehr als Platzhalter erkannt; man muß statt dessen *{datei>* schreiben.

*.KET* und dahinter ein einzelner Buchstabe, führt für den Rest der Kommandofolge dazu, daß nicht mehr das Größer-Zeichen > zur Beendigung von Platzhaltern dient. Statt dessen müssen Platzhalter für Variablen jetzt mit dem angegebenen Buchstaben enden. *Nach .KET }* wird also *<datei>* nicht mehr als Platzhalter erkannt; man muß statt dessen *<datei}* schreiben.

Stehen am Anfang einer Kommandofolge also (in zwei getrennten Zeilen!) die beiden Befehle *.BRA {* und *.KET }*, müssen in den darauffolgenden Zeilen Platzhalter in geschweifte und nicht in spitze Klammern eingeschlossen werden.

Die Änderung der beiden Sonderzeichen < und > ist besonders dann notwendig, wenn Sie in einer Kommandofolge die Ein/Ausgabeumleitung verwenden wollen. Auch dafür werden ja dieselben Zeichen verwendet, was EXECUTE bei der Ausführung einer Kommandofolge »verwirren« kann. Durch Umdefinition von < und > auf zwei andere Zeichen können Sie das vermeiden.

*.DOL* oder *.DOLLAR* und dahinter ein einzelner Buchstabe, führen für den Rest der Kommandofolge dazu, daß nicht mehr das Dollar-Zeichen *$* zur Trennung von Parameternamen und Defaultwerten in Platzhaltern dient. Statt dessen müssen Parameternamen und Defaultwerte in Platzhaltern jetzt durch den angegebenen Buchstaben getrennt werden. Nach dem Befehl *.DOL * wird also *<datei$text>* nicht mehr als Platzhalter mit Defaultwert erkannt; man muß statt dessen *<datei\text>* schreiben.

Schauen Sie sich diese Punktbefehle sorgfältig an! Bis auf die Kommentare werde ich sie zwar in den folgenden Beispiel-Kommandofolgen nicht verwenden, sollten Sie aber einmal eine Kommandofolge lesen müssen, die jemand anderes geschrieben hat, ist es gut, diese Befehle zu kennen.

Neben den mit KEY-Befehlen definierten Parametern gibt es übrigens auch noch einen ganz speziellen Parameter, der in allen Kommandofolgen verwendet werden kann. Der Platzhalter <$$> wird bei der Ausführung einer Kommandofolge immer gegen die Nummer des Shell-Prozesses ersetzt, von dem aus die Kommandofolge gestartet wurde. Das kann man dazu verwenden, eindeutige Dateinamen für temporäre Dateien zu erzeugen. (Mehr zum Thema temporäre Dateien erfahren Sie in den folgenden Abschnitten dieses Kapitels.)

### 8.4.6 Die IF-ELSE-ENDIF-Anweisung

Im vorangegangenen Abschnitt haben Sie schon eine recht komfortable Kommandofolge konstruiert. Sie hat aber immer noch einige Mängel oder Schwächen. Wäre es zum Beispiel nicht besser, wenn die Kommandofolge auch selbst merken würde, ob die Datei, die sie kopieren und ausdrucken will, auch existiert und eine entsprechende Fehlermeldung ausgeben würde, falls nicht? Im Augenblick bricht die Kommandofolge *ZeigeText* in diesem Fall nämlich nur mit einer (englischen!) Fehlermeldung ab, die auch dann, wenn man der englischen Sprache mächtig ist, recht nichtssagend ist und wenig über die Art des Fehlers aussagt.

In diesem Abschnitt werden Sie erfahren, wie Sie Fehler, die bei der Ausführung einer Kommandofolge auftreten, »abfangen« und dem Benutzer sinnvoll melden können. Dazu werden Verzweigungen beziehungsweise Programmsprünge verwendet.

Die erste Kommandofolge, die Sie bis jetzt kennengelernt haben, ist ja extrem geradlinig. Egal was geschieht, folgt sie stur der Abfolge von Shell-Befehlen, wie Sie sie eingegeben haben, bis die Befehlsfolge zu Ende ist oder ein Fehler auftritt. Das ist nicht immer wünschenswert. Wenn während der Ausführung einer Kommandofolge zum Beispiel ein Fehler auftritt, kann es sinnvoll sein, die Kommandofolge nicht einfach abzubrechen. Manche Fehler können vielleicht ausgebügelt werden. Zumindest sollte dem Benutzer in sinnvoller Form die Fehlerursache mitgeteilt werden. Unter Umständen kann es auch

sinnvoll sein, zwei Fliegen mit einer Klappe zu schlagen, wenn man eine Kommandofolge schreibt – also zwei verschiedene Aufgaben zu erledigen, je nachdem welche Parameter man ihr übergibt. Beides ist nur dann möglich, wenn man Entscheidungspunkte in eine Kommandofolge einbauen kann, an denen man, abhängig von gewisssen Bedingungen, entscheiden kann, ob die eine oder andere Folge von Befehlen durchgeführt wird. Zunächst soll die Kommandofolge *ZeigeText* deshalb nachsehen, ob die auszugebende Datei überhaupt existiert. Die dafür nötige Änderung ist relativ simpel. Sie zeigt eine einfache Anwendung des IF-Befehls (englisch *if* = wenn/falls):

```
.KEY Datei
.DEF Datei "Text"
ECHO "Start von ZeigeText"
IF EXISTS <Datei>
  COPY <Datei> TO kopie
  TYPE <Datei> OPT N
  DELETE kopie
  ECHO "ZeigeText erfolgreich beendet!"
ELSE
  ECHO "Fehler: <datei> existiert nicht!"
ENDIF
```

(Die Einrückungen in der Kommandofolge dienen dazu, Befehle zu Gruppen zusammenzufassen und die Befehle, die den Ablauf steuern, wie IF, ELSE und ENDIF hervorzuheben. Wenn Ihnen das zu lästig ist, können Sie die entsprechenden Leerstellen am Anfang der Zeile aber auch weglassen – der Shell ist's egal.)

Die neue geänderte Version von *ZeigeText* zeigt nun immer noch den Inhalt der Datei im Shell-Fenster, deren Name als Parameter übergeben wird – falls es diese Datei überhaupt gibt. Falls es keine Datei dieses Namens gibt, bricht nicht etwa die Kommandofolge ab, sondern es wird eine (hoffentlich) verständliche Fehlermeldung ausgegeben, die dem Anwender das Problem verdeutlicht. Der normale Ablauf der Kommandofolge, wie Sie ihn bisher kannten, wird dazu in die Befehle IF und ELSE eingeschlossen. Der Fehlerfall wird hinter ELSE behandelt.

Umgangssprachlich sieht die Vorgehensweise der neuen Kommandofolge etwa folgendermaßen aus: Wenn (IF) die auszugebende Datei vorhanden ist, kopiere sie und gib sie aus, sonst (ELSE) melde einen Fehler (ENDIF). Die Bedingung EXISTS <datei> ist dabei nur eine von mehreren Bedingungen, die Sie hinter IF verwenden können. Weitere Bedingungen werden Sie im nächsten Abschnitt kennenlernen.

Obwohl IF, ELSE und ENDIF einzelne Shell-Befehle sind, werden sie immer zusammen verwendet und sollen deshalb auch hier zusammen als »IF-Anweisung« erläutert werden. Das Format einer IF-Anweisung lautet:

```
IF <Bedingung>
 Folge von Shell-Befehlen...
 ...
ELSE
 andere Folge von Shell-Befehlen...
 ...
ENDIF
```

Wenn die Liste der Shell-Befehle hinter ELSE leer ist, so kann man ELSE auch ganz wegfallen lassen und die Anweisungsfolge sofort mit ENDIF beenden. Die Vorgehensweise von EXECUTE, wenn bei der Ausführung einer Kommandofolge auf ein IF getroffen wird, ist die folgende:

Die Bedingung hinter dem IF wird überprüft. Falls sie zutrifft, werden die Anweisungen hinter dem IF ausgeführt, bis ein ELSE oder ENDIF gefunden wird, je nachdem, was zuerst eintritt. Trifft EXECUTE auf ein ELSE, so werden alle Kommandos dahinter ignoriert, bis der Befehl ENDIF auftaucht. Hinter dem ENDIF wird aber auf alle Fälle wieder mit der Ausführung der Kommandos fortgefahren.

Trifft die Bedingung hinter IF nicht zu, so werden alle Anweisungen ignoriert, bis der Befehl ELSE oder ENDIF auftaucht, je nachdem was zuerst eintritt. Dahinter wird auf alle Fälle wieder mit der Ausführung der Kommandos fortgefahren.

Eine IF-Anweisung hat also zwei »Zweige«, die alternativ durchlaufen werden. Entweder die Kommandos in dem Zweig hinter dem IF oder die Kommandos in dem anderen Zweig hinter dem ELSE werden durchgeführt – niemals beide. Fällt der Zweig hinter dem ELSE ganz fort, so kann man auch von einer bedingten Anweisungsfolge sprechen. Nur wenn die Bedingung zutrifft, wird sie ausgeführt – sonst nicht.

IF-ELSE-ENDIF-Anweisungen können übrigens auch geschachtelt werden. Deshalb gehört nicht immer das unmittelbar nächste ELSE zu einem IF. Das ist zum Beispiel dann nicht der Fall, wenn im IF-Zweig erneut ein IF-Befehl auftaucht.

```
IF <Bedingung>                  ; äußeres IF
 andere CLI-Befehle...
.
  IF <Bedingung>          ;     inneres IF
    andere CLI-Befehle...
  ELSE                    ;     inneres ELSE
    andere CLI-Befehle...
  ENDIF                   ;     inneres ENDIF
.
ELSE                             ; äußeres ELSE
 andere CLI-Befehle...
 ...
ENDIF                            ; äußeres ENDIF
```

Die Zuordnung von einem ELSE oder ENDIF zu einem bestimmten IF funktioniert ähnlich wie mit geschachtelten Klammern und bedarf wohl keiner ausführlichen Erläuterung. Beachten Sie in dem obigen Beispiel die Einrückung der verschiedenen Befehle, die die Gruppierung und Zuordnung zu den beiden verschiedenen IF-ELSE-ENDIF-Anweisungen deutlich macht.

### 8.4.7 Mögliche Bedingungen für den IF-Befehl

Entscheidend für die »Mächtigkeit« des IF-Befehls sind die Bedingungen, die abgeprüft werden können. Die Bedingung EXISTS <datei> ist nur eine von mehreren möglichen Arten von Bedingungen. Weitere Bedingungen sind zum Beispiel WARN, ERROR und FAIL, die

dann zutreffen, wenn der letzte Shell-Befehl fehlschlug und der »Returncode« größer oder gleich 5 (bei WARN), 10 (bei ERROR) oder 20 (bei FAIL) war. Jeder fehlgeschlagene Shell-Befehl setzt einen solchen Returncode (für den man mit dem Shell-Befehl WHY auch eine Erläuterung erhalten kann). Trat bei der Ausführung des Befehls kein Problem auf, ist der Fehlercode gleich Null.

Mit den Bedingungen WARN, ERROR und FAIL kann man abprüfen, ob ein Fehler passiert ist, und dann geeignete Gegenmaßnahmen ergreifen oder wenigstens den Benutzer ausführlicher informieren als es der Befehl selbst tut. Bei allen drei Bedingungen muß man aber berücksichtigen, daß sie abhängig von dem aktuellen »Fehlerabbruchlevel« sind. Ist der Returncode größer als der Fehlerabbruchlevel, so wird die Ausführung einer Kommandofolge sofort abgebrochen und eine eventuell folgende IF-Anweisung wird überhaupt nicht mehr ausgeführt.

Bei der Ausführung einer Kommandofolge ist Höhe des Fehlerabbruchlevels normalerweise gleich 10. Sobald der Returncode eines Shell-Befehls 10 erreicht oder übersteigt, wird deshalb die Kommandofolge abgebrochen. Manchmal möchte man aber, daß nur bei sehr schwerwiegenden Fehlern abgebrochen wird oder man möchte den Fehler selbst erkennen und innerhalb der Kommandofolge darauf reagieren. In diesem Fall muß man den Fehlerabbruchlevel heraufsetzen.

Hierzu dient der Befehl FAILAT, der einen einzigen Parameter <n> hat. Durch FAILAT wird Fehlerabbruchlevel auf <n> gesetzt. Vor jedem Befehl, der einen Fehler verursachen könnte, den man selbst behandeln will, sollte man deshalb mit FAILAT den Fehlerabbruchlevel hochsetzen. Mehr dazu aber weiter unten in diesem Kapitel.

Mit diesen Bedingungen können Kommandofolgen vom Erfolg vorangehender Befehle abhängig gemacht werden. Damit solche Bedingungen sinnvoll sind, muß unbedingt das Fehlerabbruchlevel mit FAILAT hochgesetzt werden. Ist das Fehlerabbruchlevel zum Beispiel kleiner als 10, wird ein IF-Befehl mit der Bedingung ERROR überhaupt nie ausgeführt, weil die Kommandofolge schon vorher abgebrochen wird.

Neben der Überprüfung des Returncodes beziehungsweise des Vorhandenseins einer Datei kennt der IF-Befehl auch noch eine Reihe von Bedingungen, mit denen zwei Texte verglichen werden können:

Die Bedingungen <Text1> EQ <Text2> trifft dann zu, wenn <Text1> gleich <Text2> ist (wobei Groß- beziehungsweise Kleinschreibung keine Rolle spielen. <Text1> GT <Text2> trifft dann zu, wenn <Text1> in alphabetischer Sortierung nach <Text2> kommt. Und <Text1> GE <Text2> trifft dann zu, wenn <Text1> in alphabetischer Sortierung nach <Text2> kommt oder gleich <Text2> ist (wobei Groß- beziehungsweise Kleinschreibung auch hier keine Rolle spielen.

Bei diesen drei Formen einer IF-Bedingung wird üblicherweise wohl entweder <Text1> oder <Text2> ein Parameter der Kommandofolge sein. Zwei Texte, deren Inhalt beim Schreiben der Kommandofolge schon feststeht, miteinander zu vergleichen, ergibt ja kaum Sinn.

Die beiden Bedingungen GT (englisch *greater than* = größer als) und GE (englisch *greater or equal* = größer oder gleich) vergleichen <Text1> und <Text2> als Texte. Die Bedingung »3 GT 12« würde deshalb zutreffen (3 kommt bei alphabetischer Sortierung nach 12). Wenn Sie einen numerischen Vergleich wünschen, müssen Sie zusätzlich das Schlüsselwort VAL

angeben. Dann werden <Text1> und <Text2> als Zahlen verglichen und » VAL 3 GT 12« würde nicht mehr zutreffen.

Vor alle bisher genannten Bedingungen können Sie zudem auch noch das Schlüsselwort NOT setzen (zum Beispiel IF NOT EXISTS <Text>). Dieses Schlüsselwort kehrt den Sinn der Bedingung um. (IF NOT EXISTS <Text> würde also dann zutreffen, wenn die Datei <Text> nicht existiert.)

### 8.4.8 Die Sprunganweisungen SKIP und QUIT

Eine IF-Anweisung wird oft dazu verwendet, einen Teil einer Kommandofolge zu umgehen beziehungsweise zu überspringen. Sie kann deshalb unter Umständen auch als ein Sprung aufgefaßt werden. Trifft eine Bedingung nicht zu, werden ja die Anweisungen zwischen IF und ELSE oder ENDIF übersprungen. Noch deutlicher wird der Sprung-Character von IF, wenn der ELSE-Zweig fortfällt.

Eine etwas direktere Möglichkeit, eine Folge zu überspringen, stellt der SKIP-Befehl dar. Trifft man innerhalb einer Kommandofolge auf ein SKIP, so bedeutet dies ein Überspringen (englisch *skip*) aller Kommandos in der Folge, bis zu einem LAB-Kommando (LAB für *label*; zu deutsch Marke). Sowohl SKIP als auch LAB haben einen Parameter, der aus einem beliebigen Text bestehen kann. SKIP springt immer bis zu einem LAB, das denselben Parameter hat. Dazwischenliegende LAB mit anderen Parametern werden ebenfalls übersprungen, wie alle anderen Befehle auch.

```
SKIP Ende
 bel. CLI-Befehle...
 ...
LAB  Ende          ; Sprung bis hier
 bel. CLI-Befehle...
```

Wenn Sie sich das Ausdenken eines Namens ersparen wollen, können Sie statt dem Befehl LAB auch ENDSKIP verwenden. Hinter ENDSKIP fährt die von einem SKIP-Befehl unterbrochene Abarbeitung der Befehle in einer Kommandofolge immer fort, egal welchen Parameter (Marken-Namen) der SKIP-Befehl hatte. Deshalb sollte man den ENDSKIP-Befehl auch mit Vorsicht verwenden, falls eine Kommandofolge mehr als einen SKIP-Befehl enthält. Hinter dem ENDSKIP-Befehl ist nämlich nicht mehr klar, von welchem SKIP-Befehl der Sprung ausgegangen ist beziehungsweise was die Ursache für den Sprung zu ENDSKIP war.

Die letzte »Sprunganweisung«, die Sie in Kommandofolgen verwenden können, ist schließlich QUIT. Dieser Befehl stellt einen recht »brutalen« Sprung dar, mit dem die gesamte Kommandofolge verlassen wird, als wäre ihr Ende erreicht worden. Der Befehl QUIT besitzt einen optionalen Parameter, der ein Returncode (eine Zahl) sein sollte. Ist er nicht vorhanden, wird der aktuelle Returncode nach Aufruf von QUIT auf Null gesetzt, als wäre kein Fehler passiert. Besonders wenn die Möglichkeit besteht, daß eine Kommandofolge von anderen Kommandofolgen aufgerufen wird, sollte man im Fehlerfall immer einen Fehlercode mit QUIT »nach außen geben«, damit die andere Kommandofolge eine Chance hat, auf einen eventuellen Fehler zu reagieren. Bei Kommandofolgen, die nicht von anderen Kommandofolgen aufgerufen werden, spielt der Returncode keine Rolle.

### 8.4.9 Wie Sie Fehler in Kommandofolgen abfangen

Die Theorie zum Thema Bedingungen und Sprünge kennen Sie jetzt. Nun soll Ihr neues Wissen aber auch in die Praxis umgesetzt werden! Dazu wird die Fehlerbehandlung von *ZeigeText* verbessert.

Die aktuelle Version von ZeigeText meldet zwar korrekt, wenn die auszugebende Datei nicht existiert; andere Fehler können aber passieren, werden nicht erkannt und führen zum Abbruch der Kommandofolge. Die kritische Stelle ist dabei der COPY-Befehl, der vor der Ausgabe des Textes mit TYPE stattfindet. Wenn die Diskette voll oder schreibgeschützt ist, wird er fehlschlagen. Diese Stelle soll in der folgenden Version von *ZeigeText* nun etwas »entschärft« werden.

```
.KEY Datei
.DEF Datei "Text"
ECHO "Start von ZeigeText"
FAILAT 25
COPY <Datei> TO kopie   ; GEFÄHRLICH
IF ERROR
  SKIP Fehler
ENDIF
FAILAT 10           ; Abbruchlevel wieder auf »Normal« setzen
.
.    Alles OK
.
TYPE <datei> NUMBER
DELETE kopie
ECHO "ZeigeText erfolgreich beendet!"
QUIT
.
.    Fehler beim Kopieren
.
LAB Fehler
IF NOT EXISTS <datei>
  ECHO "Fehler: <datei> existiert nicht!"
ELSE
  ECHO "Fehler: Kopieren von <datei> fehlgeschlagen!"
  ECHO "        Diskette voll oder schreibgeschützt ?"
ENDIF
QUIT 10
```

Die einfache, kurze Kommandofolge, mit der wir begonnen haben, ist nun schon recht lang geworden. Sie nutzt fast alle Möglichkeiten, die eine Kommandofolge überhaupt nutzen kann.

Zunächst wird die kritische Stelle, der COPY-Befehl, mit einem FAILAT abgesichert. Selbst wenn der COPY-Befehl nun fehlschlägt, läuft die Kommandofolge weiter. Unmittelbar hinter COPY wird nun geprüft, ob der Befehl mißlungen ist und falls ja, zur Marke *Fehler* gesprungen. Falls kein Fehler auftrat, wird die schon bekannte Befehlsfolge durchlaufen, die Datei am Bildschirm aufgelistet und schließlich mit QUIT die Kommandofolge verlassen.

Vorher aber wird der Abbruchlevel mit FAILAT 10 wieder auf den Normalzustand zurückgesetzt. Das ist in diesem Fall nicht unbedingt nötig aber »guter Stil«, den man sich angewöhnen sollte. Man sollte es stets tun, wenn der Abbruchlevel zuvor hochgesetzt wurde. Sonst werden spätere Fehler, die man nicht vorhergesehen hat, nämlich einfach ignoriert und nicht gemeldet.

Zur Marke *Fehler* kommt die Kommandofolge nur dann, wenn beim COPY-Befehl ein Fehler aufgetreten ist. Dieser Fehler kann verschiedene Ursachen haben. Zunächst wird deshalb geprüft, ob die Datei existiert oder nicht. Wenn nicht, wird eine entsprechende Fehlermeldung ausgegeben. Wenn die Datei existiert, ist die Fehlerursache unklar. Dem Benutzer werden dann zwei wahrscheinliche Fehlermöglichkeiten genannnt. Zum Abschluß wird schließlich noch ein Fehlercode mit QUIT ausgegeben. Auch das ist im Moment für eine einzelne Kommandofolge nicht unbedingt nötig, aber guter Stil, den man sich »für später« angewöhnen sollte.

### 8.4.10 Schleifen

Die »Programmiersprache« von Kommandofolgen enthält im Prinzip alles, was auch in richtigen Programmiersprachen wie Basic oder Pascal zu finden ist. Dazu gehören neben Sprüngen und bedingten Anweisungen auch die Schleifen. Leider kennt die Shell aber keine FOR-NEXT- oder WHILE-Anweisungen wie sie in Basic zu finden sind. Statt dessen muß man sich Schleifen mit Hilfe des SKIP-Befehls »zusammenbauen«. Dabei ist entscheidend, daß SKIP nicht nur vorwärts, sondern auch rückwärts springen kann, wenn Sie hinter dem Namen des Sprungziels (der Marke) das Schlüsselwort BACK (englisch für zurück) angeben.

Am einfachsten ist eine Endlosschleife zu konstruieren. Probieren Sie das gleich einmal aus, indem Sie die folgende Kommandofolge mit Ihrem Lieblingseditor eingeben und unter dem Namen *Schleife1* abspeichern.

```
. Diese Kommandofolge enthält eine Endlosschleife
LAB Start
ECHO "Das ist eine Endlosschleife!"
SKIP Start BACK
```

Diese Kommandofolge kennt kein »natürliches Ende«. Wenn Sie sie ausprobieren, werden Sie feststellen, daß der Text »Das ist eine Endlosschleife!« immer und immer wieder im Shell-Fenster erscheint. (Bedenken Sie beim Ausprobieren, daß Sie auf den EXECUTE-Befehl nur dann verzichten und *ZeigeText* wie einen Shell-Befehl verwenden können, wenn Sie mit dem PROTECT-Befehl das S-Flag der Datei *ZeigeText* gesetzt haben.)

Um eine solche Endlosschleife abzubrechen, müssen Sie die Tastenkombination [Ctrl]+[D] drücken. Die Tastenkombination [Ctrl]+[C] können Sie nicht verwenden; diese bricht nur einen einzelnen Shell-Befehl, nicht aber eine Kommandofolge ab.

Bei der Verwendung des Schlüsselworts BACK ist sehr wichtig, daß Sie nur bis maximal zum letzten, dem SKIP-Befehl vorangehenden EXECUTE-Befehl zurückspringen können. Nur dann, wenn die Kommandofolge vor dem SKIP keinen EXECUTE-Befehl enthält, ist ein Rücksprung bis zum Anfang möglich!

Schleifen in Kommandofolgen einzubauen, die keine Endlosschleifen sind, ist nicht ganz so einfach. Die Shell sieht beispielsweise keinen Befehl vor, mit dem man eine Schleife eine vorher festgelegte Anzahl von Durchläufen ausführen lassen kann. Solche Schleifen kann man aber mit Hilfe von sogenannten »Umgebungsvariablen« konstruieren.

### 8.4.11 Umgebungsvariablen

Eine Umgebungsvariable ist ein Behälter für einen Text, den »Wert« der Variablen, der einen Namen hat. Umgebungsvariablen verhalten sich ähnlich wie Parameter einer Kommandofolge, sind aber dauerhafter. Wird beim Aufruf einer Kommandofolge ein bestimmter Parameter angegeben, so hat dieser nur während des Ablaufs dieser Kommandofolge Gültigkeit. Beim nächsten Aufruf kann der Parameter einen anderen oder auch gar keinen Wert haben. Umgebungsvariablen werden hingegen mit einem bestimmten Shell-Befehl namens SETENV definiert und mit einem Wert versehen. Diesen Wert behalten sie, bis er geändert wird – zum Beispiel mit einem neuen Aufruf von SETENV. Der folgende Befehl definiert zum Beispiel eine Umgebungsvariable namens *X* und weist ihr den Wert »Variable Nummer 1« zu.

```
SETENV X "Variable Nummer 1"
```

Sie können sich den Wert einer solchen Umgebungsvariablen anschauen, indem Sie den Befehl GETENV verwenden. Der folgende Befehl gibt zum Beispiel den Wert von *X* im Shell-Fenster aus:

```
GETENV X
Variable Nummer 1
```

Für Kommandofolgen ist darüber hinaus auch wichtig, daß der Wert einer Umgebungsvariablen in IF-Befehlen wie der Wert eines Parameters verwendet werden kann. Dazu muß dem Namen der Variablen ein Dollarzeichen vorangestellt werden. Die folgenden beiden Befehle würden, wenn man sie in einer Kommandofolge verwendet, zum Beispiel den Wert von *X* benutzen:

```
IF VAL $X GT 0
...
IF $X EQ "Variable Nummer 1"
...
```

Die Werte von Umgebungsvariablen werden in Dateien im logischen Laufwerk ENV: (das sich üblicherweise auf der RAM-Disk RAM: befindet) gespeichert. Der Wert der Umgebungsvariablen *X* wird zum Beispiel in der Datei ENV:X gespeichert. Umgebungsvariablen bleiben deshalb auch nach dem Ende einer Kommandofolge erhalten und haben sogar in verschiedenen Shell-Fenstern denselben Wert. Ihr Wert geht allerdings verloren, wenn Sie den Amiga ausschalten, da ENV: üblicherweise auf der RAM-Disk liegt. Sie können aber durch eine Änderung der Startup-Sequence (siehe unten) auch jederzeit das logische Laufwerk ENV: auf eine Diskette oder Festplatte »umleiten«. Umgebungsvariablen und ihre Werte überstehen dann auch das Ausschalten.

Da die Werte von Umgebungsvariablen in Dateien gespeichert sind, können Sie diese Werte auch mit Befehlen wie TYPE, COPY und ECHO verändern oder Sie durch Eingabeumleitung als Parameter für andere Befehle verwenden. Der folgende Befehl würde zum Beispiel

den Wert der Umgebungsvariablen X am Bildschirm ausgeben (genauso wie der Befehl GETENV):

```
TYPE ENV:X
```

Das Wissen, daß Umgebungsvariablen in Form von Textdateien gespeichert werden, läßt sich innerhalb von Kommandofolgen sehr gut nutzen – unter anderem, um Schleifen aufzubauen. Dazu wird allerdings in recht raffinierter Form der EVAL-Befehl verwendet. Besonders einfach zu verstehen ist diese Art des Aufrufs von EVAL leider nicht. Betrachten Sie einmal das folgende Beispiel zu diesem Thema, das ich der Commodore-Original-Dokumentation zur Workbench 1.3 entnommen und leicht abgeändert habe:

```
.KEY Num
.DEF Num 3
.BRA (
.KET )
SETENV Zahl 1       ; Initialisieren des Schleifenzählers
LAB Start
  ECHO "Schleife Nr.: " NOLINE
  TYPE ENV:Zahl
  EVAL <ENV:Zahl >NIL: TO ENV:Zahl2 OP + VALUE2 1 ?
                    ; Zähler um 1 hochsetzen
  COPY ENV:Zahl2 ENV:Zahl
  IF NOT VAL $Zahl GT (num)
    SKIP Start BACK
  ENDIF
ECHO "Das war's"
```

Diese Kommandofolge führt die innere Schleife (zwischen dem ersten LAB-Befehl und dem Befehl SKIP Start BACK) eine genau definierte Anzahl von Durchgängen mal aus und gibt dabei jedesmal den Text »Schleife Nr. x« aus. Die Anzahl von Durchgängen ergibt sich dabei aus dem Parameter *Num,* der eine Zahl sein sollte. Wird *Num* beim Aufruf nicht angegeben, wird die Schleife deshalb dreimal durchlaufen (wegen des DEF-Befehls in der zweiten Zeile).

Das »Geheimnis« dieser Kommandofolge liegt in der EVAL-Anweisung, die bei jedem Schleifendurchlauf den Wert der Umgebungsvariablen *Zahl* um 1 erhöht. Ist dieser Wert gleich *Num*, wird die Kommandofolge dann beendet. Solange das nicht der Fall ist, springt sie mit SKIP immer wieder zur Marke *Start* zurück. Es ist allerdings etwas schwierig zu verstehen, warum dieser EVAL-Befehl *Zahl* um *1* hochsetzt. Der »Trick« liegt in der geschickten Verwendung der Schlüsselworte, die EVAL kennt. Der EVAL-Befehl und der folgende COPY-Befehl führen im Prinzip die folgenden Anweisungen – formuliert in einer abstrakten Programmiersprache – aus:

```
Zahl2 := Zahl + 1
Zahl  := Zahl2
```

Mit den uns zur Verfügung stehenen Mitteln muß man das leider so formulieren:

```
EVAL <ENV:Zahl >NIL: TO ENV:Zahl2 OP + VALUE2 1 ?
COPY ENV:Zahl2 ENV:Zahl
```

Mittels *TO ENV:Zahl2* wird zunächst einmal das Ergebnis der Berechnung in die Datei *ENV:Zahl2* umgeleitet. Der zweite und der dritte Parameter (der Operator OP und die zweite Zahl VALUE2) werden mit Hilfe ihres Schlüsselworts explizit angegeben. Den ersten Parameter hingegen läßt man das Programm erfragen, indem man an das Ende des Befehls ein Fragezeichen schreibt. Die Frage selbst erscheint aber nicht am Bildschirm, da alle Ausgaben mit >NIL: zum Gerät NIL: umgeleitet werden. Gleichzeitig sorgt man mit *<ENV:Zahl* dafür, daß die Antwort auf die Frage (der erste Wert) nicht von der Tastatur erwartet wird, sondern daß sich das Programm diesen Wert selbständig aus der Datei *ENV:Zahl* holt.

Der COPY-Befehl nach dem EVAL ist notwendig, weil das Ergebnis von EVAL nicht direkt wieder nach *ENV:Zahl* geschickt werden kann. Diese Datei ist ja schon durch die Eingabeumleitung »belegt«. Deshalb wird das Ergebnis erst in eine andere Datei geschrieben und diese dann nach *ENV:Zahl* kopiert.

Wie Sie sehen, ist die Benutzung von Umgebungsvariablen nicht gerade einfach. Das hat hauptsächlich damit zu tun, daß im Moment nur der IF-Befehl Umgebungsvariablen »versteht«. Könnte man auch in EVAL-Befehlen Umgebungsvariablen wie Parameter verwenden, wäre Alles viel einfacher. Im Moment müssen Sie den EVAL-Befehl genauso verwenden, wie es hier beschrieben wurde. Die Zahl (VALUE2) und der Operator (OP) können aber ohne weiteres geändert werden. Mit allen anderen Änderungen sollten Sie sehr vorsichtig sein. Der EVAL-Befehl liefert sehr schnell die Fehlermeldung *Bad Args* (falsche Argumente).

### 8.4.12 Kommandofolgen automatisch erstellen lassen

Ähnlich lückenhaft wie die Realisierung der Umgebungsvariablen ist momentan die Bedeutung der sogenannten »Muster«, die es erlauben, mit Hilfe von »Jokerzeichen« eine ganze Reihe von Dateinamen zu beschreiben. Solche Muster können bei einigen Shell-Befehlen statt eines einzelnen Dateinamens oder eines Directory verwendet werden – bei anderen aber nicht, auch wenn es eventuell sinnvoll wäre.

Diese Problematik läßt sich allerdings mit Hilfe des LIST-Befehls und der beiden Kommandofolgen SPAT und DPAT weitgehend aus dem Weg räumen. Dazu wird in allen drei Fällen der Parameter LFORMAT des LIST-Befehls verwendet, mit dem es unter anderem möglich ist, Kommandofolgen automatisch erstellen zu lassen. Geben Sie zum Beispiel einmal den folgenden Befehl im Shell-Fenster ein, um sich mit dieser Möglichkeit etwas vertraut zu machen:

```
LIST C:c#? LFORMAT "RENAME %s%s AS %s%s.bak"
```

Dann wird auf dem Bildschirm eine Reihe von RENAME-Befehlen aufgelistet. Diese Befehle würden – wenn sie ausgeführt würden – alle Dateien, deren Namen in das Muster »C:c#?« passen, umbenennen, indem an den alten Namen ».bak« angehängt wird. LIST gibt nämlich, wenn LFORMAT angegeben wird, nicht die normalen Dateienliste aus. Statt dessen wird für jede ansonsten aufgelistete Datei eine Zeile entsprechend dem LFORMAT-Parameter ausgegeben. In diese Zeile kann mit Hilfe des Platzhalters »%s« der Dateiname und mit »%s%s« der vollständige Pfadname der Datei eingebaut werden. Mehr zu den exakten Möglichkeiten, die LFORMAT hinsichtlich der Gestaltung der Ausgabezeilen bietet, können Sie in Kapitel 7 erfahren.

Entscheidend ist, daß man die Ausgabe des LIST-Befehls natürlich in eine Textdatei umleiten und auf diesem Wege eine Kommandofolge erzeugen lassen kann. Diese braucht man dann nur noch aufrufen,und die erzeugten Befehle werden ausgeführt. Da der LIST-Befehl mit Namensmustern arbeiten kann, bietet dieser kleine Umweg die Möglichkeit, Muster zusammen mit jedem anderen Befehl zu verwenden.

Die Kommandofolgen SPAT und DPAT, die sich auf Ihrer Workbench-Diskette (im S-Directory) befinden, und die Sie wie Shell-Befehle verwenden können, machen diesen Prozeß noch einfacher. Sie erzeugen (mit Hilfe von LIST) die entsprechende Kommandofolge und führen Sie auch gleich aus. Am einfachsten ist die Verwendung von SPAT. Dieser »Befehl« kann ähnlich wie RUN vor jeden Shell-Befehl gesetzt werden – sofern dieser nur einen Parameter hat, der durch ein Muster ersetzt werden soll. Der folgende Befehl würde zum Beispiel alle Dateien im Directory *Batch*, deren Name mit einem X beginnt, zu Kommandofolgen machen, die ohne Verwendung von EXECUTE aufgerufen werden können:

```
SPAT PROTECT Batch/X#? +s
```

Beachten Sie bitte, daß der erste Parameter des SPAT-Befehls immer ein Shell-Befehl und der zweite ein Muster sein muß. Danach können vier weitere Schlüsselwörter und Parameter für den Shell-Befehl kommen. Die Anwendungsmöglichkeiten von SPAT sind aber beschränkt, da nur ein Parameter eines Befehls durch ein Muster ersetzt werden kann. Das oben beschriebene Bespiel mit dem RENAME-Befehl wäre also mit SPAT nicht zu realisieren.

Vielseitiger ist in dieser Hinsicht der DPAT-Befehl. Er »verträgt« auch Befehle mit zwei variablen Parametern. DPAT selbst benötigt mindestens drei Parameter. Der erste und der zweite sind wie bei SPAT ein Shell-Befehl und ein Muster. Der dritte, namens *Dir*, ist ein beliebiger Text, wobei es sich in vielen Fällen aber um den Namen eines Directorys handeln wird. Danach können noch weitere vier Parameter und/oder Schlüsselwörter kommen, die dem Shell-Befehl im ersten Parameter übergeben werden.

DPAT erzeugt eine Reihe von Befehlen, die mit dem Namen des Shell-Befehls beginnen. Der erste Parameter dieser Befehle ist immer der vollständige Pfadname einer der Dateien, die in das Muster passen. Der zweite Parameter der erzeugten Befehle besteht aus dem (Kurz-)Namen dieser Datei, dem allerdings der DIR-Parameter des DPAT-Befehls vorangestellt wird. Die zu diesem Zweck von DPAT verwendete LIST-Anweisung sieht im Prinzip wie folgt aus:

```
LIST >T:temp <Muster> LFORMAT="<Befehl> %s%s <Dir>%s <Rest>"
```

Mit DPAT und dem Shell-Befehl RENAME können Sie zum Beispiel eine ganze Reihe von Dateien von einem Directory in ein anderes bewegen. Der folgende Befehl »bewegt« zum Beispiel alle Dateien aus dem C-Directory in das Wurzel-Directory der Diskette im aktuellen Laufwerk (Sie sollten diesen Befehl deshalb nicht ausführen; er ist schwer wieder rückgängig zu machen):

```
DPAT RENAME C:#? DF0:
```

Auch mit DPAT wäre es übrigens nicht möglich, den zu Anfang dieses Abschnitts als Beispiel beschrieben RENAME-Befehl auszuführen. Dem zweiten Parameter der erzeugten Befehle wird ja immer ein weiterer Text voran- und nicht nachgestellt. Es sollte Ihnen aber, wenn Sie sich einmal in Ruhe die beiden Kommandofolgen SPAT und DPAT anschauen (sie befinden sich im S-Directory der Workbench-Diskette), nicht besonders schwerfallen,

selbst eine weitere Kommandofolge zu schreiben, die dem zweiten Parameter der von LIST erzeugten Befehle einen Text nachstellt oder andere raffinierte Manipulationen durchführt.

### 8.4.13 Eine paar besondere Kommandofolgen

Zum Abschluß dieses Abschnitts über das Thema Kommandofolgen möchte ich Ihnen einige der »wichtigsten Kommandofolgen überhaupt« vorstellen. Dabei handelt es sich um die Kommandofolgen im S-Directory der Workbench-Diskette, deren Namen alle »Start« enthalten. Einige dieser Kommandofolgen werden bei jedem Neustart (also zum Beispiel nach dem Einschalten) automatisch durchgeführt. Andere werden immer dann ausgeführt, wenn ein neues Shell- oder CLI-Fenster geöffnet wird (mit NEWSHELL beziehungsweise NEWCLI). Um wie ein Profi mit der Shell arbeiten zu können, müssen Sie den Inhalt dieser Kommandofolgen unbedingt kennen – und verstehen. Sie können sich Ihre Arbeitsumgebung dann maßschneidern beziehungsweise an Ihre Bedürfnisse anpassen, indem Sie eine oder mehrere der Start-Kommandofolgen ändern.

#### Startup-Sequence

Die wichtigste dieser Dateien ist *Startup-Sequence*. Diese Kommandofolge wird bei einem Neustart auf alle Fälle ausgeführt und startet dann von sich aus eventuell noch andere Kommandofolgen. Wenden Sie deshalb jetzt Ihr frisch erworbenes Wissen über Kommandofolgen an und betrachten Sie die Datei *S:Startup-Sequence* mit Ihrem Lieblingseditor. Sie können dazu zum Beispiel einen der zwei folgenden Befehle verwenden (der zweite funktioniert natürlich nur dann, wenn Sie MEMACS in die Utilities-Schublade der Workbench-Diskette gelegt haben oder mit PATH für die Shell auffindbar gemacht haben):

```
NOTEPAD s:startup-sequence
MEMACS  s:startup-sequence
```

Sie werden dann in der *Startup-Sequence* die folgenden Zeilen finden (zumindest in der Version der Workbench-Diskette, die mir zum Test vorlag. Ihre Version dieser Datei sieht vielleicht ein wenig anders aus oder enthält dieselben Befehle in anderer Reihenfolge.):

```
ADDBUFFERS DF0: 10
C:SETPATCH >NIL
CD C:
ECHO "A500/A2000 Workbench disk. Release 1.3 Version 34.20*N"
SYS:System/FastMemFirst
BINDDRIVERS
SETCLOCK LOAD
FF >:NIL -0
RESIDENT CLI L:Shell-Seg SYSTEM PURE ADD
RESIDENT C:EXECUTE PURE
MOUNT NEWCON:
;
FAILAT 11
RUN EXECUTE s:StartUpII
WAIT >NIL: 5 mins
;
```

```
SYS:System/SetMap d
PATH RAM: C: SYS:Utilities SYS:System SYS:Prefs add
LOADWB DELAY
ENDCLI >NIL:
```

Diese Kommandofolge werde ich jetzt Befehl für Befehl beschreiben. Dabei wird aber nicht jeder Befehl allzu ausführlich erläutert. (Ausführliche Befehlsbeschreibungen finden Sie in Kapitel 7.)

*Startup-Sequence* beginnt damit, den Zugriff auf das Diskettenlaufwerk etwas zu beschleunigen (ADDBUFFERS), korrigiert dann (mit SETPATCH) einige Befehle in Amiga-ROM und gibt schließlich die Willkommensmeldung auf dem Bildschirm aus. Danach wird mit FASTMEMFIRST die Speicherzuteilung so geändert, daß zunächst die RAM-Bereiche außerhalb der Chip-RAM genutzt werden. Danach folgt der Befehl BINDDRIVERS, mit dem die Gerätetreiber, die sich eventuell in der *Expansion*-Schublade befinden, aktiviert werden. Manche Festplatten müssen zum Beispiel auf diese Art aktiviert werden und können zuvor nicht vom Amiga-DOS benutzt werden.

Der Befehl SETCLOCK entnimmt dann die aktuelle Uhrzeit der Uhr in der Speichererweiterung A 501 und teilt sie Amiga-DOS mit. Der Befehl FF aktiviert die FastFonts-Routinen, (die die Textausgabe beschleunigen). Der erste RESIDENT-Befehl macht einen bestimmten Teil der Amiga-Software resident (hält ihn also permanent im Speicher), der für die Arbeit mit Shell-Fenster benötigt wird. Nur wenn dieser Teil, das Shell-Segment, resident ist, können Shell-Fenster geöffnet werden. (Das Fenster, das gleich beim Start geöffnet wird und in dem diese Kommandofolge abläuft, ist ein CLI-Fenster.)

Der zweite RESIDENT-Befehl sorgt dafür, daß kommende EXECUTE-Befehle schneller ausgeführt werden. Der darauffolgende MOUNT-Befehl meldet das Gerät NEWCON: beim Amiga-DOS an, das für den Aufbau von Shell-Fenstern benötigt wird. (Dieses Gerät steht direkt nach dem Start nicht zur Verfügung, sondern erst nach diesem MOUNT-Befehl.)

Die nächsten drei Befehle nach dem Semikolon führen eine zweite Start-Kommandofolge durch. Diese wird zwar mit RUN als separater Prozeß gestartet. Danach wird aber mit dem WAIT-Befehl eine Pause eingelegt, die 5 Minuten dauert, wenn der WAIT-Befehl nicht anderweitig abgebrochen wird. Die Kommandofolge *StartupII* wird gleich noch näher beschrieben. (Sie bricht übrigens den WAIT-Befehl ab, sobald Sie beendet ist. Sonst würde die *Startup-Sequence* ja immer mehr als fünf Minuten zur Ausführung benötigen.)

Am Ende der *Startup-Sequence* wird noch die deutsche Tastaturbelegung aktiviert (SETMAP d), der Suchpfad für Programme festgelegt (PATH), die Workbench aktiviert (LOADWB) und das CLI-Fenster, daß bis dahin den ganzen Bildschirm eingenommen hat, geschlossen (ENDCLI). Erst diese vier Befehle erschaffen die Arbeitsumgebung, die Sie nach dem Start gewöhnlich vorfinden. Erst nach dem SETMAP-Befehl reagiert der Amiga richtig auf Tastendrücke (bis dahin war er der Meinung, daß Sie eine amerikanische Tastatur verwenden). Erst nach dem LOADWB ist die Workbench aktiv (bis dahin gab es nur das CLI-Fenster, in dem die Kommandofolge abläuft). Und erst der ENDCLI-Befehl macht die Workbench sichtbar, indem er das CLI-Fenster schließt.

Die hier beschriebene *Startup-Sequence* ist im Prinzip nur für den Start eines Amiga mit einem oder zwei Diskettenlaufwerken gedacht. Wenn Sie eine Festplatte besitzen, ist es unter Umständen besser, wenn Sie statt dessen die Kommandofolge *Startup-Sequence.HD*

benutzen. Wie Sie dazu genau vorgehen müssen und wann es nötig oder sinnvoll ist, *Startup-Sequence.HD* zu benutzen, erfahren Sie in Kapitel 10.

### StartupII

Viele wichtige Aufgaben werden ab der Workbench-Version 1.3 nicht mehr in der *Startup-Sequence* erledigt – wie es in früheren Versionen der Fall war – sondern in einer zweiten Kommandofolge namens *StartupII*, die von *Startup-Sequence* aus gestartet wird. Auch diese Start-Kommandofolge soll deshalb nun im Detail vorgestellt werden:

```
RESIDENT C:Resident PURE
RESIDENT C:List PURE
RESIDENT C:cd PURE
RESIDENT C:Assign PURE
RESIDENT C:Mount PURE
RESIDENT C:MakeDir PURE
; Einige Kommentare
; ...
;
MAKEDIR RAM:t
ASSIGN T: RAM:t
MAKEDIR RAM:Env
ASSIGN ENV: RAM:Env
MAKEDIR RAM:Clipboards
ASSIGN CLIPS: RAM:Clipboards
;
MOUNT SPEAK:
MOUNT AUX:
MOUNT PIPE:
;
RESIDENT C:Assign REMOVE
RESIDENT C:Mount REMOVE
RESIDENT C:MakeDir REMOVE
;
BREAK 1 C
```

Die Kommandofolge *StartupII* zerfällt im Prinzip in vier große Abschnitte (die hier durch Kommentarzeilen getrennt werden.) Im ersten Abschnitt werden eine Reihe von Befehlen resident gemacht und ihre Ausführung dadurch stark beschleunigt. Genau diese Befehle werden dann in den folgenden Zeilen auch benötigt.

In der zweiten Gruppe werden auf der RAM-Disk einige Directories angelegt und diesen mit ASSIGN logische Laufwerk zugeordnet. Einige dieser Befehle werden in der vierten Gruppe dann wieder aus der Liste der residenten Befehle gestrichen.

Die MOUNT-Befehle in der dritten Gruppe melden eine Reihe zusätzlicher Geräte beim Amiga-DOS an. Die Geräte SPEAK:, AUX: und PIPE: würde es ohne diese Befehle nicht geben (oder genauer gesagt, Amiga-DOS würde diese Gerätenamen nicht kennen).

In der vierten Befehlsgruppe werden die Befehle ASSIGN, MOUNT und MAKEDIR wieder aus der Liste der residenten Befehle entfernt.

Der letzte Befehl schließlich bricht den gerade laufenden Befehl im Prozeß mit der Nummer 1. Das ist genau der WAIT-Befehl in der Kommandofolge *Startup-Sequence*, der auf die Beendigung von *StartupII* wartet. Erst jetzt wird mit der Ausführung von *Startup-Sequence* fortgefahren und schließlich mit deren letztem Befehl (ENDCLI) das CLI-Fenster geschlossen.

### CLI-Startup und Shell-Startup

Die beiden Kommandofolgen *CLI-Startup* und *Shell-Startup* haben für ein neues CLI- beziehungsweise Shell-Fenster eine ähnliche Funktion wie es *Startup-Sequence* für die Amiga-Software als Ganzes hat. Sie werden beim Öffnen eines neuen Fensters automatisch ausgeführt und dienen vor allem dazu, gewisse Vorbereitungen beziehungsweise »Initialisierungen« vorzunehmen. Diese beiden Initialisierungsdateien werden allerdings nur dann ausgeführt, wenn Sie nicht den FROM-Parameter des NEWCLI- oder NEWSHELL-Befehls angegeben haben!

Der Inhalt von *CLI-Startup* ist extrem simpel. Es handelt sich nur um einen PROMPT-Befehl, der festlegt, wie die Eingabeaufforderung in diesem Fenster aussieht:

```
PROMPT "%N> "
```

Die Kommandofolge *Shell-Startup* ist da schon deutlich gehaltvoller, wie die folgende Auflistung zeigt:

```
PROMPT "%N.%S> "
ALIAS xcopy    COPY [] CLONE
ALIAS endshell ENDCLI
ALIAS pro      EXECUTE S:SPAT PROTECT []
ALIAS sdate    EXECUTE S:SPAT SETDATE []
ALIAS ren      EXECUTE S:DPAT RENAME  []
ALIAS clear    ECHO "*E[0;0H*E[J"
ALIAS reverse  ECHO "*E[0;0H*E[41;30m*E[J"
ALIAS normal   ECHO "*E[0;0H*E[40;31m*E[J"
```

Wie Sie sehen, enthält diese Kommandofolge außer der Festlegung der Eingabeaufforderung am Anfang vor allem eine Reihe von ALIAS-Befehlen, die dazu dienen, Ihnen die Arbeit im Shell-Fenster noch leichter zu machen.

Der erste dieser Befehle definiert XCOPY, einen Befehl der dem COPY-Befehl zusammen mit dem Schlüsselwort CLONE entspricht. XCOPY erzeugt also Kopien die (bis auf den Namen) völlig identisch mit dem Original sind – einschließlich der Schutzflags und des Modifikationsdatums.

ENDSHELL ist einfach ein Alias-Name für ENDCLI.

PRO, SDATE und REN erweitern die Möglichkeiten der Befehle PROTECT, SETDATE und RENAME auf Gruppen von Dateien, die mit Hilfe von Mustern beschrieben werden. Dazu werden die Kommandofolgen SPAT und DPAT benutzt, die in Abschnitt 8.4.9 beschrieben werden.

Die Befehle CLEAR, REVERSE und NORMAL dienen dazu, die Darstellung von Texten im jeweiligen Shell-Fenster zu ändern. CLEAR löscht das ganze Fenster, REVERSE schaltet eine »invertierte« Darstellung ein, bei der dunkler Text auf hellem Hintergrund

gezeigt wird und NORMAL stellt den Normalzustand (helle Schrift auf dunklem Grund wieder her. Diese drei Befehle benutzen den ECHO-Befehl zusammen mit speziellen »Escape-Sequenzen«, um die gewünschte Wirkung zu erzielen. Mit solchen Escape-Sequenzen können Sie auch andere Eigenschaften des Fensters verändern. Eine vollständige Liste der zur Verfügung stehenden Möglichkeiten erhalten Sie im Anhang dieses Buches.

### 8.4.14 Ändern der Start-Kommandofolgen

Die Beschreibung der verschiedenen Start-Kommandofolgen im letzten Abschnitt war natürlich kein Selbstzweck. Es ist zwar interessant zu wissen, was beim Start des Amiga oder beim Öffnen eines Fensters passiert. Die meisten Shell-Anwender dürften aber sicherlich ohne dieses Wissen auskommen. Was die Start-Kommandofolgen so interessant macht, ist die Möglichkeit, sie abzuändern. Einige Beispiele für mögliche Änderungen möchte ich Ihnen jetzt vorstellen.

Bei allen Änderungen ist wichtig, sich vorher genau zu überlegen, wo diese Änderungen am besten durchgeführt werden, in *Startup-Sequence*, in *StartupII* oder in *Shell-Startup* (*CLI-Startup* ist relativ uninteressant). Für diese Entscheidung gibt es ein paar einfache Regeln. Die erste und wichtigste besagt, daß Sie extrem vorsichtig mit Änderungen an der *Startup-Sequence* sein sollten. Eine Änderung kann hier leicht zu einem Fehler und damit zum Abbruch der Kommandofolge führen. Im Falle von *Startup-Sequence* ist das besonders unangenehm, weil der gewohnte Urzustand des Amiga dann eventuell nicht hergestellt wird. Der Suchpfad könnte beispielsweise nicht komplett sein, eventuell fehlen bestimmte Gerätenamen oder die deutsche Tastaturbelegung wird nicht aktiviert. Fehler können auch dazu führen, daß keine Shell-Fenster geöffnet werden können (weil es NEWCON: nicht gibt oder *Shell-Seg* nicht resident ist). Erzeugen Sie sich vor Änderungen an *Startup-Sequence* deshalb vorher immer eine Sicherungskopie.

Auch Änderungen an *StartupII* sind mit Vorsicht zu genießen. Ein vorzeitiger Abbruch dieser Kommandofolge kann zum Beispiel dazu führen, das die logischen Laufwerke auf der RAM-Disk nicht vorhanden sind, die von vielen Programmen benötigt werden. Trotzdem sind Änderungen an *StartupII* solchen an *Startup-Sequence* vorzuziehen. Wenn Sie zum Beispiel ein paar zusätzliche logische Laufwerke definieren wollen, sollten Sie das in *StartupII* tun. Auch das Kopieren von Programmen auf die RAM-Disk (siehe unten) und ähnliche Vorbereitungsarbeiten nach dem Start sollten in *StartupII* veranlaßt werden und nicht in *Shell-Startup*, da diese Kommandofolge ja beim Öffnen jedes neuen Shell-Fensters immer wieder ausgeführt wird. Grundsätzlich gilt, daß Befehle mit »globaler« Wirkung, also Wirkungen auf alle Shell-Fenster, in *StartupII* gelegt werden sollten. Dazu gehören beispielsweise ASSIGN-und SETENV-Befehle.

Befehle, deren Wirkung nach dem Schließen des jeweiligen Fensters aufgehoben wird, haben in *StartupII* (und *Startup-Sequence*) hingegen nichts zu suchen. Sie werden ja durch den Befehl ENDCLI am Ende von *Startup-Sequence* »vergessen«. Solche Befehle gehören in *Shell-Startup*. Hier können Sie zum Beispiel weitere ALIAS-Definitionen vornehmen (die immer nur für ein Fenster gültig sind), die Eingabeaufforderung (Prompt) verändern oder mit dem ECHO-Befehl das Aussehen des Shell-Fensters ändern. Befehle wie COPY, MOUNT, ASSIGN, SETENV und so weiter, deren Wirkung bis zum Ausschalten oder bis zu einem Neustart des Amiga erhalten bleiben, haben hingegen in *Shell-Startup* nichts zu suchen.

Damit aber genug der Vorrede. Kommen wir zu einigen konkreten Änderungsvorschlägen. Eine naheliegende Änderung für überzeugte Benutzer der Shell wäre es sicherlich, aus *Startup-Sequence* den ENDCLI-Befehl zu entfernen. Dadurch würde das beim Start geöffnete CLI-Fenster offen bleiben und sie könnten gleich mit den Shell-Befehlen arbeiten. Diese Änderung ist aber nicht zu empfehlen, da es sich bei diesem Fenster um ein CLI- und nicht um ein Shell-Fenster handelt, und das Arbeiten in einem CLI-Fenster bei weitem nicht so komfortabel ist wie in einem Shell-Fenster. Besser ist es, in der *Startup-Sequence* oder in *StartupII* ein neues Shell-Fenster zu öffnen und dieses am besten auch gleich in der gewünschten Größe am Bildschirm zu plazieren.

Mein Vorschlag zu diesem Thema lautet folgendermaßen: Fertigen Sie eine Sicherungskopie Ihrer Workbench-Diskette an. Starten Sie dann Ihren Lieblingseditor und lesen Sie S:Startup-*Sequence ein. Suchen Sie dann die beiden folgenden Zeilen:*

```
RUN EXECUTE S:StartupII
WAIT >:NIL 5 mins
```

Löschen Sie die Zeile mit dem WAIT-Befehl und ersetzen Sie die Zeile mit dem RUN-Befehl durch die folgende Zeile:

```
NEWSHELL NEWCON:0/170/640/86/AmigaShell FROM S:StartupII
```

Speichern Sie nun *Startup-Sequence* wieder ab und öffnen Sie *S:StartupII*. Suchen Sie die Zeile mit dem BREAK-Befehl am Ende der Datei und löschen Sie sie. Fügen Sie statt dessen den folgenden Befehl als letzte Zeile in die Datei ein:

```
EXECUTE S:Shell-Startup
```

Speichern Sie jetzt auch *StartupII* wieder ab und probieren Sie die beiden neuen Kommandofolgen aus, indem Sie einen Neustart machen ([Ctrl]+[*A*]+[*A*]). Wenn Sie alle Änderungen korrekt durchgeführt haben, werden Sie sehen, daß der Startvorgang jetzt wesentlich schneller abläuft. Vor allem wird auch das bildschirmfüllende Fenster nach kurzer Zeit geschlossen und die Piktogramme der Workbench tauchen auf. Sie könnten jetzt sofort damit beginnen, mit der Workbench zu arbeiten (zum Beispiel indem Sie ein Diskettenpiktogramm öffnen). Noch bevor das große Fenster verschwindet, erscheint ein neues Shell-Fenster am Bildschirm. Es befindet sich praktischerweise am unteren Bildschirmrand, wo meist keine Piktogramme auftauchen. In diesem Fenster wird die Kommandofolge *StartupII* ausgeführt – während Sie schon mit der Workbench arbeiten können. Dieses Fenster bleibt auch offen, nachdem *StartupII* beendet ist. Sie haben damit sofort ein Shell-Fenster zur Verfügung. Damit in diesem Fenster auch die gewohnten Alias-Befehle funktionieren, wird am Ende von *StartupII* auch noch die Kommandofolge *Shell-Startup* ausgeführt.

Diese Änderungen dürften gerade für diejenigen Amiga-Besitzer, die viel mit der Shell arbeiten, außerordentlich angenehme Auswirkungen haben. Sie beschleunigen den Startvorgang (zumindest subjektiv) und liefern als angenehme Nebenwirkung ein offenes Shell-Fenster. Je nach Ihrer Arbeitskonfiguration können Sie aber natürlich auch noch weitere Änderungen vornehmen. Diese sollten Sie aber sicherheitshalber auf *StartupII* und *Shell-Startup* beschränken.

Wenn Sie Besitzer eines zweiten Diskettenlaufwerks sind und in dieses üblicherweise die Extras-Diskette einlegen, bietet es sich zum Beispiel ein, den Suchpfad durch einen PATH-Befehl in *StartupII* zu erweitern. Wenn Sie dort zum Beispiel den folgenden Befehl

eingeben, können Sie die Programme in der Schublade *Tools* auf der Extras-Diskette einfacher aufrufen:

```
PATH DF1:Tools
```

Wenn Sie sofort mit dem Shell-Fenster arbeiten wollen, kann es auch praktisch sein, mit dem CD-Befehl gleich ein bestimmtes Directory aktuell zu machen. Den entsprechenden Befehl können Sie in *StartupII* plazieren. Eventuell könnte man ihn aber auch in *Shell-Startup* unterbringen.

Und falls Sie über längere Zeit hinweg immer nur mit einem Werkzeug der Workbench arbeiten, zum Beispiel mit Amiga-Basic, so können Sie dieses Werkzeug auch gleich direkt von *StartupII* aus starten und brauchen nicht erst auf der Workbench mühselig Disketten- und Schubladen-Piktogramme zu öffnen. Solche Programme sollten Sie aber am besten immer mit Hilfe des RUN-Befehls (zum Beispiel RUN DF1:AmigaBASIC) aufrufen, damit *StartupII* nach dem Aufruf des Programms fortfahren kann.

Ähnliche Änderungen können Sie auch in der Kommandofolge Shell-Startup vornehmen. Hier bieten sich vor allem ein paar zusätzliche ALIAS-Befehle an. Die folgende Auswahl zeigt ein paar Beispiele (Ihnen fallen aber bestimmt bald auch eigene ein):

```
ALIAS DEL DELETE [] ALL
ALIAS EX EXECUTE []
ALIAS MAKEEX EXECUTE S:SPAT PROTECT [] +s
ALIAS D_AN EXECUTE S:SPAT PROTECT [] +d
ALIAS D_AUS EXECUTE S:SPAT PROTECT [] -d
ALIAS SW ECHO "*E[32m"
ALIAS RT ECHO "*E[33m"
ALIAS WS ECHO "*E[31m"
```

Der erste Befehl ist eine praktische Abkürzung für DELETE (der allerdings durch das Schlüsselwort ALL nicht ganz ungefährlich ist). Der zweite ist eine Abkürzung für EXECUTE. Der dritte Befehl MAKEEX kann aus einer Datei oder einer Gruppe von Dateien sehr schnell Kommandofolgen machen (indem das S-Flag gesetzt wird). Genauso kann D_AN Dateien vor dem versehentlichen Löschen schützen. D_AUS hebt diesen Schutz wieder auf. Die letzten drei ALIAS-Befehle bewirken, daß die Schrift im Shell-Fenster schwarz (SW), rot (ROT) oder weiß (WS) wird.

Dies alles waren aber nur ein paar Vorschläge. Im Laufe der Zeit werden Sie selbst bestimmt noch viele andere nützliche Anwendungen für *Startup-Sequence*, *StartupII* und *Shell-Startup* finden. Anregungen dazu werden Sie auch in den folgenden Abschnitten dieses Kapitels noch genug finden.

## 8.5 Ein ganz spezielles Gerät: die RAM-Disk

Im folgenden Abschnitt wird ein Thema behandelt, das nicht ausschließlich zur Shell gehört und auch für reine Workbench-Benutzer von Interesse sein könnte. Es konnte aber bislang, ohne gewisse Grundkenntnisse über die Shell, nicht sinnvoll erörtert werden. Bei diesem Thema handelt es sich um die sogenannte RAM-Disk, von der schon mehrfach die Rede war.

Die Verwendung einer solchen RAM-Disk kann Ihre Arbeit wesentlich beschleunigen und vor allem auch Ihr Leben einfacher machen, wenn Sie kein externes Disketten- oder Festplattenlaufwerk besitzen und mit dem internen Diskettenlaufwerk des Amiga 500 auskommen müssen.

Die Amiga-Software kennt zwei verschiedene Formen der RAM-Disk, die die Gerätenamen RAM: und RAD: tragen. Beide werden in diesem Abschnitt beschrieben. Eine Form der RAM-Disk RAM: sollte jeder Amiga-Benutzer zumindest schon einmal auf der Workbench gesehen haben – in Form des Diskettenpiktogramms *RAM DISK*, das (ab der Workbench-Version 1.2) nach dem Start in der oberen rechten Bildschirmecke auftaucht. Wenn man dieses Piktogramm öffnet, ist es allerdings leer.

### 8.5.1 Was ist eine RAM-Disk?

Hinter dem Piktogramm *RAM DISK* auf der Workbench und dem Gerät RAM: im Shell-Fenster verbirgt sich eine sogenannte »virtuelle Diskette«, die man meist »RAM-Disk« nennt. Eine RAM-Disk ist in Wirklichkeit etwas Software im Betriebssystem, die eine Diskette im RAM-Speicher des Amiga simuliert. Wenn Sie eine Datei auf diese Diskette kopieren, wird diese Datei im RAM-Speicher abgelegt. Und wenn Sie diese Datei auf dem Bildschirm ausgeben, kommen die Daten aus dem RAM und nicht von der Diskette.

Da eine solche RAM-Diskette keine beweglichen Teile besitzt, ist sie natürlich extrem schnell. Wenn Daten auf dieser »Diskette« geschrieben oder von ihr gelesen werden, werden sie ja nur im Speicher hin und her geschaufelt. Das ist der Hauptvorteil der RAM-Disk. Ihr Hauptnachteil ist ihre Flüchtigkeit. Wenn der Amiga 500 angeschaltet wird, geht der Inhalt der RAM-Disk verloren.

Ansonsten bietet die RAM-Disk aber fast nur Vorteile. Sie ist nicht nur viel schneller als es jede »echte« Diskette oder Festplatte sein kann. Sie haben mit der RAM-Disk auch ein zusätzliches Diskettenlaufwerk zur Verfügung. Wenn alle Disketten, die sich in den wirklichen Laufwerken befinden, schon voll sind, bietet die RAM-Disk oft willkommenen Ausweichplatz. Eine (vorsichtige) Anwendung der RAM-Disk kann Ihnen deshalb viel Zeit und Mühen ersparen.

### 8.5.2 Die RAM-Disk RAM:

Die RAM-Disk RAM: ist immer vorhanden. Sie kann nur unter Umständen unsichtbar sein, das heißt, es taucht kein entsprechendes Piktogramm auf der Workbench auf. Falls bei Ihnen die RAM-Disk nicht gleich nach dem Start auf der Workbench erscheint, müssen Sie ein Shell-Fenster öffnen, um das Piktogramm zu erzeugen. Von der Workbench aus können Sie die RAM-Disk nicht sichtbar machen.

Das Sichtbarmachen der RAM-Disk ist aber sehr einfach. Sie brauchen dazu nicht mehr zu tun, als sie zu benutzen. Geben Sie zum Beispiel den folgenden Befehl ein und die RAM-Disk erscheint als Piktogramm in der Workbench:

```
COPY :Disk.Info to RAM:
```

Die RAM-Disk wird nämlich auf der Workbench sichtbar, sobald sie einmal benutzt wurde. Der obengenannte COPY-Befehl macht genau das. Er legt eine Kopie der Datei *Disk.Info*,

die unter anderem die Informationen enthält, wie ein Disketten-Piktogramm aussieht, auf die RAM-Disk, die im Shell-Fenster unter dem Namen RAM: erscheint. Sie brauchen aber nicht unbedingt eine Datei auf die RAM-Disk zu kopieren. Auch der folgende Befehl, der scheinbar nichts tut (solange die RAM-Disk) leer ist, bringt das RAM-Disk-Piktogramm auf der Workbench zum Vorschein:

```
DIR RAM:
```

Wenn Sie Dateien nach RAM: kopieren, werden Sie bald eine interessante Eigenschaft dieser »Diskette« kennenlernen: RAM: ist eine RAM-Disk die je nach Bedarf wächst und schrumpft; sie ist immer randvoll! (Bei der RAM-Disk RAD: ist das, wie Sie gleich sehen werden, anders.) Das können Sie feststellen, wenn Sie das Diskettenfenster der RAM-Disk auf der Workbench betrachten (die Füllanzeige ist immer am oberen Anschlag). Lassen Sie sich davon aber nicht irritieren. Solange die Anzeige in der Titelleiste der Workbench noch freien RAM-Speicher meldet, ist auch auf der RAM-Disk noch Platz. Ob Sie diesen Platz allerdings völlig ausnutzen sollten, ist sehr fraglich. Schließlich benötigen Sie den RAM-Speicher auch noch für andere Zwecke.

### 8.5.3 Die RAM-Disk RAD:

Die RAM-Disk RAD:, die es ab der Workbench-Version 1.3 gibt, ist etwas weniger labil als das Gerät RAM:. Der Inhalt des »Geräts« RAD: bleibt nämlich auch bei einem Neustart erhalten und geht erst beim Ausschalten des Amiga verloren. Programmfehler, die zu einem »Absturz« des Amiga führen, können RAD: deshalb nichts anhaben. RAD: wird aus diesem Grund auch »resetfeste« RAM-Disk genannt, da ein Neustart (zum Beispiel mit [Ctrl]+[*A*]+[*A*]) einem Reset der 68000-Prozessors entspricht.

Ein weiterer auffälliger Unterschied zwischen RAM: und RAD: ist, daß RAM: sofort nach dem Neustart zur Verfügung steht, während das Gerät RAD: zunächst einmal unbekannt ist. Das liegt daran, daß RAD:, wie auch SPEAK:, AUX:, NEWCON: und PIPE:, ein Gerät ist, das erst mit einem MOUNT-Befehl bei Amiga-DOS angemeldet werden muß, bevor man es benutzen kann. Die folgenden beiden Befehle erledigen diese Anmeldungsformalitäten:

```
MOUNT RAD:
DIR RAD:
```

Bevor Sie diesen Befehl ausprobieren, muß ich Ihnen unbedingt noch eine Warnung mit auf den Weg geben: RAD: ist eigentlich nur etwas für Amiga-Modelle mit 1 Mbyte RAM-Speicher oder mehr. Auf einem Amiga 500 mit 512 Kbyte RAM-Speicher sollte man RAD: nicht verwenden. Der Grund dafür ist, daß RAD: anders als RAM: eine feste Größe hat und sofort nach der Anmeldung etwa 240 Kbyte Speicher »schluckt«. Versierte Shell-Benutzer können diese Größe zwar ändern. Das ändert aber nichts daran, daß man bei der Arbeit mit RAD: auf einem Amiga mit 512 Kbyte RAM meist mit einigen Problemen zu kämpfen hat.

### 8.5.4 Die RAM-Disk RAD: benutzen

Wenn Ihr Amiga jedoch 1 Mbyte oder mehr Speicher besitzt, können – und sollten – Sie RAD: als bevorzugte RAM-Disk verwenden. Führen Sie dann am besten gleich die beiden eben genannten Befehle aus, um sich mit RAD: ein wenig vertraut zu machen. Betrachten

Sie sich RAD: auch einmal auf der Workbench. Das entsprechende Piktogramm sollte unter dem Namen RAMB0 in der rechten oberen Ecke des Bildschirms auftauchen.

Kopieren Sie jetzt den Inhalt der Schublade *Prefs* auf die neue RAM-Disk. Sie können das mit der Workbench machen oder auch den folgenden Shell-Befehl dazu verwenden:

```
COPY :Prefs RAD: ALL
```

Führen Sie nun einen Neustart durch und warten Sie, bis die Workbench wieder erscheint. Wenn Ihr Amiga mit den neuen Kickstart-ROMs der Version 1.3 (oder höher) versehen ist, erscheint jetzt sofort das Piktogramm RAMB0 am Bildschirm. Sie müssen RAD: also nicht erst mit dem MOUNT-Befehl anmelden. Bei einem älteren Amiga ohne die neuen Kickstart-ROMs müssen erst wieder die beiden folgenden Shell-Befehle eingegeben werden:

```
MOUNT RAD:
DIR RAD:
```

Spätestens dann ist auf alle Fälle das Piktogramm der RAM-Disk auf der Workbench zu sehen. Wenn Sie das Piktogramm RAMB0 öffnen, tauchen sofort die Piktogramme der Programme aus der *Prefs*-Schublade auf, die Sie vor dem Neustart dort hineingelegt haben – ein überzeugender Beweis dafür, daß RAD: seinen Inhalt auch bei einem Neustart behält.

Trotzdem ist auch RAD: natürlich kein besonders sicherer Ort für wichtige Daten! Die meisten Programmfehler, die einen Neustart des Amiga erzwingen, beeinträchtigen den Inhalt von RAD: zwar nicht. Gegen Stromausfälle oder ein versehentliches Ausschalten des Amiga ist sie allerdings nicht gefeit. Und auch fehlerhafte Programme können RAD: unter Umständen löschen – was allerdings sehr unwahrscheinlich ist. Das ein Programm RAD: löschen kann, zeigt der Shell-Befehl REMRAD, der genau zu diesem Zweck auf der Workbench-Diskette vorhanden ist. REMRAD löscht den gesamten Inhalt von RAD: und gibt den von RAD: beanspruchten RAM-Speicher fast vollständig wieder frei.

Wenn Sie über reichlich Speicher verfügen und RAD: in Zukunft ausschließlich verwenden wollen, können Sie mit ein paar einfachen Änderungen an der Start-Kommandofolge *StartupII* erreichen, daß die alte RAM-Disk RAM: überhaupt nicht mehr verwendet wird. Lesen Sie *StartupII* dazu einfach mit Ihrem Lieblingseditor ein und ersetzen sie jedes Vorkommen von RAM: gegen RAD:. Fügen Sie zum Abschluß noch vor dem Befehl *MAKEDIR RAM:t* den folgenden zusätzlichen Befehl ein:

```
MOUNT RAD:
```

Dieser Befehl wird bei einem Amiga mit neuen Kickstart-ROMs zwar nur einmal nach dem Einschalten benötigt. Es schadet aber auch nichts, ihn bei jedem Neustart noch einmal auszuführen.

Beachten Sie bitte, daß Sie – nachdem Sie diese Änderungen einmal ausgeführt haben – darauf angewiesen sind, RAD: permanent eingeschaltet zu lassen, wodurch auch permanent etwa 240 Kbyte Ihres RAM-Speichers verloren gehen. Wenn Sie nicht völlig sicher sind, daß Sie auf diese 240 Kbyte auf Dauer verzichten können, sollten Sie *StartupII* am besten unverändert lassen und RAD: nur bei Bedarf mit dem MOUNT-Befehl aktivieren. Sie können RAD: dann jederzeit mit dem Shell-Befehl REMRAD wieder entfernen. Die Verwendung von REMRAD dürfte hingegen nach kurzer Zeit zu Fehlermeldungen führen, wenn Sie die logischen Laufwerke T:, ENV: und CLIPS: durch Änderungen an *StartupII* auf RAD: gelegt haben.

### 8.5.5 Die Größe von RAD: ändern

Wenn Sie über mehr als 1 Mbyte RAM-Speicher verfügen oder RAD: sehr intensiv nutzen, kann es sein, daß Ihnen die voreingestellte Größe diesre »Diskette« von etwa 240 Kbyte nicht ausreicht. (RAD: kann ja nicht wie RAM: je nach Bedarf wachsen.) In diesem Fall können Sie die Größe von RAD: auch ändern, indem Sie die Datei *Mountlist* im logischen Laufwerk DEVS: ändern. Dabei sollten Sie aber mit äußerster Vorsicht umgehen und am besten vorher eine Sicherungskopie der Datei anlegen.

Lesen Sie dann die Datei DEVS:Mountlist mit dem von Ihnen bevorzugten Editor ein und suchen Sie in dieser Datei nach einer Zeile, die mit »RAD:« beginnt. Diese und die folgenden Zeilen »beschreiben« das Gerät RAD: für den Befehl MOUNT.

```
RAD: Device    = ramdrive.device
     Unit      = 0
     Flags          = 0
     Surfaces  = 0
     BlockPerTrack= 11
     Reserved  = 2
     Interleave     = 0
     LowCyl = 0; HighCyl = 21
     Buffers   = 5
     BufMemType     = 1
#
```

Anhand dieser Daten kann MOUNT das entsprechende Gerät beim Amiga-DOS anmelden und unter anderem auch feststellen, welche Speicherkapazität RAD: hat. Entscheidend hierfür ist die folgende Zeile aus Mountlist:

```
LowCyl = 0; HighCyl = 21
```

Die Zeile gibt an, aus wievielen »Zylindern« oder Spuren eine Diskette dieser Art besteht. *LowCyl* ist die Nummer des ersten, *HighCyl* die Nummer des letzten Zylinders. Ein Zylinder entspricht dabei einer Speicherkapazität von 11 Kbyte. Die voreingestellten Werte von *LowCyl* und *HighCyl* beschreiben eine Diskette mit 22 Zylindern (Zylinder 0 bis Zylinder 21), die dadurch eine Speicherkapazität von 242 Kbyte hat. Sie können diese Kapazität vergrößern, indem Sie die Zahl hinter *HighCyl* ändern. Wenn Sie hier zum Beispiel 45 eintragen, ergibt dies eine Kapazität von 506 Kbyte (46*11).

Wenn Sie die geänderte *Mountlist* abgespeichert haben, müssen Sie den Amiga ausschalten, einen Moment warten und dann wieder einschalten. Wenn Sie dann (nach der Beendigung des Startvorgangs) wieder den Befehl MOUNT RAD: eingeben, wird die RAM-Disk in der gewünschten Größe erzeugt. (Die Änderung der *Mountlist* wirkt sich also nicht auf eine schon bestehende RAM-Disk aus).

### 8.5.6 Eine RAM-Disk richtig benutzen

Beide RAM-Disks des Amiga können Sie in einem Shell-Fenster verwenden wie jedes andere Laufwerk auch. Verwenden Sie dazu zum Beispiel in Kopierbefehlen nur den Laufwerksnamen RAM: oder RAD: statt DF0: oder DF1:. Auch auf der Workbench können

Sie ein RAM-Disk-Piktogramm verwenden wie jedes andere Disketten-Piktogramm. So ist es möglich, Projekte und Werkzeuge daraufzulegen, Schubladen daraufzulegen, Programme zu starten und so weiter.

Berücksichtigen Sie aber, daß die RAM-Disk im RAM-Speicher Ihres Amiga 500 existiert. Jedes Piktogramm, das Sie auf die RAM-Disk legen, und jede Datei, die Sie nach RAM: kopieren, kostet Sie RAM-Speicher. Das können Sie auch direkt verfolgen, wenn Sie die Anzeige des freien Speicherplatzes in der Titelleiste der Workbench während eines Kopiervorgangs betrachten. Die RAM-Disk RAD:, die auf der Workbench unter dem Namen RAMB0 auftaucht, benötigt sogar von vorneherein 240 Kbyte RAM (solange Sie ihre Größe nicht ändern.) Verwenden Sie RAM: und RAD: deshalb mit Vorsicht und Planung! Besonders auf dem Amiga 500 ohne Speichererweiterung kann es rasch passieren, daß Sie nicht mehr genügend Speicher für Programme freihaben, wenn Sie viele Daten auf die RAM-Disk legen.

Wenn Sie die RAM-Disk benutzen wollen, sollten Sie deshalb einige Punkte beachten, um nicht böse Überraschungen zu erleben. Der wichtigste dieser Punkte ist die Tatsache, daß RAM-Disk und Programme (Werkzeuge) um den RAM-Speicher des Amiga konkurrieren. Wenn Sie also vorhaben, aufwendige große Programme zu benutzen, sollten Sie möglichst wenige Daten auf die RAM-Disk RAM: legen und RAD: am besten gar nicht benutzen. Einige Programme (zum Beispiel *Deluxe Paint*) bieten einen erhöhten Leistungsumfang, wenn mehr RAM-Speicher zur Verfügung steht. Wenn Sie solche Programme benutzen, sollten Sie die RAM: vorher leeren und/oder RAD: mit dem Befehl REMRAD entfernen.

Besonders extrem ist dieses Problem naturgemäß auf dem Amiga 500 ohne Speichererweiterung. Hier sollten Sie nie oder nur vorübergehend mehr als etwa 200 Kbyte (400 Blocks) Daten auf die RAM-Disk RAM: legen, wenn Sie »echte« Anwendungsprogramme benutzen wollen. RAD: sollten Sie besser überhaupt nicht verwenden.

Großzügiger können Sie da schon arbeiten, wenn Sie die Speichererweiterung eingebaut haben. Ihnen stehen dann insgesamt 1024 Kbyte zur Verfügung. Sie können dann den Inhalt einer halben Diskette auf die RAM-Disk RAM: legen oder die Größe von RAD: auf 500 Kbyte hochsetzen und haben immer noch mehr RAM-Speicher zur Verfügung als mit einem Amiga 500 ohne Speichererweiterung. Die meisten Programme sollten dann noch problemlos arbeiten.

Das zweite große Problem der RAM-Disk RAM: ist Ihre Flüchtigkeit (bei RAD: ist das nicht ganz so problematisch). Es ist sehr gefährlich, Daten auf einer RAM-Disk abzulegen, von denen man nicht eine Kopie auf Diskette hat. Sie sollten deshalb nur Programme sowie solche Dateien auf die RAM-Disk legen, von denen eine Kopie existiert. Wenn Sie zum Beispiel einen Brief eingetippt haben und nun abspeichern wollen, legen Sie ihn niemals auf die RAM: und am besten auch nicht auf RAD:, wenn Sie wirklich auf Nummer sicher gehen wollen. Wichtige Daten oder Programme, von denen es keine Kopie auf einer normalen Diskette gibt, auf RAM: abzulegen, ist hingegen so gut wie versuchter »elektronischer Selbstmord«.

RAM-Disks haben aber natürlich nicht nur Nachteile – sonst würde es sie kaum geben. Eine RAM-Disk bietet zwei große Vorteile: eine hohe Geschwindigkeit und ein zusätzliches Diskettenlaufwerk. Entsprechend sollten Sie sie auch nutzen. Die hohe Geschwindigkeit können Sie zum Beispiel dann nutzen, wenn ein Programm viele Diskettenoperationen macht. Wenn Sie zum Beispiel ein Programm benutzen, das Bilder von der Diskette einliest

und sie in möglichst rascher Folge auf dem Bildschirm zeigen soll, kann es sinnvoll sein, eine Bildfolge auf die RAM-Disk zu kopieren. Von dort kann das Programm die Bilder sehr schnell einlesen und deshalb auch in sehr schneller Folge zeigen. Auch andere Dateien, die Sie nur vorübergehend benötigen, sollten Sie auf die RAM-Disk legen – am besten in das Directory T, das dem logischen Laufwerk T: entspricht. Solche »temporären« Dateien werden dann beim nächsten Neustart automatisch gelöscht.

Die zweite wichtige Art, eine RAM-Disk zu benutzen, ist es, sie als zusätzliches Diskettenlaufwerk zu verwenden. Besonders Besitzer eines Amiga 500 ohne externes Laufwerk werden über dieses »virtuelle« Diskettenlaufwerk bald sehr dankbar sein. Wollen Sie nämlich mit nur einem Diskettenlaufwerk mehrere Dateien von einer auf eine andere Diskette kopieren, so bedeutet das sehr viele lästige Diskettenwechsel – egal ob Sie im Shell-Fenster oder auf der Workbench arbeiten. Kopieren Sie jedoch alle Dateien zunächst von der einen Diskette auf die RAM-Disk, wechseln dann die Disketten aus und kopieren dann die Dateien von der RAM-Disk auf die andere Diskette, kommen Sie mit ein beziehungsweise zwei Diskettenwechseln aus – zumindest dann, wenn die zu kopierenden Daten alle auf die RAM-Disk passen. Nach Abschluß des Kopiervorgangs sollten Sie – zumindest auf einem Amiga mit weniger als 1 Mbyte Speicher – die RAM-Disk aber wieder leeren oder mit REMRAD entfernen, um den dadurch belegten Speicher zurückzugewinnen.

Aber nicht nur beim Kopieren kann man ein zusätzliches Laufwerk gut gebrauchen. Wenn Sie bei der Benutzung eines Programms ständig zwei Disketten benötigen, so können Sie den Inhalt einer dieser beiden Disketten (am besten die mit den wenigsten Daten darauf) auf die RAM-Disk kopieren. Sie bekommen so entweder ein Laufwerk frei oder sparen sich lästige Diskettenwechsel, falls Sie nur ein Laufwerk besitzen. Sie können auf diese Weise sogar die Workbench-Diskette selbst einsparen. Mehr zu diesem Thema in den folgenden Abschnitten.

### 8.5.7 Programme und Dateien in der RAM-Disk

Jedes Programm, das in die RAM-Disk kopiert wurde, kann ungleich schneller gestartet werden. Auch die Befehle der Shell sind ja Programme. Jedesmal, wenn Sie ein solches Kommando aufrufen, muß dieses Programm erst von der Diskette geladen werden. Deshalb dauert es oft einen kleinen Moment, bis die Shell auf einen Befehl reagiert. Eine mögliche Nutzung der RAM-Disk wäre es deshalb, die häufigsten benutzten Shell-Befehle auf die RAM-Disk zu kopieren. Die nachstehende Folge von Shell-Befehlen (aus der man natürlich eine Kommandofolge machen könnte) leistet genau das:

```
MAKEDIR RAM:c
PATH RAM:c
COPY C:COPY     RAM:c; copy ist nun schneller
COPY C:DIR      RAM:c
COPY C:CD       RAM:c
COPY C:DELETE   RAM:c
COPY C:LIST     RAM:c
COPY C:ASSIGN   RAM:c
COPY C:EXECUTE RAM:c
COPY C:RUN      RAM:c
.....           ; und so weiter für alle wichtigen Shell-Befehle
```

Wichtig ist bei dieser Folge von Befehlen der PATH-Befehl in der zweiten Zeile, der der Shell sagt, wo die Befehle zu finden sind. Würde er wegfallen, lägen die ausgesuchten Kommandos zwar auf der RAM-Disk, die Shell würde aber nach wie vor die Kommandos auf der Diskette benutzen. Mit dem PATH-Befehl sieht die Lage anders aus: Wenn Sie nun einen Shell-Befehl eingeben, sucht die Shell zunächst im aktuellen Directory nach dem entsprechenden Programm dann in den anderen Directories des Suchpfades und schließlich auf der RAM-Disk.

Wenn Sie häufiger eine Gruppe von Shell-Befehlen in der RAM-Disk unterbringen wollen, sollten Sie sich die oben vorgestellte Befehlsfolge in eine Kommandofolge im Directory S legen. Nennen Sie die Kommandofolge zum Beispiel CRAM und setzen Sie mit dem PROTECT-Befehl ihr S-Flag. Sie können diese Befehlsfolge dann mit einem einzigen Shell-Befehl ausführen. Diese Kommandofolge würde die ausgewählten Shell-Befehle allerdings auf die »flüchtige« RAM-Disk RAM: legen. Wenn Sie statt dessen lieber RAD: verwenden wollen, brauchen Sie nur alle Vorkommen von RAM: gegen RAD: ersetzen.

### 8.5.8 Kombination von RAM-Disk und Startup-Sequence

Falls Sie eine bestimmte Gruppe von Shell-Befehlen immer in der RAM-Disk haben wollen, können Sie sich die oben gezeigte Befehlsfolge natürlich auch an das Ende der Kommandofolge *StartupII* legen. Der Startvorgang dauert dann zwar etwas länger. Dafür geht aber danach alles viel schneller.

Speziell dann, wenn Sie gleich beim Start eine RAM-Disk füllen, empfiehlt es sich – sofern Sie mindestens 1 Mbyte Speicher besitzen – RAD: statt RAM: zu verwenden. Das zeitraubende Kopieren braucht dann nämlich nur einmal nach dem Einschalten durchgeführt werden. Damit die Kommandofolge CRAM aber auch merkt, daß Ihre Arbeit getan ist, muß Sie etwas modifiziert werden. Die folgende Auflistung zeigt die neue Kommandofolge, die speziell für die Verwendung zusammen mit RAD: optimiert ist:

```
MOUNT >NIL: RAD:      ; sicherstellen, daß RAD: vorhanden ist
IF NOT EXISTS RAD:C   ; RAD ist leer!
  MAKEDIR RAD:C
  PATH RAD:C
  COPY C:COPY     RAD:C
  COPY C:DIR      RAD:C
  COPY C:CD       RAD:C
  COPY C:DELETE   RAD:C
  COPY C:LIST     RAD:C
  COPY C:ASSIGN   RAD:C
  COPY C:EXECUTE RAD:C
  COPY C:RUN      RAD:C
  ......          ; und so weiter für alle wichtigen Shell-Befehle
ENDIF
```

Wenn Sie diese Kommandofolge unter dem Namen CRAM im Directory S speichern, und an das Ende der Kommandofolge *StartupII* den Befehl EXECUTE S:CRAM setzen, läuft die Initialisierung der RAM-Disk beim Neustart vollautomatisch ab. CRAM prüft zunächst, ob das Directory C auf RAD: schon vorhanden ist. Nur dann, wenn das nicht der Fall ist, wird RAD:C angelegt und dann mit einer Auswahl von Shell-Befehlen gefüllt.

CRAM kann von jedem Amiga-Benutzer, der viel mit der Shell arbeitet, sinnvoll eingesetzt werden. Für Programmierer ist die Kombination von RAM-Disk und *Startup-Sequence* beziehungsweise *StartupII* jedoch noch in einer weiteren Hinsicht interessant. Viele Compiler-Sprachen benötigen bei der Übersetzung eines Programms eine ganze Reihe von Hilfs-Dateien (bei der Programmiersprache C zum Beispiel die *h*-Dateien und die *lib*-Bibliotheken). Wenn man diese Dateien schon beim Start in die RAM-Disk legt, so führt das zu einem wesentlich beschleunigten Übersetzungsvorgang – und den weiß jeder Programmierer zu schätzen. (Auch die temporären Dateien, die von einem Compiler meist während der Übersetzung angelegt werden, sollte man, sofern das möglich ist, auf die RAM-Disk schicken. Hierzu müssen Sie meist nur mit dem ASSIGN-Befehl ein bestimmtes logisches Laufwerk auf die RAM-Disk umlegen.)

Das Kopieren von Dateien in die RAM-Disk sollten Sie aber – gerade beim Start – auch nicht übertreiben. Das dabei auftretende Problem ist naheliegend: Die RAM-Disk verschlingt kostbaren Speicherplatz, den man vielleicht anderweitig benötigen würde. So werden viele Programme zwar schneller, wenn sie Dateien auf der RAM-Disk lesen und schreiben. Dafür kann es aber sein, daß nicht mehr so viele Programme gleichzeitig in den Speicher passen. Im Extremfall kann man, wie auf den meisten anderen Computern, nur noch mit einem Programm gleichzeitig arbeiten. Will man nun von einem Programm zum anderen wechseln, so muß man erst das eine beenden und das andere starten. Diese Vorgänge können sehr langwierig sein und einen großen Teil der durch die RAM-Disk gewonnenen Zeit wieder verschlingen.

## 8.6 Tips für Shell-Anwender

In diesem Abschnitt werden Sie einige Informationen und Tips finden, die in keinen der anderen Abschnitte über die Shell so recht hineinpaßten. Sie werden sehen, daß im Shell-Fenster einige Dinge möglich sind, zu denen Sie auf der Workbench einfach nicht fähig sind. Zum großen Teil handelt es sich dabei um Kommandofolgen und besonders um die Modifikation der Kommandofolgen *Startup-Sequence* und *StartupII*.

Viele der hier aufgeführten Tips sind nur für fortgeschrittene Anwender gedacht und deshalb mit Vorsicht zu genießen. Vom Shell-Fenster aus ist es wirklich recht einfach möglich, Disketten unbrauchbar zu machen, während man sich auf der Workbench dafür schon einige Mühe geben muß.

### 8.6.1 Wie Sie auf der Workbench-Diskette Platz schaffen

Die Shell wird nicht nur durch Kommandofolgen (wie die *Startup-Sequence*) allein interessant. Ein weiterer Vorteil der Shell gegenüber der Workbench ist es auch, daß Sie dort Dateien »sehen«, die auf der Workbench verborgen sind. Auf der Workbench-Diskette sind viele dieser Dateien mehr oder weniger überflüssig oder werden nur sehr selten benötigt. Indem Sie sie löschen, können Sie wertvollen Platz auf der Diskette freimachen, den Sie für andere Zwecke einsetzen können. Das ist natürlich besonders für diejenigen Besitzer des Amiga 500 interessant, die kein zweites Diskettenlaufwerk und keine Festplatte besitzen. Auf der Workbench-Diskette wird dadurch nämlich Platz frei.

## Das Directory der Workbenchdiskette

Bevor Sie aber an das Löschen »unsichtbarer« Dateien gehen, benötigen Sie einen Überblick darüber, wo welche Dateien auf der Workbench-Diskette zu finden sind und wofür sie benötigt werden. Glücklicherweise ist der Inhalt der Workbench-Diskette recht übersichtlich in Directories aufgeteilt, die die verschiedenen Dateien logisch gruppieren. Die folgende Übersicht stellt den Aufbau des Dateienbaums auf der Workbench-Diskette dar. In ihr sind nur Directories und keine Dateien aufgeführt.

```
Trashcan
c
Prefs
System
l
devs
     clipboards
     printers
     keymaps
s
t
Fonts
     diamond
     emerald
     garnet
     opal
     ruby
     sapphire
     topaz
libs
Empty
Utilities
Expansion
```

Das Directory *Trashcan* kennen Sie schon von der Workbench her. Es entspricht dem Mülleimer-Piktogramm. Dieses Directory enthält nur dann Dateien, wenn Sie auf der Workbench Piktogramme in den Mülleimer gelegt haben. Diese Dateien werden alle gelöscht, wenn Sie den Mülleimer anklicken und dann den Befehl *Empty Trash* auswählen. Für die Arbeit mit der Shell ist *Trashcan* unwichtig.

Das Directory *c* enthält – wie Sie ja schon erfahren haben – Programme, die Sie als Shell-Befehle verwenden können. Viele dieser Befehle werden Sie nie benötigen. Speziell dann, wenn Sie üblicherweise auf der Workbench arbeiten, benötigen Sie nur die wenigen Programme aus dem Directory *c*, die in der *Startup-Sequence* und in *StartupII* vorkommen.

Das Directory *Prefs* enthält das Programm *Preferences* und ein paar weitere Piktogramme, mit denen man direkt zu einem bestimmten Fenster von *Preferences* gelangen kann.

Das Directory *System* kennen Sie bereits von der Workbench her. Es entspricht der Schublade *System* und enthält eine Reihe von Hilfsprogrammen, die sowohl von der Workbench aus wie auch mit einem Shell-Befehl gestartet werden können. Beim Start wird System mit einem PATH-Befehl in den Suchpfad gelegt.

Das Directory */* ist eines der für Amiga-DOS besonders wichtigen Directories. Es enthält die folgenden Dateien:

```
Aux-Handler
Disk-Validator
FastFileSystem
NewCon-Handler
Pipe-Handler
Port-Handler
Ram-Handler
Shell-Seg
Speak-Handler
```

Einige dieser Dateien können sie löschen, wenn Sie auf bestimmte Möglichkeiten verzichten können, andere sollten auf keinen Fall gelöscht werden.

Zu den wichtigen Dateien, die auf keinen Fall gelöscht werden sollten, gehören *Disk-Validator* und *Port-Handler*, die für ein einwandfreies Funktionieren Ihres Amiga-Systems unbedingt benötigt werden. *Disk-Validator* wird benötigt, um nach dem Einlegen einer Diskette feststellen zu können, ob sie »in Ordnung« ist. Port-Handler wird für die Steuerung der Schnittstellen benötigt.

Die Datei *Ram-Handler* ist für die Steuerung der RAM-Disk RAM: zuständig. Ohne diese Datei kennt Amiga-DOS das Laufwerk RAM: nicht. In gleicher Weise sind Aux-Handler, Pipe-Handler, Speak-Handler und NewCon-Handler für die Geräte AUX:, PIPE:, SPEAK: und NEWCON: zuständig. Auf die ersten drei dieser Dateien kann man im allgemeinen ganz gut verzichten. Ohne NewCon-Handler wird der Arbeitskomfort in Shell-Fenstern aber stark eingeschränkt. Der Befehl NEWSHELL verwendet dann nämlich das primitivere Gerät CON: beim Öffnen eines neuen Shell-Fensters. Wenn nicht nur *NewCon-Handler,* sondern auch die Datei *Shell-Seg* fehlt, können überhaupt keine Shell-Fenster mehr geöffnet werden. Sie können dann lediglich mit der Workbench oder dem CLI arbeiten.

Die Datei *FastFileSystem* wird nur dann benötigt, wenn Sie eine Festplatte besitzen, die (beziehungsweise deren Partionen) mit dem *FastFileSystem* formatiert wurde(n). Mehr zu diesem Thema können Sie in Kapitel 10 erfahren.

Auch das Directory *devs* ist ein sehr wichtiges Directory, daß auf jeder Workbench-Diskette vorhanden sein muß. Es enthält Dateien, die Amiga-DOS zur Ansteuerung der verschiedenen Geräte benötigt, aus denen Ihr Amiga-System besteht oder die Sie an den Amiga anschließen. Neben einer Reihe weiterer Directories enthält es die folgenden Dateien:

```
clipboard.device
Mountlist
narrator.device
parallel.device
printer.device
ramdrive.device
serial.device
system-configuration
```

Die Datei *clipboard.device* wird von einigen Programmen benötigt, um eine »Zwischenablage« zu realisieren. Lesen Sie die Beschreibung von *Notepad*, wenn Sie mehr über das

Prinzip einer Zwischenablage erfahren wollen. Die Dateien *parallel.device* und *serial.device* werden benötigt, um die parallele und serielle Schnittstelle an der Rückseite Ihres Amiga 500 zu steuern. Ähnlich wird *printer.device* zur Steuerung eines Druckers benötigt. Und *narrator.device* wird immer dann benötigt, wenn Sie die Sprachsynthese benutzen (zum Beispiel mit dem Programm *Say*). Die Datei *Mountlist* wurde bereits bei der Beschreibung der RAM-Disk RAD: erläutert. Sie wird vom MOUNT-Befehl benötigt. Und in der Datei *system-configuration* speichert *Preferences* die Einstellungen (für Workbench-Farben, Mausübersetzung und so weiter), die Sie in *Preferences* gewählt haben.

Das Directory *devs* enthält außer diesen Dateien noch drei weitere Directories. *Clipboards* wird von *clipboard.device* benötigt und braucht Sie nicht weiter zu interessieren (dieses Directory ist fast immer leer). *Printers* enthält die verschiedenen »Druckertreiberdateien«, die Sie in *Preferences* auswählen können. Sie benötigen wahrscheinlich nur eine davon, da Sie ja auch nur einen Drucker besitzen. Und das Directory *keymaps* im Directory *devs* enthält die verschiedenen Tastaturbelegungen. Auch davon benötigen Sie nur eine (ich nehme einmal an, die deutsche). Damit wäre der Überblick über *devs* und seine Unterdirectories aber auch schon abgeschlossen.

Das Directory *s* enthält Kommandofolgen. Hier (und im aktuellen Directory) sucht der Befehl EXECUTE die Kommandofolgen, deren Namen ihm als Parameter angegeben werden. Auf der Original-Workbench-Diskette enthält *s* unter anderem die Dateien *Startup-Sequence*, *StartupII*, DPAT und SPAT, die alle bereits in den vorangehenden Abschnitten beschrieben wurden.

Das Directory *t* ist für temporäre Dateien gedacht. Programme legen in dieses Directory Dateien, die nur vorübergehend benötigt werden (wie zum Beispiel MEMACS, das unter Umständen die Datei edit.backup hier anlegt). Auf der originalen Workbench-Diskette ist *t* zunächst einmal leer und bleibt es meist auch, da das logische Laufwerk T:, in dem temporäre Dateien üblicherweise abgespeichert werden, beim Start auf die RAM-Disk umgelegt wird.

*Fonts* hingegen enthält wieder eine Reihe von Dateien und Unterdirectories. Im Directory *Fonts* werden die verschiedenen Schriftarten abgespeichert, wie Sie wahrscheinlich schon an den Namen der Dateien darin feststellen werden. Zu jeder Schriftart existiert hier eine Datei und ein Directory. Beide zusammen werden benötigt, um Schrift in dieser Schriftart darstellen zu können. Wenn Sie zusätzliche Schriftarten erwerben – es gibt inzwischen schon viele Disketten voll davon – müssen Sie für jede neue Schriftart eine Datei und ein Directory in das Directory *Fonts* der Workbench-Diskette legen. In vielen Programmen und auch bei der Arbeit im Shell-Fenster können Sie aber auf Schriftarten und damit auf den Inhalt von *Fonts* weitgehend verzichten.

Das Directory *libs* ist hingegen wieder sehr wichtig. Es enthält Bibliotheken mit kleinen Programmroutinen, die von anderen Programmen aus benutzt werden können. Die Namen dieser Bibliotheken enden alle mit »library« (englisch für Bibliothek). Die Datei *diskfont.library* wird benötigt, wenn Programme die im Directory *Fonts* liegenden Zeichensätze verwenden wollen. Die Bibliotheken *mathtrans.library, mathieeedoubtrans.library* und *mathieeedoubbaslibrary* enthalten mathematische Routinen (transzendentale Funktionen und mathematische Operationen doppelter Genauigkeit), die nur von wenigen Programmen (aber unter anderem auch vom Amiga-Basic) benötigt werden. *Icon.library* enthält Routinen zur Darstellung von Piktogrammen und *info.library* wird für Fälle, in denen man

mehr zu einer Datei, als nur ihren Namen, wissen will, benötigt. Der *Info*-Befehl der Workbench benötigt sie zum Beispiel. *Translator.library* wird zur Sprachsynthese benötigt und die *version.library* teilt Programmen mit, welche Version der Workbench-Sofware sie benutzen.

Das Directory *Empty* ist – wie sein Name schon sagt – leer. Es entspricht der Schublade *Empty* auf der Workbench und dient zur Erzeugung neuer, leerer Schubladen mit dem Befehl *Duplicate*.

Auch das Directory *Expansion* ist üblicherweise leer, solange Sie keine Hardware-Erweiterungen kaufen, die spezielle Treiberdateien benötigen. Solche Treiberdateien müssen, wenn sie einer Festplatte oder einem anderen Gerät beiliegen, in dieses Directory gelegt werden, damit das Gerät benutzt werden kann.

Das Directory *Utilities* ähnelt dem oben beschriebenen Directory *System*. Genau wie *System* ist es auch als Schublade auf der Workbench zu sehen und enthält eine Reihe zusätzlicher Programme, die sowohl von der Shell wie auch von der Workbench aus verwendet werden können. Sie kennen diese beiden Programme aus Kapitel 4. Auch andere Programme, die im Shell-Fenster und auf der Workbench nützlich sein können (wie zum Beispiel MEMACS) sollten Sie in dieses Directory legen. Auch das Directory *Utilities* wird, wie *System*, beim Start in den Suchpfad eingehängt.

### Worauf Sie gut verzichten können

Wie Sie im letzten Abschnitt gesehen haben, besitzt die Workbench-Diskette einen recht komplexen Aufbau. Eine Unzahl von Dateien und ineinander verschachtelten Directories tummelt sich darauf. Auf viele davon können Sie aber gut verzichten. Oder anders gesagt: Sie können diese Dateien und Directories löschen, wenn Sie auf bestimmte Funktionen der Shell oder der Workbench verzichten können. Dazu finden Sie im folgenden ein paar Tips. Wenn Sie diese Tips ausprobieren wollen, sollten Sie das aber unbedingt auf einer Kopie der Workbench-Diskette machen, mit der Sie normalerweise arbeiten (die ja auch schon eine Kopie sein sollte).

Sie müssen sich übrigens auch nicht für immer und ewig entscheiden, welche Dateien Sie verwenden wollen und welche nicht. Es macht auch durchaus Sinn, sich mehrere verschiedene Workbench-Disketten zu schaffen, die ganz verschiedene Dateien und Directories enthalten, je nachdem, was Sie gerade mit Ihrem Amiga machen wollen. Das heißt, für das Programmieren mit Basic schaffen Sie sich eine spezielle Workbench-Diskette und für die Textverarbeitung eine andere. Eine wirklich universelle Workbench-Diskette zu schaffen, die man immer und für jeden Zweck verwenden kann, ist nahezu unmöglich. Dazu ist eine Diskette einfach zu klein. Festplattenbesitzer sind da etwas glücklicher dran.

Achten Sie beim Löschen von Dateien, die man auch auf der Workbench sehen kann, unbedingt darauf, daß Sie immer die Datei zusammen mit der zugehörigen Info-Datei löschen. Löschen Sie nur eine Datei und nicht deren Info-Datei, so ist die gelöschte Datei auf der Workbench scheinbar noch (als Piktogramm) vorhanden. Das kann zu verwirrenden Fehlermeldungen führen. Dateien, die als Piktogramme sichtbar sind, sollten Sie deshalb am besten immer auf der Workbench löschen. Dabei kann ein solches Versäumnis nämlich nicht vorkommen. Mehr zu Info-Dateien erfahren Sie aber auch noch in einem späteren Abschnitt dieses Kapitels.

Nach dieser langen Vorrede aber nun ans Werk: Falls Sie nicht ein intensiver Anwender eines Textverarbeitungsprogramms sind oder immer nur mit einer Schrift arbeiten, sollten Sie zunächst einmal sämtliche Schriftart-Dateien löschen. Das ist recht einfach mit dem folgenden Shell-Befehl möglich:

```
DELETE :Fonts/#? ALL
```

Dieser Befehl löscht alle Dateien und Directories im Directory *Fonts* (ohne dabei das Directory *Fonts* selbst zu löschen). Das Directory *Fonts* bleibt sicherheitshalber bestehen, weil einige Programme auf seine Existenz Wert legen – selbst dann, wenn sie es leer vorfinden.

Wenn Sie nur einen Zeichensatz löschen wollen – weil er Ihnen zum Beispiel nicht gefällt – können Sie dazu die beiden folgenden Befehle verwenden:

```
DELETE :fonts/<name>.font ALL
DELETE :fonts/<name> ALL
```

(Dabei müssen Sie <name> natürlich gegen den Namen der Schriftart ersetzen, die Sie löschen wollen.) Aber nicht nur das Schriftarten-Directory eignet sich gut dazu, Platz zu schaffen. Auch aus dem Directory *devs/printers* können Sie alle Dateien bis auf eine löschen. (Falls sich auf Ihrer Workbench-Diskette überhaupt mehr als eine Datei in diesem Directory befindet.) Die eine Datei die Sie übriglassen sollten, ist natürlich die, deren Namen Sie im Programm *Preferences* ausgewählt haben und die zu dem Drucker paßt, den Sie verwenden. Wenn Sie (zumindest mit dieser Workbench-Diskette) überhaupt nicht drucken wollen, können Sie *devs/printers* mit dem folgenden Befehl komplett löschen:

```
DELETE :devs/printers ALL
```

Nicht immer will man gleich so radikal vorgehen, wie bisher beschrieben, und ganze Directories auf einmal löschen. Für das Löschen einzelner Dateien eignet sich hingegen der DIR-Befehl in seiner interaktiven Form sehr gut. Geben Sie dazu den folgenden Befehl ein:

```
DIR :devs/printers INTER
```

Sie bekommen dann die Dateien im Directory *devs/printers* einzeln aufgelistet. Nach jedem Namen haben Sie die Möglichkeit, die Datei zu löschen, indem Sie »del« und [Return] eingeben oder sich mit [Return] den nächsten Namen zeigen lassen.

Diese Option des DIR-Befehls eignet sich natürlich nicht nur zum selektiven Löschen von Dateien im *printers*-Directory. Sie können damit natürlich auch ganze Disketten nach Löschkandidaten »durchkämmen«. Lesen Sie dazu im Zweifelsfall noch einmal die Beschreibung des DIR-Befehls im Kapitel 7 nach.

Nun aber zurück zum Thema Drucken. Wenn Sie gar nicht drucken wollen, können Sie im Directory *devs* außer dem Directory *printers* auch noch die Dateien *serial.device*, *parallel.device* und *printer.device* löschen. Dann können Sie allerdings auch keine anderen Geräte an die serielle oder parallele Schnittstelle auf der Rückseite des Amiga 500 anschließen. Und wenn Sie gerade sowieso im Directory devs »zugange sind«, können Sie auch gleich noch die Datei *narrator.device* löschen. *Narrator.device* benötigen Sie nur, wenn Sie Sprachausgabefähigkeiten des Amiga nutzen wollen. Darauf kann man schon einmal verzichten, um etwas Platz auf der Diskette zu schaffen.

Viele Möglichkeiten, Platz zu schaffen, bietet auch das Directory *c*, das die Shell-Befehle enthält. Viele dieser Befehle benötigen Sie wahrscheinlich nie oder nur äußerst selten. Sie können in diesem Directory deshalb nach Herzenslust löschen – besonders dann, wenn Sie sich eine Diskette schaffen wollen, die nur für das Arbeiten mit der Workbench gedacht ist. (Eine vollständige Version der Workbench-Diskette haben Sie ja sowieso aufgehoben – oder?) Beachten Sie bitte, daß einige der Befehle im Directory *c* auch dann benötigt werden, wenn Sie selber nie die Shell verwenden. Damit die *Startup-Sequence* und *StartupII* fehlerfrei ablaufen kann, müssen die in ihr verwendeten Kommandos verfügbar sein.

Damit Sie zur Not auch einmal mit der Shell arbeiten können, sollten Sie die folgenden Befehle auf alle Fälle nicht löschen.

```
ADDBUFFERS
BINDDRIVERS
CD
COPY
DATE
DIR
ECHO
ENDCLI
ELSE
ENDIF
FAILAT
IF
LAB
LIST
LOADWEB
MOUNT
PATH
PROMPT
RESIDENT
SETCLOCK
SETMAP
SETPATCH
SKIP
TYPE
VERSION
```

Ihre Version der Workbench-Diskette kann unter Umständen aber eine *Startup-Sequence* mit noch mehr anderen Kommandos enthalten. Betrachten Sie sich die *Startup-Sequence* und *StartupII* deshalb am besten, bevor Sie mit der Lösch-Orgie beginnen und notieren Sie sich alle darin vorkommenden Shell-Befehle. Diese Befehle dürfen Sie auf keinen Fall löschen!

Verwenden Sie zum Löschen am besten wieder den DIR-Befehl mit der Option INTER. Legen Sie dieses Buch dazu neben Ihre Tastatur und schlagen Sie Kapitel 7 auf. Löschen Sie dann die Befehle, von denen Sie meinen, daß Sie darauf verzichten können. Lesen Sie die Funktion der Befehle im Zweifelsfall im Kapitel 7 nach. Und löschen Sie vor allem keine Befehle, die in der Startup-Sequence vorkommen!

Nach dieser riskanten Operation nun wieder eine einfache. Löschen Sie alle – zumindest für diese Workbench-Diskette – überflüssigen Programme aus dem Directory *Utilities*. Gehen Sie dazu am besten auf die Workbench. Es ist auf der Workbench wesentlich leichter, die Programme und Directories zusammen mit Ihren Piktogrammen zu löschen. Klicken Sie die gewünschten Piktogramme einfach nacheinander an, wobei Sie die [Shift]-Taste gedrückt halten und wählen Sie dann *Discard* aus dem Menü *Workbench*.

Auch im Directory *System* können Sie auf diese Weise gefahrlos alle selten benötigten Programme (wie zum Beispiel *CLI*, *FixFonts*, *MergeMen*, *InitPrinter*, *Format*, *DiskCopy* und so weiter) löschen. Eventuell können Sie auch die Schublade *Preferences* komplett löschen. Preferences ist zwar wichtig, wird aber sehr selten benutzt. (Bedenken Sie aber, daß Sie ohne *Preferences* unter anderem den Mauszeiger und die Workbenchfarben nicht mehr ändern können und keinen anderen Drucker auswählen können.)

Kommen wir nun zu den beiden Directories *l* und *libs*. In beiden ist das Löschen von Dateien etwas riskant. Lesen Sie die folgenden Erläuterungen deshalb gut durch! Im Directory *l* können Sie unter Umständen die Datei *Port-Handler* löschen, wenn Sie die Schnittstellen an der Rückseite des Amiga nicht benötigen. Die Amiga-DOS-Geräte SER:, PAR: und PRT: funktionieren dann nicht mehr. (In diesem Fall sollten Sie auch *serial.device*, *parallel.device* und *printer.device* löschen, wie oben beschrieben.) Wenn Sie nicht über den Luxus einer Festplatte verfügen, können Sie auch *FastFileSystem* löschen.

Die einigermaßen risikolosen Fälle im *libs*-Directory sind *version.library* und *translator.library*. Die Datei *version.library* wird in fast keinem Programm benötigt. Lediglich der Shell-Befehl VERSION und der gleichnamige Workbench-Befehl funktionieren nicht mehr, wenn *version.library* fehlt. *Translator.library* wird nur zur Sprachsynthese gebraucht. Wenn Sie darauf verzichten können, Ihren Amiga sprechen zu hören: weg mit *translator.library*, die recht groß ist.

### 8.6.2 Die Mini-Workbench für Sparsame

Nachdem Sie nun erfahren haben, auf welche Dateien auf der Workbench-Diskette Sie »unter Umständen« verzichten können, möchte ich Ihnen nun eine Art Mini-Workbench vorstellen. Diese Diskette enthält nur das Nötigste für die Arbeit mit der Shell und auf der Workbench und enthält rund 230 Kbyte Daten. Mehr als 600 Kbyte sind frei. Die folgende Liste zeigt Ihnen einen Überblick über den Inhalt dieser Diskette, ähnlich wie sie der Befehl DIR ALL ergeben würde.

```
Trashcan (dir)
c (dir)
     AddBuffers          assign
     BindDrivers         CD
     copy                delete
     dir                 echo
     Else                EndCLI
     EndIf               Execute
     FailAt              If
     Info                List
     LoadWB              MakeDir
```

```
     Mount                    NewShell
     Path                     Prompt
     Resident                 Run
     SetClock                 SetPatch
System (dir)
     .info                    SetMap
     SetMap.info
l (dir)
     Disk-Validator           FastFileSystem
     Newcon-Handler           Ram-Handler
     Shell-Seg
devs (dir)
     keymaps (dir)
        d
     printers (dir)
     clipboards (dir)
     clipboard.device         Mountlist
     ramdrive.device          system-configuration
s (dir)
     DPAT                     Shell-Startup
     SPAT                     Startup-Sequence
     StartupII
t (dir)
fonts (dir)
libs (dir)
     diskfont.library         icon.library
     info.library
Empty (dir)
     .info
Expansion (dir)
     .info
.info                         Disk.info
Empty.info                    Expansion.info
Shell                         Shell.info
System.info                   Trashcan.info
```

Eine so sparsam gefüllte Diskette können Sie am einfachsten herstellen, indem Sie die normale Workbench-Diskette kopieren und dann auf der Kopie alle hier nicht aufgezählten, überflüssigen Dateien und Directories löschen.

Damit eine Workbench-Diskette mit dieser Ausstattung einwandfrei starten kann, müssen Sie ein paar kleine Änderungen in Startup-Sequence und StartupII vornehmen. Aus *StartupII* müssen die MOUNT-Befehle für AUX:, PIPE: und SPEAK: gelöscht werden. Aus *Startup-Sequence* müssen Sie die Befehle FASTMEMFIRST und FF löschen. Und die Befehle BREAK in *StartupII* sowie RUN und WAIT in *Startup-Sequence* sollten Sie ebenfalls entfernen, wie es in Abschnitt 8.4.14 beschrieben wurde.

Diese Diskette enthält sicherlich noch nicht das absolute Minimum an Dateien. Für meine eigenen Zwecke ist sie aber ideal und ermöglicht ein Arbeiten sowohl im Shell-Fenster wie

auch auf der Workbench. Wo das absolute Minimum an Dateien liegt, ist auch schwer zu sagen. Es hängt von den Programmen ab, die Sie verwenden. Verschiedene Programme machen nämlich verschiedene Annahmen über die Dateien und Directories, die Sie im logischen Laufwerk SYS: (also auf der Workbench-Diskette) erwarten. Die meisten von mir selbst benutzten Programme (zum Beispiel *Notepad* und *Deluxe Paint*) arbeiteten einwandfrei, wenn ich meinen Amiga 500 mit dieser Diskette startete. Es ist aber nicht ausgeschlossen, daß anderen Programmen gewisse Dateien fehlen (zum Beispiel die mathematischen Bibliotheken im *libs*-Directory). Auch ist mit dieser Diskette keine Sprachausgabe möglich, weil die entsprechenden Bibliotheken darauf fehlen. Bevor Sie selbst damit beginnen, ständig mit einer solchen Mini-Workbench zu arbeiten, sollten Sie die Programme, mit denen Sie später arbeiten wollen, ausführlich testen, nachdem Sie einen Neustart mit dieser Workbench-Diskette durchgeführt haben.

Diese Mini-Workbench ist aber kein Selbstzweck. Im täglichen Einsatz werden Sie die rund 650 Kbyte, die darauf noch frei sind, wahrscheinlich nicht frei lassen. Statt dessen können Sie Programme und Daten auf diese Diskette legen und so mit einer Diskette auskommen, wo Sie sonst zwei benötigen würden. Für Besitzer nur eines Diskettenlaufwerks (des internen) bedeutet das weniger Diskettenwechsel und für Besitzer von zwei Laufwerken wird ein Laufwerk frei für zusätzliche Datendisketten.

Fertigen Sie deshalb sofort dann, wenn Sie sich eine Mini-Workbench zusammengelöscht haben, eine oder mehrere Kopie(n) dieser Diskette an und legen Sie das Original beiseite. Auf diese Kopien können Sie nun gut ein oder zwei Programme legen und müßten dann noch genug Platz für ein paar Projekte (Daten) haben. (Dies ist bei kopiergeschützten Programmen leider schwierig oder unmöglich.)

Auch für Programmierer ist eine solche Diskette sehr interessant. Um zum Beispiel mit Amiga-Basic zu arbeiten, benötigen Sie normalerweise zwei Disketten (und damit idealerweise auch zwei Diskettenlaufwerke). Wenn Sie das Piktogramm Amiga-Basic aber auf die Mini-Workbench-Diskette legen, haben Sie noch reichlich Platz für eigene Programme und Hilfsdateien. (Um mit Basic zu arbeiten, sollten Sie allerdings die Bibliotheken *mathtrans.library*, *mathieedoubtrans.library* und *mathieedoubbaslibrary* im Directory *libs* belassen.)

### 8.6.3 Die RAM-Workbench

Eine andere Möglichkeit, die Mini-Workbench zu benutzen, ist es, sie vollständig in eine RAM-Disk zu legen und diese RAM-Disk dann als sehr schnelle »Workbench-Diskette« zu benutzen. Die folgende Kommandofolge würde das zum Beispiel unter Verwendung der RAM-Disk RAM: tun:

```
IF NOT EXISTS RAM:C
  ECHO "Kopiere Workbench-Diskette nach RAM ...."
  COPY DF0: RAM: ALL BUFFER=20 CLONE
  ASSIGN C:     RAM:c
  ASSIGN S:     RAM:s
  ASSIGN L:     RAM:l
```

```
  ASSIGN LIBS:  RAM:Libs
  ASSIGN FONTS: RAM:Fonts
  ASSIGN DEVS:  RAM:Devs
  ASSIGN SYS:   RAM:
  CD SYS:
  PATH SYS: SYS:c SYS:System RESET
  ECHO "RAM-Workbench komplett!"
ENDIF
```

Besonders wichtig an dieser Kommandofolge sind die ASSIGN-Befehle. Erst diese machen wirklick die RAM-Disk zur Workbench-Diskette. Diese Diskette hat allerdings einen Nachteil: Starten können Sie von ihr nicht. Zum Neustart muß immer die echte Workbench-Diskette – in diesem Fall die Mini-Workbench – eingelegt werden.

Wenn Sie jedoch einen Amiga 500 mit den neuen Kickstart-ROMs (Version 1.3 oder höher) besitzen, können Sie von einer RAM-Workbench-Diskette, die Sie auf RAD: anlegen, sogar starten! Aber auch mit den alten Kickstart-ROMs können Sie RAD: als Workbench-Diskette verwenden. Erfassen Sie dazu zunächst die folgende Kommandofolge mit Ihrem Lieblingseditor:

```
MOUNT >NIL: RAD:
IF NOT EXISTS RAD:C
  ECHO "Kopiere Workbench-Diskette nach RAD ...."
  COPY DF0: RAD: ALL BUFFER=20 CLONE
ENDIF
FAILAT 20
ASSIGN C:     RAD:c
ASSIGN S:     RAD:s
ASSIGN L:     RAD:l
ASSIGN LIBS:  RAD:Libs
ASSIGN FONTS: RAD:Fonts
ASSIGN DEVS:  RAD:Devs
ASSIGN SYS:   RAD:
CD SYS:
PATH SYS: SYS:c SYS:System RESET
ECHO "RAD-Workbench komplett!"
```

Speichern Sie diese Kommandofolge dann unter dem Namen RADWB im Directory *s* der Mini-Workbench ab. Besitzer des Amiga 500 mit neuen Kickstart-ROMs brauchen diese Kommandofolge dann nur einmal aufzurufen und können dann die Diskette aus dem Laufwerk nehmen. Den nächsten Neustart (zum Beispiel nach [Ctrl]+[*A*]+[*A*]) führt der Amiga dann schon von RAD: aus durch – was blitzschnell abläuft. Auch diese Kommandofolge kann man natürlich automatisch bei jedem Neustart ausführen lassen, indem man einfach den Befehl EXECUTE RADWB als letzte Zeile in *StartupII* einfügt.

Besitzer des Amiga 500 mit alten Kickstart-ROMs müssen noch eine weitere Modifikation in der Startup-Sequence durchführen, um die RAD-Workbench richtig nutzen zu können. Löschen Sie die folgende Zeile aus *Startup-Sequence*:

```
NEWSHELL NEWCON:0/170/640/86/AmigaShell FROM S:StartupII
```

Und fügen Sie statt dessen dort diese sieben Zeilen ein:

```
MOUNT >NIL: RAD:
IF EXISTS RAD:C
CD RAD:C
NEWSHELL NEWCON:0/170/640/86/AmigaShell FROM RAD:S/StartupII
ELSE
NEWSHELL NEWCON:0/170/640/86/AmigaShell FROM S:StartupII
ENDIF
```

Dieser Teil von *Startup-Sequence* überprüft jetzt, ob RAD: bereits initialisiert wurde und führt *StartupII* dann gleich von dort aus. In *StartupII* werden dann die ASSIGN-Befehle ausgeführt, die RAD: zur Workbench-Diskette machen.

Bei der Verwendung von RAD: muß aber unbedingt beachtet werden, daß diese RAM-Disk eine feste Kapazität hat. Die oben beschriebene Mini-Workbench würde zwar gerade eben auf die normalgroße RAM-Disk RAD: (mit 242 Kbyte Kapazität) passen. Viel Platz würde dabei aber nicht mehr bleiben. Schon, wenn Sie MEMACS mit auf die Mini-Workbench legen würden, würde RAD: beim Füllen überlaufen. In diesem Fall müßten Sie die Kapazität von RAD:, wie in Abschnitt 8.5.5 beschrieben, durch eine Änderung in der *Mountlist* heraufsetzen. Dabei ist auf jeden Fall Vorsicht geboten. Auch auf einem Amiga mit 1 Mbyte RAM sollte RAD nicht größer gemacht werden als maximal 300 Kbyte. Ansonsten gibt es sehr rasch Probleme mit zu knappem Speicher.

Wenn Sie glücklicher Besitzer einer Speichererweiterung auf 2 Mbyte sind, gibt es diese Probleme nicht. Dann können Sie sogar eine ganz normale Workbench-Diskette (also nicht nur die Mini-Workbench) auf eine RAD:-RAM-Disk legen und mit den neuen Kickstart-ROMs auch von dieser Diskette starten. Dazu müssen Sie lediglich in der *Mountlist* im Abschnitt RAD: die Zahl hinter *HighCyl* auf 79 setzen. RAD: bekommt dadurch dieselbe Kapazität wie eine normale Diskette (880 Kbyte). Sie können dann sogar DISKCOPY verwenden, um die Workbench-Diskette nach RAD: zu kopieren, was schneller geht als mit COPY.

### 8.6.4 Schriften löschen und zusätzliche Schriftarten installieren

Wie Sie gerade gesehen haben, ist eine optimale Nutzung einer RAM-Disk fast nur vom CLI aus möglich. Auch andere Aufgaben lassen sich nahezu ausschließlich im Shell-Fenster erledigen. Dazu gehört auch die Installation zusätzlicher Schriften, die nicht schon auf der Standard-Workbench-Diskette vorhanden sind.

Sie können auf zwei verschiedene Arten erreichen, daß *Notepad* und andere Programme Ihnen eine größere oder eine andere Auswahl von Schriftarten bietet. Die erste Methode macht sich die besondere Bedeutung des logischen Laufwerks FONTS: zunutze. Programme, die verschiedene Schriftarten verwenden können (wie zum Beispiel *Notepad*), suchen nach diesen Schriften immer im logischen Laufwerk FONTS: (das beim Start zunächst dem Directory *Fonts* auf der Workbench-Diskette zugeordnet wird).

Wenn Sie mit dem ASSIGN-Befehl vor dem Start von *Notepad* das logische Laufwerk FONTS: umdirigieren, sucht *Notepad* an anderer Stelle nach den Schriftarten. Angenommen, Sie besitzen eine Diskette namens *Extras1.3D*, die in einem Directory namens *Fonts* eine Auswahl verschiedener Schriftarten enthält. (Das ist bei der Extras-Diskette übrigens

wirklich der Fall.) Dann können Sie mit dem folgenden Befehl erreichen, daß Ihnen diese Schriften im *Notepad* zur Verfügung stehen (aber nicht mehr die »normalen« von der Workbench-Diskette).

```
ASSIGN FONTS: Extras1.3D:Fonts
```

Nachteil dieser Methode ist, daß dann ausschließlich die Schriften auf der anderen Diskette zur Verfügung stehen. Wenn Sie aber wollen, daß Ihnen die neuen Schriften immer und zusätzlich zu den normalen Schriften zur Verfügung stehen sollen, müssen Sie die Schriftart-Dateien in das Directory *Fonts* auf der Workbench-Diskette kopieren. Dazu muß auf der Diskette natürlich auch ausreichend Platz sein, was sie vorher überprüfen sollten!

Wenn das Directory *Fonts* auf der anderen Diskette aus dem obigen Beispiel nur Schriften und keine anderen Dateien enthält, können Sie diese mit dem folgenden Befehl der normalen Auswahl von Schriften hinzufügen:

```
COPY Extras1.3D:Fonts SYS:Fonts ALL
```

Wenn Sie nur einzelne Schriften kopieren wollen, können Sie natürlich auch das tun. Beachten Sie aber, daß eine Schriftart immer zwei Bestandteile hat. Im *Fonts*-Directory auf der Extras-Diskette ist die Schrift *Times* zum Beispiel durch die Datei *Times.font* und durch das Directory *Times* vertreten. Jede Datei im Directory *Times* entspricht einer Größe der Schrift. Wollen Sie diese Schrift auf einer anderen Diskette kopieren, müssen Sie sowohl die Datei *Times.font* wie auch das komplette Directory *Times* (samt den darin enthaltenen Dateien) kopieren. Nur dann kann die zusätzliche Schrift auch genutzt werden.

Am wichtigsten für die Schriftauswahl, die Ihnen in Programmen angeboten werden, in denen mehrere Schriften verwendet werden können, sind die Dateien im logischen Laufwerk FONTS: mit der Endung ».font«. Diese Dateien enthalten eine Liste der im zugehörigen Directory vorhandenen Schriftgrößen und weitere Informationen zu der jeweiligen Schrift. Der Inhalt dieser Dateien muß unbedingt zum wirklichen Inhalt der zugehörigen Directories passen. Wenn Sie beispielsweise eine Datei (entsprechend einer Schriftgröße) aus dem Directory *diamond* löschen, muß auch die Datei *diamond.font* entsprechend korrigiert werden. Diese Korrektur kann leider nicht mit einem Texteditor ausgeführt werden, da die Font-Dateien nicht nur Text sondern auch sogenannte »Binär-Daten« enthalten.

Das ist aber nicht weiter tragisch. Für die Anpassung der Font-Dateien an geänderte Verhältnisse in den Schrift-Directories gibt es einen speziellen Shell-Befehl namens FIXFONTS. Dieser Befehl (er entspricht dem Programm *FixFonts* aus der System-Schublade) hat keine Parameter und kann jederzeit aufgerufen werden. Er überprüft dann den kompletten Inhalt des logischen Laufwerks FONTS: und korrigiert die Font-Dateien entsprechend. Rufen Sie FIXFONTS am besten jedesmal auf, wenn Sie irgendwelche Änderungen an den Schriftdateien vorgenommen haben.

### 8.6.5 Piktogramme vom Shell-Fenster aus gesehen

Zum Abschluß dieser kleinen Sammlung von Tips und Tricks möchte ich mich noch etwas um die Verbindung zwischen Shell-Fenster und Workbench verdient machen. Bis jetzt waren die Tips ja zu einem großen Teil nur für Anwender gedacht, die vermehrt in der Shell tätig werden wollen. Sie werden nun sehen, wie man eine Brücke von der Shell zur Workbench schlagen und zum Beispiel »unsichtbare« Dateien sichtbar machen kann.

In diesem und den vorangegangenen Kapiteln sind ja schon mehrfach die Info-Dateien angesprochen worden. Diese tragen denselben Namen wie jeweils eine andere Datei – nur mit der zusätzlichen Endung ».info«. Auch Disketten besitzen eine solche Info-Datei. Diese heißt aber immer Disk.Info und liegt im Wurzeldirectory der Diskette. Die Info-Dateien sind hauptsächlich für die Workbench gedacht. Sie enthalten unter anderem das Piktogramm, das auf der Workbench die entsprechende Datei oder das Directory repräsentiert, die Position, an der dieses Piktogramm innerhalb des Schubladen-Fensters dargestellt wird und alle anderen Informationen, die Sie mit dem *Info*-Befehl aus dem *Project*-Menü betrachten und ändern können.

Für Programmierer steht eine Bibliothek zur Verfügung (info.library), mit deren Hilfe diese Informationen zu einer beliebigen Datei ermittelt werden können, ohne daß man dazu das Format der Info-Datei kennen muß. Nur wenige Programme benutzen aber im Moment diese Informationen überhaupt – hauptsächlich dienen sie zu Ihrer Information und dazu, eine Verbindung zwischen Programmen und ihren Projekten herzustellen.

Wenn Sie aber gar keine komplizierten Operationen mit einer Info-Datei vorhaben, sondern einfach nur eine Datei (oder ein Dateiverzeichnis) sichtbar machen wollen, die Sie zwar mit dem DIR-Befehl im Shell-Fenster, nicht aber auf der Workbench als Piktogramm sehen, ist das verhältnismäßig einfach und ohne großen Programmieraufwand möglich.

Dazu brauchen Sie dieser Datei nur eine entsprechende Info-Datei zu verschaffen. Der einfachste Weg, eine solche Datei zu erzeugen, ist das Kopieren einer anderen Info-Datei. Der folgende Befehl macht zum Beispiel eine Datei namens DIR (deren Name Ihnen bekannt vorkommen sollte), die sich im Dateiverzeichnis *c* befindet (fast) sichtbar.

```
COPY :Utilities/Notepad.info C:DIR.info
```

Der »Haken« an diesem Befehl ist nur, daß damit zwar DIR ein Piktogramm besitzt, aber in einem Directory liegt, das nicht über ein Piktogramm verfügt. Auf der Workbench gilt ja immer die Regel, daß Sie eine Schublade erst öffnen müssen, bevor Sie die darin befindlichen Objekte sehen. Und die Schublade (das Directory) *c* können Sie nicht öffnen, da es kein Piktogramm hat. Aber auch dieser Zustand läßt sich relativ schnell beheben, indem Sie den folgenden Befehl im Shell-Fenster eingeben:

```
COPY :Utilities.info :c.info
```

Dadurch wird die Schublade *c* sichtbar. Sie können sich davon überzeugen, indem Sie auf die Workbench gehen, das Diskettenfenster der Workbench-Diskette gegebenenfalls schließen und wieder öffnen. An der Stelle, an der zuvor nur die Schublade *Utilities* lag, liegen nun zwei Schubladen, *Utilities* und *c*, übereinander. Ergreifen Sie die oberste davon mit der Maus und legen Sie sie etwas zur Seite. Sie können nun beide Schubladen in voller Pracht und Schönheit sehen. Öffnen Sie nun *c* mit einem Doppelklick und nach geraumer Zeit, in der das Diskettenlaufwerk heftig arbeitet, erscheint ein Piktogramm namens DIR, daß wie *Notepad* aussieht. Von der Vielzahl der Programme im Directory *c* haben Sie damit also eines, DIR, sichtbar gemacht.

Alle anderen Shell-Befehle sind noch unsichtbar (aber Sie wissen jetzt ja, wie Sie sie sichtbar machen können). Es wäre aber zum Beispiel schon möglich, fast alle Shell-Befehle auf einen Schlag zu löschen, indem Sie die Schublade *c* in den Mülleimer werfen und diesen dann leeren. Davor sollten Sie sich aber hüten – denn dann haben Sie keine Shell-Befehle mehr und können diese Diskette dann auch nicht mehr als Workbench-Diskette verwenden!

Versuchen Sie aber bitte nicht, DIR zu starten, indem Sie einen Doppelklick auf das entsprechende Piktogramm machen. Der Shell-Befehl DIR hat es nicht gerne, wenn er von der Workbench aufgerufen wird und reagiert darauf nur mit einem Programmabsturz, der von einem System-Requester dokumentiert wird.

Mit der oben vorgestellten Methode können unsichtbare Dateien und Schubladen wenigstens schon einmal sichtbar gemacht werden. Die Auswahl für die Piktogramme, mit denen zuvor unsichtbare Dateien auf der Workbench auftauchen, ist jedoch beschränkt – auf genau die Piktogramme, die schon in irgendeiner Schublade zu finden sind. Für den, der eigene Piktogramme erstellen will, gibt es den Piktogramm-Editor *IconEd*, der bereits in Kapitel 5 ausführlich beschrieben wurde.

Auch mit *IconEd* können Sie übrigens Dateien und Directories sichtbar machen. Dazu müssen Sie nur das Piktogramm eines sichtbaren Objekts laden und dann unter dem Namen eines noch unsichtbaren Objekts (einer Datei oder eines Piktogramms) wieder abspeichern. Dieser Vorgang ist genau betrachtet auch nichts anderes als das Kopieren einer Info-Datei.

Sie müssen beim Kopieren einer Info-Datei – egal ob mit dem COPY-Befehl oder mit *IconEd* – aber immer darauf achten, daß Sie Piktogramme der richtigen Art kopieren. Wenn Sie einer Schublade ein Piktogramm verschaffen wollen, müssen Sie dazu ein Schubladen-Piktogramm kopieren. Und für Programme und andere (nicht lauffähige) Dateien müssen Sie Piktogramme von Werkzeugen beziehungsweise Projekten kopieren. Wenn Sie diese Regel nicht beherzigen, können bei einem Doppelklick auf die neuen Piktogramme die merkwürdigsten Effekte und Fehlermeldungen auftauchen.

### 8.6.6 Kommandofolgen von der Workbench aus starten

Wie Sie eben erfahren haben, können Sie einem Shell-Befehl zwar recht einfach ein Piktogramm verschaffen, ihn dann aber immer noch nicht von der Workbench aus verwenden. Das liegt daran, daß Programme, die für die Shell und für die Workbench gedacht sind, teilweise recht unterschiedlich programmiert werden müssen. Es gibt allerdings auch Programme, die für beide Arbeitsumgebungen geeignet sind (zum Beispiel DISKCOPY, FORMAT, MEMACS und so weiter).

Es ist aber ohne weiteres möglich, fast jede beliebige Kommandofolge auch von der Workbench aus zu starten. Speziel für diesen Zweck gibt es einen Shell-Befehl namens ICONX. Dieser Befehl nutzt die Verbindung zwischen einem Projekt (einer Datei) und einem Werkzeug (Programm), die von der Workbench im Info-Fenster des Projektpiktogramms verwaltet wird. Bei einem Doppelklick auf ein Projektpiktogramm startet die Workbench das Programm, dessen Pfadname im Feld *DEFAULT TOOL* des Info-Fensters angegeben wird und übergibt diesem Programm den Namen des Projekts als Parameter.

Um eine Kommandofolge von der Workbench aus starten zu können, müssen Sie diese zunächst einmal mit einem Piktogramm versehen. Einige Methoden dazu haben Sie ja bereits kennengelernt. Eine dritte ist es, die Kommandofolge mit *Notepad* einlesen und dann als *Notepad*-Projekt in einem Directory abspeichern, das auch als Schublade auf der Workbench zu sehen ist. Die Kommandofolge erhält dadurch ein Piktogramm (das Sie natürlich jederzeit mit *IconEd* ändern können). Dieses Piktogramm müssen Sie dann auf der Workbench anklicken und den Befehl *Info* aus dem Menü *Workbench* aufrufen. Daraufhin erscheint das Info-Fenster dieses Piktogramms. In diesem Fenster wird unter anderem im

Feld *DEFAULT TOOL* vermerkt, zu welchem Programm das Dokument gehört. Klicken Sie dieses Feld an, löschen Sie den Namen von *Notepad* und tragen Sie statt dessen »C:ICONX« ein. Wenn Sie jetzt das Fenster mit einem Klicken auf *SAVE* schließen, ist – für die Workbench – die Verbindung zwischen der Kommandofolge und ICONX geschaffen.

Wenn Sie jetzt einen Doppelklick auf das Piktogramm machen, wird die darin gespeicherte Kommandofolge ausgeführt. Vor Ausführung der Kommandofolge macht ICONX die Schublade, in der sich die Kommandofolge befindet, zum aktuellen Directory und öffnet ein Fenster, indem die Ausgaben der Kommandofolge erscheinen (zum Beispiel die Ausgaben von ECHO-Befehlen). In diesem Fenster läuft die Kommandofolge ab, als hätten Sie im Shell-Fenster den EXECUTE-Befehl verwendet. Nach Beendigung der Kommandofolge bleibt das Fenster noch einen kleinen Moment stehen und wird dann automatisch wieder geschlossen.

Probieren Sie ICONX am besten jetzt gleich einmal anhand einer einfachen Kommandofolge (mit ein paar ECHO-Befehlen) aus. Sie werden dabei sehen, daß es recht einfach ist, eine Kommandofolge mit einem Piktogramm zu versehen und über das Feld *DEFAULT TOOL* die Verbindung zu ICONX herzustellen. Nachdem Sie das erste Mal das Info-Fenster einer Kommandofolge auf diese Weise manipuliert haben, brauchen Sie den Info-Befehl übrigens im Prinzip nie wieder. Für alle weiteren Kommandofolgen reicht es aus, die Info-Datei der ersten Kommandofolge zu kopieren, wie es der folgende Befehl demonstriert:

```
COPY :Utilities/Folge1.info :Utilities/Folge2.info
```

In mit ICONX gestarteten Kommandofolgen können Sie alle auch sonst in Kommandofolgen erlaubten Befehle verwenden. Wenn in der Kommandofolge auch Eingaben stattfinden sollen (mit Hilfe des ASK-Befehls), muß die erste Zeile allerdings ».KEY ""« lauten. Über ICONX gestartete Kommandofolgen können aber auch Parameter besitzen, die innerhalb der Kommandofolge auf die übliche Art und Weise (mit KEY-Anweisungen beschrieben) werden. ICONX übergibt der Kommandofolge die Namen aller anderen Piktogramme, die bei ihrem Start ebenfalls markiert waren, als weitere Parameter. Sie müssen diese anderen Piktogramme dazu bei gedrückter [Shift]-Taste anklicken. Da diese Art der Verwendung von Parametern aber recht umständlich ist, sollten Sie am besten nur Kommandofolgen ohne Parameter verwenden.

Position und Größe des von ICONX geöffneten Fensters und die Wartezeit nach Beendigung der Kommandofolge können Sie selbst festlegen. Dazu müssen Sie im Info-Fenster des Piktogramms lediglich zwei Einträge im Feld *TOOL TYPES* vornehmen. (Wie diese Einträge aussehen müssen, können Sie in Kapitel 7 nachlesen.)

ICONX ist nicht so sehr für den Shell-versierten Amiga-Anwender gedacht. Es ist statt dessen ein ideales Werkzeug, um auch andere, die lediglich mit der Workbench vertraut sind, die Vorteile der Shell nutzen zu lassen. Sie können zum Beispiel mit ICONX auch für Freunde Kommandofolgen schreiben, die diese mit einem Doppelklick von der Workbench aus starten können ohne jemals im Leben ein Shell-Fenster geöffnet zu haben. Wenn Sie Interesse daran haben, sich eine raffiniert geschriebene Kommandofolge dieser Art einmal anzusehen, betrachten Sie sich am besten einmal die Datei *InstallPrinter* im Directory *Utilities* auf der Workbench. Diese nutzt äußerst raffiniert Ein/Ausgabeumlenkung und Umgebungsvariablen, um einen regelrechten Dialog mit dem Benutzer zu führen. Wenn Sie diese Kommandofolge verstehen, sind Sie wirklich schon ein sehr versierter Shell-Anwender – und damit hätte dieses Kapitel seinen Zweck erfüllt.

# 9 Ausbaumöglichkeiten

Mit diesem Kapitel beginnt ein Großangriff auf Ihr Bankkonto! Denn dieses und das folgende Kapitel beschäftigen sich mit dem Ausbau des Amiga 500, speziell dem Hardware-Ausbau. Solche Erweiterungen haben bei allen Vorteilen, die sie mit sich bringen, aber auch einen großen Nachteil: sie kosten Geld! Eine Festplatte, ein guter Monitor oder auch ein Farbdrucker kosten in etwa genausoviel wie der Amiga 500 selbst.

Dieses Kapitel soll Ihnen helfen, die Vor- und Nachteile der verschiedenen Erweiterungsmöglichkeiten gegeneinander abzuwägen. Sie sollten danach in der Lage sein, selbst zu entscheiden, was Sie noch benötigen und was nicht. Überläßt man diese Entscheidung dem freundlichen Händler an der Ecke, kommt nämlich oft ein wenig zuviel des Guten dabei heraus – was ja auch durchaus verständlich ist.

## 9.1 Ausbau des Amiga 500 – Ein Überblick

In diesem Abschnitt erhalten Sie einen kleinen Überblick darüber, wie man einen Computer im allgemeinen – und den Amiga 500 im speziellen – erweitern kann. Dazu gibt es ausgesprochen viele Möglichkeiten, die man aber vielleicht nicht immer unter den Begriff »Erweiterung« oder »Ausbau« fassen würde.

### 9.1.1 Jeder Computer ist erweiterbar

Zunächst einmal eine Behauptung vorneweg: Jeder Computer ist erweiterbar! Diese Behauptung ist aus zwei Gründen heraus wahr. Zum einen ist – von ganz wenigen Ausnahmen abgesehen – das, was Sie unter der Bezeichnung »Computer« kaufen, nur das Kernstück eines kompletten Computer-Systems. Zu diesem System gehört noch eine ganze Reihe Komponenten, die Sie zum Teil sogar benötigen, um überhaupt mit dem Computer arbeiten zu können. Bildschirme, zusätzliche Speichersysteme und Drucker sind zum Beispiel solche Komponenten, mit denen Sie den »nackten« Computer erweitern können oder vielleicht auch müssen. Ja, einige Computer-Firmen gehen sogar soweit, Computer ohne Tastatur und Maus zu verkaufen. Das ist in etwa dasselbe, als würden Sie ein Auto ohne Lenkrad kaufen.

### 9.1.2 Externe Erweiterungen

Die eben beschriebenen Erweiterungen faßt man meistens unter dem Begriff »Peripherie« zusammen. Mit solch »peripheren« oder »externen« Erweiterungen kann man nahezu alle Computer problemlos ausstatten. Meist genügt es, sich das entsprechende Zusatzgerät zu kaufen und es mit einem Kabel an den Computer anzuschließen.

Anschlüsse für diesen Zweck – sogenannte »Schnittstellen« – besitzt der Amiga 500 reichlich. An seiner Rückseite befinden sich vier mehrpolige Schnittstellen für Drucker, externe Diskettenlaufwerke und so weiter, sowie zwei Tonausgänge (damit Sie Stereo hören können). Daneben sind noch zwei weitere, kleinere Schnittstellen zu sehen. Sie dienen zum Anschluß einer Maus, eines Joysticks oder eines ähnlichen Steuergeräts. Alle diese Schnittstellen sind sehr einfach zu handhaben. Sie brauchen nur das jeweilige Gerät mit dem passenden Kabel anzuschließen. Dabei können Sie kaum Fehler machen, da die verschiedenen Schnittstellen fast alle verschiedene Steckerformen aufnehmen. Verwechslungen sind kaum möglich.

### 9.1.3 Interne Erweiterungen

Aus technischer Sicht problematischer als der externe Ausbau mit Peripheriegeräten sind da schon Erweiterungen, bei denen zusätzliche elektronische Bauteile angeschlossen werden müssen. Speichererweiterungen (mit RAM-Chips) fallen zum Beispiel in diese Kategorie. Solche elektronischen Erweiterungen können meist nicht über ein Kabel angeschlossen, sondern müssen in den Computer eingebaut werden. Wer mit dem Lötkolben und dem Oszilloskop gut umgehen kann und das Schalten und Walten der verschiedenen Bauteile im Innern seines Computers gut versteht, der schreckt natürlich vor solchen Erweiterungen nicht zurück.

Und tatsächlich blieb Computer-Besitzern in der Frühzeit des Personalcomputers oft nichts anderes übrig, als zum Lötkolben zu greifen. Heute würden wir diesen Menschenschlag wohl eher in die Kategorie Computer-Bastler einordnen. Die meisten Computer-Besitzer – mich eingeschlossen – haben mit Lötkolben nicht mehr allzuviel im Sinn. Der heutige Computer-Besitzer hat üblicherweise nicht mehr Interesse an den Details der Funktionsweise seines Computers als der Besitzer eines Farbfernsehers am Aufbau eines Fernsehempfängers.

### 9.1.4 Steckkarten

Deshalb bieten heute die meisten Hersteller Computer an, die auch ohne Lötkolben und tiefergehende Kenntnisse der digitalen Elektronik intern erweitert werden können. Hierzu hat sich bei den großen PCs das »Steckkarten-Prinzip« durchgesetzt, das als erster die Firma Apple im heute schon fast legendären Apple II einsetzte. Zur Erweiterung dienen in solchen Computern kleine Platinen (das sind die Plastikkarten, auf die die Chips, Widerstände und so weiter aufgelötet werden), eben die sogenannten »Steckkarten«. Für diese Platinen werden auf der Hauptplatine im Computer (die die wichtigsten Bauteile enthält) eine Art Steckdosen mit vielen Leitungen vorgesehen, die sogenannten »Steckplätze«. Sowohl das Format der Steckkarten wie auch die Eigenschaften der Leitungen über die die Steckkarten mit der Hauptplatine kommunizieren, werden exakt festgelegt. Meist bieten Steckplätze

Zugriff auf fast alle elektrischen Signale, die in dem jeweiligen Computer erzeugt werden (der Fachmann nennt die Gesamtheit dieser Signale den »System-Bus«). Man kann mit Steckkarten deshalb nahezu jede Erweiterung vornehmen, zu der man sonst einen Umbau mit Zange und Lötkolben vornehmen müßte.

Der Amiga 2000, sozusagen der große Bruder des Amiga 500, ist mit solchen Steckplätzen reichlich gesegnet. Deshalb besitzt er auch ein großes, zunächst einmal weitgehend leeres Gehäuse. Bei einem kompakten, preiswerten Computer wie dem Amiga 500 verbietet sich die Verwendung von Steckkarten schon aus Kostengründen.

### 9.1.5 Die Erweiterungsschnittstellen des Amiga 500

Damit aber auch der Amiga 500 mit Zusatzgeräten ausgestattet werden kann, die nicht über ein Kabel angeschlossen werden können, hat man sich bei Commodore eine ähnliche und deutlich weniger aufwendige Lösung einfallen lassen. Der Amiga 500 besitzt an seiner Unterseite sowie an der linken Kante des Gehäuses zwei sogenannte »Erweiterungsschnittstellen«, die beide mit Plastikschutzkappen verdeckt sind. Es handelt sich dabei um schmale Plastikleisten, die an der Ober- und Unterseite mit einer Vielzahl von Kontaktpunkten versehen sind. Entfernen Sie am besten einmal die beiden eben erwähnten Plastikabdeckungen, um sich diese Kontaktkleisten anzuschauen (falls Sie das noch nie gemacht haben). Dabei kann nicht tragisches passieren. Achten Sie aber darauf, daß Sie möglichst nicht mit den Fingern die Kontakte berühren.

An diese Erweiterungsschnittstellen können Geräte angeschlossen werden, die sehr schnell große Datenmengen mit dem Amiga austauschen müssen und/oder eine sehr weitgehende Kontrolle über die Amiga-Hardware benötigen. Der Fachmann nennt diese Art von Kontrollmöglichkeiten auch »Zugriff auf den Systembus des Amiga«.

Die beiden Erweiterungsschnittstellen dienen meist verschiedenen Zwecken. Die an der Unterseite des Gehäuses nimmt üblicherweise die Speichererweiterung A 501 (oder eine vergleichbare Erweiterung) auf, die dem Amiga 500 512 Kbyte mehr RAM und eine batteriegespeiste Hardware-Uhr spendiert. Die Erweiterungsschnittstelle an der linken Seite ist für alle anderen Geräte gedacht. Für diese Schnittstelle werden heutzutage unter anderem Festplatten und umfangreiche Speichererweiterungen, die nicht in das kleine Fach an der Unterseite hineinpassen, angeboten. Daneben gibt es aber auch noch eine Reihe anderer Zusatzgeräte für die verschiedensten Anwendungsbereiche, die ebenfalls über diese Schnittstelle mit dem Amiga 500 verbunden werden.

Es ist sogar möglich (und oft nötig), an dieser Schnittstelle mehr als ein Zusatzgerät zu verwenden. Dazu besitzen viele Zusatzgeräte ihrerseits wieder eine Erweiterungsschnittstelle am linken Rand des Gehäuses, die identisch wie die des Amiga 500 aufgebaut ist. Man sagt auch, daß diese Geräte die Erweiterungsschnittstelle des Amiga 500 »durchschleifen«. Beim Kauf eines Zusatzgeräts sollten Sie – falls Sie die Wahl haben – immer eines mit durchgeschleifter Erweiterungsschnittstelle nehmen, da Sie sich nur so alle Erweiterungsmöglichkeiten offen halten.

# 9.2 Bildschirme für den Amiga 500

Nach dieser kleinen Exkursion in die Theorie des Computerausbaus wende ich mich nun erst einmal wieder dem zu, was man auch von außen sehen kann: den externen Erweiterungsmöglichkeiten. Eine dieser Erweiterungsmöglichkeiten ist oder war für Sie wahrscheinlich schon vor dem Kauf des Amiga selbst wichtig: der Monitor oder Bildschirm. Denn obwohl viele Händler den Amiga im Paket-Angebot zusammen mit Bildschirm verkaufen, können Sie ihn auch »oben ohne« erhalten. Das ist besonders dann interessant für Sie, wenn Sie knapp bei Kasse sind oder besondere Ansprüche hinsichtlich des Bildschirms haben.

## 9.2.1 Wie das Bild aus dem Amiga kommt

Fast jeder Computer- oder Video-Monitor und mancher Fernseher läßt sich an den Amiga anschließen. Das Anschließen ist jedoch mit unterschiedlichen Schwierigkeiten verbunden, je nachdem, für welche Art Monitor Sie sich entschließen. Der Amiga kann nämlich – ohne weitere Zusatzeinrichtungen – nur an sogenannte RGB-Monitore angeschlossen werden. RGB-Monitore haben ihren Namen von den drei Grundfarben Rot, Grün und Blau, aus denen sich bei Verwendung einer Farbbildröhre die Farbe jedes einzelnen Bildpunktes zusammensetzt. Ein RGB-Monitor erhält über ein relativ dickes Kabel mit getrennten Adern für Rot, Grün und Blau die jeweiligen Rot-, Grün-, und Blauintensitäten für jeden Bildpunkt mitgeteilt. (Dieses dicke Kabel übermittelt allerdings auch noch weitere Informationen, so zum Beispiel die sogenannten VSYNC- und HSYNC-Signale, und enthält deshalb meist mehr als nur drei Adern.)

Videomonitore und Videorecorder erhalten hingegen ihre Bildinformationen üblicherweise durch ein sogenanntes FBAS- oder Composite-Signal, das mit nur zwei Adern übermittelt werden kann und alle nötigen Informationen für farbige Bilder in einem einzigen raffiniert modulierten Signal enthält. Dabei können die Informationen für Rot, Grün und Blau (und die restlichen für den Bildaufbau benötigten Signale) aber nicht so gut voneinander getrennt werden, wie beim RGB-Prinzip. Ein Monitor, der ein Bild aus einem FBAS-Signal darstellt, ist deshalb meist nicht so brilliant und trennt die verschiedenfarbigen Flächen im Bild nicht so klar voneinander wie ein RGB-Monitor. Das Bild wirkt deshalb oft unscharf und unruhig.

Die höhere Bildqualität, die nur mit RGB-Monitoren erzielt werden kann, ist Ursache dafür, daß der Amiga 500 zwar einen RGB-Ausgang besitzt, aber keinen Videoausgang, über den man farbige Bilder direkt an einen Videorecorder oder Videomonitor weiterleiten könnte. Lediglich ein Schwarzweiß-Signal wird an der mit *Mono* beschrifteten Buchse an der Rückseite des Gehäuses zur Verfügung gestellt. Der Nachteil dieser Lösung dürfte aber auch klar sein: Wer ein farbiges Bild sehen möchte, benötigt einen RGB-Monitor. Und wer noch keinen hat, muß sich für teures Geld einen kaufen.

## 9.2.2 Der Standard: Modell 1084 S

Die meisten Amiga 500 dürften mit dem Monitor A 1081, A 1084 oder A 1084 S ausgerüstet sein; das aktuelle Modell ist der 1084 S, dessen Vorgänger der 1081 war. Es handelt sich hierbei um einen Videomonitor mittlerer Qualität, den die Firma Philips für Commodore fertigt. Dieser Monitor kann – über den RGB-Eingang – die volle Farbpalette (immerhin

mehr als 4000 Farben), zu der der Amiga fähig ist, problemlos darstellen und bietet eine hohe Bildschärfe bis in die Ecken des Bildes hinein.

Der A 1084 S ist gleichzeitig auch ein ungewöhnlich vielseitiger Monitor. Er besitzt nämlich auch einen FBAS-Eingang und einen S-VHS-Eingang mit Cinch- beziehungsweise SCART-Steckern und kann deshalb auch als Videomonitor verwendet werden. Zudem besitzt er auch einen Audio-Eingang und einen eingebauten Lautsprecher. Dieser Lautsprecher ist wichtiger, als Sie vielleicht denken werden; der Amiga 500 besitzt nämlich keinen eingebauten Lautsprecher. Wenn Sie einen Monitor ohne Audio-Eingang verwenden, müssen Sie deshalb extra einen kleinen Lautsprecher an die Audio-Ausgänge des Amiga anschließen, wenn Sie mehr von ihm hören wollen, als das gelegentliche Schnarren der Diskettenlaufwerke.

Alles in allem ist der A 1084 S bestimmt kein schlechter Kauf – vor allem, wenn Sie ihn zu einem günstigen Preis zusammen mit dem Amiga erworben haben. Es gibt jedoch noch preiswertere Lösungen, ein Bild aus dem Amiga zu bekommen, wie Sie gleich sehen werden. Und auch in manchen anderen Fällen dürfte der 1084 nicht die ideale Lösung sein. Er ist nämlich keineswegs ein High-End-Gerät. Für den, der höhere Ansprüche an die Bildqualität stellt, können andere Alternativen interessant sein.

### 9.2.3 Für Sparsame: Anschluß eines Fernsehers

Zunächst aber zu einer sehr preiswerten Alternative. Sie können Ihren Amiga 500 nämlich relativ einfach an einen Fernseher anschließen. Dazu brauchen Sie lediglich die mit *Mono* beschriftete Buchse an seiner Rückseite über ein geeignetes Kabel mit dem Videoeingang Ihres Fernsehers verbinden. Diese Buchse liefert aber lediglich ein sogenanntes »BAS-Signal« ohne Farbinformation, weshalb Sie sich in diesem Fall mit einem Schwarzweiß-Bild begnügen müssen (in dem alle Farbtöne durch verschiedene Grautöne simuliert werden). Genügt Ihnen das nicht, können Sie dem Amiga aber auch relativ leicht »beibringen«, ein FBAS-Signal mit Farbinformationen zu erzeugen. Die preiswerteste Version der Kombination Amiga-Farbfernseher stellt der Adapter A 520 dar. Es handelt sich dabei um ein kleines Kästchen, das auf den RGB-Ausgang an der Rückseite des Amiga gesteckt wird. Auch dieses Kästchen hat natürlich eine Rückseite, auf der sich zwei Buchsen befinden, die ein FBAS- und ein HF-Signal liefern.

Wenn Sie einen Fernseher ohne Video-Eingang besitzen, können Sie mit dem HF-Signal an den Antenneneingang Ihres Fernsehers gehen. So können Sie Amiga-Bilder auf jedem (!) Fernseher betrachten. Besonders gut sind diese Bilder allerdings nicht – besonders im Vergleich mit einem RGB-Monitor. Nach längeren Computersitzungen werden Ihnen mit Sicherheit die Augen tränen und auch Kopfschmerzen werden nicht lange auf sich warten lassen. Textverarbeitung sollten Sie so zum Beispiel nicht betreiben. Ein etwas besseres Bild erhalten Sie schon, wenn Sie einen Fernseher mit Video-Eingang besitzen. Dann können Sie das FBAS-Signal des A-520-Adapters verwenden und erhalten damit ein deutlich besseres Bild als über den Antennen-Eingang.

Beide Lösungen sind aber wirklich nur etwas für sehr Sparsame oder für Amiga-Anwender, die (auch) einen Videorecorder an ihren Computer anschließen wollen. Alle auf dem Markt erhältlichen Videorecorder (außer professionellen Studiogeräten) können mit dem vom Amiga erzeugten RGB-Signal nämlich ohne den A 520 herzlich wenig anfangen.

Wenn Sie allerdings einen guten Fernseher mit SCART-Eingang besitzen, steht Ihnen eventuell noch eine weitere Möglichkeit offen. Viele dieser Fernseher können nämlich durchaus ein RGB-Signal vertragen und liefern dann ein wesentlich besseres Bild als wenn Sie das vom A 520 erzeugte FBAS-Signal verwenden würden. Sie benötigen – wenn Sie einen solchen Fernseher besitzen – lediglich ein geeignetes Verbindungskabel, daß den RGB-Ausgang des Amiga mit dem RGB-geeigneten Eingang Ihres Fernsehers verbindet. Dieses Kabel sollten Sie sich bei jedem Amiga-Händler besorgen können. Es ist genau das Kabel, über das auch der A 1084S mit dem Amiga verbunden wird.

Ob Sie Ihren Fernseher auf diese Art und Weise als Computer-Monitor mißbrauchen können, sollte sich der Bedienungsanleitung entnehmen lassen. Im Zweifelsfall fragen Sie Ihren Fernsehhändler. Tatsache ist, daß die meisten modernen Fernseher in der Preisklasse oberhalb von 2500,– DM (und auch viele darunter) mindestens einen RGB-geeigneten SCART-Eingang besitzen. Oft ist aber nur einer von zwei vorhanden SCART-Eingängen RGB-geeignet. Achten Sie also darauf, daß Sie den richtigen erwischen.

### 9.2.4 Andere RGB-Monitore

Wenn Sie, bevor Sie sich den Amiga gekauft haben, schon einen anderen Computer besessen haben, steht bei Ihnen vielleicht auch schon ein anderer Farbmonitor herum. Was läge in diesem Fall näher, als den schon vorhandenen Monitor an den Amiga anzuschließen – zumindest dann, wenn Sie sparen wollen (oder müssen) und nicht gleichzeitig an beiden Computern arbeiten wollen. Gegen diese Überlegung ist auch grundsätzlich nichts einzuwenden. Sie müssen allerdings einige Punkte beachten.

Der Amiga erzeugt nämlich ein »analoges RGB-Signal« und kann insgesamt 4096 verschiedene Farben auf einem geeigneten Monitor darstellen. Viele RGB-Monitore – zum Beispiel nahezu alle preiswerten RGB-Monitore für IBM-kompatible Computer – sind nur für die Wiedergabe von 8 (oder höchstens 16) Farben ausgelegt. Sie benötigen dazu ein »digitales RGB-Signal« (wobei die Spannungen des Signals im sogenannten »TTL-Bereich« liegen). Wenn Sie einen solchen Monitor mit einem Kabel, das Sie speziell dazu anfertigen müssen, an Ihren Amiga anschließen, gibt das im besten Fall ein relativ schlechtes Bild. In vielen Fällen wird das Bild aber so stark gestört sein, daß eine Benutzung des Computers nahezu unmöglich ist. Es gibt allerdings auch RGB-Monitore, die Sie sowohl an einen IBM-kompatiblen (und andere) Computer wie auch an den Amiga anschließen können.

### 9.2.5 Eine Klasse für sich: MultiSync-Monitore

Immer beliebter werden in letzter Zeit nämlich die sogenannten »MultiSync-« oder »MultiScan-Monitore«. MultiSync-Monitore – für die man im Augenblick je nach Qualität und Größe zwischen 1200,– und 2500,– DM bezahlen muß – vertragen nicht nur eine Vielzahl verschiedener Bildsignal-Formate, sondern passen sich auch automatisch an das gerade empfangene Signal an. Wenn Sie das Kabel eines solchen Monitors aus einem IBM-Computer herausziehen und in einen Amiga stecken, brauchen Sie höchstens noch einen Schalter umzulegen und können schon mit Ihrer Arbeit fortfahren. Einige MultiSync-Monitore lassen sich zudem auch als – ausgesprochen hochwertige – Video-Monitore verwenden.

Einen MultiSync-Monitor kann man aber nicht nur wegen dieser Flexibilität jedem Amiga-Besitzer wärmstens empfehlen. Sein Preis liegt zwar deutlich über dem des A 1084 S, dafür bietet er aber auch ein deutlich besseres Bild. Wer viel und lange vor seinem Amiga sitzt, wird das zu schätzen wissen.

## 9.3 Schwarz – oder bunt – auf weiß: Drucker

Ähnlich vielfältig wie bei den Monitoren sehen die Wahlmöglichkeiten auch bei den Druckern aus. Bei den Druckern kommt aber noch erschwerend hinzu, daß es nicht nur Modelle mit unterschiedlichen Preisen und Druckqualitäten, sondern auch eine ganze Reihe unterschiedlicher Arbeitsprinzipien gibt, die alle ihre Vor- und Nachteile haben.

### 9.3.1 Die verschiedenen Druckprinzipien

Die meisten der heute an Computern verwendeten Drucker gehören in die Kategorie der »Matrix-Drucker«. Matrix-Drucker setzten jeden Buchstaben – und jedes Bild – aus einzelnen Punkten zusammen. Zur Darstellung eines Buchstabens wird typischerweise eine rechteckige Matrix aus 8 * 8, 9 * 8 oder 24 * 18 Punkten verwendet. Je mehr und je kleinere Punkte verwendet werden, um einen Buchstaben darzustellen, desto schärfer und klarer sieht er aus.

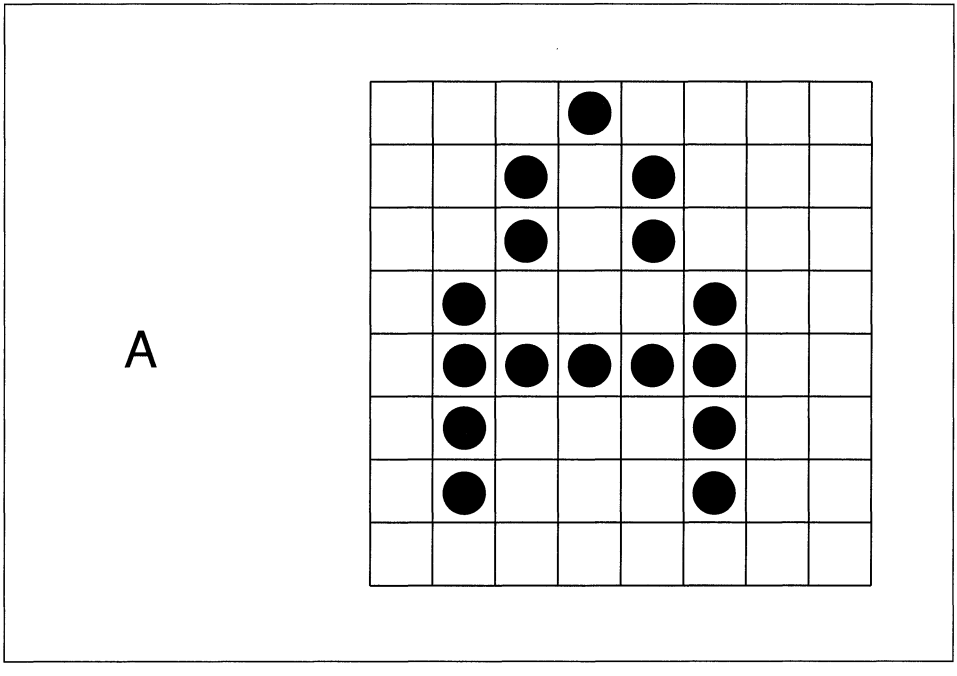



***Bild 9.1:*** *Das Prinzip des Matrixdruckers*

Bei den Matrixdruckern gibt es aber erhebliche Unterschiede, wie die Punkte, aus denen ein Buchstabe besteht, auf das Papier kommen. Die gebräuchlichste Methode verwenden die Nadeldrucker. Bei diesen schlagen feine Stahlnadeln, von denen sich 8, 9 oder 24 im sogenannten Druckkopf befinden, auf ein mit Tinte getränktes Gewebeband, das dicht über dem Papier vorbeigeführt wird. Jeder Schlag der Nadel ergibt so einen farbigen – meist schwarzen – Punkt auf dem Papier. Gleichzeitig erzeugt dieser Schlag aber auch ein Geräusch, eine Art kleinen Knall. Und da bei Nadeldruckern die Nadeln in sehr schneller Folge auf das Papier schlagen, wird aus den vielen kleinen Anschlägen ein zusammenhängender »sägender« Ton, der charakteristisch für diese Art von Druckern ist. Die recht brutale Art und Weise, in der Nadeldrucker die Tinte zu Papier bringen, hat neben dem Nachteil hoher Geräuschentwicklung aber auch einen entscheidenden Vorteil: Nadeldrucker sind die einzigen Drucker, die Durchschläge erzeugen können. Alle im folgenden noch beschriebenen Druckerarten können keine Durchschläge anfertigen und müssen statt dessen jede Seite mehrfach drucken.

Eine ganz andere (leisere) Möglichkeit, Punkte auf das Papier zu bringen, als die Nadeldrukker verwenden die Thermotransfer-Drucker. Sie besitzen anstatt der kleinen Stahlnadeln eine Reihe von Heizelementen, die sehr schnell an- und ausgeschaltet werden können. Diese Heizelemente werden dicht hinter einem Plastikband hergeführt, an dessen Vorderseite sich eine farbgetränkte Wachsschicht befindet. Diese Wachsschicht befindet sich dicht über der Oberfläche des zu bedruckenden Papiers. Wenn sich nun ein Heizelement erwärmt, schmilzt das Wachs und lagert sich zusammen mit der darin enthaltenen Farbe auf dem kühleren Papier ab. Auch so kann man also Farbe zu Papier bringen. Thermotransfer-Drucker sind im Gegensatz zu Nadeldruckern übrigens meist sehr leise (und etwas langsamer).

Hochwertige Mehrfarbdrucker verwenden oft das Thermotransfer-Verfahren. Das farbige Wachs ergibt auf dem Papier eine leicht glänzende Schicht mit sehr brillianten Farben. Alle anderen Druckverfahren ergeben meist deutlich trübere, blassere Farbausdrucke.

Ähnlich leise wie die Thermotransfer-Drucker sind Tintenstrahl-Drucker. Ihr Druckkopf enthält eine Reihe kleiner Düsen, durch die flüssige Tinte auf das Papier »geschossen« wird. Diese Düsen arbeiten so schnell und präzise, daß sie winzige Tropfen erzeugen können. Jeder Tintentropfen ergibt auf dem Papier dann einen Punkt. Tintenstrahl-Drucker sind mindestens so schnell wie Nadeldrucker und so leise wie Thermotransfer-Drucker. Sie ergeben allerdings ein nicht ganz so klares Druckbild wie die beiden Konkurrenten und es gibt sie erst seit relativ kurzer Zeit zu akzeptablen Preisen. Hauptproblem der Tintenstrahl-Drucker ist es, daß die Tinte auf dem Papier sehr schnell trocknen muß, vorher aber nicht so schnell trocknen darf, da dadurch die Düsen verstopft würden. Dieses Problem ist erst vor etwa drei bis vier Jahren gelöst worden.

Etwa genausolange gibt es die sogenannten Laserdrucker, die in gewisser Weise auch zu den Matrixdruckern gehören, meist aber als »Klasse für sich« betrachtet werden. Laserdrucker setzen Buchstaben ebenfalls aus Punkten zusammen. Diese Punkte sind aber so klein (ihr Durchmesser liegt unter einem Zehntelmillimeter), daß man sie mit bloßem Auge kaum noch unterscheiden kann. Um das Verfahren des Laserdruckers genau zu beschreiben, müßte hier etwas zu weit ausgeholt werden. Vielleicht genügt es aber auch, zu sagen, daß ein Laserdrucker im wesentlichen wie ein Kopierer funktioniert. Die Vorlage und das Blitzlicht, mit dem in einem normalen Kopierer die Kopie gemacht wird, wird in einem Laserdrucker nur durch einen sehr präzise gesteuerten Laserstrahl ersetzt.

Laserdrucker kommen von der Druckqualität her dem Zeitungsdruck nahe, sind sehr schnell (bis zu 8 oder 12 Seiten pro Minute) und leise. Ihr Hauptnachteil ist der verhältnismäßig hohe Preis (zwischen 5 000,– und 15 000,– DM).

Ein ganz anderes – weitaus herkömmlicheres – Verfahren als die eben beschriebenen Matrixdrucker verwenden die Typenraddrucker. Das Typenrad, von dem sie ihren Namen haben, enthält alle Buchstaben und Sonderzeichen des Alphabets als Metallrelief am Ende dünner Metall- oder Plastik-Stäbe, die kranzförmig um ein zentrales Loch angeordnet sind. Jeder dieser Stäbe sieht etwa so aus wie ein Typenhebel in einer ganz normalen mechanischen Schreibmaschine. Auch ansonsten funktionieren Typenraddrucker fast genauso wie Schreibmaschinen. Um einen Buchstaben zu Papier zu bringen, wird das Ende des entsprechenden Stabs (mit dem Metallrelief des Buchstabens daran) durch ein Farbband gegen das Papier geschlagen. Mit einem guten Farbband ergibt dies ein hervorragendes Druckbild, das selbst das eines Laserdruckers noch übertrifft – und macht einen Höllenlärm.

### 9.3.2 Text und Bild

Nun sollten Sie einen recht guten Überblick der verschiedenen Arten von Druckern haben, die Sie an Ihren Amiga 2000 anschließen können. Welche dieser Arten ist aber nun für Sie die ideale? Commodore selbst bietet zwar auch einige Matrixdrucker für den Amiga an (unter anderem den farbfähigen MPS 1500C). Daneben können Sie an den Amiga aber auch nahezu alle Computerdrucker anschließen, die im Handel angeboten werden. Die Software des Amiga unterstützt nämlich eine ungewöhnlich breite Palette von Druckern. Zwei wichtige Gesichtspunkte müssen Sie aber bei der Auswahl des Druckers beachten.

Der erste davon ist die Grafikfähigkeit. Die Software des Amiga ist stark grafikorientiert. Auf dem Bildschirm erscheinen ja nicht nur Zahlen und Buchstaben, sondern meist auch Symbole, Piktogramme, Skizzen und Bilder. Und wenn Sie selbst mit Mal- oder Zeichenprogrammen arbeiten, werden noch weitaus mehr Bilder und Zeichnungen am Bildschirm erscheinen. Wenn Sie diese auch auf Papier sehen wollen, brauchen Sie einen grafikfähigen Drucker. Fast alle heutzutage angebotenen Matrixdrucker (auch die, die Commodore für den Amiga anbietet) sind grafikfähig. Typenraddrucker oder zu Druckern umfunktionierte Schreibmaschinen sind jedoch nicht grafikfähig. Auch einige preiswerte Laserdrucker sind nicht grafikfähig, während die Spitzenmodelle ganz hervorragende Grafikfähigkeiten bieten. Wenn Sie nicht absolut sicher sind, daß Sie immer nur Texte ausdrucken wollen, sollten Sie auf keinen Fall einen solchen nicht-grafikfähigen Drucker kaufen.

### 9.3.3 Schwarzweiß oder bunt?

Der zweite wichtige Gesichtspunkt ist die Farbfähigkeit des Druckers. Der Amiga kann auf einem guten Monitor bis zu 4096 verschiedene Farben darstellen, während die meisten Drucker nur in einer Farbe drucken können. Wer die Bildschirmfarben auch zu Papier bringen will, muß also einen Mehrfarbdrucker kaufen. Diese Drucker haben jedoch drei entscheidende Nachteile: sie sind entweder recht teuer oder bringen nur eine dürftige Qualität zu Papier und sind außerdem meist noch recht langsam. Auch die laufenden Kosten – also die Kosten pro ausgedruckte Seite – sind bei Mehrfarbdruckern deutlich höher als bei einfarbigen Matrixdruckern.

Falls Sie sich aber auch von diesen Tatsachen nicht abschrecken lassen, berücksichtigen Sie bitte auch noch, daß heute kein zu einem Preis unter etwa 12 000,– DM erhältlicher Drucker auch nur in etwa das zu Papier bringen kann, was Sie auf dem Bildschirm sehen. Drucker mit akzeptabler Qualität wie zum Beispiel der Xerox 4020 liegen bei etwa 5 000,– DM. Diese Drucker können dann schon einen Großteil (aber nicht alle) der Farben zu Papier bringen, die der Amiga darstellen kann. Die meisten Mehrfarbdrucker verfügen allerdings nur über etwa ein Dutzend unterschiedlicher Farben; ein Bruchteil dessen, was der Bildschirm zuwege bringt. Und den gedruckten Farben mangelt es meist auch an Leuchtkraft und Brillanz. Für den Privatbedarf kann ein Mehrfarbdrucker schon akzeptable Ergebnisse liefern. Für den professionellen Einsatz sind seine Erzeugnisse aber meist zu dürftig und entstammen gar zu deutlich dem Computer.

Commodore bietet zur Zeit zwei farbfähige Drucker an. Der MPS 2000C ist ein recht robuster farbfähiger und natürlich grafikfähiger Drucker, der zugleich auch als schneller Textdrucker dienen kann. Der MPS 1500C ist etwas leichter und kompakter gebaut, besitzt ansonsten aber in etwa dieselben Farbfähigkeiten. (Er ist zudem auch preiswerter.) Daneben gibt es eine ganze Reihe farbfähiger Drucker anderer Hersteller. Fast jeder wichtige Druckerhersteller (unter anderem Epson, NEC, C. Itoh, Apple, HP, Xerox und Fujitsu) hat einen oder mehrere Mehrfarbdrucker im Angebot.

## 9.4 Diskettenlaufwerke und Festplatten

Der Amiga 500 ist als Grundmodell recht spärlich mit »externer Speicherkapazität« ausgestattet. Falls Sie sich fragen sollten, was »externe Speicherkapazität« ist: darunter versteht man Speichermedien wie Disketten oder Festplatten, die viel langsamer sind als der Hauptspeicher (RAM), deren Inhalt aber auch nach dem Ausschalten des Computers erhalten bleibt. Im Amiga 500 ist standardmäßig ein 3,5-Zoll-Diskettenlaufwerk mit einer Kapazität von 880 Kbyte eingebaut. Das scheint auf den ersten Blick gar nicht einmal so wenig zu sein. Sobald Sie aber größere Datenmengen (Datenbanken, viele Bilder und ähnliches) zu verwalten haben, werden Sie zum Diskettenjockey. Anders gesagt: Eine Ihrer Hauptbeschäftigungen wird dann das Austauschen von Disketten sein. Auch das Kopieren von Disketten ist mit nur einem Laufwerk eine recht mühselige Angelegenheit.

### 9.4.1 Welche Alternativen zur Wahl stehen

Wenn Sie die externe Speicherkapazität Ihres Amiga erweitern wollen, stehen Ihnen – wie üblich – eine Reihe von Alternativen zur Wahl. Sie können zum Beispiel ein oder zwei zusätzliche Diskettenlaufwerke ergänzen (mehr Diskettenlaufwerke sind wenig sinnvoll). Diese können nicht in den Amiga 500 eingebaut werden, sondern müssen neben ihn gestellt werden. Die meisten externen Diskettenlaufwerke verwenden dieselben 3,5-Zoll-Disketten wie das interne Laufwerk. Sie können allerdings auch Laufwerke für 5,25-Zoll-Disketten ergänzen. 5,25-Zoll-Disketten und -Laufwerke sind zwar billiger, haben aber nur etwa die halbe Kapazität von 3,5-Zoll-Disketten, sind empfindlicher und umständlicher in der Handhabung. Sie sind für den Amiga 500 nicht zu empfehlen.

Falls Sie sich nun fragen sollten, wieso diese Laufwerke überhaupt für den Amiga angeboten werden: diese Laufwerke ergeben beim Amiga 2000 einen Sinn. Dieser kann nämlich auch

Programme verarbeiten, die eigentlich für IBM-kompatible PCs gedacht sind. Und diese PCs arbeiten immer noch sehr viel mit den größeren 5,25-Zoll-Disketten.

Disketten sind aber nicht das einzige externe Speichermedium. Wenn Sie sehr große Mengen von Daten zu verwalten haben und viel Wert auf eine hohe Verarbeitungsgeschwindigkeit legen, sollten Sie sich die Anschaffung einer Festplatte überlegen. Auf Festplatten werden Daten nach einem ähnlichen Prinzip wie auf Disketten gespeichert. Sie besitzen jedoch eine wesentlich größere Kapazität (mindestens 20 Mbyte, was etwas mehr als 22 Disketten entspricht) und sind viel schneller als Disketten. Festplatten werden – wie ihr Name schon sagt – dafür aber fest installiert und können nicht wie Disketten ausgetauscht werden. (Wenn Ihre 20-Mbyte-Festplatte voll ist, können Sie sie also nicht einfach aus dem Laufwerk nehmen und eine neue leere Festplatte einlegen.)

### 9.4.2 Externe Diskettenlaufwerke

Die einfachste und preiswerteste Art, sich mehr externe Speicherkapazität zu verschaffen, ist ein externes Diskettenlaufwerk. Commodore bietet solche externen Diskettenlaufwerke unter den Bezeichnungen 1010 (3,5-Zoll-Laufwerk) und 1020 (5,25-Zoll-Laufwerk) an. (Wie schon gesagt, ist die Anschaffung des 5,25-Zoll-Laufwerks 1020 für den Amiga 500 aber wenig sinnvoll.) Um ein solches Laufwerk zu benutzen, brauchen Sie nur Ihren Amiga abschalten, das entsprechende Kabel an den mit *Disk Drive* beschrifteten Diskettenanschluß an der Rückseite der Systemeinheit stecken und den Amiga wieder einschalten. Wenn Sie dann eine Diskette in dieses Laufwerk einlegen, erscheint sie ohne weitere Umstände auf der Workbench.

Ein 1010-Laufwerk besitzt übrigens selbst wieder einen Diskettenanschluß, an den Sie weitere Diskettenlaufwerke anschließen können. Dies geht jedoch nur dann, wenn diese weiteren Laufwerke eine eigene Stromversorgung besitzen (über die das 1010 nicht verfügt). Solche Laufwerke mit eigener Stromversorgung bietet Commodore nicht an. Bei verschiedenen Zubehörfirmen sind sie aber zu bekommen. Sie sind oft meist etwas preiswerter als das Commodore-Laufwerk oder vereinen gar zwei Laufwerke in einem gemeinsamen Gehäuse.

Sie brauchen natürlich auch nicht mit dem 1010-Laufwerk beginnen, sondern können auch als erstes externes Laufwerk ein »Fremdfabrikat« anschließen. Achten Sie aber darauf, daß der Hersteller Ihnen garantiert, daß das Laufwerk ohne weitere Einstellungen oder gar Lötarbeiten am Amiga 500 funktioniert.

### 9.4.3 Festplatten

Die Installation einer Festplatte ist im Prinzip ähnlich einfach wie das Anschließen eines externen Diskettenlaufwerks. Da der Amiga 500 von Hause aus aber nicht mit einer Schnittstelle ausgerüstet ist, die für den Anschluß einer Festplatte geeignet wäre, müssen Sie ihn zunächst einmal mit einer solchen Schnittstelle versehen. Dazu wird in die Erweiterungsschnittstelle an der linken Seite ein kleines Kästchen gesteckt, das etwas Steuerelektronik, den sogenannten »Kontroller«, enthält und an seiner Außenseite eine Anschlußmöglichkeit für die Festplatte besitzt. Die Festplatte selbst wird in einem separaten Gehäuse mit eigenenr Stromversorgung untergebracht und über ein Kabel mit dem Kontroller verbunden.

Aus Vereinfachungsgründen bieten einige Hersteller auch Festplatten an, die die Steuerelektronik gleich enthalten und deren Gehäuse mit einem Stecker für die Erweiterungsschnittstelle des Amiga versehen ist. Eine solche Festplatte kann direkt (ohne weiteres Kabel) in die linke Seite des Amiga 500 gesteckt werden. Sie besitzt aber auch eine eigene Stromversorgung. Das Netzteil des Amiga ist nicht leistungsfähig genug, neben dem Amiga selbst auch noch eine Festplatte zu versorgen.

Für welche Art von Festplatte Sie sich entscheiden, eine mit integriertem oder eine mit separatem Controller, ist im Grunde Geschmackssache. Beides hat Vor- und Nachteile. Viel wichtiger ist, ob die Festplatte »autobootfähig« ist. Von einer autobootfähigen Festplatte kann der Amiga starten – sie müssen nach dem Einschalten also keine Diskette mehr einlegen. Viele ältere Festplatten können das nicht. Wenn Sie eine solche Festplatte besitzen, müssen Sie – obwohl Sie jetzt stolzer Besitzer einer schnellen Platte sind – mit der langsamen Diskette starten. Das dauert wesentlich länger als ein Start von der Festplatte. Zudem muß die Start-Diskette dann speziell für die Zusammenarbeit mit der Festplatte vorbereitet werden.

Darauf und auf die Unterschiede zwischen den verschiedenen Festplattenmodellen möchte ich aber an dieser Stelle aber nicht näher eingehen, da sich das übernächste Kapitel ausschließlich und ausführlich mit Festplatten, ihrer Installation und ihrer Handhabung beschäftigen wird.

## 9.5 Speichererweiterungen

Genau wie beim »externen« Speicher ist der Amiga auch in Hinsicht auf seinen internen Arbeitsspeicher etwas spartanisch ausgestattet. Er wird im Augenblick standardmäßig mit 512 Kbyte RAM ausgeliefert. Das reicht für viele Anwendungen sicherlich zunächst vollkommen aus. Wenn Sie aber anfangen, wirklich anspruchsvolle Programme zu nutzen oder die RAM-Disk zu verwenden, wird es bald eng. Dann muß eine Speichererweiterung her.

### 9.5.1 Die einfachste Speichererweiterung A 501

Die erste Speichererweiterung, die sich jeder Besitzer eines Amiga 500 zulegen sollte, ist Commodore's A 501 oder ein vergleichbares Produkt eines anderen Herstellers (das meist etwas billiger ist). Diese Speichererweiterung besteht aus einer kleinen Plastikkarte mit ein paar Chips darauf und wird in den Erweiterungssteckplatz an der Unterseite des Amiga gesteckt, der sich hinter einer leicht herausnehmbaren Plastikabdeckung verbirgt. Diese Abdeckung können (und sollten) Sie wieder anbringen, sobald die Speichererweiterungskarte installiert ist. Sie wird dadurch geschützt und kann nicht mehr versehentlich herausrutschen.

Die Speichererweiterung A 501 erweitert den RAM-Speicher des Amiga von 512 Kbyte um weitere 512 Kbyte auf 1 Mbyte. Das bedeutet mehr als eine Verdoppelung des Ihnen zur Verfügung stehenden freien Speichers, da ein Teil des RAM-Speichers immer vom Betriebssystem des Amiga beansprucht wird. Darüber hinaus erhalten Sie mit der A 501 auch noch eine batteriegespeiste Uhr, die dafür sorgt, daß Ihr Amiga 500 nie wider die Uhrzeit

vergißt, wenn sie einmal richtig eingestellt wurde. (Am einfachsten geht das mit dem Programm *Preferences*.) Die in dieser Uhr gültige Zeit fragt der Amiga bei jedem Neustart automatisch ab und verwendet die Uhrzeit danach bei allen Anzeigen und auch beim Versehen von Dateien mit einem Modifikationsdatum. Dazu wird der Befehl SETCLOCK verwendet, der in der Kommandofolge Startup-Sequence aufgerufen wird. (Siehe auch Kapitel 8.)

Mit 1 Mbyte RAM-Speicher, wie es Ihnen die A 501 oder eine vergleichbare Speichererweiterung verschaffen, sollten Sie eine ganze Zeitlang auskommen können – sofern Sie nicht extrem speicherhungrige Programme benutzen. Damit ist es unter anderem auch problemlos möglich, die RAM-Disk zu benutzen und mehrere Programme gleichzeitig im Speicher zu halten. Wenn Sie allerdings anspruchsvolle Grafikprogramme benutzen oder gerne eine sehr große RAM-Disk verwenden wollen, wird auch 1 Mbyte RAM-Speicher bald zu wenig sein.

### 9.5.2 Externe Speicherkarten

Das ist aber nicht weiter tragisch – zumindest dann, wenn Ihr Portemonnaie immer noch gut gefüllt ist. Mit der Erweiterungsschnittstelle an der Unterseite des Amiga 500 können Sie zwar den RAM-Speicher jetzt nicht noch mehr erweitern. Hier mehr RAM-Chips unterzubringen dürfte auch wegen der Größe des zur Verfügungs stehenden Hohlsraums problematisch sein.

Es gibt aber auch Speichererweiterungen für die Erweiterungsschnittstelle am linken Rand des Gehäuses. Platzprobleme gibt es hier keine und deshalb können Sie so richtig in die Vollen greifen. Von diversen Herstellern werden zum Beispiel 2-MByte-RAM-Erweiterungen angeboten, die Sie an dieser Schnittstelle einstecken können. Aber auch eine Erweiterung darüber hinaus ist theoretisch möglich (um 4, 6 oder 8 Mbyte). Sie sollten sich aber ernsthaft überlegen, ob Sie noch den richtigen Computer verwenden, wenn Sie Anwendungen benutzen, die so große Speichermengen benötigen. Das soll keine Kritik am Amiga 500 bedeuten. Es ist nur so, daß der Amiga 2000 wesentlich einfacher in solchen Dimensionen ausgebaut werden kann als der Amiga 500 und der Preisunterschied (prozentual) immer geringer wird, je mehr zusätzlichen Speicher und andere Erweiterungsgeräte sie zusätzlich zu ihrem Amiga 500 kaufen. Allzu viele Erweiterungen, die an der linken Seite kaskadenförmig aus dem Gehäuse des Amiga herausragen, dürften Ihren Arbeitsplatz auch bald wie ein Elektroniklabor aussehen lassen. Auch in ästhetischer Hinsicht ist der Amiga 2000, der viele Erweiterungen in seinem Gehäuse aufnehmen kann, dann die bessere Lösung.

Eine Festplatte und eine externe Speichererweiterung kann aber sicherlich auch der Amiga 500 vertragen und diese Arbeitskonfiguration ist auch bei vielen Benutzern anzutreffen. Das haben wohl auch die Hersteller von Festplatten gemerkt. Einige Festplattenmodelle (beziehungsweise die zugehörigen Kontrollergehäuse) bieten deshalb auch die Möglichkeit, auch eine RAM-Erweiterung vorzunehmen. Üblich ist es, zusammen mit der Festplatte eine RAM-Erweiterung von bis zu 2 Mbyte anzubieten. Commodore selbst bietet zum Beispiel eine Festplatte mit einer solchen integrierten RAM-Erweiterung an, das Modell A 590.

Sowohl bei den in einer Festplatte integrierten wie auch bei den eigenständigen RAM-Erweiterungen hat man oft auch die Möglichkeit, leere Platinen zu kaufen, die noch nicht oder nur zu einem Teil mit RAM-Chips bestückt sind. Der Grund dafür ist der hohe Preis

der RAM-Chips. Die Festplatte A 590 von Commodore selbst ist beispielsweise zunächst mit exakt 0 (Null) Kbyte RAM-Chips bestückt, kann aber vom Endanwender (also Ihnen) oder vom Händler mit bis zu 2 Mbyte Chips bestückt werden. Sie können mit einer solchen, zunächst nur teilweise bestückten Speichererweiterung Ihre RAM-Kapazität jederzeit an Ihre aktuellen Bedürfnisse und Ihren Kontostand anpassen und immer erst dann Speicher zulegen, wenn Sie ihn wirklich benötigen und ihn sich auch leisten können.

## 9.6 Erweiterungen für Video-Fans

Das Angebot an Erweiterungsmöglichkeiten für den Amiga 500 ist groß – nicht zuletzt deshalb, weil ein Großteil der Zusatzgeräte, die für den Amiga 1000 und 2000 entwickelt wurden, auch zusammen mit dem Amiga 500 verwendet werden kann. Zusammen stellen diese drei Modelle einen sehr großen Markt dar, der für viele Produzenten von Zusatzgeräten sehr interessant ist. Neben den schon beschriebenen Erweiterungsmöglichkeiten, die für viele, wenn nicht sogar die meisten Amiga-500-Besitzer interessant sind, möchte ich nun noch einige Ausbaumöglichkeiten vorstellen, die speziell für Anwender wichtig sind, die den Amiga im Video- oder Musik-Bereich einsetzen wollen. Den Anfang machen die Erweiterungsmöglichkeiten für Video-Fans.

### 9.6.1 Der Mediencomputer und das Fernsehen

Wußten Sie schon, daß einige Video- und Fernseh-Studios den Amiga einsetzen, um durchaus professionell gestaltete Eröffnungsbilder und kurze Trickfilme zu gestalten? Wenn nicht, ist diese Information vielleicht eine Anregung für Sie, selbst etwas auf diesem Gebiet tätig zu werden. Der Amiga hat nämlich in dieser Hinsicht einige Fähigkeiten zu bieten, die ihn aus der Masse der konventionellen PCs deutlich herausragen lassen. Seine Farbgrafik ist schnell und brilliant und kann bei geschickter Benutzung der Möglichkeiten schnell vergessen machen, daß Bilder aus einzelnen Punkten aufgebaut werden müssen und meist nur 32 verschiedene Farben enthalten können. Es gibt hervorragende Malprogramme (zum Beispiel DeluxePaint) und ebensogute Videoprogramme, Titelgeneratoren und Animationsprogramme, mit denen sich Titel und kurze Trickfilme schnell und einfach gestalten lassen. Entscheidend für viele Amiga-Anwender im Video-Bereich ist aber vor allem auch, daß leistungsfähigere Geräte, wie sie die großen Fernsehstudios verwenden, mindestens das Zehnfache eines Amiga kosten.

Ein Zeitlang hat Commodore ja auch versucht, den Amiga als »Medien-Computer« zu vermarkten, wobei vor allem seine grafischen Fähigkeiten herausgestellt wurden. Diese Marketingphase ist inzwischen allerdings vorbei, weil man bei Commodore offensichtlich bemerkt hat, daß es sich dabei nur um einen Nischenmarkt handelt. Gerade für den semiprofessionellen und Hobby-Bereich sind die grafischen Fähigkeiten des Amiga aber sicherlich nach wie vor ein wichtiges Kaufargument. Berücksichtigt man weiterhin die wachsende Verbreitung von Videorecordern und -kameras, wird eine Zusammenarbeit des Computers mit diesen Geräten zunehmend wichtiger. Und der Amiga hat hier wirklich einige Vorteile gegenüber den meisten anderen preiswerten PCs.

In dieser Hinsicht hat der Amiga aber zumindest auch einen entscheidenden Nachteil, der für einen »Medien-Computer« fast unverzeihlich ist: Das Grundgerät läßt sich nicht ohne

Erweiterungen an einen Fernseher oder Videorecorder anschließen – zumindest wenn man Wert auf Farbe legt. Der einzige für den Videorecorder geeignete Anschluß liefert nur ein Schwarzweißbild. Denn – wie weiter oben in diesem Kapitel schon beschrieben wurde – ist der Amiga für den Anschluß an einen RGB-Monitor konzipiert worden. Mit den nach den Grundfarben Rot, Grün und Blau getrennten Signalen, die am Bildschirmstecker zur Verfügung stehen, können die wenigsten Fernsehgeräte und Videorecorder etwas anfangen. Diese erwarten die Bildinformationen meist in gemischter Form (als sogenanntes »Composite-« oder »FBAS-Signal«). Aufzeichnungsgeräte, die über einen RGB-Eingang verfügen, sind wirklich nur für Profis (oder solche die es werden wollen) interessant, da sie zwar sehr gut, aber auch sehr teuer sind und zudem mit den normalen Videokassetten vom »Videoverleih um die Ecke« nichts anzufangen wissen.

### 9.6.2 Die Verbindung zum Videorecorder

Wer seinen Amiga 500 an einen »normalen« Videorecorder oder Farbfernseher anschließen will, dem bleibt nichts anderes übrig, als ihm beizubringen, wie man ein FBAS-Signal erzeugt. Hierzu stehen im Prinzip zwei Alternativen zur Verfügung. Die preiswerteste und einfachste davon wurde oben schon vorgestellt: das kleine Zusatzgerät A 520 kann ohne Probleme auch an den Bildschirmausgang des Amiga gesteckt werden. Dieses kleine und preiswerte (weniger als 100,– DM) Gerät liefert dann ein Composite-Video-Signal für den Videoeingang und ein HF-Signal für den Antenneneingang eines Farbfernseher oder Videorecorders. Im Zweifelsfall sollte man übrigens – falls möglich – immer den Videoeingang und nicht den Antenneneingang verwenden, da dieser das bessere Bild liefert.

Der Anschluß eines Videorecorders über den A 520 hat allerdings einen entscheidenden Nachteil. Der RGB-Ausgang Ihres Amiga wird dadurch belegt. Wenn Sie selbst jetzt noch sehen wollen, welches Bild zum Videorekorder geht, müssen Sie einen Farbfernseher an den Videorecorder anschließen und sich auf dem Fernsehbildschirm das vom Amiga erzeugte Bild betrachten. Als Kontroll-Möglichkeit mag das auch ganz praktisch sein. Wenn Sie jedoch länger vor einem Fernsehbildschirm arbeiten, werden Ihnen sicherlich bald die Augen tränen. Das müssen Sie entweder akzeptieren oder den A-520-Adapter aus Ihrem Amiga herausziehen und wieder einen RGB-Monitor anschließen. Diese ewige Stöpselei wird natürlich mit der Zeit lästig und ist für professionelles oder semiprofessionelles Arbeiten unmöglich.

Für den etwas professionelleren Einsatz gibt es Zusatzgeräte, die im Prinzip eine ähnliche Leistung erbringen, wie der A-520-Adapter (ein RGB- in ein FBAS-Signal umwandeln) aber komfortabler sind und meist auch eine bessere Bildqualität bieten. In der Preisklasse zwischen 300,– und 400,– DM bieten diese »PAL-Encoder« oder »PAL-Modulatoren« üblicherweise neben dem FBAS-Ausgang auch noch einen S-VHS-Eingang, einen RGB-Anschluß zur Bildkontrolle auf dem RGB-Monitor und ein paar Regler zur Beeinflussung von Farbsättigung, Helligkeit und Kontrast.

Für den Anschluß eines Videorecorders an den Amiga benötigen Sie jedoch neben dem PAL-Modulator auch noch ein passendes Kabel. Leider ist auf diesem Gebiet die Standardisierung noch nicht soweit fortgeschritten, daß man hier eindeutig beschreiben könnte, wie ein solches Kabel auszusehen hätte. Es gibt BNC, Cinch, DIN, SCART und einige andere wenig gebräuchliche Stecker-Normen. Am besten gehen Sie mit den Unterlagen Ihres Videorecorders und des A 520 beziehungsweise der PAL-Karte zu einem Computer- oder

Video-Händler, der nicht nur weiß, wie ein Videorecorder aussieht, sondern auch, wo das heiße Ende eines Lötkolbens ist und bitten ihn, Ihnen ein passendes Kabel anzufertigen. Das kostet vielleicht 100,– oder 200,– DM, ist aber auf jeden Fall besser, als tagelang herumzuprobieren und sich dabei vielleicht vor Ärger ein Magengeschwür zu holen.

### 9.6.3 Video-Digitizer

Wenn es Ihnen dann endlich gelungen ist, Amiga und Videorecorder zu koppeln, stellt sich vielleicht die – nicht gerade unwichtige – Frage, welche Bilder Sie denn aufnehmen wollen. Als talentierter Computermaler setzen Sie sich vielleicht vor den Bildschirm und malen die wunderschönsten Bilder allein mit Maus und Tastatur. Dabei wird es vielleicht oft nötig sein, Bilder, die schon in gedruckter Form oder fotografiert vorliegen, am Bildschirm nachzuempfinden. Einfacher wäre es natürlich, wenn man diese Bilder in irgendeiner Form in den Amiga einlesen und dort mit einem Malprogramm nachbearbeiten könnte. Man könnte auch sagen, daß Sie nun vor dem gegenteiligen Problem stehen, wie in den letzten Abschnitten. Es geht nicht darum, wie Sie Bilder aus dem Computer herausbekommen, sondern darum, wie Sie sie hineinbekommen, ohne sie neu malen zu müssen.

In diesem Fall benötigen Sie einen sogenannten »Video-Digitizer« oder »Framegrabber«. Mit beiden Bezeichnungen meint man oft dasselbe, nämlich ein Zusatzgerät, das Bilder, die es über einen Videoeingang erhält, so »in den Amiga holt«, daß sie von Malprogrammen bearbeitet werden können. Woher diese Bilder kommen, ist dem Digitizer dabei egal. Sie können eine beliebige »Videoquelle«, also sowohl eine Videokamera wie natürlich auch einen Videorecorder oder einen separaten Fernsehtuner an einen Digitizer anschließen.

In dieser Hinsicht sind fast alle Video-Digitizer gleich. Ansonsten teilt man diese Geräte in zwei große Gruppen ein: schnelle und langsame. Schnelle (und meist auch teure) Video-Digitizer können 20 Bilder und mehr pro Sekunde »digitalisieren« also in das Computer-Format umwandeln. Das heißt, sie können die Bilder ungefähr so schnell in den Amiga holen, wie sie von der Videoquelle geliefert werden (25 Bilder pro Sekunde). Beachten Sie aber bitte, daß dabei sehr große Mengen von Daten anfallen. Wenn Sie diese Bilderflut aufzeichnen wollen, brauchen Sie jede Sekunde eine neue Diskette (bei 320 Zeilen mit 200 Punkten in maximal 32 verschiedenen Farben). Selbst die meisten Festplatten sind nach etwa einer halben Minute voll. Als Videorecorder eignet sich der Amiga also nicht. Zudem können die meisten Digitizer dieser Art die Daten auch gar nicht schnell genug an den Amiga liefern. Sie werden meisten an der parallelen Schnittstelle angeschlossen und diese erlaubt keine Datenübertragung mit so hohen Geschwindigkeiten.

Trotzdem haben solche schnellen Digitizer eine Daseinsberechtigung. Ihr Vorteil liegt darin, daß auch aus einem sich bewegenden Bild Schnappschüsse gemacht werden können. Würde der Digitizer länger als eine 25stel Sekunde für seine Arbeit benötigen, und das zu digitalisierende Bild würde sich in dieser Zeit ändern, würde es verschwommen oder verzerrt im Computer ankommen.

Wer keine Verwendung für einen schnellen Digitizer hat, weil er vor allem einzelne Bilder digitalisieren möchte, kann es etwas preiswerter haben. Langsame Digitizer können zwar höchstens 1 bis 5 Bilder pro Sekunde digitalisieren – manche benötigen pro Bild sogar einige Sekunden – sind dafür aber wesentlich preiswerter. Der bekannteste Digitizer dieser Art dürfte wohl DigiView sein, der mit geringem Aufwand sehr gute Bilder liefert – dafür aber auch sehr lange braucht. Bei DigiView nimmt eine einfache Schwarzweiß-Videokamera

nacheinander drei Bilder von einem Motiv (zum Beispiel einem Foto) auf, wobei jedes der drei Bilder durch einen anderen Filter hindurch aufgenommen wird. Dadurch erhält man praktisch mit einer billigen Kamera drei sogenannte »Farbauszüge« (ähnlich denen, die eine Druckerei anfertigen würde), aus denen im Amiga wieder ein Farbbild gemacht wird. Für die Digitalisierung einzelner Bilder – besonders dann, wenn sie nur gelegentlich einmal anfällt – ist ein solches Gerät fast eine ideale Lösung. Sie hat allerdings den Nachteil, daß es unmöglich ist, ein bewegtes Objekt oder auch nur einen ruhig dasitzenden Menschen aufzunehmen. Extrem langsame Digitizer wie DigiView sind nur für unbewegliche Objekte und Fotos oder Abbildungen geeignet.

### 9.6.4 GenLock – Bilder im Gleichschritt

Eine Anwendung, für die Digitizer sicherlich oft eingesetzt werden, ist ein mit einer Kamera aufgenommenes Bild etwas zu verändern oder zusätzliche Bildteile darüberzulegen und das veränderte Bild dann wieder zu zeigen oder aufzunehmen. Wenn das die Hauptanwendung ist, für die Sie meinen, einen Digitizer zu benötigen, brauchen Sie in Wirklichkeit vielleicht ein ganz anderes Gerät: ein sogenanntes »GenLock-Interface«. Das ist ein kleines Gerät, mit dem Sie zwei Video-Signale mischen können. Dabei werden aus zwei Bildern ein überlagertes, wobei das Bild der einen Videoquelle das Bild der anderen Videoquelle nur an den Stellen ersetzt, an denen es eine bestimmte Farbe hat. Das Prinzip ist ähnlich wie beim sogenannten »Bluebox-Effekt«, der im Fernsehen oft verwendet wird, um Personen vor einem ganz anderen Hintergrund erscheinen zu lassen, als dem, vor dem sie wirklich stehen. Wenn hinter dem (der) Tagesschau-Sprecher(in) zum Beispiel das Bild eines Politikers oder eine Landkarte erscheint, sitzt er (sie) in Wirklichkeit vor einer blauen Wand.

Ähnlich funktionieren ein GenLock-Interface für den Amiga. Es ersetzt alle Flächen am Bildschirm, die normalerweise in der Hintergrundfarbe erscheinen würden (auf der Workbench ist das meist ein tiefes Blau) durch das Bild aus einer externen Videoquelle. Sie können damit zum Beispiel einen Text im *Notepad* schreiben, während dahinter – statt des blauen Workbench-Bildschirms eine Dallas-Folge oder die Tagesschau abläuft. Das ist aber zugegebenermaßen eine Spielerei ohne viel Sinn und Zweck. Andererseits ermöglicht es Ihnen ein GenLock-Interface aber auch, »über« einen Videofilm einen Text (zum Beispiel einen Filmtitel) oder ein Symbol (zum Beispiel ein Firmenzeichen) zu legen. Oder Sie können auf dem Amiga gezeichnete Figuren zusammen mit echten Personen in einem selbstgedrehten Videofilm agieren lassen. Beachten Sie aber bitte, daß Sie, wenn Sie solche Filme nicht nur am Amiga-Monitor sehen wollen, zusätzlich zum reinen GenLock-Interface noch einen PAL-Modulator benötigen, um die kombinierten Bilder aufzeichnen zu können. Die meisten GenLock-Interfaces enthalten allerdings gleich einen PAL-Modulator.

Der Vorteil, den ein GenLock-Interface gegenüber einem schnellen Digitizer hat, mit dem die oben beschriebenen Effekte grundsätzlich auch möglich sind, ist die höhere Bildqualität. Wird ein Bild digitalisiert, geht nämlich immer etwas Bildqualität verloren. Zudem kann das digitalisierte Bild dann nur noch mit den grafischen Fähigkeiten des Amiga am Bildschirm dargestellt werden. Diese Möglichkeiten sind zwar ganz beachtlich. Zur Darstellung eines Bildes in Fernsehqualität reichen Sie aber bei weitem nicht aus – egal was Ihnen ein begeisterter Amiga-Fan oder der »freundliche Händler an der Ecke« erzählen möchte. Mit dem GenLock-Interface können Sie hingegen das qualitativ hochwertige Videobild mit einzelnen computererzeugten Bildteilen kombinieren.

Die Qualität, die das vom GenLock gemischte Bild (aus realen und Amiga-erzeugten Bildteilen) hat, hängt sehr stark vom verwendeten GenLock ab. Die Qualität des realen Bildteils ist zwar immer sehr hoch. Die Übergänge zwischen Realbild und der vom Amiga erzeugten Grafik fransen aber bei preiswerten GenLocks oft aus und flimmern stark. Hier gibt es sehr große Unterschiede in Qualität und Preis. Wirklich gute GenLocks – die auch den Ansprüchen eines professionellen Video-Studios standhalten können – gibt es leider momentan nur für den Amiga 2000. Dafür muß man dann aber auch bis zu 5000,– DM anlegen. Für den Heimgebrauch tun es die preiswerten GenLocks, die zwischen etwa 400,– und 1500,– DM für den Amiga 500 angeboten werden aber sicherlich auch.

## 9.7 Erweiterungen für Musiker

Aber nicht nur grafisch interessierte Anwender und Video-Fans können eine Menge Geld beim Kauf von Zusatzgeräten ausgeben. Ähnliche Möglichkeiten stehen auch musikalisch interessierten Menschen offen. Auch hier steht man im wesentlichen vor dem Problem, wie man Musik und andere Töne in den Amiga hinein- und wieder herausbekommt. Daneben ist der Amiga für die »Noten-Verarbeitung« – sprich Komposition – aber ebensogut geeignet wie zur Textverarbeitung oder zum Malen und zeichnen.

### 9.7.1 Der Amiga als Kompositionshilfe

Gerade diesen Aspekt, also die Verwendung eines Computers als Hilfe beim Komponieren und Arrangieren, sollte man nicht vernachlässigen. Hierbei stehen einem Möglichkeiten zur Verfügung, von denen der vor dem Klavier sitzenden Komponist mit Bleistift und Notenpapier nur träumen kann. Mit Programmen wie Aegis Sonix oder DeluxeMusic sind kleinere Änderungen blitzschnell (und sauber) durchzuführen und selbst große Änderungen (Transpositionen und Änderungen in der Instrumentierung verlangen meist nur einen einzigen Menübefehl. Zugleich kann man sich die frisch komponierten beziehungsweise geänderten Stücke auch jederzeit anhören. Der Amiga simuliert dazu – ähnlich wie eine elektronische Orgel – die verschiedenen Instrumente.

### 9.7.2 Der Amiga als Instrument

Auch als Instrument hat der Amiga 2000 einiges zu bieten. In vieler Hinsicht bietet er wirklich die Fähigkeiten einer einfachen elektronischen Orgel – und sogar einiges darüber hinaus. Einfache Orgeln (zum Beispiel die in letzter Zeit immer beliebter werdenden Geräte, die nur noch aus einem Keyboard bestehen) bieten nur eine beschränkte Auswahl an »Instrumenten« (Klangformen, die beim Abspielen einer Melodie verwendet werden). Zusätzliche Instrumente müssen nachgekauft werden und werden in Form von Chips in das Keyboard gesteckt.

Mit dem Amiga (und der passenden Software) steht Ihnen hingegen eine fast unbegrenzte Zahl von Klangformen zur Verfügung. Wenn Sie wollen – und sich dazu berufen fühlen – können Sie sogar eigene Klangformen, also Instrumente, schaffen. Sie müssen solche Klangformen aber nicht »von Grund auf neuschaffen« sondern können natürlich auch den Klang von Instrumenten, die das jeweilige Programm schon simulieren kann, abändern.

Trotz dieser ungewöhnlich hohen Flexibilität, die schon der nackte und nicht erweiterte Amiga bietet, hat er als Instrument einige Schwächen (die sich aber beheben lassen). So läßt die Klangqualität zum Beispiel arg zu wünschen übrig. In den meisten Fällen dürften die beiden Audio-Ausgänge des Amiga ja an die Lautsprecher in einem Videomonitor angeschlossen sein. Da viele Monitore Lautsprecher enthalten, bietet sich eine solche Lösung auch an. Diese in den Monitor eingebauten Lautsprecher sind aber recht klein, von minderwertiger Qualität und oft auch noch schlecht und ungünstig eingebaut. Für einen Piepston reichen Sie aus, aber kaum für die Wiedergabe von Musik.

Es hält Sie aber auch niemand davon ab, die beiden Stereo-Ausgänge Ihres Amiga mit einer HiFi-Anlage zu verbinden. Das ist wesentlich problemloser als der Anschluß eines Videorekorders an den Amiga. Alles, was Sie dazu benötigen, ist ein passendes, ausreichend langes Kabel. Die beiden Audio-Ausgänge des Amiga vertragen Standard-Stecker im Cinch-Format. Entsprechende Adapter für den Anschluß an die DIN-Eingänge, über die manche deutschen HiFi-Anlagen verfügen, sind im Fachhandel zu haben. Die Spannungspegel des vom Amiga gelieferten Tonsignals sind aber verhältnismäßig hoch. Wenn Ihre HiFi-Anlage dies nicht vertragen sollte und immer ein tiefes Brummen zu hören sein sollte, können Sie dieses durch einen passenden Adapter beseitigen. Dieser Adapter muß einen Vorwiderstand enthalten, der die hohen Spannungspegel des Amiga auf die Spannungen reduziert, die Ihre HiFi-Anlage erwartet.

Wenn Ihre HiFi-Anlage im Wohnzimmer steht und der Amiga in einem anderen Zimmer, ist diese Lösung aber vielleicht nicht ideal. Dann können Sie sich aber ohne weiteres mit zwei kleinen Aktivboxen, wie Sie zum Beispiel für Walkmans angeboten werden, behelfen. Diese kleinen Boxen sind recht preiswert und liefern einen erstaunlich guten Klang – zumindest einen erheblich besseren als der in den Monitor eingebaute Lautsprecher.

Selbst mit diesen Zusätzen kann der Amiga allerdings einem CD-Player keine Konkurrenz machen – ja, er erfüllt nicht einmal die bescheidenen Kriterien der deutschen HiFi-Norm DIN 45500. Dazu ist der Synthesizer-Chip, mit dem er Töne erzeugt, einfach nicht leistungsfähig genug.

### 9.7.3 Der MIDI-Standard

Wenn Sie den Amiga wirklich als Instrument verwenden wollen, stehen Ihnen deshalb weitere Anschaffungen ins Haus. Die wichtigste davon ist ein »MIDI-fähiger Synthesizer« und ein MIDI-Adapter für den Amiga. Das klingt aber schlimmer als es in Wirklichkeit ist. Während ein Synthesizer noch vor wenigen Jahren ein riesiges, teures und kompliziert zu bedienendes Gerät war, bekommen Sie heute bereits gleichwertige Modelle zu Preisen um etwa 1000,– DM. Und für etwa 2000,– oder 3000,– DM sind schon sehr gute Geräte zu bekommen. Es handelt sich dabei um die modernen elektronischen Orgeln in Form eines handlichen Keyboards. Ein solches Keyboard können Sie – wenn es MIDI-fähig ist und Sie an den Amiga einen MIDI-Adapter angeschlossen haben – über ein entsprechendes Kabel mit dem Amiga verbinden. Mit der passenden Software (zum Beispiel DeluxeMusic) können Sie den im Keyboard eingebauten Synthesizer dann vom Amiga aus steuern oder dessen Tasten auch für die interaktive Noteneingabe benutzen. Sie können zum Beispiel eine Melodie auf dem Keyboard spielen, sie dabei gleich vom Amiga auf Diskette aufzeichnen lassen, die Melodie am Bildschirm (in Notenform) ändern und dann wieder abspielen lassen, ohne daß Sie ein zweites Mal in die Tasten greifen müssen.

Das alles ermöglicht der MIDI-Standard! MIDI ist eine Abkürzung für ***M**usical **I**nstruments **D**igital **I**nterface*; oder zu deutsch digitale Schnittstelle für Musikinstrumente. Dieser Standard erlaubt es, daß elektronische Musikgeräte, wie Synthesizer, Keyboards und Computer über genormte Stecker und Kabel miteinander verbunden werden können. Auch das Format der Informationen, die über diese Kabel übermittelt werden, zum Beispiel Noten, Pausen und Klanginformationen, ist weitgehend genormt.

Wenn Sie einen Synthesizer vom Amiga aus steuern, verschwinden so ziemlich alle Probleme und Einschränkungen, die der Amiga, wie oben beschrieben, mit sich bringt. Ihre Musikanlage ist dann so gut wie der Synthesizer, den Sie anschließen. Das bedeutet eine hervorragende Klangqualität und – mit einem entsprechend ausgestatteten Synthesizer – endlich auch Vielstimmigkeit.

### 9.7.4 Sound-Digitizer

Gerade im Zusammenhang mit dem Anschluß eines Synthesizers ist eine andere Erweiterungsmöglichkeit sehr interessant: ein Sound-Digitizer. Ein Sound-Digitizer ist ein kleines Kästchen mit einem Audio-Eingang, das Sie an Ihren Amiga (meist an die parallele Schnittstelle) anschließen können. Schließen Sie eine Tonquelle – zum Beispiel einen Kassettenrekorder, eine Schallplatte oder auch ein Mikrofon – in den Audio-Eingang an, können die von dieser Quelle kommenden Töne mit der geeigneten Software digitalisiert werden. Das bedeutet, daß die Töne in eine Zahlenfolge umgewandelt werden, die abgespeichert, modifiziert und (natürlich als Ton) wiedergegeben werden kann. Die Wiedergabe kann dabei über den Audio-Ausgang des Amiga erfolgen, sie kann aber auch einem MIDI-fähigen Ausgabegerät überlassen werden, indem man die digitalisierten Tonfolgen wie Klangcharakteristika eines Instruments behandelt.

Mit einem Sound-Digitizer können Sie den Amiga wirklich wie einen Kassettenrekorder verwenden. Ähnlich wie beim Digitalisieren von Bildern gilt jedoch, daß dabei enorme Datenmengen anfallen; bei guter Klangqualität mehr als 10 Kbyte pro Sekunde. Das sind zwar nicht ganz so große Mengen wie beim Digitalisieren von Bildern aber immer noch genug, um in kürzester Zeit jede Diskette und die meisten Festplatten zum Platzen zu bringen. Da auch die Klangqualität des Amiga für echte HiFi-Fans wenig überzeugend ist, sollten Sie ihn besser doch nicht als Kassettenrekorder mißbrauchen. Sound-Digitizer sind für kurze Sound-Effekte oder einzelne gesprochene Sätze am besten geeignet.

Zudem kann man Sie dazu verwenden, die Klangcharakteristik eines natürlichen Instruments aufzuzeichnen. Dazu muß man keine stundenlangen Aufnahmen machen, sondern es genügt, ein oder zwei Noten mit dem Instrument zu spielen und zu digitalisieren. Diese kurze Zeitspanne reicht aus, um den »typischen Klang« eines Instruments einzufangen (der Fachmann nennt das »samplen«). Mit einem solchen digitalisierten Klang kann der Amiga dann Noten beliebiger Tonhöhe und Dauer spielen. Er kann die Klangcharakteristik aber auch an einen Synthesizer übermitteln und so dessen Repertoire an Instrumenten vergrößern.

Wenn Sie ein Freund der »echten« elektronischen Musik sind, brauchen Sie natürlich nicht unbedingt einen natürlichen Klang aufzuzeichnen, um »ein neues Instrument zu schaffen«. Sie können die Klangcharakteristik des neuen Instruments auch direkt am Bildschirm beschreiben oder mathematisch berechnen lassen. Dafür brauchen Sie aber keine zusätzliche Hardware, sondern nur das passende Programm (wie zum Beispiel Aegis Sonix).

## 9.8 Beschleuniger

Der Amiga 500 ist gewiß kein langsamer Computer. Gerade beim Aufbau von Grafiken ist er schneller als jeder andere Computer seiner Klasse. Andererseits ist aber kein Computer schnell genug, wie Ihnen jeder bestätigen wird, der schon einmal längere Zeit mit einem Computer gearbeitet hat. Größe oder Schnelligkeit des jeweiligen Computers spielen dabei interessanterweise überhaupt keine Rolle; jeder Computer ist »zu langsam« für den ungeduldigen Benutzer. Deshalb gibt es auch für den Amiga sogenannte »Beschleuniger-« oder »Turbo-Karten«.

Wie aber kann man einen Computer schneller machen? Die einfachste und effektivste Art ist es, die CPU, das Herz des Computers, auszutauschen. Dafür gibt es beim Amiga, grundsätzlich zwei Möglichkeiten. Die preiswerteste Möglichkeit ist es, den 68000-Chip mit höherer Taktfrequenz zu verwenden. Im Amiga arbeitet der 68000 normalerweise mit einer Taktfrequenz von 7,19 MHz (Megahertz). Inzwischen gibt es aber Chips annähernd gleicher Bauart, die mit der doppelten oder gar der dreifachen Frequenz arbeiten können. Die meisten Aufgaben werden von diesen Chips dann auch doppelt oder dreifach so schnell verrichtet. Die zweite Beschleunigungsmöglichkeit stellen verbesserte Versionen des 68000, der 68020 und 68030 dar. Diese Chips arbeiten »von Natur aus« etwa doppelt so schnell wie der 68000, weil sie viele Arbeiten, für die der ältere Chip zwei Schritte benötigt, in einem Schritt erledigt. Zusammen mit einer höheren Takfrequenz ergibt sich meist eine Beschleunigung um mindestens den Faktor 4.

Leider genügt es aber nicht, einfach den 68000-Chip gegen einen neueren Chip auszutauschen und/oder diesen mit höherer Taktfrequenz laufen zu lassen. Auch alle anderen Bauteile im Amiga – vor allem die RAM-Chips – sind auf die normale Taktfrequenz der CPU abgestimmt. Die RAM-Chips zum Beispiel können gar nicht ohne weiteres mit einer schnelleren CPU zusammenarbeiten. Und wenn Sie einen 68020 verwenden, so verlangt dieser zudem auch noch eine andere Organisation des RAM-Speichers. Deshalb enthalten Turbokarten neben der schnelleren CPU meist auch eigenen RAM-Speicher. Der schon vorhandene Speicher (außer den unteren 512 Kbyte Chip-RAM) wird lahmgelegt.

Turbokarten werden üblicherweise direkt im Gehäuse des Amiga 500 eingebaut. Dazu müssen meist schon vorhandene Chips entfernt oder auch neue Kabelverbindungen geschaffen werden. Alles das sind Aufgaben, die – wenn man nicht selbst beträchtliches Löttalent besitzt – am besten dem Fachhändler überlassen werden.

Beschleunigerkarten werden von einer Vielzahl von Herstellern angeboten (Commodore selbst bietet für den Amiga 500 allerdings keine solche Karte an). Besonders die kalifornische Firma CSA hat sich auf diesem Feld sehr hervorgetan und bietet sowohl Steckkarten für den Amiga 2000 wie auch spezielle Konstruktionen für den Amiga 500 an, die auch diesen um einiges schneller machen. Aber egal, bei wem eine solche Turbokarte gebaut wird, grundsätzlich gilt bei diesem Thema Vorsicht! In mancher Hinsicht entspricht eine Turbokarte einem Porschemotor im Käfer. Sie kann einen Riesenspaß machen – und natürlich auch die Freunde beeindrucken – ist aber auch gefährlich. Einige wenige Programme vertragen die hohe Geschwindigkeit nämlich nicht. Sie verursachen schwerwiegende Fehler oder funktionieren nicht richtig. Schnelle Aktion-Spiele werden zum Beispiel regelrecht unspielbar. Bevor Sie sich eine solche Karte zulegen, sollten Sie deshalb am besten beim Händler ausprobieren, ob die Programme, auf die Sie angewiesen sind, auch wirklich einwandfrei damit funktionieren!

# 10 Festplatten für den Amiga 500

Der Amiga 500 wird in den meisten Fällen sicherlich in einer Standard-Konfiguration mit 512 Kbyte RAM-Speicher und einem eingebauten 3,5-Zoll-Diskettenlaufwerk gekauft. In diesem spartanisch ausgestatteten Zustand dürfte er aber wohl bei vielen Anwendern nicht lange bleiben. Die Arbeit mit nur einem Diskettenlaufwerk ist einfach zu mühselig. Jedem, der intensiv mit dem Amiga arbeiten möchte, kann nur wärmstens empfohlen werden, sich zumindest ein zweites Diskettenlaufwerk zuzulegen. Die tägliche Arbeit wird dadurch nicht nur beschleunigt, sondern auch sicherer gemacht (weil das Anfertigen einer Sicherungskopie viel einfacher ist). Im allgemeinen arbeitet der professionelle PC-Anwender heutzutage aber mit einer Festplatte. Und genau um diese »Platten« dreht sich auch dieses Kapitel.

Sie werden in den folgenden Abschnitten erfahren, welches die Vorteile von Festplatten gegenüber Disketten sind, welche Fabrikate Ihnen zur Wahl stehen, wie Sie die Festplatte, für die Sie sich dann entschieden haben, einbauen, installieren und in Betrieb nehmen. Und schließlich werden Sie auch den einen oder anderen Tip zur Organisation der Daten auf der Festplatte finden. Diese unterscheidet sich nämlich meist grundlegend von der Dateienverwaltung auf Disketten. Auf Festplatten findet eine Unzahl von Dateien Platz, weshalb es sehr leicht ist – bei falscher Planung und Organisation – den Überblick zu verlieren.

## 10.1 Grundsätzliches zum Thema Festplatte

Der Unterschied zwischen einer Festplatte und einer Diskette ist – rein technisch gesehen – nicht sehr groß. In beiden Fällen werden Informationen auf einer rotierenden Scheibe aus magnetisierbarem Material aufgezeichnet. Die Oberfläche der Scheibe ist dabei in konzentrische Ringe, »Spuren« genannt, unterteilt, die ihrerseits in sogenannte »Sektoren« unterteilt sind. Jeder dieser Sektoren kann eine bestimmte Menge Daten fassen. Die Kapazität eines solchen Datenträgers ergibt sich deshalb aus der Anzahl vorhandender Spuren und der Anzahl der Sektoren pro Spur.

Der Schreib-Lese-Kopf einer Diskette oder Festplatte kann die verschiedenen Spuren erreichen, indem er näher zum Mittelpunkt oder zum Rand bewegt wird; die Sektoren einer Spur hingegen kommen während einer Umdrehung dann automatisch unter dem Kopf

vorbei und können gelesen oder beschrieben werden. Dieses Prinzip garantiert, daß innerhalb kurzer Zeit jeder Sektor erreicht werden kann und ist der Hauptvorteil von Festplatten und Disketten gegenüber Magnetbändern.

Damit hören die Gemeinsamkeiten aber schon auf. Während Disketten im wesentlichen aus einer dünnen flexiblen Plastikscheibe bestehen, die magnetisierbar beschichtet wurde, ist die Grundlage einer Festplatte eine harte Scheibe aus Metall oder Glas. Diese Scheibe rotiert wesentlich schneller als eine Diskette und ist auch in mehr Spuren eingeteilt. Um die schmaleren Spuren richtig lesen zu können, ist der Magnetkopf sehr klein und »fliegt« in sehr geringer Höhe über die Scheibe. Das alles hat zur Konsequenz, daß Festplatten wesentlich schneller als Disketten arbeiten und viel mehr Daten fassen können. Magnetkopf und die magnetisierbare Scheibe einer Festplatte sind dafür aber so empfindlich, daß schon winzige Staubpartikel auf der Platte genügen würden, um den Kopf während seines »Fluges« zu zerstören. (Der Kopf fliegt tatsächlich mit so hoher Geschwindigkeit über die Oberfläche der Platte hinweg, daß er nach aerodynamischen Gesichtspunkten konstruiert wird.) Das Innere einer Festplatte ist deshalb absolut staubfrei und hermetisch versiegelt, damit das auch so bleibt. Eine aufwendige Mechanik sorgt ferner dafür, daß von außen kommende Stöße und Bewegungen (bis zu einer bestimmten Intensität) niemals dazu führen können, daß Magnetkopf und Platte sich berühren. Eine solche Berührung während der Rotation hätte wegen der hohen Geschwindigkeit des Kopfes katastrophale Folgen (genauer gesagt: die Festplatte wäre hinüber).

Die meisten Festplattenlaufwerke enthalten nicht nur eine Magnetscheibe, sondern gleich mehrere. Gebräuchlich sind Festplatten mit zwei bis sieben Scheiben, die sowohl an der Ober- wie auch an der Unterseite magnetisch beschichtet sind und Daten tragen. Eine Festplatte hat deshalb üblicherweise doppelt so viele Schreib-Lese-Köpfe wie sie Platten enthält. (Auch die Diskettenlaufwerke des Amiga entsprechen diesem Prinzip. Sie enthalten zwei Schreib-Lese-Köpfe, mit denen die Ober- und Unterseite einer Scheibe, der Diskette, bearbeitet werden kann.) Alle Schreib-Lese-Köpfe sind fest miteinander verbunden. Sie können nur gemeinsam nach innen und nach außen bewegt werden. Das heißt, wenn der oberste Kopf sich gerade über Spur Nummer 8 der obersten Platte befindet, »fliegen« auch alle anderen Schreib-Lese-Köpfe über Spur Nummer 8 der jeweils zugehörigen Oberfläche. Aus diesem Grund fast man die Spuren gleicher Nummer (gleicher Entfernung von Mittelpunkt) in den verschiedenen Platten auch zu einem sogenannten »Zylinder« zusammen. Auch eine Diskette wird in solche Zylinder eingeteilt, da auch sie eine Ober- und Unterseite besitzt, die von zwei starr gekoppelten Schreib-Lese-Köpfen beschrieben oder abgetastet wird.

Das sind genau die Zylinder (engl. *cylinder*), die der Shell-Befehl FORMAT während des Formatiervorgangs nennt. Die Formatierung einer Diskette und einer Platte geschieht nämlich meist zylinderweise, das heißt, alle Köpfe formatieren gleichzeitig einen Zylinder der Plattenoberfläche, für die sie »zuständig« sind. Bei allen anderen Operationen ist üblicherweise immer nur ein Schreib-Lese-Kopf tätig, und liest oder schreibt einen bestimmten Sektor in einer bestimmten Oberfläche eines bestimmten Zylinders.

Da Festplattenlaufwerke mehrere Platten enthalten und hermetisch abgeschlossen sein müssen, sind sie meist wesentlich klobiger als Diskettenlaufwerke, teurer und auch empfindlicher. Sie werden sicherheitshalber (von wenigen Ausnahmen abgesehen) fest in den Computer oder ein separates Gehäuse eingebaut. Anders als ein Diskettenlaufwerk, das

im Laufe der Zeit mit vielen verschiedenen Disketten zusammenarbeiten kann, gehen Festplatte und Festplattenlaufwerk also eine feste Verbindung ein – was einer der Gründe für den Namen »Festplatte« ist. Wegen dieser engen Verbindung unterscheidet man meist auch nicht zwischen »Festplatte« und »Festplattenlaufwerk«, sondern verwendet diese Begriffe meist wie Synonyme. Wenn eine Festplatte voll ist, können Sie deshalb nicht einfach die volle Platte aus dem Laufwerk nehmen und eine andere einlegen. Immer, wenn Sie mehr Speicherkapazität benötigen, müssen Sie ein komplettes (recht teures) Festplattenlaufwerk kaufen.

### 10.1.1 Wozu Festplatten gut sind

Was Sie jetzt gerade erfahren haben, waren aber nur die rein technischen Unterschiede zwischen Disketten und Festplatten. Natürlich sind die höhere Arbeitsgeschwindigkeit und die größere Kapazität ein Vorteil – wie aber wirkt sich das auf Ihre Arbeit aus?

Der wichtigste Vorteil ist für die meisten Anwender sicherlich die größere Speicherkapazität. Dieser Kapazitätsunterschied zwischen Diskette und Festplattte ist allerdings in gewisser Hinsicht ein zweischneidiges Schwert, denn die höchste Speicherkapazität hat sicherlich eine große Kiste mit Disketten. Wie Sie ja eben erfahren haben, können Festplatten nicht ohne weiteres ausgetauscht werden, während ein Diskettenwechsel zwar oft lästig, aber jederzeit möglich ist. Ein Diskettenlaufwerk und eine Kiste mit 100 Disketten haben also eine wesentlich höhere Speicherkapazität als die meisten Festplatten (üblich sind bei Festplatten für den Amiga Speicherkapazitäten zwischen 20 und 80 Mbyte). Sind Sie deshalb nicht mit einem zweiten Diskettenlaufwerk besser bedient, als mit einer Festplatte?

Nein! Der entscheidende Vorteil des Datenbestands auf einer Festplatte ist nämlich, daß er jederzeit verfügbar ist. Um an eine bestimmte Datei heranzukommen, müssen Sie also nicht erst die Diskette aus dem Laufwerk nehmen, langwierig in Ihrer Diskettensammlung blättern und dann schließlich eine andere einlegen (sofern Sie sie gefunden haben). Alle Dateien und Programme auf einer Festplatte sind schnell mit wenigen Mausklicks (zum Öffnen der entsprechenden Schubladen) erreichbar. Grundvoraussetzung für die optimale Nutzung einer Festplatte ist aber natürlich eine sinnvolle Ordnung der einzelnen Schubladen beziehungsweise (für Shell-Anwender) Directories auf der Platte. Nur dann können Sie den Kapazitätsvorteil wirklich ausnutzen.

Der zweite Vorteil einer Festplatte ist ihre gegenüber der Diskette deutlich höhere Arbeitsgeschwindigkeit. (Sie ist allerdings noch etwas langsamer als eine RAM-Disk.) Diese höhere Geschwindigkeit hat zwei Aspekte. Festplatten können einerseits einen bestimmten Sektor schneller auffinden und mit seiner Übertragung in den RAM-Speicher beginnen, als Disketten. Sie verfügen aber auch über eine höhere Datenübertragungsrate. Das heißt, Sie können große Datenmengen sehr schnell in den RAM-Speicher des Amiga übertragen oder auf der Platte sichern.

Diese beiden Faktoren wirken sich unter anderem so aus, daß der Start eines Programms, das sich auf einer Festplatte befindet, blitzschnell abläuft (manchmal zehnmal schneller als mit Disketten). Aber auch fast alle andere Operationen, wie zum Beispiel das Einlesen und Abspeichern einer Datei oder Sortieroperationen in großen Datenbanken, gehen merklich schneller vonstatten. Kurzum: die unproduktiven und frustrierenden Zeiten, während denen Sie nichtstuenderweise darauf warten, daß Ihr Amiga mit einer bestimmten Aufgabe fertig wird, werden deutlich verkürzt!

### 10.1.2 Aus welchen Teilen eine Festplatte besteht

Bevor ich Ihnen die Wahlmöglichkeiten, die Ihnen beim Kauf einer Festplatte zur Verfügung stehen, vorstellen kann, muß ich Ihnen zunächst erläutern, aus welchen Komponenten eine Festplatte besteht, beziehungsweise welche Teile Sie kaufen und in den Amiga einbauen müssen, um mit einer Festplatte arbeiten zu können. Dazu soll dieser – für den technisch nicht so interessierten Leser wahrscheinlich recht langweilige – Abschnitt dienen.

Das, was man gemeinhin »Festplatte« nennt, besteht aus drei Komponenten: dem Festplattenlaufwerk, der eigentlichen Festplatte und einem »Festplatten-Controller«. Da Festplattenlaufwerk und Festplatte meist zu einer untrennbaren Einheit verschmolzen sind (siehe oben), kann man diese beiden Komponenten aber auch getrost als einen einzigen Bestandteil auffassen. Anders ist es mit dem Festplatten-Controller (oft auch einfach »Controller« genannt). Dies ist eine kleine elektronische Schaltung, die das Festplattenlaufwerk steuert. Sie regelt die Umdrehungsgeschwindigkeit der Platte und die Position des Schreib-Lese-Kopfes, liest die Daten von der Platte und stellt sie dem Computer in einem bestimmten Format bereit oder schreibt bereitgestellte Daten in einem bestimmten Sektor auf der Platte.

Bevor Sie eine Festplatte, die Sie sich gekauft haben haben, benutzen können, müssen Sie deshalb zusätzlich auch einen solchen Controller installieren. Auch Diskettenlaufwerke benötigen übrigens einen Controller. Ein Disketten-Controller für bis zu vier Laufwerke ist beim Amiga 500 aber schon fest eingebaut und braucht deshalb beim Anschluß weiterer Diskettenlaufwerke nicht dazugekauft werden.

Das wirft natürlich die Frage auf, warum Commodore dann nicht auch gleich auch einen Festplatten-Controller in den Amiga eingebaut hat. Die Antwort ist recht einfach: Ein Festplatten-Controller ist wesentlich komplizierter und teurer als ein Disketten-Controller. Er würde deshalb den Preis für die Grundaustattung des Amiga 500 deutlich in die Höhe treiben. Damit würden auch diejenigen Käufer »bestraft«, die sich vielleicht nie eine Festplatte kaufen werden. Und schließlich ist der Amiga ja problemlos erweiterbar, wie Sie im letzten Kapitel erfahren haben. Nicht jede wünschenswerte Komponente muß deshalb schon im Grundgerät enthalten sein.

Festplatten-Controller werden auch tatsächlich in Form von Zusatzgeräten für den Erweiterungssteckplatz an der linken Seite des Amiga 500 angeboten. Ein solcher Controller (ein kleines Kästchen) wird einfach in den Erweiterungssteckplatz gesetzt, über ein Kabel mit dem Festplattenlaufwerk verbunden und schon können Sie Ihre neue Festplatte auch benutzen. Diese Beschreibung der »Beziehung« zwischen Festplatte und Controller ist aber nur die abstrakte Theorie. In der Praxis sind viele Hersteller dazu übergegangen, den Controller indem selben Gehäuse unterzubringen wie die Festplatte selbst und dieses Gehäuse mit einem geeigneten Stecker zu versehen, damit man es direkt in den Erweiterungssteckplatz stecken kann. Dadurch wird ihr kompakter, kleiner Amiga 500 allerdings an der linken Seite mit einem meist ziemlich wuchtigen Klotz fest verbunden, für den ja auch Platz vorhanden sein muß. Festplatten, bei denen Controller und Festplatte getrennt sind, haben deshalb auch ihre Vorteile. In den Amiga muß dann nur ein relativ kleines Kästchen gesteckt werden, daß über ein flexibles Kabel mit dem Festplattengehäuse verbunden wird. Die eigentliche Festplatte kann dann auch etwas entfernt vom Amiga aufgestellt werden. Das ist meist ganz praktisch. Wenn Sie hinter Ihrem Amiga zum Beispiel noch etwas Platz haben, können Sie die Festplatte oft zum Beispiel sehr gut als Bildschirmständer verwenden.

### 10.1.3 Festplatten und Controller

Auch, wenn Festplatte und Controller getrennt sind, heißt das noch lange nicht, daß man einen beliebigen Controller mit einer beliebigen Platte kombinieren kann. Leider (oder zum Glück – je nachdem welchen Standpunkt man einnimmt) gibt es nicht nur eine Art von Festplatten-Controllern. So hat sich zum Beispiel bei IBM-kompatiblen PCs der sogenannte ST-506-Standard für den Anschluß von Festplatten durchgesetzt. Dieser Standard bestimmt zum einen das Format der Kabel und Stecker, die Festplatten und Controller verbinden, und zum anderen auch Art und Bedeutung der elektrischen Signale, mit denen der Controller die Festplatte steuern kann. Zum ST-506-Standard gibt es heute eine große Zahl von Festplatten und Controllern auf dem Markt, die untereinander nahezu beliebig austauschbar sind.

Es gibt allerdings zwei Arten von ST-506-Controllern mit verschiedenen Vor- und Nachteilen. Früher wurden die sogenannten MFM-Controller verwendet, während sich heutzutage mehr und mehr die sogenannten RLL-Controller durchsetzen. RLL-Controller können dieselben Festplatten steuern wie MFM-Controller, bringen auf diesen Platten durch ein effizienteres Aufzeichnungsformat aber bis bis 50 Prozent mehr Daten unter. Eine Festplatte, die an einem MFM-Controller eine Kapazität von 20 Mbyte hat, kommt mit einem RLL-Controller also auf etwa 30 Mbyte. Nachteile der RLL-Controller sind ihr höherer Preis und die Tatsache, daß nicht jede Festplatte das neue Aufzeichnungsformat verträgt. Besonders preiswerte Platten älterer Bauart sind durch das RLL-Verfahren oft überfordert. Wenn Sie aber bereit sind, ein paar Mark mehr auszugeben, sind Sie mit einem RLL-Controller bestimmt besser bedient.

Damit keine Mißverständnisse aufkommen: RLL- und MFM-Controller sind zwar unterschiedlich, können aber über dieselben Kabel dieselben ST-506-Festplatten steuern.

Anders ist das bei ESDI- und SCSI-Controllern. ESDI ist ein Controller-Standard, der bislang vor allem in der Welt der IBM-kompatiblen PCs Verbreitung gefunden hat und im Amiga-Bereich bislang keine große Rolle spielt. Es handelt sich dabei um eine Weiterentwicklung des ST-506-Standards.

SCSI (ausgesprochen »Skazzy«) ist ein neuer Standard zur Steuerung von Festplatten, der sich in jüngster Zeit mehr und mehr durchsetzt. Während ein ST-506-Controller immer nur zwei Festplatten steuern kann, ist es möglich, bis zu 7 Geräte an einen SCSI-Controller zu hängen. Diese Geräte müssen nicht alle Festplatten sein. Auch andere Geräte, die in kurzer Zeit viele Daten mit dem Computer austauschen müssen, können an einen SCSI-Controller angeschlossen werden – selbst dann, wenn schon zwei Festplatten daranhängen. Scanner zum Digitalisieren von Bildvorlagen und schnelle Laserdrucker sind zum Beispiel zwei Klassen von Geräten mit denen der Datentransfer gar nicht schnell genug ablaufen kann. Auch Wechselplatten oder Bandlaufwerke für die Datensicherung können zusätzlich zu einer Festplatte über einen einzigen SCSI-Controller mit dem Amiga verbunden werden.

Für andere Computer (vor allem den Apple Macintosh) gibt es schon eine Zeitlang SCSI-Geräte zu kaufen. Viele dieser Geräte – zum Beispiel Festplatten – können mit einem SCSI-Controller auch am Amiga angeschlossen werden. SCSI scheint auch für einige andere Geräte der künftige Standard zu werden. Immer mehr Hersteller von Zusatzgeräten bieten ihre Produkte auch mit SCSI-Schnittstelle an. Auf den ersten Blick sieht das ja alles so aus, als hätte der SCSI-Standard nur Vorteile. Das stimmt natürlich nicht. Hauptnachteil eines SCSI-Controllers – vor allem dann, wenn man ihn nur zum Anschluß einer Festplatte

verwenden will – ist sein Preis. Der Preis für SCSI-Festplatten ist gleichfalls höher als der für ST-506-Festplatten.

Ich hoffe, daß ich Sie mit dieser Übersicht der Wahlmöglichkeiten beim Plattenkauf nicht allzusehr verwirrt habe. Das Thema ist leider etwas komplizierter – zumindest in der Theorie. In der Praxis werden Controller und Festplatte meist zusammen verkauft und stammen auch von einer Firma, die für das einwandfreie Zusammenwirken der beiden Komponenten garantiert (wenn sich Controller und Platte nicht sogar in ein und demselben Gehäuse befinden). Für sparsam Veranlagte und Bastler können die Informationen zum Thema Controller und Festplatten aber vielleicht trotz dieser Tatsache eventuell sehr wichtig sein. Einige Firmen (vor allem Versandunternehmen) bieten nämlich ausgesprochen günstig (zwischen 250,– und etwa 380,– DM) einzelne Controller für den Amiga 500 an. Diese Controller kann man dann mit einer (fast) beliebigen ST-506-Festplatte verbinden und erhält auf diese Weise ein ausgesprochen preiswertes Festplattensystem, da einzelne Festplatten (die eigentlich für IBM-kompatible PCs gedacht sind) im Versandhandel meist wesentlich günstiger zu haben sind, als Komplett-Festplattensysteme für den Amiga.

Der Nachteil einer solchen Lösung ist allerdings, daß die im Versandhandel erhältlichen IBM-Festplatten kein Gehäuse und kein Netzteil besitzen. Beides muß man sich anderweitig besorgen und zusammenbauen. Deshalb haben solche Bastelaktionen nur selten ein besonders ansehnliches Ergebnis. Wer auch nur ein klein wenig Wert auf ein Mindestmaß an Ästhetik und einen aufgeräumten Arbeitsplatz legt, ist mit komplett montierten Festplatten, die speziell für den Amiga 500 entwickelt wurden, sicherlich besser bedient. In diesem Fall muß man sich auch um Details wie Controller- und Festplatteneigenschaften dann auch nicht kümmern. Viel wichtiger sind dann Kapazität und Preis der Platte.

### 10.1.4 Autoboot oder nicht

Eine bestimmte Eigenschaft des Festplattencontrollers beziehungsweise der Festplatte ist aber für alle Amiga-Besitzer wichtig, egal ob es sich um hardwareinteressierte Bastler oder technikfremde »Nur-Anwender« mit zwei linken Daumen handelt. Diese Eigenschaft ist die »Autobootfähigkeit« der Festplatte. (Eigentlich ist diese Fähigkeit eine Eigenschaft des Controllers/Adapters. Der Einfachheit halber unterscheidet man aber oft nicht zwischen Eigenschaften des Controllers und denen der Festplatte – weil beides oft auch in einem Gehäuse eingebaut ist.)

Von einer »autobootfähigen« Festplatte kann der Amiga starten. Das heißt, Sie brauchen nach dem Einschalten keine Workbench-Diskette mehr einzulegen. Der Amiga verwendet die Festplatte wie die Workbench-Diskette. Da die Festplatte aber wesentlich schneller als ein Diskettenlaufwerk ist, läuft auch der Startvorgang wesentlich schneller ab. Bei Verwendung einer nicht-autobootfähigen Festplatte müssen Sie hingegen mit einer speziell für die Festplatte konfigurierten Workbench-Diskette starten, die während des Startvorgangs »die Kontrolle an die Festplatte« übergibt. Dazu führt die *Startup-Sequence* auf einer solchen Diskette unter anderem den MOUNT-Befehl (siehe Kapitel 7 und 8) aus, der die Festplatte erst bei der Amiga-Systemsoftware anmeldet. Erst nach diesem MOUNT-Befehl »kennt« die Amiga-Systemsoftware überhaupt die Festplatte. Deshalb ist es auch unmöglich, von der Platte zu starten, sie existiert sozusagen nicht, bevor der entsprechende MOUNT-Befehl ausgeführt wurde.

Die Erstellung einer geeigneten Startdiskette für eine Festplatte ist – wenn der Festplattenhersteller nicht sehr gute Hilfsprogramme mitliefert – oft nicht besonders einfach. Im allgemeinen muß man dazu gute Kenntnisse des CLIs beziehungsweise der Shell besitzen.

Eine autobootfähige Festplatte ist deshalb auf alle Fälle einer nicht-autobootfähigen Festplatte vorzuziehen. Sie macht das Leben in vieler Hinsicht leichter und beschleunigt den Startvorgang deutlich. Einziger Nachteil den autobootfähige Festplatten beziehungsweise Controller haben, ist ihr etwas höherer Preis. Diese Preisdifferenz (ca. 100,– bis 200,– DM) macht sich bei der täglichen Arbeit mit der Platte und durch eingesparten Ärger beim »Installieren« aber sehr schnell bezahlt.

Die Autobootfähigkeit einer Festplatte ist erst mit der Version 1.3 der Workbench- und Kickstart-Software eingeführt geworden. Genauer gesagt, ist sie »eigentlich« nur dann möglich, wenn Ihr Amiga mit der Kickstart-ROMs Version 1.3 (oder höher) ausgerüstet ist. Die Workbench-Diskette mit den Versionsnummer 1.3 allein genügt nicht. Einige Hersteller bieten allerdings besonders trickreiche Festplatten beziehungsweise Controller an, die es auch zusammen mit den älteren Kickstart-ROMs möglich machen, einen Neustart von der Festplatte auszuführen und auf die Workbench-Diskette zu verzichten (deshalb das Wörtchen »eigentlich«). Wenn Sie einen schon etwas älteren Amiga mit den 1.2-Kickstart-ROMs besitzen, sollten Sie beim Kauf einer autobootfähigen Festplatte unbedingt beachten, ob diese auch mit den alten Kickstart-ROMs autobootfähig ist. Falls nicht, müssen Sie sich die neuen Kickstart-ROMs besorgen und in Ihren Amiga einbauen – was auf alle Fälle eine sinnvolle Maßnahme darstellt.

### 10.1.5 Das FastFileSystem

Eine andere Neuerung, die mit der Version 1.3 der Workbench-Software eingeführt wurde, ist das sogenannte »FastFileSystem« (kurz FFS genannt). Hierbei handelt es sich um eine neue Methode der Dateienverwaltung auf einer Festplatte. Die FFS-Methode ist wesentlich schneller als die zuvor verwendete Methode. Welche Aspekte der Dateiverwaltung im einzelnen beschleunigt werden, ist von Festplatte zu Festplatte unterschiedlich und im einzelnen nur von Fachleuten zu verstehen. Der Endanwender kann die Geschwindigkeitssteigerung aber sofort feststellen, wenn er auf der Workbench eine Schublade öffnet oder im Shell-Fenster eine Dateienliste mit dem DIR-Befehl erzeugen läßt: die Piktogramme beziehungsweise Dateinamen erscheinen viel schneller. Je nach verwendeter Festplatte werden einige Arbeiten bei Verwendung des FFS bis zu zehnmal, andere bis zu vierzigmal schneller ausgeführt.

Das FFS besteht im Prinzip »nur« aus etwas zusätzlicher Software, die sich in Form der Datei *FastFileSystem* im Directory *L* der Workbench-Diskette befindet. (Das FFS kann deshalb auch ohne die Kickstart-ROMs der Version 1.3 genutzt werden.) Es ist allerdings lediglich für Festplatten gedacht und kann den Zugriff auf Disketten nicht beschleunigen. Damit eine Festplatte in den Genuß des schnelleren Datenzugriffs über das FFS kommen kann, müssen zwei Grundvoraussetzungen erfüllt sein. Erstens muß die Festplatte über den MOUNT-Befehl bei der Amiga-Systemsoftware (Amiga-DOS) angemeldet werden. Der MOUNT-Befehl entnimmt die Daten der Festplatte der Datei *Mountlist* im Directory Devs. In dieser *Mountlist* wird festgelegt, daß auf der Platte das FFS verwendet werden soll und nicht die normale Dateiverwaltung, die für Disketten benutzt wird. Zweitens muß die Platte auch

entsprechend der FFS-Methode formatiert werden. Dazu sollte man beim FORMAT-Befehl im Shell-Fenster das Schlüsselwort FFS angeben.

Der Nachteil des FFS-Systems ist, daß von Festplatten, die mit diesem Dateiverwaltungssystem formatiert wurden, nicht gestartet werden kann. (Denn zuerst muß ja ein MOUNT-Befehl durchgeführt werden, bevor die Festplatte überhaupt genutzt werden kann.) Das scheint auf den ersten Blick ein großer Haken des sonst so vorteilhaften FFS-Systems zu sein, da es eine Kombination von FFS und autobootfähiger Festplatte scheinbar ausschließt.

### 10.1.6 Festplatten-Partitionen

Glücklicherweise ist es wirklich nur scheinbar ausgeschlossen, diese beiden angenehmen Eigenschaften einer Festplatte miteinander zu kopieren. Um das FFS auf einer autobootfähigen Festplatte benutzen zu können, bedient man sich eines sehr einfachen Tricks, der auch in anderer Hinsicht eine Menge Vorteile mit sich bringt. Dabei wird eine Festplatte in mehrere sogenannte »Partitionen« aufgeteilt. Diese Partitionen nehmen zwar jeweils nur einen Teil der Festplatte ein (einen bestimmten Bereich von Zylindern; siehe oben), verhalten sich aber auf der Workbench und auch im Shell-Fenster wie selbstständige Festplatten. Mit entsprechenden Hilfsprogrammen ist es möglich, eine Festplatte in nahezu beliebig viele solche Partitionen zu zerlegen.

Um das FFS-System benutzen zu können, formatiert man nun die erste Partition einer autobootfähigen Festplatte mit dem normalen Dateiverwaltungssystem und bringt auf dieser die Workbench-Software unter, die beim Start benötigt wird. Für diesen Zweck reicht im allgemeinen eine Partition mit ein oder zwei Mbyte Größe. Der Rest der Festplatte (das können eine oder auch mehrere Partitionen sein) wird mit dem schnelleren FFS formatiert. Um diese anderen Partitionen nutzen zu können, muß allerdings bei den meisten Festplattenfabrikaten die *Startup-Sequenz* und auch die Datei *Mountlist* im Directory *Devs* in der ersten Partition geändert werden. Dazu sind zumindest Grundkenntnisse der Shell nötig. Teilweise liegen den Festplatten aber auch Dienstprogramme bei, die die notwendigen Änderungen mehr oder weniger automatisch durchführen.

Auch die von der Commodore-Software vorgegebene Einschränkung, daß von mit FFS formatierten Partitionen aus kein Neustart ausgeführt werden kann, wird von einigen Festplattenherstellern trickreich umgangen. Es gibt also Festplatten, die lediglich aus einer startfähigen Partition bestehen, die mit dem FFS-System formatiert ist. Oder die Platte besteht aus mehreren FFS-formatierten Partitionen, von denen eine beim Neustart als »Workbench-Diskette« verwendet wird. Festplatten beziehungsweise Controller, die das ermöglichen, sind sicherlich am empfehlenswertesten. (Sie kosten im allgemeinen übrigens nicht einmal mehr als andere autobootfähige Festplatten.)

Partitionen werden oft auch bei nicht-autobootfähigen Festplatten verwendet. Durch die Aufteilung wird nämlich der Umgang mit der fast unübersichtlich großen Speicherkapazität einer Festplatte oft einfacher. Auch Fehler – die immer einmal vorkommen können – bei denen eine Partition beschädigt wird, zerstören dann nicht den gesamten Datenbestand auf der Festplatte. Man muß dann nur die eine defekte Partition neu formatieren.

Mit diesen Anmerkungen zum Thema Partitionen möchte ich die grundsätzlichen Erläuterungen zum Thema Festplatten zunächst einmal abschließen.

### 10.1.7 Welche Festplatte für welchen Zweck?

Welcher Controller und welche Festplatte für Sie letztendlich die ideale Lösung darstellt, hängt stark davon ab, was Sie von der Festplatte im Detail erwarten, welche Programme Sie benutzen und welche Datenmengen dabei anfallen.

Zunächst einmal sollten Sie versuchen, Ihren Bedarf an Speicherkapazität abzuschätzen. Die preiswertesten Festplatten haben heute meist eine Kapazität von etwa 20 Mbyte. Für viele Anwender wird eine solche 20-Mbyte-Platte mehr als ausreichend sein. Sie können darauf eine Vielzahl von Programmen, Tausende von Textseiten, ein paar Dutzend Bilder, Ihre Buchhaltungsdaten und einige Spiele ablegen und haben dann immer noch Platz frei. Wenn Sie jedoch planen, einige Romane oder ein Archiv mit Hunderten von digitalisierten Bilder auf der Platte anzulegen, dürften 20 Mbyte schon etwas eng werden. Datenbanken benötigen zwar im Gegensatz zu »Volkes Meinung« nicht unbedingt viel Speicherplatz. Eine Adreßdatei mit tausend Einträgen wird zum Beispiel selten mehr als 1 Mbyte benötigen. Wenn Sie hingegen einige 10 000 Adressen sowie weitere Daten dazu ablegen wollen, werden Sie rasch die Schallmauer von 20 Mbyte durchbrechen.

Für Geschwindigkeitsfanatiker ist sicherlich bei der Entscheidung für eine bestimmte Plattenkapazität auch wichtig, daß Festplatten üblicherweise auch um so schneller werden, je größer sie sind. Ein wichtiges Merkmal für die Geschwindigkeit einer Festplatte ist die sogenannte »mittlere Zugriffszeit«, die in Millisekunden gemessen wird. Je kleiner die mittlere Zugriffszeit, desto schneller ist die Platte. Die meisten 20- und 30-Mbyte-Festplatten sind bei mittleren Zugriffszeiten von 40 bis 60 Millisekunden relativ langsam – für Festplattenverhältnisse. Festplatten mit 40 Mbyte und mehr Kapazität sind oft deutlich schneller und große Festplatten über 80 Mbyte können mittlere Zugriffszeiten von 20 Millisekunden und weniger erreichen. Dafür muß man dann aber schon recht tief in die Tasche greifen.

Vielen Amiga-500-Besitzern wird eine Festplatte mit 20 oder 30 Mbyte völlig ausreichen. Exakt abschätzen kann man seinen genauen Speicherbedarf aber wohl nur recht schwer. In den meisten Fällen kann man aber nicht viel falschmachen, wenn man zunächst einmal mit einer 20- oder 30-Mbyte-Platte beginnt. Falls diese später wirklich einmal »überlaufen« sollte, kann man ja eine zweite Platte gleicher oder größerer Kapazität dazukaufen.

Noch wichtiger als die Kapazität ist eigentlich die Frage, ob die Festplatte autobootfähig ist. Die Vorteile einer solchen Platte wurde ja bereits ausführlich erläutert. Ich möchte aber auf alle Fälle noch einmal wiederholen, daß autobootfähige Festplatten in vieler Hinsicht die beste Lösung darstellen, vor allem für Anwender, die nicht in die Kategorie »Löter und Bastler« gehören. Und wenn Sie sich schon dazu durchgerungen haben, eine autobootfähige Platte zu kaufen, sollten Sie am besten gleich auch eine der Platten nehmen, bei der auch FFS-formatierte Partitionen bootfähig sind.

### 10.1.8 Was Sie bei der Arbeit mit einer Festplatte beachten müssen

Zum Schluß dieses einleitenden Abschnitts zum Thema Festplatten möchte ich Ihnen die dunkle Seite« von Festplatten vorstellen. Damit ist nicht das Innere der Festplatte gemeint – obwohl es im Inneren des hermetisch versiegelten Plattengehäuses sicherlich pechschwarz ist. Mit der dunklen Seite meine ich vielmehr die Probleme, die Festplatten mit sich bringen

können. Festplatten haben nämlich nicht nur Vorteile, sondern – insbesondere bei undisziplinierter Arbeitsweise – auch einige Nachteile. Fast alle diese Nachteile entspringen einem Vorteil: es ist die große Speicherkapazität!

Auf einer Festplatte können Sie zum Beispiel sehr viele einzelne Dateien unterbringen – unvergleichlich mehr als auf einer Diskette. Wenn Sie nicht von vornherein eine Schubladen- beziehungsweise Directory-Struktur planen, in der Sie die Dateien unterbringen, kann es sehr schnell passieren, daß Sie den Überblick verlieren. Wenn Sie zum Beispiel zuviele Piktogramme direkt in das Diskettenfenster der Festplatte legen, wird ein einzelnes Piktogramm dabei leicht übersehen. Und da die Fensterfläche für Hunderte von Piktogrammen wohl kaum ausreicht, werden Sie den Fensterinhalt häufig mit den Rollbalken verschieben müssen.

Aber auch das umgekehrte Extrem ist nicht sinnvoll. Wenn Sie zu viele Schubladen einrichten und diese immer wieder ineinander verschachteln, werden Sie, um zu einem bestimmten Piktogramm zu gelangen, oft sehr viele Schubladen öffnen müssen, was ebenfalls lange dauert. Im Shell-Fenster hat eine solche Schubladen-Manie ähnliche Folgen. Hier werden nämlich die Pfadnamen, die Sie eingeben müssen, länger und länger. Das kostet Zeit und erhöht die Fehlerwahrscheinlichkeit.

Wie Sie die Ordnung auf Ihrer Festplatte gestalten sollen, kann ich Ihnen leider nicht beschreiben. Dazu sind die Bedürfnisse (und die Geschmäcker) der Anwender zu verschieden. In einem der folgenden Abschnitte finden Sie aber zumindest einige Tips dazu, wie Sie die Schubladen der Workbench-Diskette in übersichtlicher Form auf einer Festplatte unterbringen können. Ferner werden die Einschränkungen beschrieben, die Sie bei Anwendungsprogrammen oft beachten müssen. Diese Programme und deren Hilfsdateien können nämlich leider meist nicht in beliebige Schubladen gelegt werden.

Festplatten haben aber noch ein weiteres Problem, das seine Ursache ebenfalls in der hohen Speicherkapazität hat: Das Anfertigen von Sicherungskopien ist viel schwieriger als bei Disketten. Wie Sie ja in den ersten Kapiteln dieses Buches erfahren haben, müssen Sie von allen wichtigen Disketten immer Sicherheitskopien haben. Das gilt sowohl für Disketten mit Programmen darauf wie auch für reine Daten-Disketten. Sicherungskopien der Disketten, mit denen Sie ständig arbeiten, sollten Sie zudem mindestens täglich erneuern. Wenn eine solche Diskette dann einmal einen Defekt bekommen sollte, können Sie einfach mit der Kopie vom Vortag weiterarbeiten.

Wenn Sie nicht gerade eine zweite Festplatte besitzen, die Sie ansonsten nicht benötigen, ist die Anfertigung einer Sicherungskopie von einer Festplatte bei weitem nicht so einfach, wie bei einer Diskette. (Bei Disketten können Sie den Workbench-Befehl *Duplicate* oder den Shell-Befehl DISKCOPY verwenden.) Eine professionelle Lösung wäre es sicherlich, sich einen sogenannten »Streamer« (ein Magnetbandlaufwerk) für diesen Zweck anzuschaffen. Wenn Sie eine Festplatte mit SCSI-Controller besitzen, ist der Anschluß eines solchen Streamers kein Problem, kein technisches zumindest, sondern höchstens noch ein finanzielles, da ein Streamer etwa genausoviel wie eine 20-Mbyte-Festplatte kostet!

Sie können sich natürlich auch damit behelfen, Sicherungskopien der Festplatte auf einer Vielzahl von Disketten zu machen. Auf der Workbench und auch im Shell-Fenster ist eine vollständige Kopie einer fast vollen 20-Mbyte-Festplatte allerdings ein langwieriges und umständliches Unterfangen. Sinnvoll ist es, solche Kopien schubladenweise anzufertigen, solange die Schubladen nicht allzu viele Daten enthalten.

## 10.2 Installation und Inbetriebnahme einer Festplatte

In den folgenden Abschnitten wird überblickartig beschrieben, wie Sie nach der Anschaffung einer Festplatte vorgehen müssen, um diese nutzen zu können und was dabei zu beachten ist. Bei diesen Beschreibungen konnte ich leider nicht immer so sehr auf einzelne Details eingehen, wie es derjenige sich wohl erhoffen dürfte, der gerade mehr oder weniger verzweifelt vor einem offenen Karton und einer unleserlichen Bedienungsanleitung steht. Dies ist deshalb nicht möglich, weil die Unterschiede zwischen den Festplatten einfach zu groß sind. Speziell die Installation der zum Betrieb benötigten Software auf der Festplatte selbst und auf der eventuell nötigen Start-Diskette ist nicht immer ganz einfach. Viele Festplattenhersteller sind aber glücklicherweise dazu übergegangen, Installationsprogramme mit ihren Platten auszuliefern, die diesen an sich recht komplizierten Vorgang zu einem simplen Frage-und-Antwort-Spiel vereinfacht.

Wenn ein solches Installationsprogramm Ihrer neuen Festplatte nicht beiliegt, sollten Sie zumindest solide Grundkenntnisse im Umgang mit der Shell besitzen, um einige der Informationen in den folgenden Abschnitten nutzen zu können. Besitzen Sie diese nicht, sollten Sie vielleicht besser einen Fachmann bei der Einrichtung der Festplatte zu Rate ziehen – besonders dann, wenn diese nicht autobootfähig ist. Die Installation einer autobootfähigen Platte ist meist wesentlich einfacher.

Bei der Installation und Inbetriebnahme einer Festplatte müssen im Allgemeinen die folgenden Schritte ausgeführt werden:

- Installation von Controller und Festplatte
- Vorbereitung und physikalische Formatierung der Festplatte
- Installation der Festplattensoftware
- Partitionieren der Festplatte (Ändern der Datei *Mountlist*)
- Logische Formatierung der Festplatte durch Amiga-DOS
- Kopieren der Workbench-Software auf die Festplatte
- Erzeugen einer speziellen Start-Diskette (nur bei nicht autobootfähigen Platten)

Diese Schritte werden in den folgenden Abschnitten ausführlich beschrieben. Sie werden aber nicht immer jeden dieser Schritte auch selbst durchführen müssen. Einige wird oft schon der Festplattenhersteller für Sie erledigt haben, andere werden vielleicht von Installationsprogrammen mehr oder weniger automatisch erledigt.

### 10.2.1 Installation von Controller und Festplatte

Wichtig bei der Installation der Plattenhardware ist vor allem, daß sowohl die Festplatte wie auch ein Controller vorhanden sein müssen. Speziell dann, wenn Sie eine Festplatte im Versandhandel bestellen, müssen Sie auf diese Tatsache achten. Wie schon in einem vorangehenden Abschnitt erwähnt wurde, können sich Festplatte und Controller zwar auch in einem einzigen Gehäuse befinden. Es kann sich aber auch um zwei getrennte Geräte handeln, die über ein Kabel miteinander verbunden werden. Entweder das Festplattengehäuse selbst oder das (meist recht kleine) Kästchen mit dem Controller muß einen Anschlußstecker besitzen, den Sie in die Erweiterungsschnittstelle an der linken Seite des Amiga stecken können. (Diese verbirgt sich hinter einer Plastikkappe, die mit einem kurzen Griff entfernt werden kann.)

Bevor Sie das tun, sollten Sie natürlich erst den Amiga ausschalten. Verbinden Sie am besten die Festplatte auch noch nicht mit dem Stromnetz und achten Sie darauf, daß Sie die kleinen Kontakte der Erweiterungsschnittstelle und die entsprechenden Kontakte des Steckers an der Festplatte nicht mit den Fingern berühren. Bei einer solchen Berührung dürfte zwar in 99% aller Fälle nicht Tragisches passieren – vielleicht erleben aber gerade Sie das eine Prozent, bei dem ein kleiner Funken statischer Elektrizität empfindliche Bauteile zerstört.

Ist diese Erweiterungsschnittstelle schon von einem anderen Gerät belegt, muß dieses seinerseits eine Erweiterungsschnittstelle besitzen – sonst können Sie die Festplatte nicht anschließen. Oder die Festplatte beziehungsweise der Controller verfügen über eine »durchgeschleifte« Erweiterungsschnittstelle, in die Sie dann das erste Gerät stecken können.

Wenn Sie die Festplatte beziehungsweise den Controller mit der Erweiterungsschnittstelle verbunden haben (was ohne großen Kraftaufwand möglich sein muß), sollten Sie kurz überprüfen, ob es eine Möglichkeit gibt, die Platte oder den Controller an der Schnittstelle festzuschrauben. Dafür sind zwei kleine Bohrlöcher an der Erweiterungsschnittstelle vorgesehen. Eine solche Befestigung erhöht wesentlich die Sicherheit der Verbindung. Ansonsten erhalten Sie ein recht labiles Gebilde, das nur durch Reibung gehalten wird. Bei einer nicht verschraubten Verbindung sollten Sie Ihren Amiga in Zukunft nur noch vorsichtig bewegen und nach jeder Verlagerung des Arbeitsplatzes die Steckverbindung überprüfen.

Wenn Sie eine Festplatte mit integriertem Controller besitzen, müssen Sie diese jetzt nur noch mit dem Stromnetz verbinden (Festplatten verfügen immer über eine eigene Stromversorgung) und einschalten.

Bei Festplatten mit getrenntem Controller müssen Sie vor dem Einschalten Platte und Controller verbinden. Dazu werden bei den verschiedenen Festplattentypen unterschiedliche Kabel verwendet. Meist handelt es sich um ein flaches breites Kabel mit kleinen Plastiksteckern an den Enden oder um ein dickes rundes Kabel mit stabilen, breiten Steckern. Diese zweite Kabelsorte hat den Vorteil, daß sie etwas besser aussieht und die Stecker festgeschraubt werden können und dann nicht mehr versehentlich herausgezogen werden können.

### 10.2.2 Vorbereiten der Festplatte

Nach der Installation der Festplatte können Sie diese und dann den Amiga einschalten. Gewöhnen Sie sich am besten gleich die richtige Reihenfolge an, erst die Platte und dann den Amiga einzuschalten. (In manchen Fällen erkennt der Amiga sonst die Platte nicht.) Falls Sie nicht eine autobootfähige Platte gekauft haben, die gleich mit der nötigen Software versehen wurde, »fragt« der Amiga nun nach einer Workbench-Diskette.

Legen Sie diesmal nicht Ihre »normale« Workbench-Diskette ein, mit der Sie bisher immer gestartet sind, sondern verwenden Sie die Workbench-Diskette, die der Festplatte beilag – falls eine Workbench-Diskette beilag. Ist das nicht der Fall, können Sie Ihre normale Workbench-Diskette benutzen. Was jetzt genau zu tun ist, um die Festplatte nutzen zu können, hängt sehr stark vom konkreten Festplattenmodell ab. Wenn Sie eine autobootfähige Platte gekauft haben und diese schon vom Hersteller mit der Workbench-Software ausgestattet wurde, brauchen Sie vielleicht überhaupt nichts weiter zu tun als Platte und Amiga einzuschalten und die Workbench erscheint kurz darauf. Bei vielen anderen Festplatten sind aber einige Vorbereitungsschritte notwendig.

Grundsätzlich muß die Platte zunächst einmal formatiert und initialisiert werden (wie eine Diskette), bevor Sie Daten aufnehmen kann. Die echte »physikalische« Formatierung einer Festplatte kann üblicherweise nur von einem speziellen Programm vorgenommen werden. Der Workbench-Befehl *Initialize* oder der *Shell*-Befehl FORMAT sind dafür nicht geeignet. Lesen Sie am besten gleich nach dem ersten Start im Handbuch Ihrer Festplatte nach, wie das Formatierprogramm heißt, wo es zu finden ist und wie es bedient wird. Es kann sein, daß Sie dieses Programm jetzt erst einmal starten müssen, um die Platte zu formatieren. Viele Platten werden aber bereits »ab Werk« physikalisch formatiert. Auch diesen Platten liegt aber immer ein Formatierprogramm bei, weil bei manchen seltenen Fehlern eine Neuformatierung nötig ist.

Das Formatierprogramm benötigt immer die Angabe eventuell auf der Festplatte vorhandener defekter Spuren. (Diese können auf jeder Festplatte vorhanden sein und stellen keinen Qualitätsmangel dar.) Die Nummern dieser defekten Spuren und Zylinder sind üblicherweise auf der Festplatte selbst vermerkt oder auf einem Zettel, der der Platte beiliegt. Diesen Zettel sollten Sie sehr sorgfältig verwahren. Sie benötigen ihn unbedingt, um eine zuverlässige physikalische Formatierung der Platte vornehmen zu können.

Der nächste Schritt, der nach der physikalischen Formatierung der Platte nötig ist, um eine Festplatte für den Amiga »sichtbar zu machen« ist die Aktivierung spezieller Treibersoftware. Wenn der Festplatte eine Workbench-Diskette beilag, dann ist auf dieser die Treibersoftware wahrscheinlich schon korrekt installiert und wird beim Start auch automatisch aktiv. Eine formatierte Festplatte sollte bei Verwendung dieser Workbench-Diskette dann gleich nach dem Start auf der Workbench auftauchen.

Bei Festplatten, denen keine Workbench-Diskette beiliegt, muß die Treibersoftware üblicherweise von einer der Festplatte beiliegenden Diskette auf die Workbench-Diskette kopiert werden. Diese Aufgabe übernimmt oft ein Installationsprogramm. Manchmal muß man es aber auch »von Hand« machen, wozu meist Kenntnisse der Shell beziehungsweise des CLI benötigt werden. Lesen Sie im Handbuch Ihrer Festplatte nach, welche Schritte unternommen werden müssen, um die Festplattensoftware auf einer anderen Workbench-Diskette zu installieren. Üblich ist, daß zusätzliche Dateien in die Directories (beziehungsweise Schubladen) *Expansion*, *Devs*, *L* und/oder *Libs* gelegt werden und die Dateien *DEVS:Mountlist* und *S:Startup-Sequence* geändert oder ersetzt werden müssen.

Um den Sinn dieser Änderungen verstehen zu können, müssen Sie Grundkenntnisse der Shell und einiger Shell-Befehle besitzen – die die folgenden Erläuterungen voraussetzen.

Um eine Festplatte steuern zu können, muß der Amiga einen sogenannten Festplattentreiber, eine Art ganz spezielles Programm, nutzen. Dieser Treiber muß auf der Workbench-Diskette, mit der der Amiga gestartet wird, vorhanden sein. Er muß oft in das Directory *Expansion* oder *Devs* gelegt werden. Einige Treiber werden beim Start automatisch durch den Befehl BINDDRIVERS in der Startup-Sequenz aktiviert. Andere müssen mit einem zusätzlichen MOUNT-Befehl aktiviert werden, der dazu in der Datei *Startup-Sequenz* oder *StartupII* untergebracht wird. Erst nach diesem MOUNT-Befehl kennt der Amiga die neue Festplatte.

Der MOUNT-Befehl entnimmt der Datei *Mountlist* im Directory *Devs*, welche Eigenschaften die Festplatte hat und mit welchem Treiber sie gesteuert wird. Deshalb muß bei der Installation von Festplatten-Software üblicherweise auch diese Datei ersetzt oder geändert

werden. Dazu müssen Sie den Aufbau und die Bedeutung der Einträge in der Datei *Mountlist* nicht kennen oder gar verstehen. Es genügt, wenn Sie den Anweisungen im Handbuch der Festplatte folgen – wenn Ihnen nicht sogar ein Installationsprogramm diese Arbeit abnimmt. Sie brauchen die *Mountlist* nur dann genauer zu kennen, wenn Sie die Festplatte in »Partitionen« aufteilen wollen.

### 10.2.3 Festplatten-Partitionen

Solche Festplatten-Partitionen stellen eine logische Aufteilung der Festplatte dar. Sie bewirken, daß eine Festplatte wie mehrere verschiedene Festplatten auf der Workbench erscheint und es auch im Shell-Fenster mehrere verschiedene »Festplatten-Geräte« gibt. Eine solche logische Aufteilung kann die Arbeit mit einer Festplatte – unter anderem auch das Anfertigen von Sicherungskopien – vereinfachen. Sie kann aber bei manchen autobootfähigen Festplatten auch notwendig sein, wenn man das *FastFileSystem* verwenden will. Prinzipiell kann man eine Festplatte in eine fast beliebig große Anzahl von Partitionen aufteilen. Meist werden aber nur zwei oder drei Partitionen auf einer Platte angelegt – sonst würden beim Start auch gleich sehr viele Festplattenpiktogramme auf der Workbench erscheinen.

Um eine Festplatte in mehrere Partitionen aufzuteilen, muß im Prinzip die Datei *Mountlist* so abgeändert werden, daß Sie für jede einzelne Partition einen eigenen Eintrag enthält (der genauso aufgebaut ist wie der Eintrag für eine komplette Festplatte). Es kann aber sein, daß Sie nicht selbst diese Einträge in der Datei *Mountlist* vornehmen müssen, sondern daß das ein spezielles Programm macht, dem Sie dazu lediglich ein paar Fragen beantworten müssen. Diese Fragen werden Sie mit ein paar Grundkenntnissen über den Inhalt der *Mountlist* wesentlich besser verstehen und auch leichter beantworten können.

Wichtig ist im Zusammenhang mit der Partitionierung vor allem, ob die Platte autobootfähig ist. Bei einer autobootfähigen Platte kann die erste Partition meist nicht mit dem *FastFileSystem* formatiert werden, wenn man diese als Workbench-Diskette verwenden möchte. Solche Platten sollten auf alle Fälle in Partitionen aufgeteilt werden. Und zwar sollte die erste Partition nur so groß sein, daß sie den Inhalt der normalen Workbench-Diskette aufnehmen kann. Der Rest der Platte kann dann von einer oder auch von mehreren Partitionen eingenommen werden. Wichtig ist, daß die erste Partition nicht zu groß gemacht wird, da diese ja die langsamere alte Dateiverwaltungsmethode benutzen muß. Mehr zum Sinn und Zweck dieser besonderen Art der Partitionierung erfahren Sie ab Abschnitt 10.2.6 dieses Kapitels.

### 10.2.4 Die Mountlist

Die folgende Darstellung zeigt einen typischen Eintrag, mit dem eine nicht partitionierte Festplatte in der *Mountlist* verzeichnet sein könnte. Ein solcher Eintrag beginnt immer mit dem Amiga-DOS-Gerätenamen der Platte und endet mit einem Doppelkreuz (#):

```
HD0: Device = hddisk.device
     Unit = 0
     Flags = 0
     Surfaces = 4
     BlocksPerTrack = 17
```

```
    Reserved = 2
    Interleave = 0
    LowCyl = 0; HighCyl = 621
    Buffer = 5
    BufMemtype = 0
#
```

Diesen Eintrag könnte man mit dem folgenden Befehl im *Shell-Fenster* oder in der *Startup-Sequence* aktivieren:

```
MOUNT HD0:
```

Danach gäbe es das Gerät HD0:. Bevor diese Festplatte genutzt werden kann, muß sie aber eventuell noch »logisch« mit dem FORMAT-Befehl formatiert werden. Ansonsten erscheint nach dem MOUNT-Befehl ein Requester mit einer Fehlermeldung.

Die einzelnen Zeilen in dem eben aufgelisteten Mountlist-Eintrag haben die folgende Bedeutung: Die erste Zeile legt fest, daß zur Steuerung der Treiber *hddisk.device* verwendet werden soll. Eine Datei dieses Namens sollte dazu im Dierectory *Devs* liegen. Die zweite Zeile besagt, daß es sich um die Platte mit der »Unit-Nummer« *0* handelt. Die meisten Festplattencontroller können mindestens zwei Festplatten steuern. Die zweite Festplatte am selben Controller hätte die Unit-Nummer 1. Der Wert für *Flags* wird intern von Amiga-DOS benötigt. Hier steht fast immer eine 0. Die Werte für *Surfaces* und *BlocksPerTrack* können jedoch bei verschiedenen Platten verschieden sein. *Surfaces* gibt an, wieviele Oberflächen (und Köpfe) die Festplatte enthält, *BlocksPerTrack* beschreibt, wieviele Sektoren jede Spur enthält (siehe Abschnitt 10.1.3). Beide Angaben können dem Datenblatt der Festplatte entnommen werden. Kleine Platten mit 20 Mbyte haben meist 4 *Surfaces*. Bei Platten, die von MFM-Controllern gesteuert werden, ist *BlocksPerTrack* üblicherweise gleich 17; bei RLL-Controllern gleich 26.

*Reserved* gibt an, wieviele Blocks der Platte für Amiga-DOS reserviert sind und nicht zum Speichern von Daten zur Verfügung stehen. Hier steht immer eine 2. *Interleave* kontrolliert den sogenannten »Interleave-Faktor« der Festplatte. (Diesen zu erläutern würde hier etwas weit führen.) Er ist bei fast allen Amiga-Festplatten gleich 0. Sehr wichtig sind die Angaben *LowCyl* und *HighCyl*. Sie beschreiben, welche Zylinder der Festplatte vom Gerät HD0 benutzt werden können. In diesem Beispiel sind es die Zylinder 0 bis 621 und damit alle Zylinder einer 20-Mbyte-Platte mit vier Köpfen. Würde HD0: nur eine Partition der Platte beschreiben, könnten hier auch andere Zahlen stehen. (Mehr dazu weiter unten.)

*Buffer* gibt an, wieviele Pufferspeicher Amiga-DOS bei der Datenverwaltung auf dieser Platte verwenden soll (siehe auch den Shell-Befehl ADDBUFFERS). Fünf oder zehn Puffer sind für eine Festplatte übliche Werte. Die Zahl *BufMemType* ist eine sehr wichtige Angabe. Sie besagt, welcher Art der RAM-Speicher sein muß, in dem die Puffer angelegt werden. An anderer Stelle wurde ja bereits einmal erwähnt, daß der Amiga-RAM in »Chip-RAM« und »Fast-Mem« aufgeteilt ist. Die Zahl 0 an dieser Stelle bedeutet, daß die Speichertype keine Rolle spielt. 2 oder 3 besagt, daß Chip-RAM nötig ist, 4 oder 5 verlangt Fast-Mem.

Dieser Beispieleintrag beschreibt eine Festplatte, die vollständig, also als eine einzige Partition genutzt wird. Bei einer Festplatte, die in zwei Partitionen aufgeteilt ist, sieht die *Mountlist* etwas anders aus. Die folgende Liste teilt zum Beispiel eine Festplatte gleicher Bauart wie in obigem Beispiel in zwei annähernd gleich große Partitionen auf:

```
HD0: Device = hddisk.device
     Unit = 0
     Flags = 0
     Surfaces = 4
     BlocksPerTrack = 17
     Reserved = 2
     Interleave = 0
     LowCyl = 0; HighCyl = 310
     Buffer = 5
     BufMemtype = 0
#
HD1: Device = hddisk.device
     Unit = 0
     Flags = 0
     Surfaces = 4
     BlocksPerTrack = 17
     Reserved = 2
     Interleave = 0
     LowCyl = 311; HighCyl = 621
     Buffer = 5
     BufMemtype = 0
#
```

Wie Sie an diesem Beispiel sehen können, sind lediglich der Gerätename und die Angaben für *HighCyl* und *LowCyl* bei den beiden Partitionen anders. Die erste Partition hört bei Zylinder 310 auf, die zweite beginnt bei 311. (Das wichtigste bei Einträgen dieser Art ist, daß sich die Zylinderbereiche nie überschneiden!) Um diese beiden Partitionen nutzen zu können, müssen Sie beide einzeln mit dem Shell-Befehl MOUNT angemeldet werden:

```
MOUNT HD0:
MOUNT HD1:
```

Für die genaue Berechnung der Partitionsgröße in Kilobyte gibt es übrigens eine recht einfache Faustformel, mit der Sie sich Partitionen regelrecht maßschneidern können. Sie lautet:

Partitionsgröße = Zylinderanzahl * Anzahl Oberflächen * Block pro Sektor / 2

Bei einer Platte mit vier Oberflächen und 17 Sektoren pro Spur (*BlocksPerTrack*) hat also jeder Zylinder 34 Kbyte Speicherkapazität. Um eine Partition mit 5 Mbyte Kapazität einzurichten, müßte diese demnach etwa 148 Zylinder umfassen. Und eine Partition mit 26 Zylindern könnte etwas mehr als 880 Kbyte und damit eine komplette Diskette aufnehmen.

Alle bisher gezeigten Beispiel für Mountlist-Einträge gingen davon aus, daß für die Verwaltung der Partitionen das normale Dateiverwaltungssystem verwendet wird. Wenn Sie das FastFileSystem verwenden wollen, müssen Sie dem Eintrag (vor dem #) drei Zeilen hinzufügen:

```
     GlobVec = 1
     FileSystem = L:FastFileSystem
     DosType = 0x444F5301
#
```

Was diese drei Einträge zu sagen haben, ist im einzelnen nicht so wichtig, und nur für echte »Bitbohrer« interessant. Sie müssen genauso wie hier gezeigt an das Ende des Mountlist-Eintrags einer Partition gesetzt werden, damit das FastFileSystem mit dieser Partition verwendet werden kann (nachdem die Partition formatiert wurde).

Beachten Sie aber bitte, daß es mit den meisten autobootfähigen Controllern nicht möglich ist, von einer solchen Partition (die das FastFileSystem verwendet) zu starten. Deshalb wird die erste Partition solcher Platten von dem alten Dateiverwaltungsystem verwaltet.

Diese erste Partition einer autobootfähigen Platte wird sofort nach dem Einschalten des Amiga vom Kickstart-ROM (ab der Version 1.3) erkannt. Sie muß nicht mit dem MOUNT-Befehl angemeldet werden und es gibt deshalb für diese Partition auch keinen Eintrag in der *Mountlist*. Um die Größe dieser ersten Partition festzulegen, können Sie deshalb auch nicht die *Mountlist* verwenden. Statt dessen wird die Größe der Startpartition bei autobootfähigen Festplatten mit einem speziell für diesen Zweck gedachten Programm festgelegt. Manchmal kann dieses Programm auch dazu verwendet werden, die anderen Partitionen zu beschreiben. Manchmal muß dafür aber wie üblich die *Mountlist* geändert werden.

Das folgende Beispiel zeigt einen kompletten Mountlist-Eintrag für die zweite Partition einer autobootfähigen Festplatte:

```
HD1: Device = hddisk.device
     Unit = 0
     Flags = 0
     Surfaces = 4
     BlocksPerTrack = 17
     Reserved = 2
     Interleave = 0
     LowCyl = 29; HighCyl = 621
     Buffer = 5
     BufMemtype = 0
     GlobVec = 1
     FileSystem = L:FastFileSystem
     DosType = 0x444F5301
#
```

Dieser Eintrag läßt die Partition erst bei Zylinder 29 beginnen, weil davon ausgegangen wird, daß in den Zylindern 0 bis 28 die erste autobootfähige Partition liegt. Diese hat in etwa die Kapazität einer Diskette und dient meist auch dazu, den Inhalt der Workbench-Diskette aufzunehmen.

### 10.2.5 Die Festplatte formatieren

Bevor Sie eine Festplatte oder eine Partition einer Festplatte benutzen können, müssen Sie sie aber erst formatieren. Damit ist diesmal nicht die physikalische Formatierung gemeint, die ja nur von einem Spezialprogramm ausgeführt werden kann, sondern die »logische« Formatierung, die Amiga-DOS voraussetzt, um auf der Platte Daten ablegen zu können. Dazu verwenden Sie am besten den Shell-Befehl FORMAT.

Sie müssen jede Partition der Platte formatieren, bevor Sie sie benutzen können – nicht nur die erste. Sie sollten auch bei der Formatierung noch einmal angeben, ob das FFS-System benutzt werden soll oder nicht. Die folgenden beiden Befehle mounten und formatieren zum Beispiel eine Nicht-FFS-Partition:

```
MOUNT HD0:
FORMAT DRIVE HD0: NAME Platte NOFFS
```

Bei einer FFS-Partition müßte der entsprechende FORMAT-Befehl lauten:

```
MOUNT HD1:
FORMAT DRIVE HD1: NAME SchnellePlatte FFS
```

Nachdem Sie den FORMAT-Befehl eingegeben haben, fordert Sie dieser genau wie beim Formatieren einer Diskette auf, eine Diskette im entsprechenden Laufwerk einzulegen. Das ist bei einer Festplatte natürlich nicht nötig. Drücken Sie einfach auf [Return] und der Formatiervorgang beginnt. Danach können Sie dann beginnen, Daten auf die Festplatte zu legen. Beachten Sie aber bitte, daß die Festplatte oder die Partition Ihnen nach dem nächsten Neustart nur dann wieder zur Verfügung stehen, wenn Sie wieder den entsprechenden MOUNT-Befehl verwenden (oder diesen ein- für allemal in der *Startup-Sequence* unterbringen. Nur autobootfähige Festplatten (von denen zumindest eine Partition nicht mit dem FFS-System formatiert wurde), tauchen nach dem Neustart automatisch wieder auf.

In diesem Zusammenhang muß allerdings erwähnt werden, daß einige besonders trickreiche Festplattenhersteller diese Einschränkung umgangen haben. Bei den Festplatten der Firma Vortex tauchen beispielweise alle Partitionen ohne MOUNT-Befehl sofort nach dem Neustart auf und können auch alle mit dem FFS-System formatiert werden. Auch die FFS-Partitionen (beziehungsweise die erste davon) kann als Workbench-Diskette verwendet werden. Derart trickreiche Platten werden überhaupt nicht mit dem MOUNT-Befehl angemeldet. Statt dessen werden in diesem Fall alle Partitionen von einem Spezialprogramm festgelegt, daß auch dazu verwendet werden kann, die Partitionen zu ändern – wonach aber immer eine Neuformatierung nötig ist.

Dasselbe gilt prinzipiell auch für alle anderen Festplatten. Sollten Sie jemals die Partitionsaufteilung in der *Mountlist* ändern, müssen alle an der Neuordnung beteiligten Partitionen neu formatiert werden! Wenn die Änderung der Partitionsaufteilung »von Hand« und nicht mit einem Spezialprogramm erfolgt, sollten Sie besonders aufpassen, daß es niemals zu Überschneidungen zwischen den Zylinderbereichen der Partitionen kommt. (Spezielle Partitionsprogramme verhindern das meist automatisch.)

### 10.2.6 Die Workbench auf die Festplatte bringen

Wenn Sie Ihre neue Festplatte und die zugehörige Software erfolgreich installiert, in Partitionen aufgeteilt und formatiert haben, können Sie damit beginnen, es sich darauf häuslich einzurichten oder anders formuliert: Programme und Dateien darauf unterzubringen. In einem ersten Schritt sollten Sie zunächst einmal den kompletten Inhalt der Workbench-Diskette, mit der Sie den Amiga üblicherweise starten, auf die erste Partition der Festplatte legen (sofern mehrere Partitionen) vorhanden sind. Verwenden Sie dazu am besten den folgenden Befehl im Shell-Fenster:

```
COPY DF0: DH0: ALL
```

Dieser Befehl geht davon aus, daß sich die Workbench-Diskette im internen Laufwerk des Amiga befindet und die Festplatte den Gerätenamen HD0: trägt – was bei Ihnen nicht unbedingt zutreffen muß. Ändern Sie gegebenenfalls die beiden Gerätenamen.

Nun steht Ihnen eine Wartezeit bevor, während der Sie ruhig eine Tasse Kaffee trinken gehen können. Der oben genannte COPY-Befehl kopiert alle Dateien einzeln von der Workbench-Diskette auf die Festplatte. Bei mir dauerte das, bei einer nicht ganz vollen Workbench-Diskette und einer 30-Mbyte-ST-506-Festplatte, etwa 4 Minuten. Danach ist die Platte fast eine exakte Kopie der Workbench-Diskette – mit dem Unterschied, daß auf der Platte meist noch etwas mehr Platz ist.

Die Workbench-Diskette, die Sie auf die Festplatte kopieren, sollte dieselbe sein, die Sie zuvor modifiziert haben, um die Festplatte anmelden zu können. Auf dieser Diskette sollte sich zum Beispiel der Festplattentreiber und auch die eventuell geänderten Dateien *Mountlist* und *Startup-Sequence* befinden. Besonders bei autobootfähigen Platten beziehungsweise Partitionen ist das sehr wichtig. Aber auch bei nicht autobootfähigen Platten ist es empfehlenswert, daß diese Dateien auf der Start-Diskette und auf der Festplatte übereinstimmen.

Wie Sie nach dem Kopieren der Workbench-Diskette weiter verfahren, hängt entscheidend davon ab, was für eine Platte Sie verwenden und wie diese in Partitionen aufgeteilt ist. Wenn Sie eine »normale« autobootfähige Platte (und einen autobootfähigen Amiga mit den neuen Kickstart-ROMs) besitzen, sollte die erste Partition immer recht klein sein und nicht mehr enthalten als die normale Workbench-Diskette. Kopieren Sie bei einer solchen Festplatte jetzt am besten gleich noch einmal den kompletten Inhalt der ersten in die zweite Partition (wobei die Partitionsnamen bei Ihnen natürlich anders lauten können):

```
COPY HD0: HD1: ALL
```

Diese zweite Partition (die mit FFS formatiert sein sollte) wird dann in Zukunft Ihre Haupt-Arbeitspartition sein. Sie können auf diese Partition jetzt auch gleich den Inhalt der Extras-Diskette legen. Dazu können Sie zum Beispiel die folgenden Befehl verwenden (wobei Sie eventuell aufgefordert werden, die Extras-Diskette einzulegen):

```
COPY Extras1.3D: HD1: ALL
```

Wenn Ihre Festplatte nicht autobootfähig ist (oder wenn Sie eine Festplatte mit autobootfähigen FFS-Partitionen besitzen), sollte die erste Partition gleich mindestens 10 Mbyte groß sein oder die ganze Platte umfassen. Dann können Sie auf diese erste Partition den Inhalt der Extras-Diskette und aller anderen Disketten, mit denen Sie für gewöhnlich arbeiten, legen. Nur bei autobootfähigen Platten, bei denen FFS-Partitionen nicht autobootfähig sind, ist es sinnvoll, mit einer speziellen kleinen Start-Partition zu arbeiten, die gerade den Inhalt der Workbench-Diskette aufnehmen kann.

### 10.2.7 Grundsätzliches zum Start mit der Festplatte

Wenn Sie eine autobootfähige Festplatte besitzen, können Sie jetzt sofort einen Neustart durchführen. Nehmen Sie vorher am besten die Diskette aus dem internen Laufwerk, damit der Amiga für den Start auch garantiert die Festplatte verwendet. Dieser Startvorgang sollte deutlich schneller ablaufen als bei Verwendung einer Diskette.

Bei einer nicht autobootfähigen Festplatte müssen Sie erst noch einige Vorarbeiten leisten, um die Festplatte optimal zu nutzen. Daß die Workbench-Software sich nun auf der Festplatte befindet, genügt nämlich dazu nicht. Trotz des Vorhandenseins der Festplatte werden vorläufig sämtliche Amiga-Programme nach wie vor wichtige Dateien (zum Beispiel Schriften) auf der Workbench-Diskette suchen. Damit die entsprechenden Dateien von der Festplatte verwendet werden, muß eine ganze Reihe von logische Geräten mit dem ASSIGN-Befehl »umgelegt« werden. Dies sollte gleich nach dem Start, in der *Startup-Sequence* erfolgen. Ja, es ist sogar sinnvoll, die *Startup-Sequence* auf der Workbench-Diskette so weit abzuändern, daß diese lediglich noch die nötigen MOUNT- und ASSIGN-Befehle ausführt und danach sofort die Kontrolle an eine Startup-Sequence auf der Festplatte übergibt. Die notwendigen Änderungen an der Workbench-Diskette, die Sie durchführen müssen, um das zu erreichen, werden gleich noch im Einzelnen beschrieben.

Eine ähnliche Vorgehensweise wie bei nicht autobootfähigen Festplatten ist aber auch bei autobootfähigen Festplatten sinnvoll, deren erste (bootfähige) Partition nicht mit dem schnelleren FFS formatiert werden kann. Auch hier sollte die *Startup-Sequence* gleich nach dem Start eine Reihe von MOUNT- und ASSIGN-Befehlen durchführen und dann die Kontrolle an die schnellere und größe FFS-Partition übergeben.

### 10.2.8 Start mit nicht autobootfähigen Festplatten

Betrachten wir aber zunächst einmal den Fall der nicht autobootfähigen Festplatte, an deren erste Partition beim Start die Kontrolle übergeben werden und die danach als Workbench-Diskette verwendet werden soll. In diesem Fall müssen Sie zum Start zwar nach wie vor eine Diskette verwenden, der Startvorgang läuft aber trotzdem fast mit Festplatten-Geschwindigkeit ab, weil ein großer Teil der beim Neustart ablaufenden Arbeiten schon unter Zuhilfenahme der Platte ausgeführt wird.

Glücklicherweise müssen Sie die für eine solche Nutzung der Festplatte nötigen Änderungen an der *Startup-Sequence* nicht »von Hand« durchführen. Commodore hat eine für die Zusammenarbeit mit einer Festplatte geeignete *Startup-Sequence* schon unter dem Namen *Startup-Sequence.hd* im S-Directory auf der Workbench-Diskette hinterlegt. Sie sieht folgendermaßen aus:

```
SETPATCH
SYS:System/FastMemFirst
BINDDRIVERS
ASSIGN >NIL: DH0: EXISTS
IF NOT WARN
  ASSIGN SYS:    DH0:
  ASSIGN C:      SYS:c
  ASSIGN L:      SYS:l
  ASSIGN S:      SYS:s
  ASSIGN FONTS:  SYS:Fonts
  ASSIGN DEVS:   SYS:Devs
  ASSIGN LIBS:   SYS:Libs
  MAKEDIR RAM:tr
  ASSIGN T:      RAM:tr
  EXECUTE S:Startup-Sequence
```

```
ELSE
  EXECUTE S:Startup-Sequence.f
ENDIF
```

Diese *Startup-Sequence* führt lediglich die wichtigen Befehle SETPATCH, FASTMEM-FIRST und BINDRIVERS aus und testet dann mit ASSIGN, ob eine Festplatte namens DH0: existiert. Wenn ja, werden alle logischen Laufwerke auf diese Festplatte umgelegt – wobei die Kommandofolge davon ausgeht, daß auf der Festplatte dieselben Directories vorhanden sind, wie auf der originalen Workbench-Diskette. Nach Abschluß der ASSIGN-Befehle wird erneut *Startup-Sequence* aufgerufen. Das ist diesmal aber die *Startup-Sequence* auf der Festplatte, da ja vorher das logische Laufwerk S: umgelegt wurde.

In der hier vorgestellten Form ist *Startup-Sequence.hd* für Sie vielleicht nicht sofort verwendbar. *Startup-Sequence.hd* geht beispielsweise davon aus, daß die Festplatte schon durch den Befehl BIDNDRIVERS bei Amiga-DOS angemeldet wird und DH0: heißt. Wenn das bei Ihnen nicht der Fall ist, müssen Sie die folgenden Änderungen vornehmen:

Ändern sie jedes Vorkommen des Gerätenamens DH0: gegen den Gerätenamen Ihrer Festplatte, wie er in der *Mountlist* festgelegt wird (zum Beispiel HD0:) und fügen Sie dann hinter BINDDRIVERS einen entsprechenden MOUNT-Befehl ein; zum Beispiel:

```
MOUNT HD0:
```

Um die Ausführung der Befehle noch etwas zu beschleunigen, sollten Sie zudem gleich hinter dem IF-Befehl auch noch einen neuen CD-Befehl einfügen. Bei einer Festplatte mit dem Gerätenamen HD0: würde die Kommandofolge dann beispielsweise wie folgt aussehen:

```
SETPATCH
SYS:System/FastMemFirst
BINDDRIVERS
MOUNT HD0:
ASSIGN >NIL: HD0: EXISTS
IF NOT WARN
  CD HD0:c
  ASSIGN SYS:    HD0:
  ASSIGN C:      SYS:c
  ASSIGN L:      SYS:l
  ASSIGN S:      SYS:s
  ASSIGN FONTS:  SYS:Fonts
  ASSIGN DEVS:   SYS:Devs
  ASSIGN LIBS:   SYS:Libs
  MAKEDIR RAM:tr
  ASSIGN T:      RAM:tr
  EXECUTE S:Startup-Sequence
  ECHO "Sie können jetzt die Diskette herausnehmen!"
ELSE
  EXECUTE S:Startup-Sequence.f
ENDIF
```

Sie können diese Änderungen mit jedem Texteditor durchführen (zum Beispiel mit *Notepad* oder MEMACS).

Danach sollten Sie zunächst eine Sicherungskopie der originalen *Startup-Sequence* anlegen und diese dann durch *Startup-Sequence.hd* ersetzen. Das leisten die beiden folgenden Shell-Befehle:

```
COPY S:Startup-Sequence S:Startup-Sequence.f
COPY S:Startup-Sequence.hd S:Startup-Sequence
```

Jetzt können Sie mit der geänderten Diskette einen Neustart durchführen. Dabei sollte nach kurzer Zeit das Diskettenlaufwerk zur Ruhe kommen und nur noch die Platte aktiv sein. Wenn dann die Meldung »Sie können jetzt die Diskette herausnehmen!« erscheint, können Sie die Workbench-Diskette wirklich aus dem Laufwerk herausnehmen – ab jetzt ist die Festplatte die »Workbench-Diskette«. Sie brauchen die echte Workbench-Diskette erst wieder, wenn Sie den nächsten Neustart durchführen.

### 10.2.9 Start mit autobootfähigen Festplatten

Bei einer autobootfähigen Platte benötigen Sie zum Start keine Diskette – genau das ist ja der Sinn einer autobootfähigen Platte. Auch hier kann aber eine *Startup-Sequence*, wie die im letzten Abschnitt beschriebene nötig sein, da die erste Partition einer solchen Festplatte meist nicht mit dem FFS-Dateiverwaltungssystem formatiert werden kann. Man legt deshalb diese erste Partition nur mit der Größe einer Diskette (etwa 880 Kbyte) an und verwendet sie nur für den Start. Damit nach dem Startvorgang die schnellere FFS-Partition für alle weiteren Aufgaben verwendet wird, muß die *Startup-sequence* in dieser Partition im Prinzip genau so aussehen, wie die einer Startdiskette für eine nicht autobootfähige Platte. Diese *Startup-Sequence* könnte zum Beispiel wie folgt aussehen:

```
SETPATCH
SYS:System/FastMemFirst
BINDDRIVERS
MOUNT >NIL: FAST:
ASSIGN >NIL: FAST: EXISTS
IF NOT WARN
  CD FAST:c
  ASSIGN SYS:    FAST:
  ASSIGN C:      SYS:c
  ASSIGN L:      SYS:l
  ASSIGN S:      SYS:s
  ASSIGN FONTS:  SYS:Fonts
  ASSIGN DEVS:   SYS:Devs
  ASSIGN LIBS:   SYS:Libs
  MAKEDIR RAM:tr
  ASSIGN T:      RAM:tr
  EXECUTE S:Startup-Sequence
ELSE
  EXECUTE S:Startup-Sequence.f
ENDIF
```

Diese Fassung der *Startup-Sequence* geht davon aus, daß die zweite – mit FFS formatierte – Partition der Festplatte den Gerätenamen FAST: trägt und ebenfalls dieselben Directories

wie eine Workbench-Diskette enthält. (Insbesondere muß im S-Directory auch eine *Startup-Sequence* vorhanden sein.) Damit diese *Startup-Sequence* funktioniert, muß die Datei *Mountlist* im Directory *Devs* der ersten Partition auch einen Eintrag enthalten, der die zweite Partition FAST: beschreibt. Diesen Eintrag benötigt MOUNT. (Beispiele dafür, wie ein solcher Eintrag aufgebaut sein könnte, finden Sie in Abschnitt 10.2.4 dieses Kapitels.)

An dieser Stelle muß unbedingt auch noch einmal erwähnt werden, daß bei einer autobootfähigen Platte der Trick mit der besonderen Start-Partition und der speziellen *Startup-Sequence* nur dann nötig ist, wenn bei dieser Platte ein Start von einer FFS-formatierten Partition nicht möglich ist. Das ist leider zwar bei vielen, nicht aber bei allen autobootfähigen Platten der Fall. Falls auch von einer FFS-formatierten Partition der Festplatte gestartet werden kann, braucht diese überhaupt nicht partitioniert werden oder sie kann in zwei, drei gleichgroße Partitionen aufgeteilt werden, die alle mit dem FFS formatiert werden. Die erste dieser Partitionen wird dann beim Neustart verwendet. Diese kann deshalb eine ganz »normale« *Startup-Sequence* – ohne Übergabe der Kontrolle an eine andere Partition – enthalten.

### 10.2.10 Festplattenstart mit Hilfe der RAM-Disk RAD:

Das Angenehme einer autobootfähigen Festplatte ist, daß auf die Workbench-Diskette beim Start verzichtet werden kann. Das hat unter anderem den Vorteil, daß der Startvorgang wesentlich schneller abläuft. Auch Besitzer einer nicht autobootfähigen Festplatte können aber in den Genuß eines wesentlich schnelleren Startvorgangs kommen, wenn ihr Amiga über die neuen Kickstart-ROMs (Version 1.3 oder größer) verfügt und sie etwas Speicher opfern können. Für diesen kleinen Trick können Sie die resetfeste RAM-Disk RAD: einsetzen.

Diese RAM-Disk hat ja, wie in Kapitel 8 beschrieben, den Vorteil, daß ihr Inhalt bei einem Neustart erhalten bleibt und daß man deshalb sogar von ihr starten kann, wenn sie die nötigen Workbench-Dateien und -Directories enthält. Ihr Nachteil ist, daß Sie dauerhaft kostbaren RAM-Speicher benötigt, weshalb es meist nicht sinnvoll ist, RAD: als Workbench-Diskette zu gebrauchen. Der Speicherbedarf läßt sich aber sehr stark minimieren, wenn man die RAM-Disk ausschließlich dafür einsetzt, beim Neustart die Kontrolle an eine (nicht autobootfähige) Festplatte zu übergeben.

Dazu müssen Sie sich zunächst einmal eine Workbench-Diskette schaffen, die wirklich nur das absolute Minimum an Dateien enthält. Eine solche Workbench-Diskette wurde ebenfalls schon in Kapitel 8 vorgestellt. Wenn Sie sich nicht mehr genau daran erinnern, sollten Sie am besten jetzt einmal nachlesen, wie und nach welchen Gesichtspunkten die Dateien auf dieser Diskette gelöscht wurden. Die »Super-Mini-Workbench«, die für den Start der Festplatte benötigt wird, ist sogar noch um einiges kleiner, als die in Kapitel 8 beschriebene Mini-Workbench.

Um diese Super-Mini-Workbench-Diskette anzufertigen, sollten Sie sich zunächst einmal eine Kopie der Workbench-Diskette erzeugen, mit der Sie normalerweise starten, wenn Sie die Festplatte benutzen wollen. Löschen Sie dann von dieser Diskette alle Dateien bis auf die allernotwendigsten. Die folgende Liste zeigt Ihnen einen Überblick des Inhalts dieser Diskette, ähnlich wie sie der Befehl DIR ALL ergeben würde.

```
c (dir)
       assign              BindDrivers
       CD                  copy
       echo                Else
       EndIf               Execute
       If                  MakeDir
       Mount               SetPatch
l (dir)
       Disk-Validator      FastFileSystem
       Ram-Handler
devs (dir)
       Mountlist           ramdrive.device
       system-configuration hddisk.device (evtl. Festplatten-
treiber)
s (dir)
       Startup-Sequence
t (dir)
fonts (dir)
libs (dir)
       icon.library
Empty (dir)
Expansion (dir)
     disk.expansion (evtl. Festplattentreiber)
```

Diese Diskette sollte insgesamt nur noch etwa 90 bis 100 Kbyte Daten enthalten, weshalb auch die RAM-Disk RAD: nicht größer werden muß. Die genaue Größe richtet sich etwas nach den zusätzlichen Festplattentreibern, die Ihre Platte benötigt. Diese Treiberdateien liegen meist in den Directories *Devs* und/oder *Expansion* und müssen natürlich auch vorhanden sein. Weiterhin muß auch die Datei *Mountlist* Einträge für die RAM-Disk RAD: und für die Partition(en) Ihrer Festplatte enthalten.

Den Eintrag für RAD: in der *Mountlist* auf dieser speziellen Startdiskette sollten Sie jetzt erst einmal abändern. Änderungsbedürftig ist lediglich die Zahl hinter *HighCyl*, die die Größe von RAD: festlegt. Tragen Sie hier eine 8 ein. RAD: erhält dadurch eine Kapazität von 99 Kbyte, was in den meisten Fällen ausreichen sollte. Wenn Sie auf Nummer Sicher gehen wollen, können Sie auch eine 9 hinter *HighCyl* eintragen, was allerdings 11 Kbyte mehr RAM kostet.

Ändern Sie nun auch die *Startup-Sequence* auf dieser Diskette bis sie wie folgt aussieht:

```
SETPATCH
SYS:System/FastMemFirst
BINDDRIVERS
MOUNT >NIL: FAST:
MOUNT >NIL: RAD:
IF NOT EXISTS RAD:C
  ECHO "Kopiere Workbench-Diskette nach RAD ...."
  COPY DF0: RAD: ALL BUFFER=20 CLONE
  ECHO "RAD-Workbench komplett!"
  ECHO "Ab jetzt Neustart immer von RAD:!"
```

```
ENDIF
CD FAST:c
ASSIGN SYS:    FAST:
ASSIGN C:      SYS:c
ASSIGN L:      SYS:l
ASSIGN S:      SYS:s
ASSIGN FONTS:  SYS:Fonts
ASSIGN DEVS:   SYS:Devs
ASSIGN LIBS:   SYS:Libs
MAKEDIR RAM:tr
ASSIGN T:      RAM:tr
EXECUTE S:Startup-Sequence
```

Auch diese *Startup-Sequence* geht – wie die in Abschnitt – 10.2.9 davon aus, daß es auf der Festplatte eine Partition mit dem Gerätenamen FAST: gibt (die mit dem FFS formatiert ist). Sollte Ihre Festplatte beziehungsweise die erste Partition Ihrer Festplatte einen anderen Namen tragen, müssen Sie jedes Vorkommen von FAST: in der *Startup-Sequence* gegen diesen Namen ersetzen.

Wenn die neue *Startup-Sequence* dann komplett ist, können Sie den ersten Neustart mit der Super-Mini-Workbench versuchen. Dieser erste Neustart wird noch verhältnismäßig lang dauern, da dabei zunächst der Disketteninhalt nach RAD: kopiert wird. Danach wird – wie schon in Abschnitt 10.2.8 und 10.2.9 beschrieben – die Kontrolle an die Festplatte beziehungsweise eine Partition der Festplatte übergeben. In dem obigen Beispiel wird diese Partition FAST: genannt. Sie sollte alle Dateien und Directories enthalten, die auch auf der normalen Workbench-Diskette zu finden sind (und kann darüber hinaus noch beliebig viele zusätztliche Dateien und Directories enthalten).

Dieser erste langwierige Ablauf von *Startup-Sequence* sollte nach dem Einschalten des Amiga nur einmal nötig sein. Danach können Sie die Diskette aus dem Laufwerk entfernen. Beim nächsten Neustart startet der Amiga nämlich von der RAM-Disk RAD:. Dabei wird dieselbe *Startup-Sequence* ausgeführt (da diese ja von der Super-Mini-Workbench kopiert wurde). Diesmal ist auf RAD: das C-Directory schon vorhanden. Die Kommandofolge springt deshalb gleich zum CD-Befehl und übergibt die Kontrolle an die Festplatte. Solange Sie den Amiga nicht ausschalten, brauchen Sie also nie wieder eine Workbench-Diskette zum Start einlegen, sondern können allein mit RAD: und der Festplatte arbeiten.

Der einzige Nachteil dieser Methode, die die Arbeit mit einer nicht autobootfähigen Festplatte wesentlich angenehmer machen kann, ist, daß sie etwa 100 Kbyte RAM-Speicher kostet. Auf einem Amiga 500 mit Speichererweiterung dürfte das im allgemeinen zu verkraften sein. Mit nur 512 Kbyte RAM-Speicher wird es aber sicherlich knapp.

## 10.3 Arbeiten mit einer Festplatte

Eine Festplatte verhält sich im täglichen Gebrauch im Grunde wie eine Diskette – nur eben eine sehr große und schnelle Diskette. Die wesentlich größere Kapazität einer Festplatte hat aber nicht nur Vorteile, sondern bringt auch die eine oder andere unangenehme Konsequenz mit sich. Da wesentlich mehr Daten auf einer Festplatte als auf einer Diskette untergebracht

werden können, muß man sich zum Beispiel ein paar Gedanken darüber machen, wie man große Mengen von Dateien anordnet, ohne den Überblick zu verlieren.

### 10.3.1 Eine Alternative zur Installation der Workbench

In Abschnitt 10.2 wurde beschrieben, wie Sie mit einem einfachen COPY-Befehl im Shell-Fenster den kompletten Inhalt der Workbench-Diskette und der Extras-Diskette auf eine Festplatte oder Plattenpartition bringen können. Diese Art der Installation der Workbench-Software auf der Festplatte hat aber einen Nachteil. Sie erhalten dadurch viele Schubladen und einige einzelne Piktogramme im Fenster der Festplatte. Dadurch geht kostbarer Platz im Fenster verloren, der für die diversen Schubladen, die Sie bald auf die Festplatte legen werden, verloren ist. Unübersichtlichkeit des Fensterinhalts und langes Suchen werden die Folgen sein. Das ist um so unangenehmer, da ein Großteil der Schubladen von der Workbench-Diskette beziehungsweise ihr Inhalt bei der täglichen Arbeit nur selten gebraucht wird. Was läge also näher, als den Inhalt der Workbench-Diskette in einer eigenen Schublade zu verstecken, die sinnigerweise den Namen *Workbench* erhalten wird. Sie haben dadurch so gut wie keine Nachteile – müssen allerdings eine Schublade mehr öffnen, wenn Sie an ein Piktogramm in dieser Schublade heranwollen.

Das geht einfacher, als Sie vielleicht denken werden. Und am einfachsten läßt sich diese Aufgabe ausnahmsweise auf der Workbench und nicht im Shell-Fenster ausführen. (Im Shell-Fenster sind allerdings einige Nacharbeiten nötig.) Als ersten Schritt müssen Sie sich zunächst einmal eine neue Schublade auf der Festplatte schaffen. Klicken Sie dazu die Schublade *Empty* an und wählen Sie den Befehl *Duplicate* im Menü *Workbench*. Warten Sie, bis die neue Schublade erscheint und geben Sie ihr dann mit dem Befehl Rename den Namen *Workbench*. Legen Sie nun nacheinander alle anderen Schubladen in diese neue Schublade. Lediglich der Mülleimer, *Amiga-Basic*, die *Shell* und die Schubladen *System* sowie *Workbench* selbst sollten im Hauptfenster des Festplattenpiktogramms zurückbleiben. Wenn Sie auch den Inhalt der Extras-Diskette – wie oben beschrieben – auf die Festplatte gelegt haben, sollten Sie die Schublade *Workbench* jetzt gleich noch einmal öffnen und ein wenig »aufräumen«. Den Inhalt der Schublade *Tools* von der Extras-Diskette können Sie zum Beispiel genausogut in die Schublade *Utilities* legen (und *Tools* dann löschen). Verteilen Sie die anderen Schubladen dann möglichst übersichtlich im Fenster und setzen Sie sie mit dem Befehl *Snapshot* fest.

Jetzt sieht es sicherlich schon viel ordentlicher auf ihrer Festplatte aus. Bevor Sie diese neue Anordnung aber nutzen können, müssen Sie aber erst auch noch die Shell beziehungsweise Amiga-DOS davon in Kenntnis setzen. Einige der Schubladen, die Sie gerade verlegt haben, erwarten wichtige Kommandofolgen der Shell nämlich an anderer Stelle. Um diesen Irrtum aus der Welt zu räumen, müssen Sie jetzt ein Shell-Fenster öffnen und dann mit einem Texteditor die Datei *Startup-Sequence* und eventuell auch *StartupII* im S-Directory der Festplatte ändern.

Lesen Sie *Startup-Sequence* mit Ihrem Lieblingseditor ein. Suchen Sie dann den PATH-Befehl am Ende der Kommandofolge und ändern Sie diesen wie folgt:

```
PATH RAM: C: SYS:Workbench/Utilities SYS:System
   SYS:Workbench/Prefs ADD
```

Speichern Sie *Startup-Sequence* dann ab und öffnen Sie *StartupII*. Überprüfen Sie, ob in

dieser Kommandofolge Befehle vorkommen, die sich auf Directories beziehen, die jetzt in der Schublade *Workbench* liegen. Setzen Sie vor die Namen dieser Directories dann »Workbench/«.

Speichern Sie nun auch *StartupII*, warten Sie einen Moment und führen Sie dann einen Neustart durch. Wenn alle Änderungen korrekt ausgeführt wurden, sollte dieser Neustart problemlos ablaufen. Wenn Sie während des Startvorgangs Fehlermeldungen erhalten, haben Sie höchstwahrscheinlich an einer oder mehreren Stellen vergessen, »Workbench/« vor einen Directorynamen zu setzen.

### 10.3.2 Schubladenanordnung auf einer Festplatte

Wahrscheinlich ist es bei Ihnen aber mit dem Installieren der Workbench-Software auf der Platte nicht getan. Vor allem, wenn Sie schon eine Zeitlang mit dem Amiga gearbeitet haben, werden Sie sicherlich den Wunsch verspüren, auch alle anderen Programme und Dokumente auf die Platte zu legen. Nichts einfacher als das, werden Sie vielleicht denken. Das ist aber aus zwei Gründen heraus falsch. Viele Programme stammen aus einer Zeit, als es noch keine Festplatten gab, oder wurden unter der Annahme geschrieben, »wer sich einen Amiga 500 kauft, kann sich sowieso keine Festplatte leisten«. Es ist nicht immer ganz einfach, solche Programme auf Festplatten zu installieren. Zu diesem Thema finden Sie ein paar Tips im Abschnitt 10.4. Andererseits erhalten Sie bald aber auch ein heilloses Chaos auf der Festplatte, wenn Sie einfach alle Dateien von den Disketten, mit denen Sie bislang gearbeitet haben, auf die Festplatte legen.

Wichtig ist es, sich eine vernünftige Disziplin zuzulegen, nach der man die einzelnen Programme und Ihre Dokumente auf Schubladen beziehungsweise Directories verteilt. Eine große Hilfe ist es dabei schon, wenn Sie – wie oben beschrieben – einen Großteil der Workbench-Software in ein großes Directory (eine Schublade) namens *Workbench* gelegt haben.

Die danach auf der Platte vorhandenen Directories sollten Sie auch zum Abspeichern neuer Daten und Programme immer nutzen. Wenn Sie zum Beispiel neue Schriften kaufen, sollten Sie diese im Directory FONTS: installieren, neue Shell-Befehle oder Programme, die Sie in der Shell benutzen wollen im Directory C: und neue Druckertreiber im Directory DEVS:Printers. Auch kleine Hilfsprogramme für die Workbench, die Sie im Laufe der Zeit erwerben werden, sollten Sie möglichst sinnvoll auf die Schubladen *Utilities* und *System* verteilen.

Wenn Sie nicht mit sehr vielen verschiedenen Programmen arbeiten, würde ich Ihnen empfehlen, dann eine Schublade für jedes Programm anzulegen. In diese Schublade kommt das Programm selbst und die damit bearbeiteten Projekte (Dateien). Kommen mit der Zeit viele Projekte zusammen, sollten Sie innerhalb der Programmschublade weitere Schubladen anlegen und die Projekte in die verschiedenen Schubladen aufteilen. Wenn Sie also zum Beispiel ein intensiver Benutzer eines Textverarbeitungsprogramms sind, sollten Sie eine Schublade *Text* anlegen, in die Sie das Programm selbst legen. In der Schublade richten Sie weitere Schubladen ein, die zum Beispiel *Briefe*, *Memos*, *Gedichte* und so weiter heißen könnten.

Nicht bei jedem Programm ist eine solche Anordnung möglich. Viele Programmierer haben

in der Zeit, als es kaum Festplatten für den Amiga gab, nämlich Fehler gemacht. Mehr dazu im Abschnitt 10.4. Sie sollten aber trotzdem versuchen, dieses oder eine ähnliche Struktur einzuhalten. Oft reicht es schon aus, wenn man das Programm selbst aus der dafür vorgesehenen Schublade in das Fenster der Festplatte legt.

### 10.3.3 Sicherungskopien anlegen

Die vernünftige Aufteilung der Dateien auf der Festplatte in Schubladen ist auch für das regelmäßige Anfertigen von Sicherungskopien sehr wichtig. Das grundsätzliche Problem, das hierbei auftritt, ist die große Kapazität der Festplatte. Es ist deshalb, wenn Sie nicht spezielle Zusatzgeräte kaufen, unmöglich, eine Kopie der kompletten Festplatte anzulegen. Das ist meist aber auch gar nicht nötig. Grundsätzlich gilt zwar die Regel, daß von jeder Datei auf der Festplatte mindestens eine – besser zwei – Kopie(n) auf Diskette existieren sollten. Verschiedene Dateien haben aber verschiedene »Sicherungsbedürfnisse«. Diese Bedürfnisse ergeben sich aus der unterschiedlichen Änderungshäufigkeit der Dateien. Die meisten Programme und Systemdateien (zum Beispiel Shell-Programme, Bibliotheken und Druckertreiber) ändern sich so gut wie nie. Wenn Sie ein neues Programm erhalten, sollten Sie zunächst einmal eine Sicherungskopie der mitgelieferten Disketten machen und dann das Programm auf der Platte installieren (lesen Sie dazu Abschnitt 10.4). Sie haben dann einen Satz Original-Disketten, einen Satz kopierte Disketten und eine Kopie auf der Festplatte. Das sollte genügend Sicherheit geben.

Anders ist es mit den Dateien (zum Beispiel Texten oder Basic-Programmen), mit denen Sie täglich arbeiten. Solche Dateien sollten täglich mindestens einmal gesichert werden. Und damit Sie sich nicht immer überlegen müssen, welche Datei Sie denn geändert haben und welche nicht, schlage ich folgende Strategie vor:

Verteilen Sie Ihre Programme und deren zugehörige Dateien auf Schubladen, wie es in Abschnitt 10.3.2 beschrieben wurde. Teilen Sie dann jeder dieser Schubladen mindestens eine Diskette zu. Machen Sie sich zur Angewohnheit, jeden Abend alle Dateien der Programme, mit denen Sie tagsüber gearbeitet haben, auf die entsprechende Diskette zu kopieren. Wenn Sie diese Dateien schön zu Schubladen zusammengefaßt haben, ist das sowohl im Shell-Fenster wie auch auf der Workbench ein geringe Mühe, die Sie nicht scheuen sollten. Als versierter Shell-Anwender können Sie mit etwas Geschick vielleicht sogar eine Kommandofolge schreiben, die diesen Ablauf noch vereinfacht.

Probleme bekommen Sie mit dieser Methode erst dann, wenn die Dateien einen Umfang von mehr als 850 Kbyte annehmen. Dann passen sie nämlich nicht mehr alle auf eine Diskette. Spätestens dann sollten Sie beginnen, die mit einem Programm bearbeiteten Projekte auf verschiedene Schubladen (in einer umfassenden Programmschublade) aufzuteilen. Wenn Sie dann jeder dieser einzelnen Schubladen eine Diskette zuordnen, können Sie mit Ihren täglichen Sicherungen weitermachen wie bisher.

### 10.3.4 Spezielle Software und Hardware zum Sichern von Festplatten

Die oben beschriebene Strategie für das Anlegen von Sicherungskopien ist zwar unbestreitbar etwas umständlich, bei kleinen Datenmengen aber noch praktikabel und nicht unzumutbar. Wenn Sie jeden Tag mit vielen Programmen arbeiten oder sehr große Dateien bear-

beiten, wird sie aber immer unpraktischer. Spätestens, wenn die ersten Dateien auftauchen, die größer als 880 Kbyte sind, ist diese Methode am Ende. Bei Datenbank-Programmen sind Dateien dieser Größe übrigens nicht weiter unüblich.

Dann müssen sie einmal wieder in die Brieftasche greifen und sich zusätzliche Software und/oder Hardware kaufen. Es gibt zum Beispiel besondere Sicherungsprogramme, die eine komplette Kopie einer Festplatte auf Disketten anlegen können. Dazu werden Sie nacheinander aufgefordert, immer wieder neue Disketten einzulegen, auf die das Programm dann automatisch die verschiedenen Dateien der Festplatte verteilt. Ein solches Sicherungsprogramm kann auch übergroße Dateien in handliche Häppchen zerlegen, die auf die Diskette passen. Sollte dann einmal eine Katastrophe über die Platte kommen, weil zum Beispiel eines Ihrer selbstgeschriebenen Programme verrückt gespielt hat, brauchen Sie die Platte nur neu zu formatieren und dann den alten Inhalt mit Hilfe des Sicherungsprogramms (das sich natürlich auch auf einer Diskette befinden sollte) wieder zurückkopieren. Dabei werden automatisch die übergroßen Dateien wieder aus ihren Einzelteilen zusammengesetzt.

Bei einer 20-Mbyte-Festplatte kann es aber schnell passieren, daß für eine Komplettsicherung 20 oder mehr Disketten benötigt werden. Jeden Abend 20 Disketten in den hungrigen Schlitz des Amiga zu stecken und wieder herauszuholen ist aber ganz schön lästig. Einige Sicherungsprogramme bieten deshalb die Möglichkeit der »inkrementellen« oder »schrittweisen« Sicherung. Sie fertigen dabei zunächst erst einmal eine Komplettsicherung an. Danach werden täglich nur die Dateien gesichert, die nach der ersten Komplettsicherung oder nach der letzten Teilsicherung verändert wurden. Dazu benötigt das Programm nur Datum und Uhrzeit der letzten Änderung an einer Datei, was beim Amiga bei jeder Schreiboperation automatisch bei der Datei vermerkt wird. Bei einer solchen Teilsicherung werden oft nur zwei oder drei neue Disketten benötigt.

Auch die Methode der schrittweisen Teilsicherungen hat aber Ihre Nachteile. Der Stapel mit den Sicherungsdisketten wächst dabei nämlich genauso schrittweise und unaufhaltsam an. Sie werden deshalb nicht darum herumkommen, zumindest einmal die Woche eine neue Komplettsicherung zu machen, wobei Sie wieder mit 20 oder mehr Disketten jonglieren müssen. Im professionellen Bereich hat sich deshalb eine andere Lösung durchgesetzt. Dort verwendet man Magnetbandgeräte, »Streamer« genannt, die den kompletten Inhalt einer Festplatte auf eine austauschbare Kassette aufzeichnen können. Eine Komplettsicherung einer 20-Mbyte-Platte (es sind auch hier inkrementelle Sicherungen der Änderungen möglich) dauert zwar auch mit einem solchen Gerät 20 Minuten. Sie kann aber unbeaufsichtigt erfolgen. Statt Disketten zu wechseln, können Sie sich also schon mit angenehmeren Dingen beschäftigen.

Schneller als Streamer und ähnlich gut für Sicherungszwecke geeignet sind Wechselplattenlaufwerke, die mit austauschbaren, von kompakten Gehäuse umgebenen Festplatten arbeiten und praktisch wie Diskettenlaufwerke sehr großer Kapazität (20 – 40 Mbyte) arbeiten.

Streamer und Wechselplatten sind zweifellos die komfortabelste Art, Sicherungskopien einer Festplatte anzufertigen, aber auch die teuerste. Sie sind nur aber dann zu rechtfertigen, wenn Sie täglich große Datenmengen sichern müssen und die Daten große Bedeutung für Sie haben (wie zum Beispiel Buchhaltungsdaten). Ansonsten ist eine Sicherung mit Disketten – eventuell zusammen mit speziellen Sicherungsprogrammen – aus Kostengründen vorzuziehen.

Wenn Sie aber sicher wissen, daß Sie einen Streamer oder ein Wechselplattenlaufwerk benötigen werden, sollten Sie von vornherein eine Festplatte mit SCSI-Controller kaufen (wie zum Beispiel die A 590 von Commodore). An diesen Controller können Sie neben der Festplatte nämlich relativ problemlos weitere Geräte wie Streamer und Festplatten anschließen.

## 10.4 Programme auf die Festplatte bringen

Bevor Sie aber die Programme auf der Festplatte sichern können, müssen Sie sie erst einmal auf die Festplatte bekommen. Und das ist leider nicht immer ganz so einfach, wie es eigentlich sein sollte. Oft wird es vorkommen, daß Sie das Piktogramm eines Programms guten Willens auf die Festplatte legen, dann starten und es erscheint nur eine Fehlermeldung oder das System stürzt ab. Noch geheimnisvoller sind die Fälle, bei denen der erste Start des Programms scheinbar problemlos klappt, einen Tag später aber nur noch Abstürze (Gurus) produziert.

### 10.4.1 Mögliche Probleme

Solche Fehler können zwei verschiedene Ursachen haben (die auch kombiniert auftreten): Zum einen gibt es Programme, die eine Reihe von (meist) unsichtbaren Hilfsdateien benötigen und diese in bestimmten Directories erwarten. Gut geschriebene Programme merken das Fehlen dieser Hilfsdateien und tun Ihnen das mit einem Requester kund. Schlecht geschriebene Programme überprüfen nicht, ob sich die Hilfsdateien dort befinden, wo sie gesucht werden, machen einfach weiter und stürzen dann meist kläglich ab.

Neben dieser Sorte von Programmen gibt es noch die kopiergeschützten Programme, die oft eine ganze Reihe von Problemen im Umgang mit Festplatten haben. In manchen Fällen ist es nahezu unmöglich, solche Programme auf die Festplatte zu bringen. Man muß sie dann weiterhin von der Diskette starten – was natürlich ärgerlich ist, wenn man frischgebackener Besitzer einer 1200,– DM teuren Festplatte ist.

Ich will damit nicht behaupten, daß alle Programme beziehungsweise Programmierer sich so »schlecht benehmen« würden. Viele Programme, wie zum Beispiel Amiga-Basic und die bekannten Grafik-Programme der Firma Aegis lassen sich völlig problemlos auf eine Festplatte legen. Viele andere Programme – speziell ältere – machen aber wirklich mehr oder weniger große Schwierigkeiten. Sie sollten zunächst natürlich immer versuchen, ein Programm auf einer Platte zu installieren, indem Sie einfach sein Piktogramm hinüberlegen. Funktioniert das Programm dann sofort und auch am folgenden Tag noch, ist alles o.k. Funktioniert es aber nicht hundertprozentig, können Ihnen die folgenden Abschnitte vielleicht helfen.

### 10.4.2 Vorsichtsmaßnahmen

Wenn Sie ein neues Programm gekauft haben, das nach der Installation auf der Festplatte nicht hundertprozentig funktioniert, sollten Sie sich zunächst einmal die Original-Programmdiskette im Shell-Fenster ansehen. (Auf der Workbench bringt eine solche Untersu-

chung leider meist nichts.) Überprüfen Sie vor allem die Directories *c*, *l* und *libs*. Wenn sich darin Dateien befinden, die auf einer normalen Workbench-Diskette nicht vorkommen, sollten Sie diese Dateien in die entsprechenden Directories auf der Festplatte kopieren und dann noch einmal versuchen, das Programm zu starten.

Das Programm *DeluxeVideo* der Firma Electronic Arts enthält zum Beispiel im Directory *libs* einige zusätzliche Bibliotheken, die bei der Installation in das logische Laufwerk LIBS: (also in das Directory*Libs* auf der Festplatte) kopiert werden müssen.

Finden Sie auf einer Programmdiskette ein Directory, das es auf der Workbench-Diskette nicht gab, müssen Sie das Directory samt Inhalt auf die Platte bringen. Es gibt zwei Stellen, an denen ein Programm dieses Directory erwarten könnte. Sie sollten diese Stellen nacheinander ausprobieren, bis Sie Erfolg haben. Die erste ist das Wurzeldirectory (:) der Festplatte. Der zweite mögliche Ort, an dem ein Programm ein Zusatzdirectory erwarten könnte, ist das Directory (die Schublade), in dem sich das Programm selbst befindet.

Wenn auch dieser Versuch kein Ergebnis bringt, sollten Sie nach unbekannten Dateien im Wurzeldirectory der Original-Diskette suchen. Legen Sie auch diese auf die Festplatte, wobei Sie nacheinander wieder die zwei möglichen Orte durchprobieren sollten, die im letzten Absatz beschrieben wurden.

Sollte das Programm nun immer noch nicht laufen, haben Sie wirklich Pech! Denn jetzt fällt mir kein weiterer Tip mehr ein. Oder höchstens dieser: Wenden Sie sich an die Firma, von der Sie das Programm gekauft haben und verlangen Sie Hilfe! Rein rechtlich ist die Firma wahrscheinlich meist nicht dazu verpflichtet, Ihnen bei solchen Problemen zu helfen, wenn auf der Programmpackung nicht ausdrücklich steht, daß das Programm auch auf einer Festplatte installiert werden kann. Freundlicherweise wird man Ihnen aber doch oft helfen (wenn man kann), um einen zufriedenen Kunden zu erhalten oder zu gewinnen. Bei Problemen mit Programmen, die Sie »bei einem Freund/Kollegen kopiert haben« sollten Sie das aber besser nicht versuchen. Da reagieren Händler oft – und das zu Recht – etwas unwirsch.

### 10.4.3 Die Verbindung zwischen Programmen und Projekten erhalten

Ein anderes Problem bei der Installation eines Programms auf der Festplatte ist oft, daß die Verbindung zwischen Projekten und Programmen zerstört wird. Das heißt, wenn Sie einen Doppelklick auf einem Projekt-Piktogramm machen, erscheint nur eine Fehlermeldung, während früher bei Diskettenbetrieb das entsprechende Programm gestartet wurde. Es ist also nicht mehr möglich, Programme über ihre Projekte zu starten.

In so einem Fall sollten Sie zunächst einmal das Info-Fenster des wiederspenstigen Projektes öffnen. Im Feld *Default Tool* ist hier nämlich gespeichert, welches Programm gestartet werden soll, wenn man einen Doppelklick auf das Piktogramm des Projekts macht. Wenn dort zum Beispiel steht *DPaint:DPaint* (wie bei den Projekten des Programms Deluxe-Paint), dann heißt das, daß die Workbench bei einem Doppelklick auf das Piktogramm versuchen soll, das Programm *DPaint* auf der Diskette *DPaint* zu starten. Sie könnten nun natürlich das *DPaint:* gegen den Pfadnamen der Schublade ersetzen, in der sich das Programm *DPaint* wirklich befindet. Das müßten Sie aber für jedes Projekt des Programms tun. Und beim nächsten Abspeichern eines Projektes würde das Programm außerdem wieder

sein *DPaint:DPaint* in das Feld *Default Tool* setzen und Ihre Arbeit zunichte machen. Sie sollten sich deshalb eine grundsätzlichere Lösung überlegen.

Das Einfachste ist es, der Workbench vorzugaukeln, daß die Schublade, in der das Programm *DPaint* liegt, in Wirklichkeit eine Diskette mit dem Namen *DPaint* wäre. Das ist einfacher als Sie vielleicht denken. Sie brauchen dazu nur den folgenden Befehl im Shell-Fenster einzugeben (wenn diese Schublade den Pfadnamen *DH0:Deluxe* hat):

```
ASSIGN DPAINT: DH0:Deluxe
```

(Lesen Sie im Zweifelsfall noch einmal die Beschreibung des ASSIGN-Befehls nach.) Wenn Sie mit diesem Programm häufiger arbeiten, sollten Sie den Befehl natürlich am besten in die Datei *Startup-Sequence* oder *StartupII* legen.

So oder so ähnlich lassen sich viele Programme überlisten. Schwieriger ist es schon bei Programmen, wie zum Beispiel *Amiga-Basic*, die bei Ihren Projekten *:AmigaBASIC* im Feld *Default Tool* hinterlegen. Im Fall von Amiga-Basic müssen Sie das Piktogramm *AmigaBASIC* wirklich in das Fenster der Festplatte (also in die oberste Directory-Ebene) legen. Wenn Sie mehrere solcher Programme haben, kann es im Festplatten-Fenster ganz schön voll werden.

### 10.4.4 Ein Traum: Selbstinstallierende Programme

Vielleicht habe ich aber in den vergangenen Abschnitten mit den geschilderten Problemen übertrieben. Vielleicht kommen die Probleme alle daher, daß sich die Programmhersteller noch nicht genügend mit Festplatten beschäftigt haben und sind bald alle aus der Welt geschafft. Zum Abschluß möchte ich Ihnen deshalb noch vorstellen, wie die Installation neuer Software auch aussehen könnte – und wie Sie auf anderen Computern, die schon längere Zeit über Festplatten verfügen, auch aussieht. Auch für den Amiga gibt es schon das eine oder andere »selbstinstallierende« Programm dieser Art. Solche Programme oder besser Programmpakete sehen folgendermaßen aus:

Zusammen mit dem eigentlichen Programm liegt auf der Original-Diskette ein Piktogramm namens *InstallierMich*. Macht man einen Doppelklick darauf, erscheint ein Fenster, in dem einige Fragen gestellt werden. Diese könnten zum Beispiel den Namen der Festplatte erfragen, die Schublade, in die das Programm gelegt werden soll und so weiter. Hat das Programm *InstallierMich* alle nötigen Informationen zusammen, legt es das zu installierende Programm und sämtliche Hilfsdateien in die richtigen Directories auf der richtigen Festplatte. Am Ende vermerkt es beim Programm auch noch, in welchem Directory es liegt, damit dieses den richtigen Pfadnamen in das Feld *Default Tool* seiner Projekte schreiben kann.

Wäre das nicht schön? Hoffentlich bleibt diese Wunschvorstellung kein Traum!

# 11 Chips und Tricks

Mit diesem Kapitel beginnt der dritte Teil des vorliegenden Buchs. Während Sie in den beiden vorangegangenen beiden Buchteilen hauptsächlich eine Anleitung zur praktischen Arbeit mit dem Amiga 500 auf der Workbench und in der Shell gefunden haben, dienen die letzten drei Kapitel mehr dazu, Ihren Wissensdurst zu stillen. Sie werden zum Beispiel in diesem Kapitel einiges über die Grafik-Hardware und -Software des Amiga 500 erfahren. Leider konnte ich mich aber nicht dazu durchringen, zu diesem Thema gleich Bauanleitungen beizulegen. Wer also nach Schaltplänen und fertigen Maschinencodeprogrammen sucht, wird das – zumindest in diesem Buch – vergeblich tun.

Wer jedoch mehr darüber erfahren möchte, was den Amiga 500 zu einem so ungemein leistungsfähigen und schnellen Computer macht, oder was noch alles »in ihm steckt«, das die bisher vorgestellte Software nicht demonstrieren konnte, der möge weiterlesen. Ich habe mir Mühe gegeben, Fachjargon weitgehend zu vermeiden, so daß vielleicht auch der »Computer-Anfänger« diesem und den folgenden Kapiteln das eine oder andere Interessante über die Innereien seines Amiga 500 entnehmen kann. Zunächst einmal geht es in diesem Kapitel um die grafischen Fähigkeiten des Amiga 500 – sowie natürlich auch seiner Brüder beziehungsweise Schwestern Amiga 1000 und Amiga 2000.

## 11.1 Grundlagen der Amiga-Grafik

Eine der prominentesten Eigenschaften des Amiga sind sicherlich seine grafischen Fähigkeiten, die – zumindest für einen Computer dieser Preisklasse – sehr ungewöhnlich sind. Vor allem durch ein paar Spezial-Chips, die jeder Amiga enthält, kann er viele Arten von Grafiken ungewöhnlich schnell aufbauen. Zudem verfügt auch schon das einfachste Modell der Amiga-Familie, der Amiga 500, über eine Vielzahl verschiedener grafischer Betriebsarten, die es dem Benutzer oder dem Programmierer erlauben, die jeweils optimale Lösung für nahezu jedes Problem grafischer Darstellung zu finden. Wer viele Bildpunkte benötigt, bekommt viele Bildpunkte und wer möglichst viele Farben möchte, kann auch dafür einen geeigneten Grafikmodus finden.

### 11.1.1 Das digitale Prinzip

Um zu verstehen, wozu der Amiga 500 in grafischer Hinsicht fähig ist (und wozu nicht), benötigt man zunächst einmal ein paar Informationen über die Grundlagen der Computergrafik »an sich« und über einige grundsätzliche Eigenschaften eines Computers. Zu diesen grundlegenden Eigenschaften gehört – zumindest bei allen gebräuchlichen Computern – das »digitale Prinzip«. Das heißt nichts anderes, als daß ein Computer alles in kleinen Schritten, man sagt dazu auch »diskret« oder »digital«, macht. Es gibt keine fließenden Bewegungen, sondern immer nur kleine Sprünge, keine wirklich runden Kreise, sondern nur Vielecke. Jeden Zustand, den ein Programm berücksichtigt oder zeigt, speichert der Computer intern als eine ganze Zahl ab.

Digitaluhren sind ein gutes Beispiel für dieses digitale Prinzip. Sie zeigen vielleicht Stunden, Minuten und Sekunden an, kennen aber keine feinere Unterteilung (von sportlichen Uhren mit Zehntel- oder Hundertstelsekunden-Teilung einmal abgesehen). Für die Sekunden ist intern in diesen Uhren ein Teil in der Schaltung der Uhr vorgesehen, der genau 60 verschiedene Zustände annehmen kann, beziehungsweise 60 verschiedene Zahlen speichern kann. Eine solche Schaltung kennt keine Zehntelsekunden und keine Zahlen zwischen 59 und 60. Sie bewegt sich nicht langsam von der 59sten auf die 60ste Sekunde zu, sondern mit einem Sprung – genau wie es die Sekundenziffern der Uhr auch anzeigen.

### 11.1.2 Pointilismus und digitale Farben

Solche Auswirkungen des digitalen Prinzips, Sprünge also, gibt es auch bei der Computergrafik. Während es in der Natur zum Beispiel nahezu unendlich viele Farben gibt (von denen das menschliche Auge aber nur endlich viele unterscheiden kann), steht einem Computer immer nur eine ganz bestimmte Anzahl von Farben zur Verfügung.

Der Amiga kann zum Beispiel etwas mehr als 4000 (genau 4096) verschiedene Farben darstellen. Er kann nicht zwei von diesen Farben nehmen, sie mischen und so eine neue erzeugen. Man ist immer auf diese 4096 Farben festgelegt. Mehr dazu aber in einem folgenden Abschnitt.

Eine ähnliche Einschränkung wie bei der Anzahl möglicher Farben gibt es hinsichtlich der Anzahl von einzelnen Punkten, aus denen ein Bild aufgebaut wird. Anders als ein Bild, das man mit Farbe auf einem Blatt Papier gemalt hat, besteht ein Computerbild nämlich nicht aus mit Farbe gefüllten Flächen, sondern aus einer Unzahl von (eventuell verschiedenfarbigen) einzelnen Punkten, die erst dann zu zusammenhängenden Flächen verschwimmen, wenn man etwas vom Bildschirm zurücktritt. Es gibt eine Kunstrichtung, den »Pointilismus«, bei der Bilder ähnlich entstehen. Auch dort werden kleine Punkte in verschiedenen Farben auf die Leinwand plaziert. Mit etwas Abstand werden aus diesen Punktwolken dann farbige Flächen.

Im Gegensatz zum Pointilismus können die Punkte einer Computergrafik aber auch nicht an beliebigen Stellen auf dem Bildschirm liegen, sondern sind in einem ganz bestimmten gitterähnlichen Muster aus Zeilen und Spalten angeordnet. Sie liegen praktisch in einer Art »Raster«, weshalb diese (die heute übliche) Art von Computergrafik auch »Rastergrafik« heißt.

Bei der echten »hochauflösenden« Rastergrafik kann jeder beliebige Punkt des Rasters einzeln mit Farbe versehen werden. Jeder Punkt kann eine andere Farbe als seine »Nachbarn« haben. Zwischen diesen Punkten aber kennt ein Computer nichts. Eine Fläche, die kleiner ist als ein solcher Punkt, kann nicht dargestellt werden.

Dies bedeutet prinzipiell noch keinen Unterschied gegenüber anderen Medien der Grafik. Auch Filmmaterial besitzt eine Körnung, und deckende Farben bestehen aus einzelnen Pigmenten in einer flüchtigen (verdunstenden) Lösung. Im Buchdruck wird auch mit einem Raster gearbeitet, in dem mit vier Grundfarben und kleinen Punkten nahezu beliebige Bilder erzeugt werden. Der Hauptunterschied zwischen diesen Grafikmedien und dem Computerbildschirm liegt nur in der extrem regelmäßigen Anordnung der Punkte und vor allem in ihrer Größe! Während man die Körnung einer Foto-Emulsion nur mit einer starken Lupe erkennen kann, sind die Punkte einer Computergrafik, man nennt sie auch »Pixel«, meist deutlich zu erkennen. Gehen Sie einfach einmal etwas näher an den Bildschirm des Amiga heran und Sie werden sehen, daß die Bilder, die er Ihnen zeigt, aus kleinen, fast rechteckigen Punkten bestehen, die deutlich zu unterscheiden sind.

### 11.1.3 Das Prinzip der »bitmapped« Grafik

Welche Farbe ein bestimmter Punkt auf dem Bildschirm hat, merkt sich der Computer in seinem Arbeitsspeicher. Wie bei Computern üblich, kodiert er die Farbe als Zahl. Das heißt, für jeden Punkt auf dem Bildschirm reserviert ein Computer, der über Rastergrafik verfügt, einen Platz in seinem Arbeitsspeicher, in dem er jederzeit nachsehen kann, welche Farbe dieser Punkt hat. Um einem Punkt eine andere Farbe zu geben, braucht ein Programmierer nur die dem Punkt entsprechende Stelle im Speicher zu finden (zu berechnen) und dort eine andere Zahl einzutragen.

Da Computer sehr sparsam mit ihrem Arbeitsspeicher umgehen müssen, reservieren sie natürlich so wenig Speicherplatz wie möglich für die Zahlen, die den Farben der einzelnen Bildschirmpunkte entsprechen. Am »sparsamsten« in dieser Hinsicht ist reine Schwarzweiß-Grafik (oder besser zweifarbige Grafik). Hierbei kann jeder Punkt am Bildschirm (jedes Pixel) nur zwei Zustände haben, Schwarz und Weiß (oder Schwarz und Grün, Schwarz und Gelb und so weiter), und der Computer braucht sich nur zwei verschiedene Zahlen für diesen einen Punkt zu merken. Und das wiederum kann er schon mit einem einzigen »Bit«, der kleinsten Speichereinheit, die ein Computer kennt.

Das legendäre Bit – Sie haben bestimmt schon einmal davon gehört – kann genau zwei Zustände annehmen, die man wahlweise mit »An« und »Aus«, »Wahr« und »Falsch«, 1 und 0 oder eben auch mit »Schwarz« und »Weiß« bezeichnen kann. Von diesen Bits, in denen sich der Computer die Farbe eines Pixels merkt, kommt der Name »bitmapped Grafik« (zu deutsch etwa »Bitkarten-Grafik«) für die spezielle Unterart der Rastergrafik, die die meisten grafikfähigen Computer heutzutage benutzen. Der Computer reserviert bei dieser Art Grafik einen bestimmten Bereich in seinem Arbeitsspeicher, der praktisch als Karte für das Bild dient. Genau wie bei einer Karte ein kleiner Punkt einer Ortschaft entspricht, entspricht bei einer Bitmap (Bitkarte) ein Bit einem Pixel am Bildschirm.

In Wirklichkeit liegen die Bits für ein Bild im Speicher natürlich nicht in einer rechteckigen Anordnung wie in Bild 11.1, sondern hintereinander. Die rechteckige Darstellung hilft einem aber, etwas leichter zu verstehen, welches Bit welchem Bildschirmpunkt entspricht.

Wie die verschiedenen Bits hintereinander im Speicher liegen, zeigt Bild 11.2. Sie sehen in dieser Abbildung zugleich, in welcher Reihenfolge der Elektronenstrahl Ihres Monitors oder Fernsehers über den Bildschirm streicht, um so ein Bild zu erzeugen. Wenn Sie den durchgezogenen Linien folgen, durchlaufen Sie die Bits (Kästchen) in genau der Reihenfolge, in der sie hintereinander im Speicher liegen. In genau dieser Reihenfolge liest die Elektronik des Amiga diese Bits und steuert anhand des jeweiligen Inhalts den Elektronenstrahl. Ist das Bit an (= 1), wird der Elektronenstrahl verstärkt und der Punkt am Bildschirm hell. Ist das Bit aus (= 0), wird der Elektronenstrahl gedrosselt und der Punkt am Bildschirm bleibt dunkel. Die gestrichelten Linien in der Abbildung entsprechen den Zeiten, in denen der Elektronenstrahl immer gedrosselt bleibt, während es zum Anfang einer neuen Zeile geht oder zum Anfang des Bildes, nachdem er es einmal komplett auf den Bildschirm gezeichnet hat.

| | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 |
| 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 |
| 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 1 |
| 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 |
| 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 1 |
| 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 |
| 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 1 |
| 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 |
| 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |

***Bild 11.1:*** *Eine Bitkarte für den Bildschirm*

### 11.1.4 Farbe ins Bild

Soll ein Computer aber nicht nur zwei Farben am Bildschirm darstellen können, sondern mehr (wie Ihr Amiga), müssen für jeden Bildschirmpunkt mehr als nur zwei verschiedene Zahlen gespeichert werden können. Auch dabei geht man natürlich so ökonomisch wie möglich vor und teilt einem Bildschirmpunkt immer eine bestimmte, möglichst kleine Anzahl von Bits zu. Das Bitkarten-Prinzip stimmt also noch, nur entsprechen mehrere Bits einem Pixel.

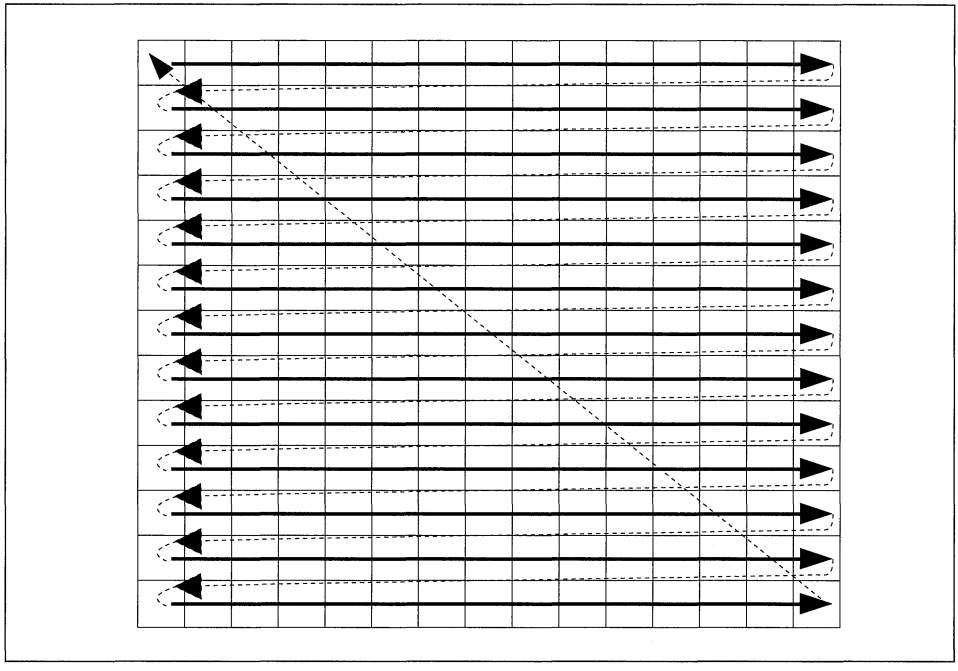


***Bild 11.2:*** *Reihenfolge, in der die Bits im Speicher liegen*

Nun müssen Sie ein wenig in die Tiefen der »Dualzahlen« eindringen, der Zahlen, die nur aus Bits oder Nullen und Einsen bestehen. Nur dann können Sie verstehen, wieviele verschiedene Zahlen (Farben) man mit einer bestimmten Anzahl Bits darstellen kann. Sie haben ja bereits gesehen, daß man in einem Bit zwei verschiedene Zustände beziehungsweise zwei Zahlen speichern kann. Man nennt diese beiden Zahlen meist nicht 1 und 2, sondern 0 und 1. Hat man statt einem Bit zwei zur Verfügung, kann man damit schon vier (2*2 oder $2^2$) verschiedene Zustände speichern, 00, 01, 10 und 11. Dies entspricht den (Dezimal-)Zahlen 0, 1, 2 und 3. Nimmt man gar drei Bits, reicht das zum Speichern von 8 verschiedenen Zahlen (2*2*2 oder $2^3$), vier Bits reichen für 16 Zahlen (2*2*2*2 oder $2^4$) und so weiter.

Diese Art der Kodierung von Zahlen (Farben) ist der Grund, warum Computer nie 5, 10 oder 100 Farben bieten, sondern immer 2, 4, 8, 16, 32 und so weiter. Der Speicher des Computers wird auf diese Weise am besten ausgenutzt. Mit vier Bit kann man nun einmal 16 Farben darstellen und mit dreien 8. Es gibt keine halben Bits und auch keine Möglichkeit, einen Speicherbereich zu reservieren, der genau 10 verschiedene Zahlen speichern kann. Das ist so unmöglich wie eine Digitaluhr, deren Sekundenanzeige halbe Sekunden anzeigt.

### 11.1.5 Wie Farben auf den Bildschirm kommen

Welche Farben kann ein Computer aber überhaupt auf den Bildschirm bringen und wie? Die erste Frage ist relativ leicht zu beantworten: Im Prinzip alle. Warum die Einschränkung »im Prinzip« gemacht werden muß, erklärt die Antwort auf die zweite Frage. Dazu muß man allerdings etwas weiter ausholen.

Grundsätzlich verwendet ein Computer (-Monitor) nämlich dasselbe Prinzip, um Farben zu zeigen, wie auch Ihr Farbfernseher. Er mischt durch »additive Farbmischung« aus den drei Grundfarben Rot, Grün und Blau die gewünschte Farbe zusammen. Betrachten Sie sich Ihren Monitor oder Farbfernseher einmal näher, dann werden Sie sofort verstehen, wie das gemeint ist. In einer scheinbar weißen Fläche werden Sie dann nämlich drei verschiedenfarbige Arten von leuchtenden Punkten erkennen, rote, grüne und blaue. Bei manchen Bildschirmen sind sie rund, bei anderen schlitzförmig. Wenn alle drei Punkte gleichzeitig leuchten, ergibt das eine weiße Fläche, sobald Sie etwas zurücktreten.

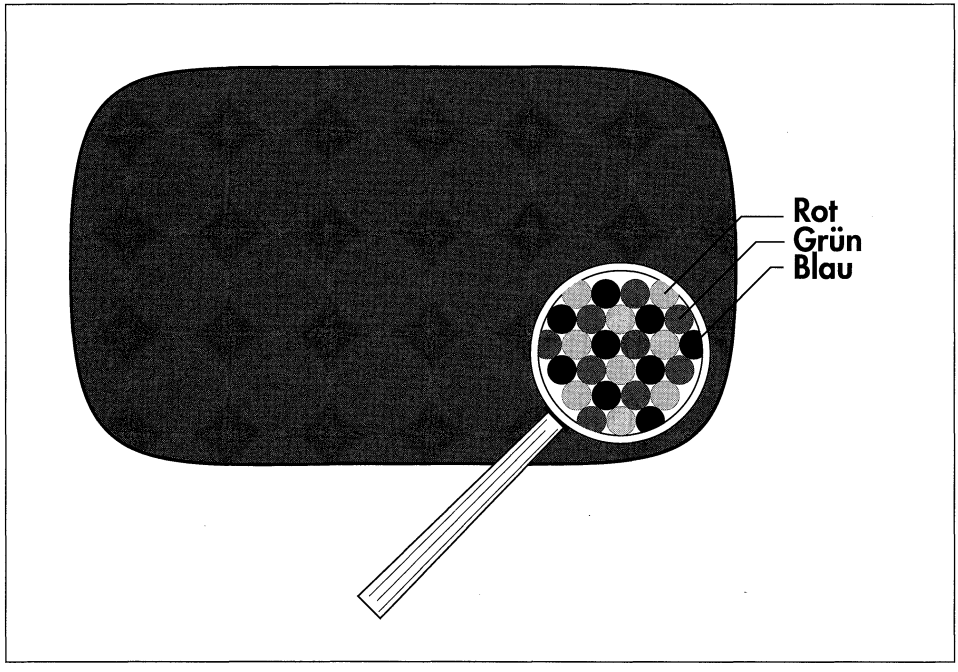



***Bild 11.3:*** *Das Grundprinzip des Farbmonitors (und des Farbfernsehens)*

Jeder Bildschirmpunkt besteht deshalb aus (mindestens) drei dieser leuchtenden Punkte. Jeder dieser drei Punkte kann von drei verschiedenen Elektronenstrahlen, mit denen ein Farbfernseher arbeitet, getrennt zum Leuchten gebracht werden. Je nachdem, wie hell jeder der drei Leuchtpunkte leuchtet, aus denen sich ein Bildschirmpunkt zusammensetzt, ergeben sich verschiedene Farben.

Leuchten alle drei Punkte gleich hell, ergibt das Weiß oder einen Grauton. Sind alle drei Punkte dunkel, bleibt der Punkt schwarz. Durch feine Abstufungen der Leuchtintensitäten kann man alle Farbtöne erzeugen, die man sich nur vorstellen kann. Mit dem Werkzeug *Preferences* und den drei Schiebereglern im Hauptfenster können Sie das ausprobieren. Diese drei Schieberegler bestimmen die Rot-, Grün- und Blau-Anteile der vier Workbench-Farben.

### 11.1.6 Malen nach Zahlen

Im Gegensatz zu einem Farbfernseher kann ein Computer aber nicht beliebige Farben erzeugen. Er arbeitet ja »digital« und speichert zu jedem Bildschirmpunkt eine Zahl, die dessen Farbe bestimmt. Welche der Zahlen oder Bitkombinationen, die für einen Bildschirmpunkt abgespeichert werden, welchen Farben am Bildschirm entsprechen, bleibt weitgehend den Ingenieuren überlassen, die den Computer konstruieren. Vielleicht haben Sie schon einmal die Bilder, mit denen angeblich jeder (vor allem Kinder und Jugendliche) ein großer Maler werden kann, gesehen. Bei diesen Bildern erhält man eine Kiste mit Farben und eine Leinwand oder ein Blatt Papier, auf dem die Konturen des Bildes – ähnlich wie bei einem Bilderbuch – schon vorgegeben sind. Jede Fläche des Bildes trägt eine Zahl und man braucht nur noch den zu dieser Zahl gehörenden Farbtopf aus der Kiste zu nehmen und damit die Fläche auszumalen. »Malen nach Zahlen« heißt das Ganze und obwohl die Bilder, die daraus hervorgehen, meist nicht gerade nach großen Kunstwerken aussehen – das Prinzip funktioniert ganz gut.

So ähnlich verfährt auch der Computer bei mehrfarbiger bitmapped Grafik. Und genau wie beim Malen nach Zahlen muß es nicht unbedingt eine logische Beziehung zwischen bestimmten Zahlen und bestimmten Farben geben. Es reicht, wenn diese Beziehung eindeutig ist und nicht dieselbe Zahl zwei verschiedenen Farben zugeordnet wird. Es gibt eine ganze Reihe von verschiedenen Prinzipien, diese Beziehung logischer aufzubauen.

Bei vielen Computern wird heute ein Prinzip angewendet, bei dem die Farbe jedes Bildpunktes in drei Bits abgelegt wird. Unterschiede gibt es nur dabei, wie die acht Zahlen (bei 3 Bit) als Farben interpretiert werden. Man kann sich natürlich acht beliebige Farben aussuchen und sie fest diesen acht Zahlen zuordnen. Bei einigen Modellen macht man sich auch die Tatsache zunutze, daß Farbfernsehbilder aus den drei Grundfarben Rot, Grün und Blau durch additive Farbmischung zusammengesetzt werden. Jedes der drei Bit, die auf einen Bildschirmpunkt abgebildet werden, wird deshalb einer dieser Grundfarben zugeordnet. Das erste Rot, das zweite Grün und das dritte Blau. Jede Farbe, deren Bit gleich 1 ist, wird dann bei dem jeweiligen Punkt »beigemischt«. Ist das Bit gleich 0, bleibt die Farbe weg. Die acht Farben, die auf diesem Wege gemischt werden können, entsprechen dabei den acht möglichen Kombinationen von drei Bit. Schwarz entspricht zum Beispiel der Bitkombination 000, Weiß der Kombination 111, Rot der Kombination 100, Violett der Kombination 101 und so weiter. Verschiedene Intensitäten der drei Grundfarben sind jedoch nicht möglich. Jede ist entweder an (wird beigemischt) oder aus (wird ganz fortgelassen) und deshalb gibt es nur acht Farben, die sich auf diese Weise mischen lassen.

Ziemlich genau dieses Prinzip wird zum Beispiel bei der Standardfarbgrafik (CGA) der IBM PCs verwendet. Es ist einfach zu realisieren, aber auch recht eingeschränkt. Mehr als acht Farben kann man nämlich so nicht darstellen, und auch diese acht Farben liegen eben »fest« und können nicht geändert werden. Beim Amiga wird ein anderes Prinzip zur Erzeugung von Farben angewendet, die sogenannte »Farbregister-Technik«, über die Sie gleich noch mehr erfahren werden. Mit dieser Technik sind bis zu 4096 verschiedene Farben darstellbar, von denen meist aber nur eine Auswahl verwendet werden kann.

# 11.2 Die Grafikbetriebsarten des Amiga

Nach diesen mehr allgemeinen Ausführungen über die Computergrafik folgen nun aber noch ein paar »harte Fakten« über die Grafikfähigkeiten Ihres Amiga. Die beiden wichtigsten Leistungsmerkmale einer Computergrafik sind die Bildauflösung und die Anzahl der verwendbaren Farben.

## 11.2.1 Bildauflösung

»Bildauflösung« nennt man die Anzahl unterscheidbarer Bildschirmpunkte auf einer bestimmten Strecke. Im übertragenen Sinne wird oft auch die Anzahl der Zeilen und Spalten, aus denen das Raster eines grafikfähigen Computers besteht, »Bildauflösung« genannt. Die Bildauflösung ist ein Maß für die »Feinheit« der Grafik, zu der ein Computer fähig ist und wird getrennt in vertikale Auflösung (=Anzahl unterscheidbarer Zeilen in einem Bild) und horizontale Auflösung (=Anzahl unterscheidbarer Punkte pro Zeile).

Der Amiga bietet die Wahl zwischen zwei vertikalen Auflösungen, 256 oder 512 Zeilen, und zwei horizontalen Auflösungen 320 oder 640 Punkten. Kombiniert ergibt dies vier verschiedene Bildauflösungen mit mindestens 81 920 und höchstens 327 680 Punkten auf dem Bildschirm. (Einige Programme nutzen oft noch größere vertikale und horizontale Auflösungen. Da diese Auflösungen aber nur sehr selten in speziellen Programmen für Videoanwendungen zur Verfügung stehen, werde ich diese nicht im Detail erläutern.)

327 680 Punkte hört sich nach viel an, ist es aber nicht. Selbst das relativ körnige Bild einer Pocketkamera hat mehr Punkte. Die folgende Tabelle gibt einen Überblick über die verschiedenen Auflösungen, und die jeweilige Anzahl von Punkten.

| Horizontale Auflösung | Vertikale Auflösung | Anzahl Punkte |
|---|---|---|
| 320 | 256 | 81 920 |
| 640 | 256 | 163 840 |
| 320 | 512 | 163 840 |
| 640 | 512 | 327 680 |

***Tabelle 11.1:*** *Grafikauflösungen des Amiga 500*

Die in den USA verkauften Modelle des Amiga, die an den PAL-Fernsehstandard angepaßt sind, besitzen nur eine vertikale Auflösung von 200 beziehungsweise 400 Zeilen. Viele der derzeit erhältlichen Programme (die ja fast alle aus den USA kommen) unterstützen die höhere vertikale Auflösung der europäischen Modelle aber nicht. Ich werde im folgenden aber trotzdem von 256 oder 512 Zeilen vertikaler Bildschirmauflösung ausgehen.

Aus dieser Tabelle ergeben sich eine ganze Reihe von Konsequenzen. Die verschiedenen Mengen von Punkten, die der Amiga in den einzelnen Betriebsarten seiner Grafik separat

färben kann, nehmen ja immer denselben Platz auf dem Bildschirm ein. So ist eine Grafik in einem Modus mit 256 Zeilen nicht etwa halb so hoch wie eine Grafik in einem Modus mit 512 Zeilen. Statt dessen ist die zweite Grafik »feiner«, weil auf derselben Strecke, auf der vorher 20 Punkte lagen, im 512er-Modus nun 40 Punkte liegen, mit denen sich feinere Details darstellen lassen.

Daraus ergibt sich aber auch zwingend, daß die Pixel (Bildschirmpunkte) nicht in allen Grafik-Betriebsarten gleich groß sind. Die folgende Gegenüberstellung zeigt Ihnen (in stark vergrößerter Form) die vier verschiedenen Pixelgrößen, mit denen Sie auf dem Amiga arbeiten können.

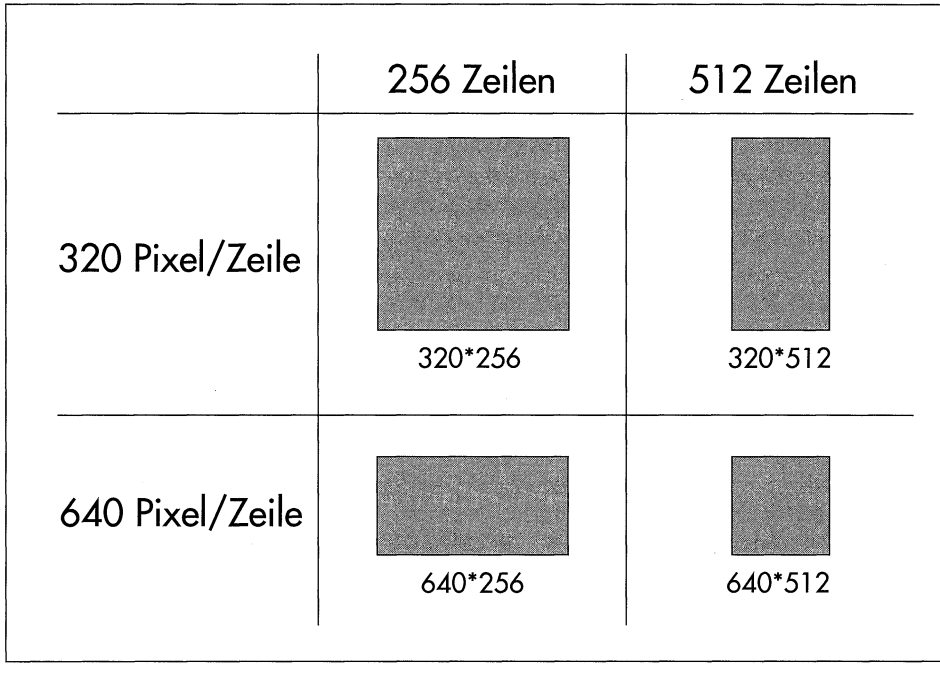



***Bild 11.4:*** *Die vier verschiedenen Pixelgrößen des Amiga 500*

Beachten Sie, daß die Pixel nicht immer quadratisch sind. (Die hier angedeutete Rechteckform der Pixel ist natürlich übertrieben. In Wirklichkeit sind die Ecken der Rechtecke so stark abgerundet, daß es auf manchen billigen Monitoren fast Ovale werden.) Wenn Sie ein Rechteck auf den Bildschirm zeichnen, das 100 Pixel breit und hoch ist, so wird es in zwei Grafik-Betriebsarten annähernd quadratisch (in einer davon jedoch nur halb so groß) sein und in den beiden anderen Betriebsarten rechteckig (einmal breiter als hoch und einmal höher als breit). Das wird spätestens dann wichtig, wenn Sie selbst Grafikprogramme schreiben oder mit einem Malprogramm ein Bild in einer anderen Auflösung betrachten als die, in der es gezeichnet wurde. Ansonsten kompensieren Mal- und Zeichen-Programme diese Unterschiede automatisch.

### 11.2.2 Die Farbpalette

Bei der niedrigsten Bildauflösung kann der Amiga bis zu 32 verschiedene Farben gleichzeitig auf dem Bildschirm darstellen, bei der höchsten Auflösung mit 512 Zeilen zu 640 Punkten sind es immerhin noch 16. Das Programm Deluxe Paint benutzt standardmäßig zum Beispiel eine Bildschirmauflösung von 200 beziehungsweie 256*320 Punkten und 32 Farben. Die Workbench hingegen arbeitet mit einer Bildschirmauflösung von 256*640 Punkten und nur vier Farben.

Wie vereinbart sich die eben genannte Zahl von maximal 32 möglichen Farben nun aber mit der weiter oben genannten Zahl von 4096 Farben, die der Amiga darstellen kann? Eine der beiden Zahlen scheint ja wohl (auf den ersten Blick) falsch zu sein.

Dies ist jedoch nicht der Fall. Die beiden Aussagen beziehen sich auf zwei unterschiedliche Tatbestände, die nun etwas näher beleuchtet werden sollen: Der Amiga ist fähig, 4096 verschiedene Farben darzustellen. Von diesen kann er aber immer nur maximal 32 gleichzeitig am Bildschirm zeigen. Man muß also aus den 4096 Farben eine Palette mit bis zu 32 Farbklecksen zusammenstellen und mit diesen ausgewählten Farben das gezeigte Bild malen. Welche 32 Farben Sie hierzu benutzen wollen steht Ihnen völlig frei – nur mehr als 32 Farben dürfen es nicht sein. (Es gibt allerdings einen besonderen Grafikmodus, der diese Einschränkung trickreich umgeht – dafür aber andere Kompromisse eingehen muß.)

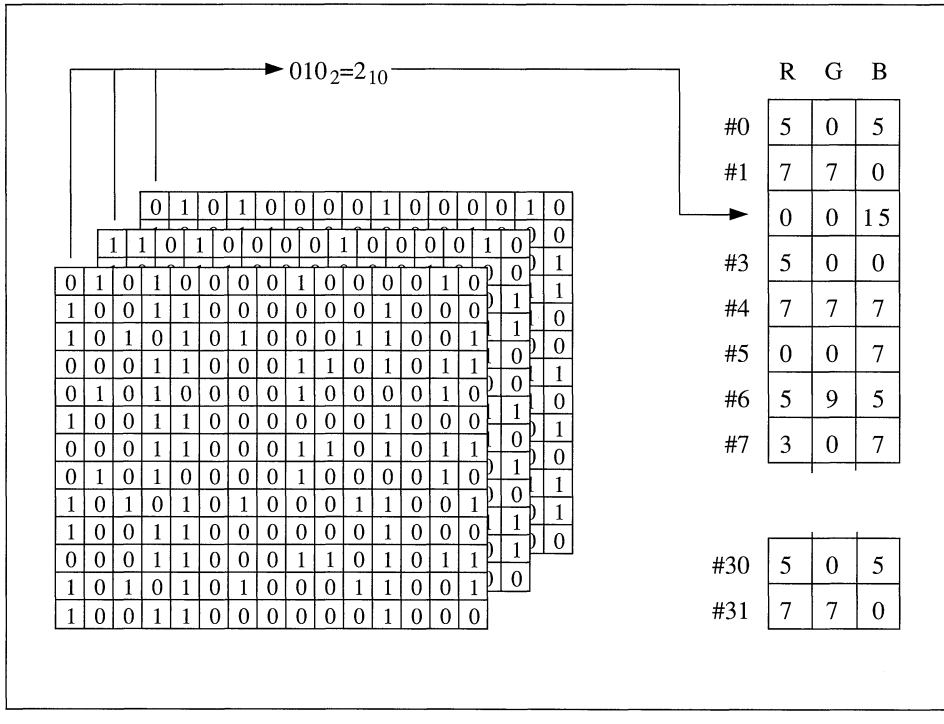



***Bild 11.5:*** *Bestimmung der Farbe eines Bildschirmpunktes*

Der Grund für die Beschränkung auf 32 gleichzeitig verwendbare Farben ist relativ einfach. Wie ja weiter oben bereits erläutert wurde, merkt sich der Amiga für jeden Bildpunkt eine Zahl in seinem Arbeitsspeicher, die bestimmt, welche Farbe dieser Punkt trägt. Die Zahl ist dabei praktisch ein Code für die gewünschte Farbe. Um 4096 verschiedene Farben kodieren zu können, benötigt ein Computer 12 Bit ($2^{12} = 4096$). Der Amiga würde also bereits bei der niedrigsten Bildschirmauflösung 81 920 * 12 = 983 040 Bit benötigen, nur um den Bildschirminhalt abzuspeichern. Dies entspricht über 120 Kbyte und damit fast einem Viertel des Arbeitsspeichers eines Amiga 500 ohne Speichererweiterung. Übrigens sprechen auch andere technische Gründe neben dieser Speicherplatzverschwendung gegen so viele Farbmöglichkeiten.

Um 32 Farben darzustellen, benötigt man hingegen nur 5 Bit; nicht einmal halb so viele wie für die volle Farbenzahl. Um mit nur 5 Bit, also einer Zahl zwischen 0 und 31, trotzdem eine von 4096 Farben bestimmen zu können, verwendet der Amiga eine zusätzliche Stufe der Kodierung. Die Zahl, die er sich zu einem bestimmten Bildschirmpunkt merkt, bestimmt nicht direkt die Farbe, die dieser Punkt hat, sondern ist nur die Nummer eines Farbtopfs in der aktuellen Palette. Die Palette wiederum enthält 32 der 4096 möglichen Farben (diese Farben müssen dazu natürlich in Form von 12 Bit abgespeichert werden). Der Computerfachmann nennt die Farbtöpfe in der Palette übrigens auch »Farbregister«.

Wie an dem obigen Bild zu sehen ist, liegen die Bits, die für einen Punkt »zuständig« sind, nicht direkt hintereinander. Die drei Bit, die hier zur Auswahl einer Farbe in der Farbpalette dienen, liegen in drei verschiedenen Speicherbereichen, die hintereinanderliegend dargestellt wurden. Zu diesen zusammenhängenden Speicherbereichen, in denen jeweils die ersten, zweiten und dritten Bits zu einem Pixel liegen, sagt man auch »Bitebenen« oder »Farbebenen«. Wenn drei solche Ebenen für eine bestimmte Grafik benötigt werden, sagt man oft auch die Grafik oder der Bildschirm ist »drei Bit« oder »drei Bitebenen tief«. Aus der Tiefe einer Grafik und ihrer Auflösung ergibt sich der Speicherbedarf eines Bildes. Dazu müssen Sie die Anzahl der Punkte in einer Grafik mit der Tiefe (der Anzahl Bit pro Punkt) multiplizieren und dann durch acht teilen (acht Bit = ein Byte). Die folgende Tabelle nennt einige Beispiele für den Speicherbedarf (in Bytes) von Bildern bei verschiedenen Auflösungen. (Zu Ihrer Information ist darin auch noch einmal die maximal mögliche Zahl unterschiedlicher Farben bei der jeweiligen Tiefe angegeben.)

| Auflösung | Tiefe | Anzahl Farben | Speicherbedarf |
|---|---|---|---|
| 320 * 256 | 1 | 2 | 10 240 |
| 640 * 256 | 2 | 4 | 40 960 |
| 320 * 256 | 3 | 8 | 30 720 |
| 640 * 512 | 4 | 16 | 163 840 |
| 320 * 512 | 5 | 32 | 102 400 |

***Tabelle 11.2:*** *Grafikauflösungen und ihr Speicherbedarf*

Damit aber genug der Wortklauberei. Wie die Grafik des Amiga genau funktioniert, dürfte wohl im Detail nur für wenige Anwender interessant sein. Die wichtigsten Grundlagen und die sich daraus ergebenden Grenzen der Grafikfähigkeiten zu kennen, ist für fast jeden Amiga-Benutzer wichtig. Der eben beschriebene »Umweg« über die Farbpalette hat zum Beispiel eine interessante Konsequenz, die im folgenden Abschnitt beschrieben wird. Damit sind Effekte ermöglicht, die mit echter Farbe und Papier nicht zu erreichen sind.

### 11.2.3 Nachträgliches Ändern der Palette

Um die Farbe eines Pixels zu ändern, gibt es auf dem Amiga nämlich nicht nur eine Möglichkeit, sondern gleich zwei. Zunächst einmal kann man natürlich die Zahl (Bitkombination) im Speicher ändern, die die Farbe des Pixels bestimmt. Dieses Pixel erscheint dadurch in einer anderen der 32 Paletten-Farben am Bildschirm. Dieser Vorgang entspricht praktisch dem Ändern der Farbe eines Punktes auf dem Papier durch erneutes Überstreichen mit einer anderen Farbe.

Alternativ dazu kann man aber auch die Farbe in der Palette ändern, die durch die Zahl ausgewählt wird, die im Computer für dieses Pixel gespeichert wird (siehe oben). Diese Art der Änderung der Pixelfarbe hat jedoch einen Nachteil (oder Vorteil – wie man es sieht). Im selben Augenblick, in dem an einer Stelle in der Palette eine neue Zahl (= Farbe) eingetragen wird, ändern alle Bildschirmpunkte, die ihre Farbe aus diesem Topf der Palette bezogen haben, ihre Farbe! Wenn Sie also die Farbe Nummer 1 in der Palette von Blau nach Braun ändern, werden alle Punkte des Bildes, die mit dem Blau der Farbe 1 gefärbt waren, schlagartig braun.

Dies liegt daran, daß ein Computer das Bild, das er am Bildschirm zeigt, nicht einmal zeichnet und dann »stehenläßt«. Wie es dem Prinzip des Monitors (und auch Ihres Fernsehers) entspricht, muß das Bild etwa 50mal in der Sekunde neu gezeichnet werden. Ändert man eine Farbe in der Palette und der Amiga sieht beim nächsten Zeichnen des Bildes in der Palette nach, welche Farbe der Zahl entspricht, die er für ein Pixel abgespeichert hat, erhält er die neue Farbe. Folglich kommt es dann am Bildschirm zu einer Farbänderung, obwohl an dem Pixel nichts geändert wurde.

Stellen Sie sich die Möglichkeiten vor, die diese Methode bietet! Sie können eine Grafik zum Beispiel zunächst in nahezu beliebigen Farben zeichnen und sich hinterher noch einmal in Ruhe vor den Bildschirm setzen und nun die Palette bearbeiten. Sie können blitzschnell neue Farbkombinationen austesten, auf Vorschläge in dieser Hinsicht von Freunden oder Kunden eingehen und so weiter.

Eine weitere, sehr interessante Anwendung dieses Konzepts ist die »Farbpalettenanimation« (engl. *color register animation*) oder »Farbenrotation«. Mit diesem Kniff kann man die Illusion von Bewegung erzeugen, indem man den Inhalt der Farbpalette ändert und so recht rasch auch die Färbung großer Flächen auf dem Bildschirm auswechseln kann.

Die Idee der Farbpalettenanimation ist relativ simpel: Ein kleines Programmstück sorgt dafür, daß eine Gruppe von Farben in der Palette automatisch in einem bestimmten Rhythmus geändert wird. Diese Gruppe tauscht die Farben zyklisch aus. Jeweils eine Farbe wird zum nächsten Farbtopf weitergereicht, und dieser gibt vorher seine Farbe an den übernächsten weiter. Der letzte Farbtopf dieser Gruppe schließlich gibt seine Farbe an den Ersten der Gruppe weiter.

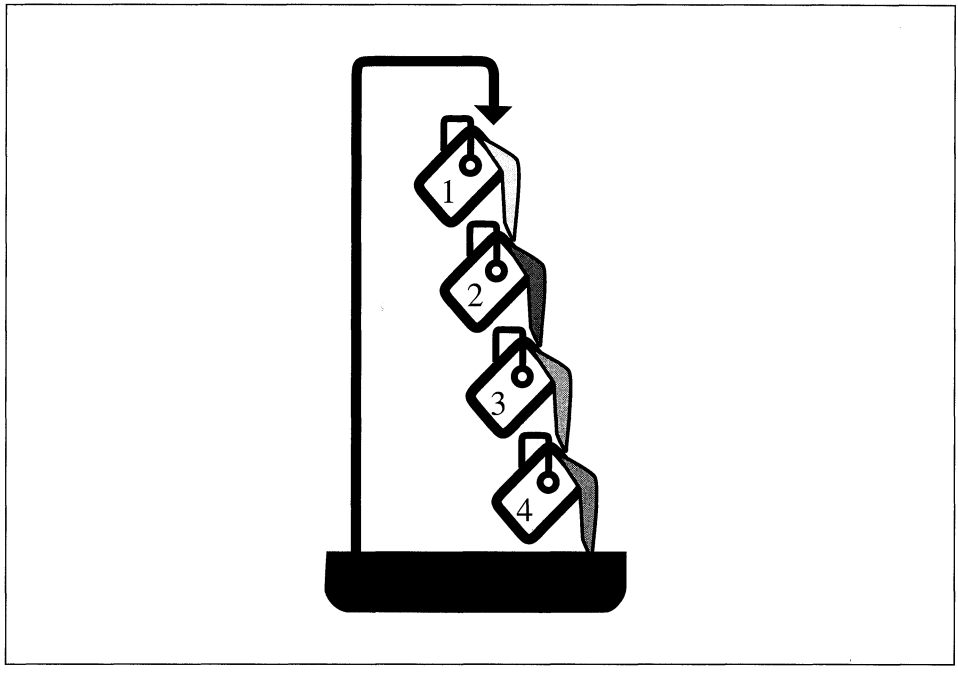



***Bild 11.6:*** *Das Prinzip der Farbpaletten-Animation*

Enthalten die Farbtöpfe, die ihre Farben austauschen, zum Beispiel verschiedene Abstufungen einer Farbe, so wirkt der Farbwechsel unter Umständen wie eine Bewegung der mit den dunklen Farben bemalten Flächen. Mit ein wenig Geschick können Sie auf diese Weise die tollsten Effekte erzielen. Ein berühmtes Beispiel hierfür ist das Bild eines Wasserfalls, bei dem das Wasser in verschiedenen Blautönen gemalt wurde. Läßt man diese Blautöne nun »rotieren«, so scheint der Wasserfall in dem vorher ruhenden Bild auf einmal zu fließen.

Fast alle Malprogramme für den Amiga bieten Möglichkeiten zur Farbpalettenanimation. Im Programm Deluxe Paint (II und III) genügt zum Beispiel ein Druck auf die [Tab]-Taste, um die Bewegung der Farben in der Palette in Gang zu setzen oder zu stoppen. Bei diesem Programm werden auch eine ganze Reihe von Beispielbildern mitgeliefert, anhand deren man ausprobieren kann, welche vielfältigen Möglichkeiten ein so simples Prinzip wie die Farbpalettenanimation schon bietet.

### 11.2.4 Der Interlace-Modus

Computergrafik hat aber nicht nur ihre schönen und interessanten Seiten. Manche Eigenschaften der Amiga-Grafik, die sich aus technischen Zwängen ergeben, die man nicht abstellen kann, sind auch unangenehm. Dazu gehört auch eine Eigenheit der hohen vertikalen Auflösung von 512 Punkten, die Sie bestimmt gleich bemerkt haben, wenn Sie die Workbench einmal in dieser Auflösung betrachtet haben: das Bild flimmert dann fast unerträglich. (Sie können die Workbench auf hohe Auflösung stellen, indem Sie *Preferen-*

*ces* aufrufen und den Schalter *Interlace* aktivieren, dann *Preferences* mit einem Klick auf *Save* verlassen und einen Neustart machen.)

Die vertikale Auflösung beträgt auf dem Bildschirm normalerweise ja 256 Punkte. Dies ist die Grenze, die von einigermaßen preiswerten Monitoren und Farbfernsehern gesetzt wird. Die Bildwiederholfrequenz beträgt bei dieser Auflösung 50 Hz. Das heißt, 50mal pro Sekunde wird das Bild neu gezeichnet. Das ist schneller als das menschliche Auge es bemerken kann, weshalb von diesem andauernden Neuzeichnen normalerweise nichts zu merken ist. Im Interlace-Modus müssen jedoch 512 (oder bei amerikanischer Software 400) Zeilen auf den Bildschirm geschrieben werden. Und das schafft der Amiga beziehungsweise ein Monitor nicht in derselben Zeit wie das Zeichnen von 256 Zeilen.

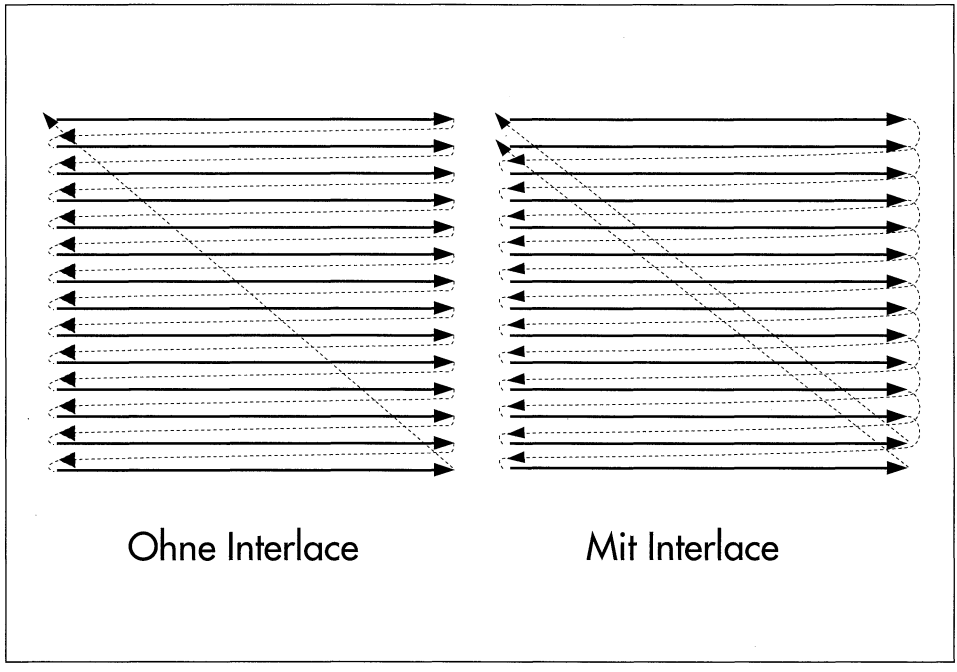



***Bild 11.7:*** *Wie der Elektronenstrahl des Monitors ein Interlace-Bild zeichnet*

Im Interlace-Modus wird das 512 Zeilen hohe Bild deshalb in zwei »Halbbilder« mit jeweils 256 Zeilen aufgeteilt. Diese Halbbilder werden mit einer Bildwiederholfrequenz von 50 Hz abwechselnd auf den Bildschirm geschrieben. Dabei ist jedes Halbbild aber nur jedes zweite Mal »dran«. Für jedes Halbbild gilt also eine Bildwiederholfrequenz von nur 25 Hz. Und das kann das menschliche Auge (leider) noch wahrnehmen – es flimmert. Damit Sie mich aber nicht falsch verstehen: die beiden Halbbilder bestehen natürlich nicht aus der oberen und unteren Bildhälfte. Ein Halbbild enthält statt dessen jede zweite Zeile des Bildes. Das erste Halbbild besteht also aus allen ungeraden, das andere aus allen geraden Zeilen des Vollbilds mit 512 Zeilen. Deutsche Fernsehtechniker nennen dieses Verfahren manchmal auch »Zeilensprung«. Die folgende Skizze zeigt in schematischer Form, wie der Amiga ein Bild

zeichnet, wenn der Interlace-Modus aktiviert ist und er Zeilensprünge ausführt (dabei werden allerdings weit weniger als 512 Zeilen verwendet):

Der Zeilensprung ist übrigens etwas ganz Alltägliches. Auch das normale Fernsehen basiert auf diesem Prinzip. Das Flimmern wirkt beim Fernsehen nur deshalb nicht so schlimm, weil die Kontraste nicht so hart sind wie bei einem computererzeugten Bild, weil das Bild sich sowieso schneller ändert und das Flimmern deshalb nicht so auffällt und schließlich auch deshalb, weil Sie nicht so nahe vor einem Fernseher sitzen wie vor dem Monitor des Amiga.

Der Interlace-Modus hat aber trotz dieser unangenehmen Eigenart durchaus seine Daseinsberechtigung. Wenn Sie nämlich unbedingt möglichst viele Informationen auf dem Bildschirm sehen wollen, bietet er zum Beispiel die doppelte Zeilenzahl. Und wenn Sie Fotos vom Bildschirm machen wollen, spielt das Flimmern sowieso keine Rolle, da das Foto ja hinterher nicht mehr flimmert. Ein anderer Weg, das Flimmern zu umgehen ist, einen Monitor zu verwenden, dessen Phosphorschicht länger nachleuchtet. Monochrome Monitore dieser Art gibt es auch reichlich. Bei Farbmonitoren ist diese Art Phosphor noch selten. Das Flimmern des Interlace wird damit auf ein Minimum reduziert. Leider aber ziehen schnell bewegte Objekte (wie der Mauszeiger) auf solchen Monitoren dann auch deutliche Spuren – wie Kometenschweife – hinter sich her, die auch sehr störend wirken können.

Eine andere Möglichkeit, das Flimmern zu vermeiden sind Zusatzgeräte, manchmal »Flickerfixer« genannt, die dem Amiga beibringen 512 Zeilen ohne Zeilensprung zu erzeugen. Diese Zusatzgeräte verlangen meist eine kleine interne Modifikation der Amiga-500-Hardware, die man am besten von einem Fachmann ausführen läßt. (Beim Amiga 2000 genügt es, eine Zusatzkarte in einen freien Steckplatz zu setzen; siehe Kapitel 9.)

Die Anschaffung eines Flickerfixers ist allerdings alles in allem ein recht teures Vergnügen, obwohl das kleine Zusatzgerät selbst meist nicht viel kostet. Die hohen Mehrkosten ergeben sich daraus, daß 512 Zeilen ohne Zeilensprung nicht auf den preiswerten Amiga-Monitoren dargestellt werden können (wie ihn Commodore beispielweise unter der Bezeichnung A 1084 S anbietet). Sie benötigen statt dessen einen sogenannten MultiSync-Monitor (siehe auch Kapitel 9). Und der kostet meist deutlich mehr als der Amiga 500 selbst.

## 11.3 Ungewöhnliche grafische Fähigkeiten des Amiga

Die bislang beschriebenen Aspekte der Amiga-Grafik waren mehr oder weniger »normal« oder »üblich«. Viele andere Computer haben in grafischer Hinsicht solche oder ähnliche Eigenschaften. Der Amiga 500 hat allerdings in mancher Hinsicht bessere »Eckdaten«. So ist es zum Beispiel bei anderen preiswerten Computern recht unüblich, 32 Farben aus einer Palette von 4096 gleichzeitig am Bildschirm verwenden zu können. In den folgenden Abschnitten werden noch einige weitere Aspekte der Amiga-Grafik behandelt, die recht »ungewöhnlich« sind. Einige der beschriebenen Fähigkeiten bietet überhaupt kein anderer Computer.

### 11.3.1 HAM: Mehr Farben aus dem Computer

Die Grafik des Amiga benötigt – wie oben gesehen – maximal 5 Bit pro Bildschirmpunkt (Pixel), um 32 verschiedene Farben gleichzeitig am Bildschirm darzustellen. Es gibt aber

auch einen Modus, der 6 Bitmaps benötigt und theoretisch alle 4096 möglichen Farben gleichzeitig am Bildschirm zeigt. Dieser Modus heißt »Hold-and-Modify-Modus«, oder kurz »HAM-Modus«. Der Name bedeutet soviel wie »festhalten und modifizieren«, und genau das macht die Video-Hardware in diesem Modus.

Die sechs Bit, die für jeden Bildschirmpunkt in diesem Modus zur Verfügung stehen, werden dabei in einer speziellen Weise genutzt, die vom üblichen Bitmap-Prinzip, das weiter oben beschrieben wurde, etwas abweicht. Die Bits Nummer 5 und 6 bestimmen in diesem Modus bei jedem einzelnen Pixel, wie die restlichen Bits (Nummer 1 bis 4) interpretiert werden.

Ist die Bitkombination der Bits 5 und 6 gleich 00, so werden die restlichen 4 Bit als die Nummer eines Farbregisters interpretiert, das die Farbe dieses Punktes enthält. In diesem Fall wirkt der HAM-Modus wie ein etwas speicherverschwendender Grafikmodus mit 4 Bitebenen (= 16 Farben).

Ist die Bitkombination 6–5 aber gleich 01, 10 oder 11, tritt der eigentliche HAM in Aktion. Im Fall 01 wird erst einmal die Farbe festgehalten (engl. *to hold* = halten), die der unmittelbar vor (links neben) dem aktuellen Punkt liegende Bildpunkt hatte. Die Blau-Intensität dieser Farbe wird jedoch verworfen und durch den Wert ersetzt, der in den Bitmaps 1 bis 4 für den neuen Punkt eingetragen ist. Ist diese Bitkombination zum Beispiel 0000, dann wird dem neuen Punkt überhaupt kein Blau mehr beigemischt. Die Bits Nummer 1 bis 4 modifizieren (engl. *to modify* = modifizieren) also den Blauwert des vorangegangenen Punktes und ergeben damit die neue Farbe für den aktuellen Punkt.

Ist die 6–5-Bitkombination gleich 10, wird auch die Farbe des »Vorgängerpunktes« festgehalten. Dann aber ersetzen die Bits 1 bis 4 des neuen Punktes die Rot-Intensität der Farbe des Vorgängerpunktes.

Und wenn die Kombination gleich 11 ist, werden die Bits 1 bis 4 des neuen Punktes dazu benutzt, die Grün-Intensität des Vorgängerpunktes zu modifizieren.

Auf diese Weise können alle 4096 Farben erzeugt werden, zu denen der Amiga 500 überhaupt fähig ist. Es ist allerdings nicht möglich, jeden beliebigen Punkt auf jede beliebige Farbe zu setzen. Spätestens nach drei Punkten kann man aber jede beliebige Farbe erhalten, indem man nacheinander alle drei Komponenten der Pixelfarbe, Rot, Grün und Blau austauscht. Die 16 Farben, die man über ein Farbregister direkt wählen kann (mit der Bit-Kombination 00 der Bits 6 und 5), können natürlich auch noch (möglichst geschickt) eingesetzt werden.

Es gibt inzwischen sogar einige Malprogramme auf dem Markt (zum Beispiel »DigiPaint« und »Prism«), mit denen Sie Bilder in diesem Modus erstellen können. Weiterhin gibt es eine Reihe zusätzlicher Peripheriegeräte (sogenannte »Digitizer«), mit denen Sie Bilder mit einer Videokamera aufnehmen und mit HAM-Modus auf dem Amiga 500 darstellen können. Die so zustande kommenden Bilder sehen oft erstaunlich gut aus – echte Fotoqualität ist damit aber nicht zu erzielen.

### 11.3.2 Grafiken, die größer als der Bildschirm sind

Der Amiga bietet aber nicht nur die Möglichkeit, unglaublich viele Farben gleichzeitig in einer Grafik zu verwenden, sondern erlaubt auch die mühelose Bearbeitung von Grafiken,

die größer sind als der gesamte Bildschirm. Solche Grafiken werden in riesigen Bitmaps gespeichert, von denen nur ein rechteckiger Ausschnitt (eine Art Fenster) am Bildschirm zu sehen ist.

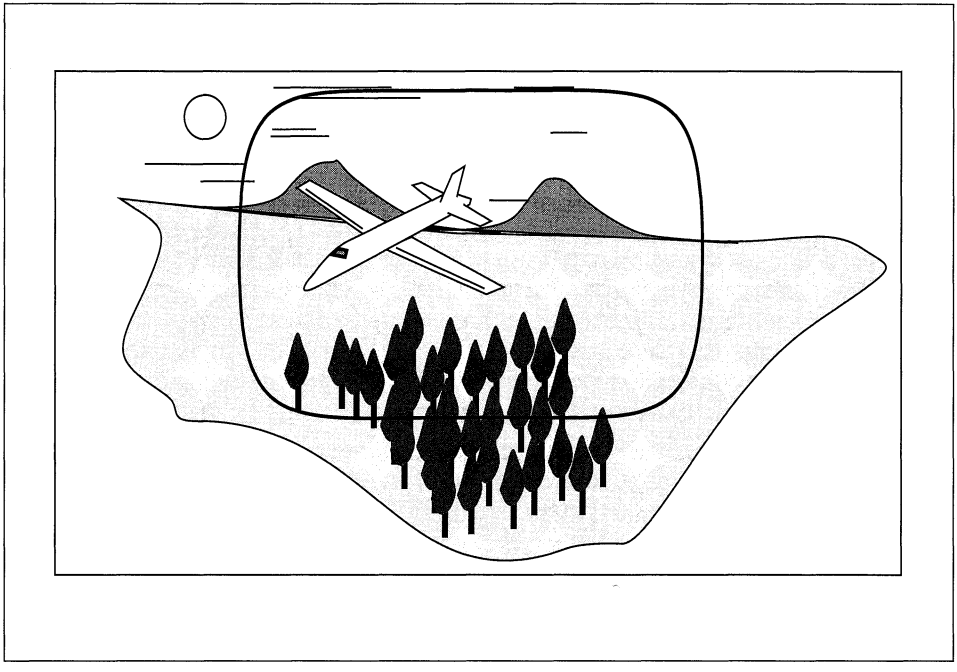


***Bild 11.8:*** *Der Bildschirm als Fenster in eine übergroße Grafik*

Das »Verschieben« des Bildschirms beziehungsweise des sichtbaren Ausschnitts über diese Riesengrafik ist für Programme sehr einfach. Sie brauchen dazu nur den Inhalt einiger weniger Speicherstellen zu ändern (für Fachleute: bei diesen Speicherstellen handelt es sich um Register des Grafikchips *Denise*). Die Bilddaten selbst brauchen im Speicher nicht bewegt werden. Die Verschiebung geht deshalb sehr schnell und ohne das – auf anderen Computern in solchen Fällen übliche – Flimmern und Rucken vonstatten.

### 11.3.3 Zwei Bilder übereinander: Der Dual-Playfield-Modus

Und schließlich verdient (mindestens) noch ein weiterer Aspekt der Amiga-Grafik Erwähnung. Erstens weil er ein ungewöhnliches Konzept darstellt, das bei vergleichbaren Geräten kaum zu finden ist. Und zweitens, weil ihn zumindest ein faszinierendes Programm, Deluxe Video, intensiv nutzt. (Deluxe Video ist ein Programm, mit dem Sie selbst kleine Zeichentrickfilme erstellen können.)

Der Amiga besitzt nämlich einen Betriebsmodus, den sogenannten »Dual-Playfield-Modus«, in dem zwei sich überlagernde Bilder gleichzeitig am Bildschirm dargestellt werden. Der Eindruck, der dabei entsteht, ist ähnlich dem einer transparenten, teilweise bemalten

Folie, die über eine andere Zeichnung gelegt wird. An den Stellen, an denen die Folie bemalt ist, wird das dahinterliegende Bild verdeckt. An den Stellen, an denen sie frei (transparent) ist, sieht man das darunterliegende Bild.

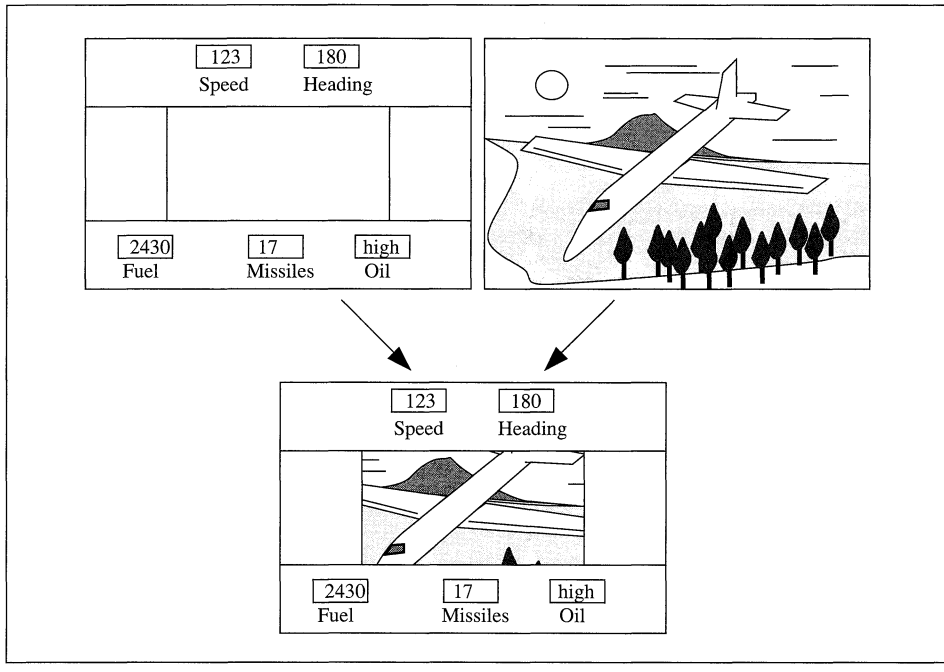



***Bild 11.9:*** *Zwei übereinanderliegende Bilder*

Der Vorteil dieser Fähigkeit des Amiga liegt darin, daß beide Bilder völlig unabhängig voneinander sind. Jedes Bild kann zum Beispiel getrennt verändert und verschoben werden, ohne daß das andere davon beeinträchtigt wird. Beide Bilder haben eine eigene Farbpalette mit bis zu acht verschiedenen Farben und können auch getrennt geändert werden.

Man kann sich leicht vorstellen, welche Möglichkeiten sich dadurch gerade für Trickfilme und Spiele ergeben. Das hintere Bild enthält den relativ statischen Hintergrund eines Bildes (der vielleicht langsam von links nach rechts wandert), das andere die beweglichen Elemente des Vordergrunds, Personen, Tiere, Fahrzeuge, Wolken und so weiter.

### 11.3.4 Sprites & BOBs

Der Dual-Playfield-Modus bietet schon eine Ahnung von den Eigenschaften, die der Amiga speziell für die bewegte Computergrafik mit sich bringt. Noch »beweglicher« sind aber »Sprites« und »BOBs«. Sprites sind selbständige kleine Bilder, die vom großen Hintergrundbild unabhängig sind. Ähnlich wie das obere Bild im Dual-Playfield-Modus sind sie völlig getrennt vom darunterliegenden Bild und werden auch an anderer Stelle im Speicher abgelegt.

Ein solches Sprite ist also ein kleines Bild, das maximal 16 Pixel breit und beliebig hoch sein kann (maximal natürlich höchstens nur so hoch wie das Hintergrundbild). Der Mauszeiger ist zum Beispiel ein kleines Sprite.

Um ein Sprite an einer bestimmten Stelle des Bildschirms erscheinen zu lassen, braucht man nur die X- und Y-Position der Stelle, an der die obere linke Ecke des Sprites sitzen soll, in eine bestimmte Speicherstelle (für Fachleute: in ein Register von Denise) schreiben. Ändert man die Werte in diesen Speicherstellen, taucht das Sprite sofort an der neuen Stelle auf. Eine »Bewegung« dieses kleinen Bildes ist so sehr leicht und auch sehr schnell möglich.

Eine gravierende Einschränkung der Sprites ist allerdings ihre Zahl: es gibt nur acht Stück. Diese acht Sprites können jeweils maximal vier verschiedene Farben verwenden. Es gibt auch die Möglichkeit, 16 Farben bei jedem Sprite zu verwenden – dabei halbiert sich die Anzahl möglicher Sprites aber auf vier. Acht oder gar nur vier bewegliche Objekte sind für viele Zwecke zu wenig. Deshalb gibt es zwar nur acht Hardware-Sprites, man kann jedes aber mehrmals benutzen. Ein und dasselbe Sprite kann mehrmals in verschiedenen Formen untereinander, aber nicht nebeneinander auf dem Bildschirm erscheinen. (Wer wissen möchte, woher die Bedingung »untereinander« kommt, muß sich sehr intensiv mit der Amiga-Hardware beschäftigen. Für die meisten Anwendungsfälle genügt es, zu wissen, daß es diese Einschränkung gibt und daß ohne Einschränkungen nur acht Sprites verwendet werden können.)

Trotz vieler Tricks und Kniffe bleiben Sprites in ihren Möglichkeiten doch recht »beschränkt«. Es gibt nur 8 Stück davon, sie können nur 16 Punkte breit (schmal) werden und verfügen nur über wenige Farben. Alle diese Einschränkungen kennen BOBs nicht. Im Gegensatz zu Sprites sind die BOBs aber kein Konzept der Amiga-Hardware, sondern der Amiga-Software. Diese Software nutzt allerdings intensiv ein besonderes Stück Hardware, den »Blitter«, von dem die BOBs (***Blitter Objects***) auch ihren Namen haben. Über diesen Blitter werden Sie im Abschnitt 11.5 noch mehr erfahren.

BOBs verhalten sich nahezu wie Sprites. Es sind kleine Grafiken, die man schnell und einfach verschieben kann, ohne die »darunterliegende« Grafik zu beeinträchtigen. BOBs können im Gegensatz zu Sprites aber genausogroß werden und genausoviele Farben verwenden wie die dahinterliegende Grafik auch. Sie sind im allgemeinen (jedoch nicht zwangsläufig) aber auch recht klein.

Bei BOBs wird eine kleine Bitmap, die das entsprechende Bild enthält, wirklich in die Bitmap des Hintergrundbildes hineinkopiert. Vorher wird aber der alte Inhalt dieser Bitmap in einem Puffer gesichert. Wird das BOB bewegt, so wird diese gesicherte Kopie des alten Inhalts wieder in die Bitmap des Hintergrundbildes zurückgeschrieben und das BOB an der neuen Stelle gezeichnet – und davor natürlich wieder der Inhalt des Hintergrundbildes an der neuen Stelle gesichert. Sie werden sich nun vielleicht fragen, wieso man sich überhaupt mit Sprites abgibt, wo es doch die viel flexibleren BOBs gibt? Die Antwort darauf ist recht einfach: BOBs sind viel langsamer und kosten viel mehr Speicherplatz als Sprites!

Das Bewegen eines BOBs geschieht nämlich durch Software (Programme). Und eine Computer-Grundregel lautet: »Hardware ist schneller (als Software)!« Je größer ein BOB wird, desto größer werden die Unterschiede zu einem Sprite. Wenn sich mehrere große BOBs über den Bildschirm bewegen, so werden diese Bewegungen schon recht langsam und die BOBs fangen an zu flimmern. BOBs sind also einerseits wesentlich flexibler und vielseitiger als Sprites, andererseits aber bei weitem nicht so »beweglich« wie diese.

### 11.3.5 AnimObjects

BOBs und Sprites eignen sich ideal für die Darstellung bewegter Dinge in einem Programm. Mit beiden Arten von beweglichen Objekten kann man Bilder (mit verschieden großen Einschränkungen) recht mühelos über den Bildschirm schieben. Es reicht jedoch für die Simulation echter Bewegung meist nicht aus, einfach ein »starres Bild« zu verschieben. Wenn in einem Zeichentrickfilm zum Beispiel ein Männlein durch das Bild geht, wird auch nicht ein Bild dieses Männleins unverändert verschoben. An jeder Stelle taucht statt dessen ein anderes Bild des Männleins auf, das verschiedene Phasen der Gehbewegung zeigt. Bein- und Armhaltung ändern sich ständig – wiederholen sich aber natürlich nach einer Anzahl von Bewegungsschritten.

Exakt solche Effekte sind mit AnimObjects möglich. AnimObjects sind eine Reihe von BOBs oder Sprites, die mit einigen zusätzlichen Daten zu einem Objekt zusammengefaßt werden. Ein AnimObject kann dann wie ein Sprite über den Bildschirm bewegt werden, wobei parallel zur Bewegung automatisch ein BOB oder Sprite gegen das nächste ausgetauscht wird.

Die Reihenfolge und Geschwindigkeit, in der die BOBs oder Sprites gegeneinander ausgewechselt werden, ist vom Programmierer frei wählbar. Gleichzeitig kann er dem AnimObject noch eine Geschwindigkeit mitteilen, mit der die BOBs sich über den Bildschirm bewegen sollen und eine Beschleunigung, die angibt, mit welcher Rate sich diese Geschwindigkeit ändern soll.

Ein AnimObject könnte also die verschiedenen Bewegungsphasen einer gehenden Person enthalten. Je mehr Phasen man verwenden würde (und je mehr Speicher man dafür verwenden würde), desto flüssiger würde diese Bewegung wirken. Gibt man dem AnimObject dann eine Geschwindigkeit, so könnte diese Person langsam oder schnell über den Bildschirm schreiten.

### 11.3.6 Kollisionserkennung

Wenn sich Objekte über den Bildschirm bewegen, ist es oft wichtig, rechtzeitig zu erkennen, wann eines mit dem anderen »zusammenstößt« oder an gewisse Grenzen (zum Beispiel den Rand des Bildschirms oder eines Fensters) stößt. Die Software und die Hardware des Amiga arbeiten deshalb zusammen und überprüfen nach jeder Bewegung eines Objektes (ob BOB, Sprite oder AnimObject spielt dabei keine Rolle), ob eine solche Kollision aufgetreten ist. Dabei kann der Programmierer bestimmen, welche Kollisionen für ihn von Interesse sind und welche nicht. Nur die interessanten Kollisionen werden dann von der Systemsoftware des Amiga an die Programme gemeldet, die sich dafür interessieren. Wie die Programme darauf reagieren, ist deren Sache – in dieser Hinsicht bekommen Sie keine Unterstützung.

Trotzdem ist dieser »Service« der Kollisionserkennung (die man natürlich auch »abschalten« kann) eine sehr praktische Angelegenheit. Das wird jeder einsehen, der nur ein wenig Ahnung von Programmierung hat. Man muß sich dazu einmal klar machen, wie schwierig es ist, mit einem Programm festzustellen, wann ein beliebig geformtes (!) Objekt mit einem anderen, das natürlich ebenfalls eine beliebige Form haben kann, zusammenstößt.

# 11.4 Klangerzeugung

Nicht ganz so große Publizität wie seine Grafikfähigkeiten haben die Möglichkeiten gefunden, die der Amiga zur Klangerzeugung bietet. Sie sind aber trotzdem auch recht beeindruckend. Der Amiga besitzt insgesamt 4 voneinander unabhängige Tonkanäle, von denen jeweils zwei zu einem Stereokanal zusammengefaßt werden können. Jeder der 4 Kanäle kann eine völlig beliebige Wellenform mit beliebiger Hüllkurve ausgeben. Dazu wird nur in sehr geringem Maß ein Eingreifen des »zuständigen« Programms benötigt – der Amiga macht fast alles allein.

Um zu verstehen, wie ein Computer aber überhaupt dazu eingesetzt werden kann, Musik (und natürlich auch andere Geräusche) zu machen, müssen Sie zunächst einen kleinen Ausflug in die Gefilde der »digitalen Musik« machen.

## 11.4.1 Töne, wie sie der Amiga sieht

Ein Ton oder Klang ist nichts anderes als eine Folge ständiger Luftdruckänderungen. Diese Luftdruckänderungen bewegen die menschlichen Trommelfelle. Die Hörnerven und das restliche Nervensystem des Menschen (vor allem auch das Gehirn) machen aus diesen Bewegungen des Trommelfells dann den Klang, den wir wahrnehmen. Ähnliche Luftdruckschwankungen muß der Amiga also zwangsläufig auch erzeugen, um einen (einigermaßen) natürlichen Klang zu simulieren. Da es aber Verstärker und Lautsprecher gibt, die aus Spannungsschwankungen Luftdruckschwankungen machen, reduziert sich das Problem für den Amiga auf die Erzeugung einer im »richtigen Rhythmus« schwankenden Spannung an den beiden Lautsprecherausgängen.

Zur Erzeugung einer Klangform haben Sie zwei Alternativen. Sie können einen Klang entweder völlig »frei erfinden« oder einen natürlichen Klang digitalisieren. Zum Digitalisieren werden die Luftdruckschwankungen, die bei einem natürlichen Klang entstehen, in sehr kurzen Zeitabständen gemessen. Diesen Meßvorgang nennt man in der Musikbranche üblicherweise »Sampling« oder »Sound-Sampling«. Entscheidend für die Qualität – also die »Natürlichkeit« – eines so digitalisierten Klangs ist die Anzahl der Meßwerte, die pro Sekunde aufgenommen werden. Dies nennt man die »Sampling-Frequenz« oder »Sampling-Rate«. Es läßt sich mathematisch nachweisen, daß für eine einigermaßen naturgetreue Digitalisierung eines Klangs eine Sampling-Frequenz nötig ist, die mindestens doppelt so hoch ist wie die höchsten Töne, die in dem Klang vorkommen.

Leider kann ein Computer aber Werte nicht beliebig genau messen und abspeichern wie zum Beispiel ein Kassettenrecorder. (Sie erinnern sich vielleicht an das »digitale Prinzip« aus dem Abschnitt 11.1?) Statt dessen ordnet er jedem gemessenen Wert eine Zahl zwischen einem ganz bestimmten Minimum und Maximum zu. Deswegen heißt dieser Vorgang auch »digitalisieren«. Aus einer kontinuierlichen (analogen) Schwingung wird eine (digitale) Folge von diskreten Zahlenwerten. Die folgende Abbildung zeigt in vereinfachter Form ein solches Sampling, bei dem jedem der gemessenen Werte eine Zahl zwischen 128 und +127 zugeordnet wird. Die dicke schwarze Linie stellt darin die Luftdruckschwankungen eines Tons dar, wie sie sich über die Zeit hinweg (die waagerechte Achse) ändern. Die gestrichelten Linien markieren die Zeitpunkte, an denen der Luftdruck jeweils gemessen wird.

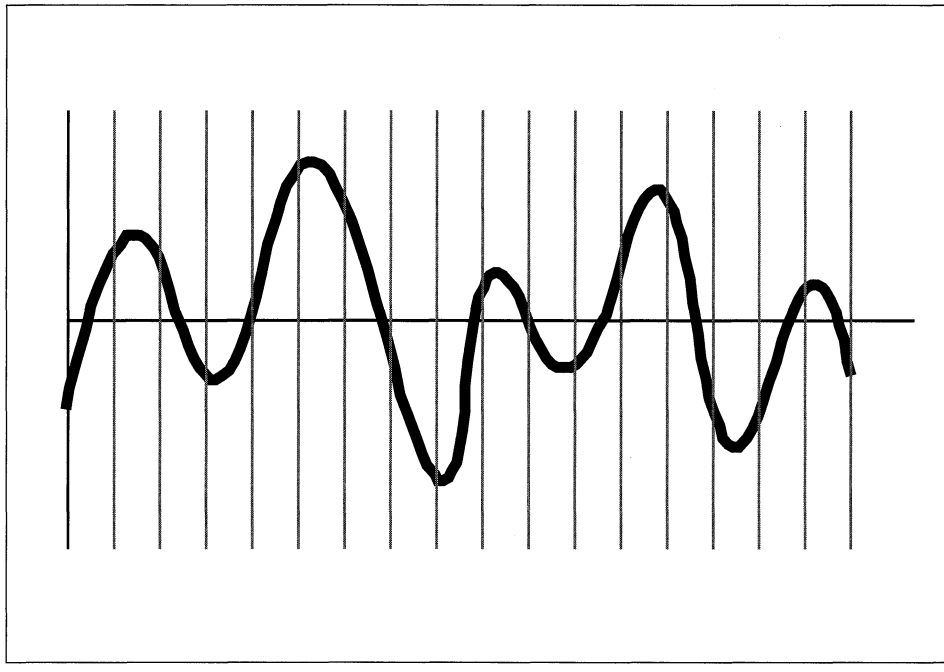


***Bild 11.10:*** *Ein Klang wird digitalisiert*

Ein solcher Meßvorgang (Sampling) ergibt eine Folge von Zahlenwerten, die dem Schwanken des Luftdrucks um die Nullachse entsprechen. Im Fall des Amiga liegen diese Werte zwischen –128 und +127, was dem Fachmann natürlich sofort sagt, daß es sich dabei um 8-Bit-Zahlen (Bytes) handelt. Man spricht dann manchmal auch von Zahlen mit 8 Bit »Breite« oder von »8 Bit breitem Sampling«.

Obwohl ein solcher Zahlenwert nur sehr kurze Zeit gültig ist, »nimmt der Computer an«, daß er bis zum nächsten Meßpunkt unverändert bleibt. Etwas anderes kann er nach dem digitalen Prinzip auch gar nicht »annehmen«, da es für ihn keine feinere Aufteilung der Zeit zwischen zwei Meßpunkten gibt. Wenn ein digitalisierter Klang wieder ausgegeben wird, bleibt die Spannung (der Luftdruck) deshalb zwischen zwei Meßpunkten unverändert, woraus sich die typische Treppchenform eines digitalisierten Klangs ergibt. Die dicken schwarzen Linien bezeichnen dabei den Verlauf der vom Computer erzeugten Spannungsschwankungen, während die dünne Linie den Verlauf der Luftdruckschwankungen des »Originalklangs« vor der Digitalisierung beschreibt. Die gestrichelten Linien markieren die Zeitpunkte, zu denen die Spannung wechseln kann.

Diese Abbildung ist allerdings nur eine recht grobe Annäherung an die tatsächlichen Gegebenheiten. Die meisten klangerzeugenden Computer – so auch der Amiga – enthalten elektronische Schaltkreise, die die plötzlichen Sprünge etwas glätten. Der digitalisierte kommt damit dem »echten« Klang wieder näher. Trotzdem sieht man hoffentlich, daß digitalisierte Klänge immer nur eine Annäherung an einen wirklichen Klang darstellen, die »der Natur« um so näher kommt, je kleiner die Abstände zwischen den einzelnen Messungen sind und je genauer der Wert gemessen wird.

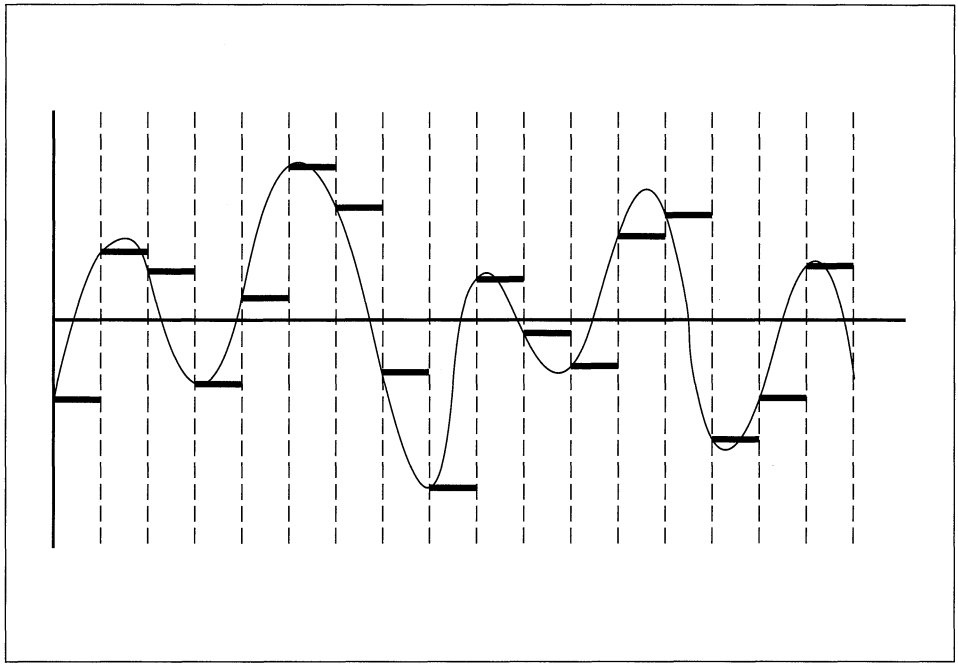


***Bild 11.11:*** *Wie der Amiga einen digitalisierten Klang wieder ausgibt*

Man darf von der Sound-Hardware des Amiga deshalb nicht zuviel verlangen. Es handelt sich beim Amiga nicht um einen Synthesizer, und seine Klangqualität kann auch zusammen mit einem guten Verstärker und guten Boxen HiFi-Tests gewiß nicht standhalten. Für die Darstellung eines Klangs werden im Amiga zum Beispiel nur 8 Bit breite Werte verwendet. HiFi-Geräte, die die Digitaltechnik nutzen (zum Beispiel CD-Plattenspieler) verwenden üblicherweise 16 Bit, was zu einer rund 250mal besseren Klangqualität führt – von anderen klangverbessernden Konstruktionen in diesen Geräten ganz abgesehen.

### 11.4.2 »Vollsynthetische« Klänge

Das eben beschriebene Digitalisierungsverfahren ist natürlich recht aufwendig. Man muß zum Beispiel einen natürlichen Klang mit einem Mikrofon aufnehmen – wobei eine gute Akustik des Aufnahmeraums sehr wichtig ist. Dies geht nur mit zusätzlicher Hardware, denn der Amiga besitzt keinen Mikrofonanschluß. Aber auch mit dieser Zusatz-Hardware wird eine Digitalisierung selten auf Anhieb gelingen. Man muß statt dessen viel herumprobieren und mit der zugehörigen Digitalisierungs-Software sehr gut vertraut sein, bis man akzeptable Ergebnisse erzielen wird.

Einfacher hat man es da mit Klangformen, die man mathematisch beschreiben kann. Bei diesen »vollsynthetischen« Klängen kann man die Zahlenfolge, die den Luftdruckverlauf des Klangs beschreibt, nach einer – meist sehr einfachen – Formel berechnen. Die meistverwendeten dieser vollsynthetischen Klangformen sind die Sinus-, die Rechteck-, die Dreieck- und die Sägezahn-Welle.

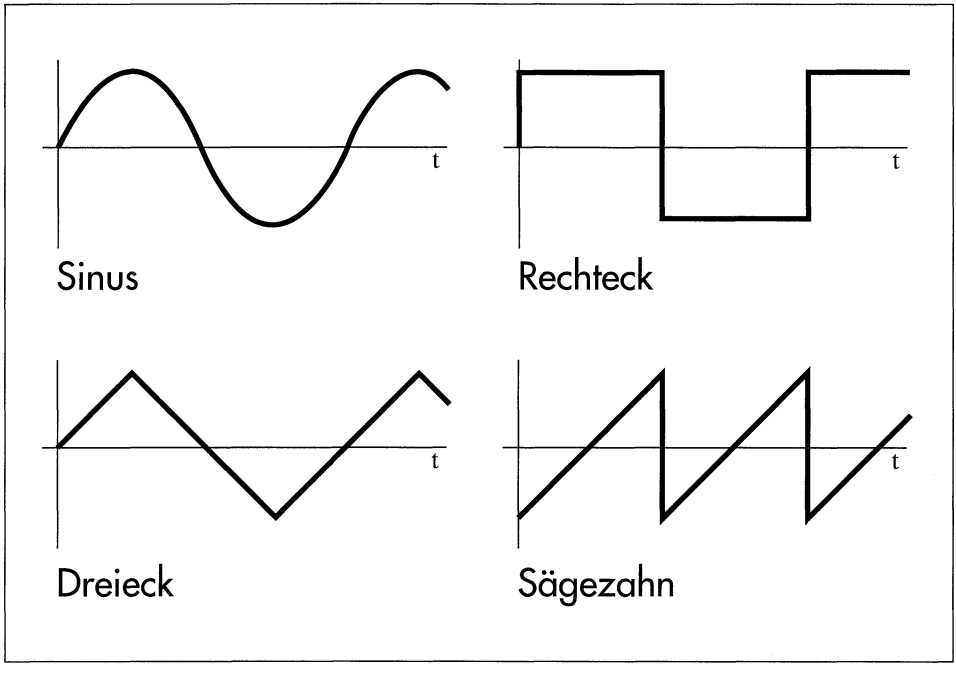



***Bild 11.12:*** *Die vier wichtigsten »vollsynthetischen« Klangformen*

### 11.4.3 Die Hüllkurve eines Tons

Selbst bei der (verhältnismäßig) niedrigen Qualität der 8-Bit-Musik des Amiga wäre es sehr aufwendig, mehrere Minuten Musik in der oben beschriebenen Form zu digitalisieren. Dabei spielt es keine Rolle, ob ein natürlicher Klang digitalisiert wird oder ein vollsynthetischer Klang berechnet wird. Jeder Wert der den Klang beschreibenden Zahlenfolge ist ja ein Byte (8 Bit). Bei einer Sampling-Frequenz von nur 10 kHz (10 000 Messungen pro Sekunde) fallen jede Sekunde 10 Kbyte Daten an. In einer Minute sind das schon 600 Kbyte oder fast eine volle Diskette. Wenn Sie also gerne Ihre Lieblings-LP aus dem Computer hören würden, tun Sie gut daran, sich eine Festplatte anzuschaffen. Eine Festplatte mit 20 Mbyte Kapazität könnte gerade eben eine LP in digitalisierter Form aufnehmen.

Glücklicherweise ist eine vollständige Digitalisierung aber für viele Anwendungen gar nicht notwendig. Die meisten – zum Beispiel die von musikalischen Instrumenten erzeugten – Töne wiederholen sich nach kurzer Zeit, der sogenannten Periode. Die Länge dieser Periode liegt weit unterhalb einer Sekunde. Man braucht deshalb nur eine Periode zu digitalisieren und kann schon stundenlang einen Ton »halten«. Die Zahlenfolge mit den Luftdruckwerten, die sich aus der Digitalisierung ergibt, wird dazu einfach ständig wiederholt.

Hat man eine Periode eines Klangs erst einmal digitalisiert, kann man zudem auch die Tonhöhe (Frequenz) variieren. Dazu läßt man die Zahlenfolge der Luftdruckwerte einfach schneller oder langsamer ablaufen. Ohne für jede Tonhöhe zu einem bestimmten Instrument

eine Sampling-Folge zu haben, kann man dieses Instrument trotzdem in ganz verschiedenen Tonhöhen simulieren.

Um ein natürliches Instrument ausreichend zu charakterisieren, reicht aber die Klangform – also die Abfolge der Luftdruckschwankungen in einer Periode – nicht aus. Eine einzelne Note hat bei jedem Instrument typischerweise auch einen sehr charakteristischen Lautstärkenverlauf. Zu Beginn steigt sie sehr schnell zu einer maximalen Lautstärke an, fällt dann etwas ab, bleibt einen Moment auf einem Plateau nahezu gleicher Lautstärke stehen, um dann langsam auszuklingen. Bei manchen Instrumenten hört man das recht deutlich. Kirchenglocken zum Beispiel können noch Minuten nachklingen, wenn man sie einmal angestoßen hat.

Den typischen Verlauf der Lautstärke eines einmal angestoßenen Tons nennt man »Hüllkurve«, weil er die eigentlichen Schwingungen des Klangs einhüllt. Die Länge und Steilheit der verschiedenen Plateaus, Steigungen und Gefälle in der Hüllkurve ist von Instrument zu Instrument sehr verschieden und trägt entscheidend zu dem »typischen Klang« bei, der ein Instrument vom anderen unterscheidet.

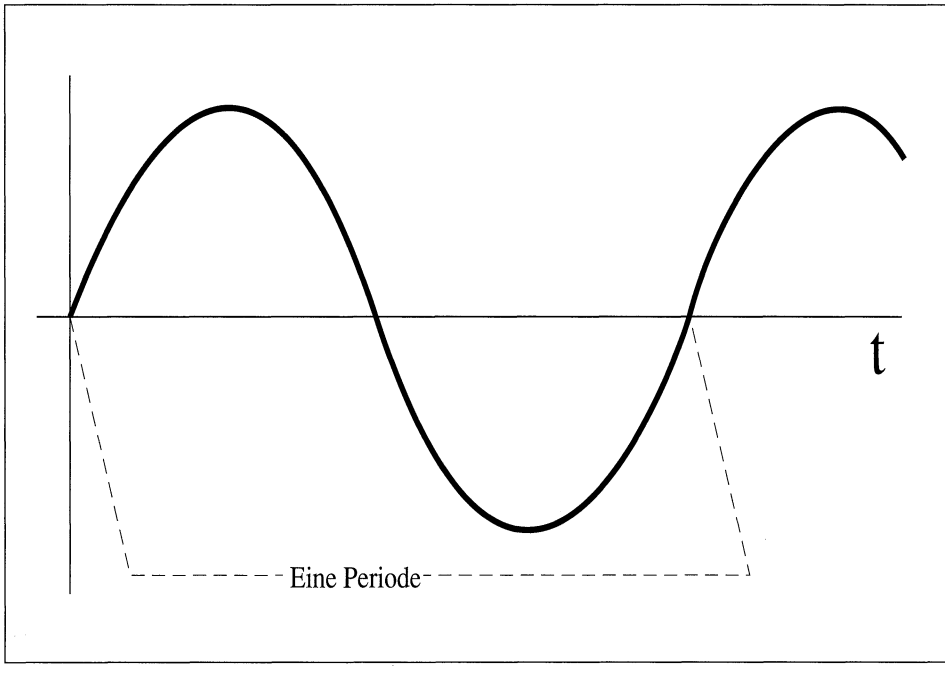



***Bild 11.13:*** *Die Periode eines Klangs*

Die Hüllkurve kann normalerweise schon durch sehr wenige Zahlenwerte genau beschrieben werden, und zwar durch drei Zeiten und drei Lautstärken. Dies sind die Zeiträume für das Erreichen der Maximal-Lautstärke, das Abfallen zum ersten »Plateau«, die Dauer des Plateaus, die Zeit, bis der Ton völlig verklungen ist und die Lautstärke, die der Ton jeweils am Ende jeder dieser Phasen hat. Der Fachmann nennt diese Zeiten *Attack*-(Anstiegs-),

*Decay* (Abfalls-), *Sustain* (Halte-) und *Release*-(Auskling-)Zeiten und -Levels. Die Hüllkurve wird nach den Anfangsbuchstaben dieser Worte manchmal auch ADSR-Kurve genannt.

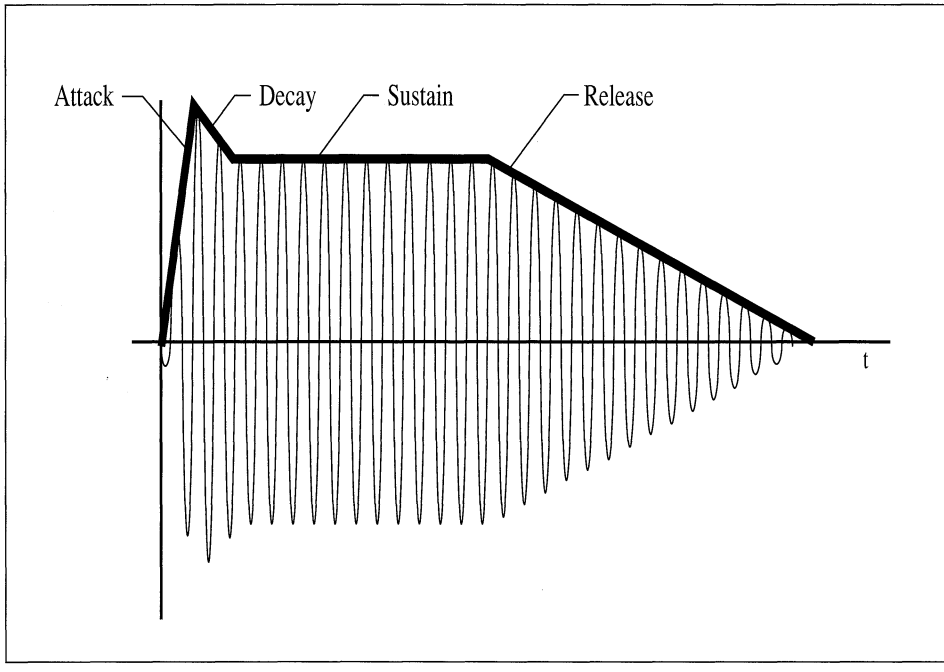



***Bild 11.14:*** *Die Hüllkurve eines Tons*

### 11.4.4 Klangerzeugung mit dem Amiga

Nach dieser mehr theoretischen Vorrede über vom Computer erzeugte Klänge werden Sie nun aber auch erfahren, was man wirklich dazu benötigt, wenn der Amiga einen Ton erzeugen soll. Wie oben beschrieben, braucht man dazu vor allem eine digitalisierte Periode des gewünschten Klangs (also eine Zahlenfolge), die im Speicher hinterlegt werden muß. Diese Zahlenfolge kann man entweder bei einem wirklichen Ton messen oder mit Bleistift und Papier erstellen. Die Zahlenfolge für einen mathematisch beschreibbaren Klang, wie zum Beispiel die Sinuswelle, kann man zum Beispiel auch berechnen. In bestimmte Speicherstellen (für Fachleute: in Register des Soundchip Paula) schreibt man danach die Adresse, an der die Zahlenfolge im Speicher zu finden ist, ihre Länge und die Zeit, nach der jeweils ein neuer Wert der Zahlenfolge ausgegeben werden soll. Diese letzte Zahl bestimmt, wie schnell eine Periode ausgegeben wird und damit auch, welche Frequenz der erzeugte Ton hat.

Ist man mit diesen Vorarbeiten fertig, muß man in eine andere Speicherstelle einen speziellen Wert schreiben. Das ist das Signal für den Amiga (genauer: den Sound-Chip Paula), sich die Zahlenwerte für den Verlauf der Kurve zu holen und daraus Töne zu

erzeugen. Wenn die Klangerzeugung erst einmal begonnen hat, braucht sich das Programm, das die Zahlenwerte für den Klang hinterlegt hat, nicht mehr darum zu kümmern. Der Amiga holt sich selbständig neue Werte aus der digitalisierten Zahlenfolge und wandelt diese in Spannungen am Lautsprecherausgang um.

Die weiter oben beschriebene Hüllkurve wird aber leider nicht durch die Hardware des Amiga eingehalten. Hier muß die Systemsoftware helfend eingeifen und während der Erzeugung des Tons die Lautstärke regeln. Dies geht natürlich auf Kosten der Leistung des Prozessors, der sich in dieser Zeit nicht um andere Programme kümmern kann. Der Leistungsverlust fällt aber sehr klein aus, da diese Lautstärke-Änderungen immer nur nach Zeiträumen anfallen, die einem schnellen Mikroprozessor wie dem 68000 als halbe Ewigkeit erscheinen.

Leider läßt sich aber nicht jeder Klang durch eine kurze Periode beschreiben, die dann mit einer bestimmten Geschwindigkeit abgespielt werden kann. Viele Geräusche besitzen keine solche Regelmäßigkeit. In diesem Fall muß man entweder die Länge der digitalisierten »Periode« auf die gesamte Dauer des gewünschten Tons ausdehnen oder einen anderen Betriebsmodus der Sounderzeugung verwenden.

In diesem zweiten Modus macht die Hardware fast gar nichts mehr alleine. In diesem Modus muß das Programm die Spannung, die am Lautsprecherausgang anliegen soll, und die Dauer, während der sie konstant gehalten werden soll, direkt in bestimmte Speicherstellen schreiben. Ist die angegebene Zeitdauer vorbei, wird dies dem Programm gemeldet (für Fachleute: durch einen Interrupt) und ein neuer Wert verlangt. Das eigentliche Programm wird dadurch immer wieder (und zwar in sehr kurzen Abständen) unterbrochen und muß die Hardware mit neuen Daten versorgen. Alle anderen Programme werden während einer so aufwendigen Klangerzeugung deshalb merklich langsamer.

### 11.4.5 Spezialeffekte

Bisher wurde nur beschrieben, wie ein Ton erzeugt wird. Der Amiga besitzt aber vier Tonerzeugungs-Kanäle. Jeder dieser Kanäle kann völlig unabhängig von den anderen betrieben werden. Sie können von den vier verschiedenen Kanälen also – ähnlich wie mit vier verschiedenen Instrumenten – zur selben Zeit vier völlig verschiedene Töne erzeugen lassen. Verwendet ein Programm allerdings alle vier Kanäle im zweiten Betriebsmodus (siehe oben), verlangsamt das alle laufenden Programme schon recht beachtlich.

Die von den verschiedenen Kanälen erzeugten vier Stimmen werden auf zwei Lautsprecherausgänge verteilt. Der Programmierer kann bei der Tonerzeugung wählen, welcher Klang auf welchen Lautsprecherausgang geleitet wird. (Es müssen aber immer zwei Kanäle links und zwei rechts ausgegeben werden. Einen Kanal links und drei rechts auszugeben geht nicht.) Dadurch sind Stereoeffekte ohne weiteres möglich. Durch geschickte Programmierung können Sie also zum Beispiel einen Zug von links nach rechts fahren lassen.

Die Sound-Hardware des Amiga erlaubt aber noch einige viel raffiniertere Tricks. Es können zum Beispiel zwei der Klangkanäle miteinander gekoppelt werden. Der eine (der sogenannte Oszillator) erzeugt dann den »eigentlichen« Ton, während die vom zweiten Kanal (dem sogenannten Modulator) erzeugte Schwingung zur Modulation des ersten Tons verwendet wird. Der Modulator besitzt dabei meist eine wesentlich niedrigere Frequenz als der Oszillator. Je nach der Einstellung der verschiedenen für diese Tricks zuständigen

Register kann der Modulator nun die Frequenz des Oszillators oder seine Lautstärke ändern. Damit kann man die typischen Effekte erzielen, zu denen auch menschliche Musiker mit ihren Instrumenten fähig sind. Dazu gehören zum Beispiel Vibratos, Tremolos, Glissandos und so weiter.

Eine Kopplung zweier Kanäle für Spezialeffekte dieser Art findet übrigens in nahezu identischer Form auch in professionellen Synthesizern statt. Nur besitzen solche Synthesizer einige Klangkanäle mehr. Die Verwendung eines Kanals zur Modulation eines anderen schränkt deshalb die Anzahl der zur Verfügung stehenden Kanäle (Stimmen) kaum ein. Beim Amiga 500 hingegen muß man sich schon sehr genau überlegen, ob man diese Modulationsmöglichkeiten wirklich braucht oder lieber mehr Stimmen für eine Melodie verwenden möchte.

Obwohl die Möglichkeiten des Amiga in dieser Hinsicht recht eingeschränkt sind, können Sie mit der passenden Software zumindest die Grundbegriffe der elektronischen Musikerzeugung kennenlernen. Das Programm Aegis Sonix zum Beispiel bietet am Bildschirm einige Regler und Einstellmöglichkeiten, die jedem, der schon einmal mit einem professionellen Synthesizer gearbeitet hat, vertraut sein werden. Wenn auch die Klangqualität des Amiga höchsten Ansprüchen nicht genügen wird, können Sie doch eine Menge damit lernen.

### 11.4.6 Sprachsynthese

Eine akustische Fähigkeit des Amiga darf man auf keinen Fall vergessen: die Sprachausgabe. Der Amiga 500 (beziehungsweise sein Vorgänger, der Amiga 1000) dürfte so ziemlich der erste Computer sein, der ohne zusätzliche Software und Hardware sofort sprechen konnte. Die Hard- und Software, die zur Sprachsynthese benötigt wird, ist im Amiga beziehungsweise im Betriebssystem »fest eingebaut«. Sprachsynthese kann deshalb in jedem Programm eingesetzt werden und ist problemlos zu programmieren. In der simpelsten Form übergibt man einfach einen ganz normalen Text an eine bestimmte System-Routine, die ihn daraufhin in akustischer Form ausgibt.

Die Aussprache kann man zusätzlich in verschiedenster Form beeinflussen. Die Stimmlage kann geändert werden. Der Amiga kann entweder im tiefen Baß oder mit einer hellen Frauenstimme sprechen. Die Geschwindigkeit der Sprache kann innerhalb weiter Grenzen geändert werden – sogar in viel weiteren Grenzen als sinnvoll ist. Und schließlich kann der Amiga entweder sehr ausdrucksvoll, mit betonten Wörtern und Satzzeichen oder so monoton sprechen, wie es sich für einen Computer gehört.

So lesen Sie es zumindest in der Dokumentation des Amiga. Wenn Sie schon ein wenig mit dem Programm *Say* herumgespielt haben, das einen direkten Zugriff auf die Sprachsynthesefähigkeiten des Amiga bietet, werden Sie schon wissen, daß die Sache nicht ganz so leicht ist – zumindest wenn Sie wirklich verständliche Sprache erzeugen wollen. Die Umsetzung von Buchstaben in zusammenhängend ausgesprochene Wörter und Sätze ist nämlich eine sehr komplexe Aufgabe, die durch viele Ausnahmeregeln erschwert wird. Überlegen Sie sich doch zum Beispiel einmal, wieviele verschiedene Möglichkeiten es gibt, einen Vokal wie *i* auszusprechen. Dieser eine Buchstabe kann völlig verschieden ausgesprochen werden, je nachdem, welche anderen Buchstaben sich in seiner »Nachbarschaft« befinden. Konsonanten und Wortbetonungen sind – zumindest für einen Computer – fast noch schwieriger zu handhaben.

Glücklicherweise kann man auch »eine Ebene tiefer« ansetzen. Die Sprachsynthese-Software des Amiga kann nicht nur Klartext aussprechen, sondern auf Wunsch auch »Phoneme«. Phoneme kennt jeder, der schon einmal in ein Wörterbuch geschaut hat. Es sind die Zeichen, mit denen die Aussprache von Wörtern beschrieben wird. Phoneme sind die »Urbausteine« der Sprache, elementare Silben und Laute, aus denen man jedes Wort (fast) jeder Sprache zusammensetzen kann.

Mit etwas Mühe – das heißt, bei manchen Wörtern müssen Sie die Phonemschreibweise im Wörterbuch nachschauen – kann man den Amiga auch dazu bringen, deutsche Sätze auszusprechen. Ein unverkennbarer (amerikanischer) Computer-Akzent bleibt allerdings. Trotzdem ist die Sprachausgabe für viele Anwendungszwecke – zum Beispiel für Fehlermeldungen oder zum Erschrecken von Freunden – recht brauchbar. Auch ohne auf den Bildschirm zu schauen, kann einem ein Programm auf diese Weise mehr Informationen mitteilen als mit einem kurzen Piepston, der sonst für solche Zwecke verwendet wird.

### 11.4.7 MIDI

Wie es inzwischen scheinbar schon guter Ton bei neuen Computern zu werden scheint, ist der Amiga natürlich auch für die Kommunikation mit MIDI-Geräten vorbereitet. MIDI (= ***M**usical **I**nstruments **D**igital **I**nterface*) ist der wohl wichtigste Standard für die Verbindung digitaler Musikinstrumente, wie Synthesizer und Keyboards. Hierzu benötigt man ein kleines Kästchen namens MIDI-Interface, das an den seriellen Ausgang des Amiga angeschlossen wird und die Signale des Amiga dem MIDI-Standard anpaßt. Dann braucht man »nur noch« einen guten Synthesizer und ein MIDI-fähiges Keyboard.

Wem also die Klangqualität seines Amiga auch dann noch nicht ausreicht, wenn er ihn an seine Stereoanlage angeschlossen hat, der sollte sich die Anschaffung des MIDI-Interfaces und der entsprechenden MIDI-Geräte überlegen.

Man mag sich nun vielleicht fragen, wozu der Computer überhaupt noch gut ist, wenn die Musik auf einer separaten Tastatur eingegeben und auf einem Synthesizer erzeugt wird. Die Antwort darauf ist wieder einmal mehr bei der Software zu finden. Es ist nämlich wesentlich komfortabler, Musik auf einem Computer zu entwickeln als auf einem Blatt Papier. Dies wird noch deutlicher, wenn Musik mit noch mehr als nur vier Stimmen erzeugt werden soll und die Bedienung eines leistungsfähigen (und komplizierten) Synthesizers durch den Musiker nötig wird.

Man könnte den Amiga deshalb sehr sinnvoll als Oberaufseher in einem System MIDI-fähiger Komponenten einsetzen. Auf seinem Bildschirm bekommt man einen Überblick über die verschiedenen Vorgänge und Verbindungen gezeigt, man komponiert aber auch gleichzeitig auf dem Computer und entwickelt neue Klänge. Die Klangformen, Hüllkurven und Noten kann man sich natürlich direkt auf dem Amiga probeweise anhören. »Gespielt« werden die Melodien dann aber auf einem Synthesizer, zu dem man die dazu nötigen Informationen über MIDI-Kabel schickt. Mit einem speziellen Keyboard kann man aber auch »direkt« den Synthesizer bedienen, ohne die Noten dazu erst in den Amiga eingeben zu müssen. Nur die Klangformen und Hüllkurven werden in diesem Fall vom Amiga geliefert. Das Programm Deluxe Music der Firma Electronic Arts bietet zum Beispiel exakt diese Möglichkeiten!

# 11.5 Eine Handvoll Chips

In diesem Abschnitt werden wir noch ein wenig tiefer in die »Innenwelt« des Amiga einsteigen. Nachdem Sie in den ersten Abschnitten dieses Kapitels die beachtlichen Fähigkeiten des Amiga bei der Erzeugung von Bildern und Tönen kennengelernt haben, werden Sie jetzt erfahren, wie diese Fähigkeiten zustande kommen. Dabei dreht sich alles um eine Handvoll spezieller Mikro-Chips, die zum Teil nur im Amiga zu finden sind. Keine Angst: Sie benötigen diese Informationen nicht unbedingt zur Arbeit mit dem Amiga. Glücklicherweise gibt es heute fast keine Computer mehr, über deren Hardware oder Systemsoftware man detailliert Bescheid wissen muß, um damit umgehen zu können. Interessant ist es aber allemal, mehr über diese Spezial-Chips zu erfahren, die – mehr als alle anderen Komponenten des Amiga-Systems – den Amiga 500 zum »Amiga« machen. Sie tragen die Namen »Paula«, »Denise« und »Agnus«. (Es gibt noch einen Spezialchip namens »Gary«, der in diesem Zusammenhang aber nicht so interessant ist, da er nur die Funktionen vieler kleinerer Chips zusammenfaßt und ansonsten nicht viel Weltbewegendes leistet.) Diese liebevollen Bezeichnungen für kleine schwarze Chips mit stacheligen Beinen lassen erahnen, wie sie den Entwicklern am Herzen liegen.

## 11.5.1 Die CPU – Herr im Hause Amiga

Objektiv betrachtet ist sicherlich die sogenannte »CPU« (*Central Processing Unit* oder auf gut deutsch Zentraleinheit) der wichtigste Chip im Innern des Amiga 500, wenn auch die anderen Chips spektakulärer sind. Die CPU ist der Chip in jedem Computer, der den Löwenanteil der Arbeit zu leisten hat. Es ist dieser Chip, der Programme wirklich ausführt – also zum Beispiel rechnet und Entscheidungen trifft. Man könnte die CPU auch als das »Herz des Computers« bezeichnen.

Die CPU des Amiga ist der 16/32-Bit-Prozessor 68000 der Firma Motorola. Er und seine schnelleren Nachfolger 68020 und 68030 werden auch in anderen Computern dieser Preisklasse sowie in professionellen Systemen für 30 000,– DM und mehr eingesetzt. Die Bezeichnung 16/32-Bit-Prozessor kommt daher, daß man sich nicht darüber einig werden kann, nach welchen Kriterien man einen Prozessor in welche Kategorie einordnet. Typischerweise ist es so, daß Prozessoren, die jeweils 8 Bit in einem Arbeitsschritt verarbeiten können (zum Beispiel zwei 8-Bit-Zahlen addieren können) 8-Bit-Prozessoren heißen. Der 6502 im Apple //e und im Commodore C64 fallen zum Beispiel in diese Kategorie. Prozessoren, die in einem Schritt 16 oder 32 Bit verarbeiten können, heißen entsprechend 16- oder 32-Bit-Prozessoren. Ein anderes Kriterium ist aber auch, wieviele Bits in einem Schritt aus dem Speicher gelesen oder in ihm abgelegt werden können. Der 68000 erfüllt bei einigen dieser Kriterien die Bedingungen für einen 32-Bit-Prozessor und bei anderen nur die für einen 16-Bit-Prozessor. (Die Nachfolgemodelle des 68000, der 68020 und der 68030, erfüllen in jeder Hinsicht die Bedingungen an einen echten 32-Bit-Chip. Diese Chips können unter anderem auch im Amiga 2000 und 2500 verwendet werden.)

Kommen wir nun aber zu den drei Spezial-Chips. Bei ihnen handelt es sich um sogenannte »Custom-Chips«, das heißt, sie wurden speziell für einen Computer-Hersteller (in diesem Falle für Commodore beziehungsweise Amiga) entwickelt und werden auch nur für ihn hergestellt. Die meisten anderen Chips, die Sie im Amiga 500 finden werden, sind Standard-Bauteile, die jeder Kunde dem Chip-Hersteller abkaufen und dann in seinen eigenen

Produkten verwenden kann. Standard-Chips sind fast immer billiger als Custom-Chips, weil sie üblicherweise in größeren Stückzahlen verkauft werden. Sie stellen andererseits aber auch immer einen Kompromiß zwischen den Wünschen vieler Anwender dar. Für wirklich einzigartige Möglichkeiten oder besonders hohe Leistungen benötigt man deshalb Custom-Chips.

### 11.5.2 Teamwork

Die Hardware des Amiga betreibt echtes Teamwork. Die drei Zusatz-Chips »Agnus«, »Paula« und »Denise« arbeiten unabhängig von der CPU aber gleichzeitig mit ihr an denselben Aufgaben. Das ist eine Behauptung, die Sie vielleicht schon einmal gehört haben, deren Bedeutung aber nicht immer sofort klar ist. Der Vorteil, den dieses Arbeitsprinzip mit sich bringt, ist auch nicht ganz so leicht zu erklären, weshalb ich etwas weiter ausholen werde.

Es gibt viele Computer, die alle anfallenden Arbeiten der CPU überlassen. Die CPU muß Grafikdarstellungen, Musik, Berechnungen und natürlich auch die Ein- und Ausgabe übernehmen. Da alle Arbeiten aber von einem Prozessor übernommen werden, müssen diese Tätigkeiten hintereinander ausgeführt werden. Jede dieser Aufgaben hält die CPU also eine Weile von den anderen Aufgaben ab.

Die Spezial-Chips des Amiga nehmen nun dem Hauptprozessor Arbeit ab. Grafikdarstellungen, Sounderzeugung und noch andere Arbeiten werden durch die drei Chips erledigt. Der Prozessor hat dann zwar immer noch etwas – aber nicht mehr sehr viel – mit diesen Aufgaben zu tun. Um diese Verlagerung von Aufgaben möglichst effektiv zu gestalten, können manche Spezial-Chips ihre Arbeit gleichzeitig mit der CPU erledigen. Es handelt sich dann um sogenannte »Koprozessoren«, die völlig unabhängig von der CPU ihrer Arbeit nachgehen. Die CPU bleibt natürlich immer noch »Herr im Hause Amiga«, gibt aber – wie ein guter Manager – »niedere Aufgaben« an ihre Mitarbeiter weiter und wendet sich dann anderen Dingen zu.

Man muß sich das so vorstellen, daß die CPU eine Aufgabe vorbereitet und den Koprozessor mit allen nötigen Informationen darüber versorgt. Dann sagt sie dem Koprozessor, daß er mit der Arbeit beginnen soll und kümmert sich um andere Dinge. Währenddessen wird auch der Koprozessor tätig und erledigt die ihm aufgetragene Arbeit. Wenn er damit fertig ist, signalisiert er das der CPU (mit einem sogenannten »Interrupt«) und kann wieder neu mit Arbeit versorgt werden oder eine Pause einlegen (auch Chips brauchen mal eine Pause). Statt sich vom Koprozessor die Beendigung einer Arbeit mitteilen zu lassen, kann die CPU aber auch »zwischendurch immer mal wieder nachschauen«, ob der Koprozessor schon mit seiner Arbeit fertig ist und gegebenenfalls darauf reagieren. (Der Computerfachmann nennt diesen Vorgang des regelmäßigen Nachschauens übrigens »Polling«.)

Bei einer solchen Konfiguration aus CPU und Koprozessoren tritt oft nur ein Problem auf. Sowohl die CPU wie auch die Koprozessoren besitzen üblicherweise nur sehr wenig eigenen Speicherplatz, die sogenannten »Register«. Für größere Aufgaben benötigen sie deshalb den RAM-Speicher, der ungleich mehr Daten fassen kann. In vielen Computern gibt es Koprozessoren und andere Hilfs-Chips, die Daten nicht selbst aus dem Speicher holen oder dort hinterlegen können, sondern dazu immer die CPU benötigen, weil diese als einziger Chip an den Speicher angeschlossen ist. Immer, wenn solche »unselbständigen« Chips

Daten brauchen oder im Speicher ablegen wollen, müssen Sie die CPU von ihren eigentlichen Aufgaben abhalten und sie als »Datenschaufel« mißbrauchen. Ein paralleles Arbeiten, das nötig ist, um einen Zusatz-Chip so richtig auszunutzen, ist also nur so lange möglich, wie der Speicher nicht benötigt wird.

Auf dem Amiga ist das anders! Hier sind nämlich auch die Koprozessoren an den Speicher angeschlossen. Das entsprechende Arbeitsprinzip heißt »DMA« (für *Direct Memory Access* oder auf deutsch direkter Speicherzugriff). Von Chips, die von selbst auf den Speicher zugreifen können, spricht man deshalb auch als DMA-Chips. Auf den Speicher des Amiga kann man über insgesamt 25 Wege oder »Kanäle« zugreifen. Einige dieser DMA-Kanäle nutzen auch die Spezial-Chips, andere sind für künftige Erweiterungen noch frei.

DMA ist ein recht fortschrittliches und teures Konzept, das erst vor relativ kurzer Zeit Eingang in preiswerte Computer gefunden hat. Es ist für viele Anwendungen nützlich. Neben Aufgaben wie Grafik und Klangerzeugung – die DMA benötigen, weil bei diesen Vorgängen Daten aus dem Speicher gelesen und in Bilder und Töne umgewandelt werden müssen – gibt es noch eine Reihe anderer Anwendungen für DMA. Dazu gehören vor allem auch die Ein- und Ausgabe von Daten an Peripheriegeräte wie Bildschirm, Drucker und so weiter. Die DMA-Chips des Amiga nehmen dem Hauptprozessor aber nicht nur Arbeit ab. Sie können die ihnen übertragenen Arbeiten auch viel schneller erledigen als der Hauptprozessor das überhaupt könnte. Die CPU ist schließlich nicht speziell für Grafikanwendungen oder für das Hin- und Herschieben von Daten im Speicher gebaut worden – wie die Spezial-Chips es sind.

Obwohl die Spezial-Chips die CPU nicht für den Speicherzugriff benötigen, tritt trotzdem ein Engpaß beim »Eingang in den Speicher« auf. Die Chips und der Prozessor können normalerweise nämlich nicht gleichzeitig auf den Speicher zugreifen. Dieser kann nämlich immer nur mit einem anderen Bauteil zur gleichen Zeit kommunizieren. Es gibt zwar besondere Speicher, die mit zwei oder mehr anderen Chips gleichzeitig Daten austauschen können (dual ported RAMs), diese sind aber sehr viel teurer als die im Amiga verwendeten Chips und für einen preiswerten Personalcomputer wie den Amiga somit kaum geeignet.

Glücklicherweise kann der Speicher jedoch mit viel höherer Geschwindigkeit Daten aufnehmen und abgeben, als die CPU dies kann. Diese Geschwindigkeit, mit der Chips arbeiten können, nennt man ihre »Taktfrequenz« und ist gleich der Anzahl von »Taktzyklen«, die in einer Sekunde durchlaufen werden. Die CPU des Amiga durchläuft in jeder Sekunde etwa 7,2 Millionen solcher Taktzyklen und hat somit eine Taktfrequenz von 7,2 MHz (Mega-Hertz). In jedem Taktzyklus kann die CPU einen Arbeitsschritt durchführen, also zum Beispiel ein Wort (16 Bit) Daten vom Speicher empfangen. Der RAM-Speicher läuft aber mit einer Taktfrequenz von 14,4 MHz, hat also in der Sekunde die doppelte Anzahl von Taktzyklen und könnte deshalb doppelt so viele Daten in jeder Zeiteinheit liefern als die CPU vertragen kann.

Diesen »Geschwindigkeitsvorsprung« des Speichers haben die Entwickler des Amiga ausgenutzt, indem sie den RAM-Speicher nur jeden zweiten Takt mit der CPU kommunizieren lassen. Alle anderen Takte werden auf die restlichen DMA-Chips verteilt. Da diese den Speicher meist nicht so intensiv benötigen wie die CPU, können so CPU und Spezial-Chips relativ ungestört voneinander und gleichzeitig ihrer Arbeit nachgehen.

Nur unter ganz besonderen Umständen benötigen die Spezial-Chips mehr als die ihnen zustehende Hälfte der Taktzyklen des Speichers. In diesen Fällen wird die CPU kurz

angehalten, wenn Sie den Speicher zur selben Zeit benötigt wie ein anderer Chip und darf wieder weitermachen, wenn dieser Chip den Speicher nicht mehr braucht. (»Kurz« heißt in diesem Fall wirklich extrem kurz. Es handelt sich dabei um Zeitspannen, in denen kein Mensch reagieren oder auch nur etwas von einer solchen Pause merken könnte.) Diese Fälle, in denen die CPU gebremst wird, kommen aber nur sehr selten vor.

Wenn Sie zum Beispiel 32 verschiedene Farben gleichzeitig am Bildschirm sehen wollen, bremst Denise (der Videochip) die CPU merklich ab, weil er wesentlich mehr Daten aus dem Speicher holen muß, um die ganze Farbenpracht darzustellen. Wenn Sie sich aber mit weniger Farben begnügen, üben die CPU und die Spezial-Chips friedliche Koexistenz.

Damit sollten Sie jetzt zumindest einen ersten Eindruck vom internen Aufbau des Amiga und der modernen arbeitsteiligen Zusammenarbeit seiner Komponenten erhalten haben. Kommen wir nun zu den Chips im einzelnen.

### 11.5.3 Paula die Musikalische

Einer der drei Spezialchips des Amiga trägt den Namen »Paula«. Das ist nicht nur irgendein Name, sondern leitet sich auf eine heute nicht mehr ganz nachvollziehbare Weise aus den englischen Worten *peripheral* und *audio* ab, was schon andeutet, wofür dieser Chip zuständig ist. Er kontrolliert die Arbeit mit den Peripheriebausteinen und ist für die Klangerzeugung zuständig. Und er kann noch mehr, denn er regelt die sogenannte »Interruptverarbeitung«.

Interrupts sind, wörtlich übersetzt, Unterbrechungen. Dieser Ausdruck bezieht sich meist darauf, daß andere Chips oder externe Geräte die CPU in ihrer Arbeit unterbrechen, weil irgendein Ereignis eingetreten ist, das die Aufmerksamkeit der CPU verlangt. Koprozessoren senden zum Beispiel Interrupts an die CPU, wenn sie mit ihrer Arbeit fertig sind, und Ein-/Ausgabe-Bausteine schicken Interrupts, wenn Daten bereitstehen und die CPU entscheiden muß, was damit gemacht werden soll. Es gibt verschiedene Interrupts, mit denen der CPU verschiedene Nachrichten geschickt werden können. Zusätzlich stellen die Chips, die Interrupts verursachen, meist aber auch noch weitere Informationen in bestimmten Registern bereit, aus denen die CPU mehr über die näheren Umstände eines Interrupts entnehmen kann.

Bei einem Interrupt wird der gesamte Zustand des gerade laufenden Programms schnappschußartig festgehalten und ein spezielles kleines Programm gestartet, das sich um die Behandlung des Interrupts kümmert. Der Fachmann nennt solche Programme »Interrupt-Handler«. Ist der Interrupt-Handler fertig, wird das unterbrochene Programm wieder fortgesetzt, wobei zuvor der alte Programmzustand mit Hilfe des gespeicherten Schnappschusses wiederhergestellt wird.

Verschiedene Arten von Interrupts besitzen zudem auch verschiedene Prioritäten, die besagen, wie wichtig die Unterbrechung ist. Es kann ja sein, daß gerade kein »normales Programm« läuft, wenn ein Interrupt eintritt, sondern ein Interrupt-Handler. Falls dieser »wichtiger« ist als der neu eingetroffene Interrupt, darf er auf keinen Fall unterbrochen werden. Genau dies ermöglichen die Prioritäten. Es gilt die einfache Regel, daß Interrupts niedrigerer Priorität keine Interrupt-Handler von gleicher oder höherer Priorität unterbrechen können.

Der Paula-Chip hilft nun der CPU dabei, die Interrupts der verschiedenen Systemkomponenten zu erkennen und richtig einzuordnen. Das Amiga-DOS nutzt diese Interruptverarbeitung auch aus, um das Multitasking zu ermöglichen. Hierzu wird in regelmäßigen Abständen ein spezieller Interrupt erzeugt, der dazu führt, daß das gerade laufende Programm unterbrochen wird. Nach dem Ende des entsprechenden Interrupt-Handlers wird aber das unterbrochene Programm nicht wieder aufgenommen, sondern ein anderes, das bislang in Wartestellung stand, wird aktiv. Kurz darauf erfolgt wieder ein Interrupt, woraufhin das nächste Programm eine Weile arbeiten darf, bis es auch wieder unterbrochen wird. Irgendwann einmal – je nachdem, wieviele Programme arbeiten wollen – ist auch das zuerst unterbrochene Programm dann wieder »an der Reihe«.

Paula hilft der CPU in folgender Weise, Interrupts einfacher handhaben zu können: sie empfängt (noch vor der CPU) alle Interrupts der für Ein-/Ausgabe zuständigen Chips und der verschiedenen DMA-Chips und speichert alle Informationen darüber in speziellen Registern. In diesen Registern kann die CPU später nachschauen, was bei einer »Unterbrechung« genau vorgefallen ist. Paula gliedert die vielen möglichen Interrupts dabei zugleich auch in unterschiedliche Gruppen auf. Jede dieser Gruppen vereinigt Interrupts ähnlicher Art und gleicher Priorität. Da Paula alle diese Vorarbeiten schon leistet, bevor die CPU etwas von einem Interrupt erfährt, braucht diese viel weniger Zeit, um einen Interrupt einzuordnen und zu entscheiden, wie »angemessen« darauf zu reagieren ist.

Eine weitere Aufgabe des Paula-Chips ist der Kontakt zwischen Außenwelt und System. Paula betreut die Ein-/Ausgabe-Bausteine, mit deren Hilfe die verschiedenen Schnittstellen angesteuert werden. Dazu gehören die parallele Schnittstelle, die serielle Schnittstelle, der Anschluß für ein externes Diskettenlaufwerk und die beiden Maus-Anschlüsse. Daß Paula, also ein von der CPU getrennter Chip, diese Aufgabe übernimmt, bedeutet wieder eine Entlastung für die CPU, die sich sonst selbst um diese Aufgaben kümmern müßte. Paula dient sozusagen als Mittelsmann (Mittelsfrau?) zwischen dem eigentlichen Computer und den an den verschiedenen Schnittstellen angeschlossenen externen Geräten.

Auch die Klangerzeugung und Klangausgabe auf dem Amiga ist (nahezu) ausschließlich eine Sache des Paula-Chips. Die Schaltungen, die die Informationen über Form, Lautstärke und Dauer eines Klangs in die Spannungsschwankungen umsetzen, aus denen ein Lautsprecher dann die Töne macht, befinden sich alle in Paula.

### 11.5.4 Denise das Grafik-Wunder

Während Paula für die Klangerzeugung zuständig ist, erledigt der Custom-Chip Denise den größten Teil der bei der Erzeugung von Bildern anfallenden Aufgaben. Der Name »Denise« hat ähnlich mysteriöse Ursprünge wie der Name Paula und stammt von der Bezeichnung Display Encoder (zu deutsch etwa Bild-Kodierer) ab. Dieser Name deutet schon darauf hin, daß Denise hauptsächlich für die Erzeugung des Bildsignals zuständig ist, das an den Monitor beziehungsweise Fernsehbildschirm geht. Beim eigentlichen Aufbau des Bildes im RAM-Speicher ist ein anderer Chip behilflich, Agnus, der gleich noch näher beschrieben wird.

Wenn Sie sich noch einmal den ersten großen Abschnitt dieses Kapitels in das Gedächtnis zurückrufen, werden Sie sich vielleicht daran erinnern, was eine Bitmap ist. In Form einer Bitmap werden im RAM-Speicher die Bilder abgespeichert, die hinterher am Bildschirm

erscheinen. Zu jedem Punkt am Bildschirm enthält die Bitmap die Information, welche Farbe dieser Punkt hat. Denise enthält die Schaltungen, die den Inhalt der Bitmap lesen und die Farbinformationen in elektrische Signale umsetzen. Diese Signale gehen dann über das Monitorkabel zu Ihrem Monitor und steuern dort den Elektronenstrahl, der das Bild auf die Bildröhre zeichnet. Hierzu liest Denise die Informationen zu den einzelnen Bildpunkten in genau derselben Reihenfolge aus der Bitmap im Speicher, in der auch der Elektronenstrahl die Bildpunkte auf die Bildröhre zeichnet.

Die eben beschriebene Umsetzung einer Bitmap in eine Bildschirm-Grafik ist statisch – also unbeweglich. Wenn irgendetwas an dem Bild am Monitor geändert werden soll, muß dafür die Bitmap geändert werden. Zumindest ist das auf anderen Computern der Fall. Auf dem Amiga macht Denise bestimmte Arten von Änderungen deutlich einfacher. Denise kennt nämlich zweierlei Arten bewegter Grafiken. Zum einen kann der Hintergrund einer Grafik (das sogenannte Playfield) in Wirklichkeit eine sehr große Fläche, größer als der Bildschirm, sein, aus dem nur ein Teil gezeigt wird. Dieser sichtbare Ausschnitt kann stufenlos und schnell über dem gesamten Playfield verschoben werden. Die zweite Möglichkeit, Bewegung ins Bild zu bringen, sind die sogenannten Sprites, kleine farbige Bilder, die sich »über« dem Playfield befinden und sehr schnell, unabhängig von diesem, bewegt werden können. Sprites erfreuen sich auch auf anderen Computern schon großer Beliebtheit und lassen sich (gegenteilig lautenden Meinungen zum Trotz) auch außerhalb von Video-Spielen sinnvoll einsetzen. Der Mauszeiger ist beispielsweise ein Sprite.

Der Denise-Chip erkennt außerdem noch Kollisionen verschiedener Sprites mit Teilen der Hintergrundgrafik und mit anderen Sprites. Programme können diese Kollisionsabfrage benutzen und eine entsprechende Reaktion darauf einleiten.

### 11.5.5 Agnus der Rechenkünstler

»Agnus« ist eine Abkürzung, die auf heute unerfindlichen Wegen aus »Adreßgenerator« entstanden ist. Agnus beinhaltet aber noch mehr als nur diesen Adreßgenerator. Weitere wichtige Bestandteile dieses Chips sind der »Blitter« und der »Copper«, über die Sie in diesem Abschnitt noch mehr erfahren werden. Eigentlich heißt der Agnus-Chip im Amiga 500 übrigens »Fat-Agnus« (dicker Agnus), weil er gegenüber dem ersten Agnus im Amiga 1000 noch eine Menge Funktionen dazubekommen hat. Eine solch unfreundliche Bezeichnung möchte ich diesem flinken Rechenkünstler aber ersparen. Agnus ist alles andere als fett und träge.

Ein Hauptbestandteil des Agnus-Chips ist ein großer Adreßgenerator. Die Custom-Chips des Amiga haben ja unabhängig von der CPU direkten Zugriff auf den RAM-Speicher. Dazu benötigen diese Chips jedoch sogenannte »Adreßgeneratoren« und DMA-Kanäle. Über DMA-Kanäle wird der Zugriff auf den RAM-Speicher abgewickelt. Ein Adreßgenerator wandelt die Signale eines Chips in vollständige Speicheradressen um oder ordnet bestimmte Speicheradressen einzelnen Registern in den Chips zu. Man kann sich das so vorstellen, daß von einem Haus statt der Postanschrift nur die Beschreibung »erste Straße nach dem Ortseingang und dann drittes Haus auf der linken Seite« bekannt ist. Ein Adreßgenerator würde daraus dann »Schillerstraße 12, 4783 Klotzhausen« machen.

Keiner der beiden anderen Chips enthält jedoch alle DMA-Kanäle oder Adreßgeneratoren. Alle Speicherzugriffe laufen deshalb über Agnus und werden von ihm kontrolliert und

koordiniert. Insgesamt hat Agnus 25 DMA-Kanäle, über die 25 verschiedene Geräte ohne Beihilfe der CPU Daten in den Speicher schreiben oder aus ihm lesen können.

Eine der wichtigsten Vorgaben bei der Konstruktion des Amiga war es, eine möglichst schnelle Erzeugung und Änderung von Grafiken zu ermöglichen. Insbesondere sollten ohne viel Aufwand »animierte« bewegte Grafiken möglich sein. Zum Teil werden diese Aufgaben von Denise erledigt. Sprites ermöglichen zum Beispiel sehr schnelle Bewegungen auf dem Bildschirm. Sprites sind von ihren Möglichkeiten her aber auch sehr eingeschränkt. Für aufwendigere Effekte muß deshalb die Bitmap, die einem Bild zugrunde liegt, geändert werden. Im Gegensatz zur Änderung von Sprites müssen zur Änderung der Hintergrund-Bitmap eines Bildes große Mengen von Daten blitzschnell im Speicher verschoben werden. Dies ist die Spezialität des »Blitters«, eines relativ selbständigen Prozessors innerhalb von Agnus.

Der Name »Blitter« hat schon eine recht lange Tradition und beruht auf der Prozedur »BitBlt« (*bit block transfer*), die es auf einem älteren Computer der Firma XEROX gab. Diese Prozedur war für das schnelle Verschieben »rechteckiger« Bereiche in einer Bitmap zuständig. Und da die für den Amiga entwickelte Hardware-Einheit innerhalb von Agnus genau dasselbe tut, lag es nahe, diesen Namen zu verwenden. Jay Miner, der Designer der drei Amiga-Chips, ist in dieser Hinsicht anderer Meinung. Der Blitter des Amiga kann auch wirklich einiges mehr, als nur Daten hin- und herzuschieben. Jay Miner nennt ihn deshalb Bimmer für *Bit Image Manipulator* (zu deutsch etwa Bit-Bild-Manipulierer). Dies soll andeuten, daß dieser Prozessor nicht nur Daten durch den Speicher schaufeln, sondern zugleich komplexe Operationen damit durchführen kann.

Der Blitter hat in vielen grafikintensiven Programmen eine ganze Reihe von Aufgaben zu übernehmen. Die erste davon ist das schnelle Füllen von Flächen (wie es zum Beispiel Graphicraft und Deluxe Paint demonstrieren). Der Blitter kann Flächen mit einer Geschwindigkeit von einer Million Bildpunkten pro Sekunde füllen. Mit derselben Geschwindigkeit kann er auch gerade Linien ziehen. Beide Fähigkeiten verknüpft er übrigens auch sehr geschickt. Um eine bestimmte Fläche auf dem Bildschirm zu füllen, muß diese durch mindestens eine dünne Linie in einer anderen Farbe begrenzt sein. Programme zeichnen polygonförmige Flächen (zum Beispiel auch Rechtecke) deshalb, indem Sie den Blitter blitzschnell die Seiten des Polygons ziehen lassen und dann das Innere wieder durch den Blitter füllen lassen. Dieser Vorgang läuft so schnell ab, daß die Fläche scheinbar schlagartig auf dem Bildschirm erscheint.

Wichtigste Aufgabe des Blitters ist aber die Bewegung von Objekten auf dem Bildschirm. Sinnigerweise heißen diese Objekte dann auch *Blitter Objects* (BOBs). Der Blitter kann extrem schnell (so schnell, wie es die Speicherchips überhaupt nur können) Daten im Speicher von einem Ort zu einem anderen bewegen. Um BOBs als eine Art eigenständiger Bilder »über der Hintergrundgrafik« zu bewegen, werden diese mit Hilfe des Blitters direkt in die Bitmap des Hintergrundbildes hineinkopiert. Vorher wird (ebenfalls mit dem Blitter) der alte Inhalt der Bitmap in einem Puffer gesichert. Wird das BOB dann bewegt, so wird diese gesicherte Kopie des alten Inhalts in die Bitmap zurückkopiert, der Inhalt des Hintergrundbildes an der neuen Position des BOBs gesichert, und dann das BOB an der neuen Stelle gezeichnet. Für diese verschiedenen Aktionen wird zwar auch viel geschickt programmierte Software benötigt, den Löwenanteil der Arbeit erledigt jedoch der Blitter – weshalb die BOBs ihren Namen zu Recht tragen.

Solche Datenverschiebungen könnte die CPU auch übernehmen – aber bei weitem nicht so schnell wie der Blitter. Eine CPU wie die M68000 kann Speicherbewegungen nämlich nur mit ganzen Gruppen von Bytes (zu 8 Bit) und Wörtern (zu 16 Bit) effizient und schnell durchführen (der Blitter ist allerdings auch bei solchen Gruppen schneller). Um ein unregelmäßig geformtes Objekt auf dem Bildschirm zu bewegen, muß man beliebige Häppchen von Wörtern und Bytes – unter Umständen sogar einzelne Bits – bewegen. Der Blitter kann Daten beliebiger Länge und an beliebigen Bit-Positionen im Speicher lesen und wieder hineinschreiben, ohne daß dies Geschwindigkeitseinbußen mit sich bringen würde. Die CPU würde bei Daten dieser Art sehr langsam werden.

Und weil der Blitter das Datenschaufeln so viel besser kann als die CPU, darf er dabei die CPU und alle anderen Chips auch vom Speicher »abschneiden«. (Dieses unhöfliche Verhalten des Blitters kann man allerdings auch abschalten.) Dadurch kann er dann Daten so schnell im Speicher hin- und herschieben wie die Speicherchips dazu überhaupt fähig sind. (Ansonsten müßte er sich ja den Speicher mit der CPU teilen, wie oben beschrieben, und würde nur die halbe Speichergeschwindigkeit nutzen können.) Das führt dazu, daß die CPU auf neue Daten aus dem Speicher warten muß, bis der Blitter fertig ist. Für Sie als Anwender oder Programmierer bedeutet das aber keine Geschwindigkeitseinbuße, wegen der man vom Gebrauch des Blitters absehen sollte. Immer, wenn der Blitter die CPU vom Speicher abschneidet, erledigt er Aufgaben, die die CPU in derselben Zeit sowieso nicht schaffen würde!

Der Blitter des Amiga 500 kann – im Gegensatz zu Blittern in anderen Computern – aber nicht nur Bereiche in einer Bitmap verschieben. Er kann bei diesem Verschiebevorgang auch Daten aus bis zu drei verschiedenen Quellen logisch miteinander kombinieren und die Ergebnisse dieser logischen Verknüpfung zurück in den Speicher schreiben. (Diese drei Quellen können verschiedene Bitmaps oder verschiedene Gebiete innerhalb einer Bitmap sein.) Erst das ermöglicht auch die Verschiebung völlig unregelmäßig geformter Flächen.

Der neben dem Blitter zweite wichtige Bestandteil von Agnus ist der Copper. »Copper« ist eine Abkürzung für »Co-Processor«. Auch der Blitter ist natürlich ein Koprozessor, wie Sie gerade erfahren haben – aber für den Copper hatte man wohl keinen besseren Namen zur Hand. Der Copper ist allerdings auch ein ganz besonderer Koprozessor. Er ist nämlich an den Elektronen-Strahl gekoppelt, mit dem das Bild im Monitor gezeichnet wird und arbeitet zu diesem Zweck eng mit Denise zusammen.

Der Copper kontrolliert, während ein Bild auf den Bildschirm gezeichnet wird, permanent, wo sich der Elektronenstrahl gerade befindet (in welcher Zeile und Spalte des Bildes). Der Programmierer kann ihm befehlen, auf eine bestimmte Position zu warten, und kann bei Erreichen dieser Position Befehle an die anderen Chips schicken (indem er bestimmte Werte in die Register dieser Chips schreibt). Dadurch ergeben sich erstaunliche Möglichkeiten. Die Register des Chips Denise bestimmen zum Beispiel, welches Bild auf dem Bildschirm erscheint, welche Farben dafür zur Verfügung stehen und so weiter. Indem der Copper diese Register ändert, wenn der Elektronenstrahl auf halber Höhe des Bildes ist, kann er zum Beispiel die Farbpalette ändern. Dadurch können zum Beispiel in der oberen Hälfte des Bildes andere Farben als in der unteren verwendet werden.

Im Extremfall kann man den Copper aber nicht nur die Farbpalette, sondern alle Eigenschaften des Bildschirms – einschließlich des angezeigten Bildes – »mittendrin« ändern lassen. Das führt dann zu den Screens, die Sie ja schon im ersten Buchteil kennengelernt haben.

Screens, eine Art virtuelle Bildschirme, erlauben es, die Ausgaben mehrerer Programme am Bildschirm zu betrachten, die völlig unterschiedliche Anforderungen an Bildschirmfarben und die Bildauflösung stellen. Als geschickter Programmierer kann man aber noch eine Menge anderer Tricks mit dem Copper anstellen, die Sie zweifellos schon in einigen Grafik-Programmen gesehen haben – ohne natürlich zu wissen, daß der Copper dafür zuständig ist.

### 11.5.6 Chip-RAM und FastMem

Die drei Spezial-Chips sind auch verantwortlich für einen ganz entscheidenden Aspekt der »Amiga-Architektur«, die Unterscheidung zwischen Chip-RAM und Fast-Mem. Dieses Thema wurde ja schon in anderen Kapiteln dieses Buches angeschnitten; unter anderem bei der Beschreibung des Programms *NoFastMem* in Kapitel 5. Der Chip-RAM ist der Bereich des RAM-Speichers, der von den drei Spezial-Chips benutzt werden kann. Früher waren das immer nur die ersten 512 Kbyte des RAM-Speichers. In einigen Ausführungen des Amiga 500 sind aber schon neue Chips zu finden, die die ersten 1024 Kbyte benutzen können. Diese Chips tragen den Namen ECS (engl. *enhanced chip set* = verbesserter Chip-Satz).

Bestimmte Daten, zum Beispiel Bitmaps, Sprite-Beschreibungen und Sound-Daten müssen unbedingt im Chip-RAM-Bereich liegen, damit die entsprechenden Spezial-Chips darauf zugreifen können. Programme, die solche Daten erzeugen, müssen deshalb aufpassen, daß sie vom Betriebssystem ausdrücklich Chip-RAM anfordern, wenn Sie weitere Bereiche des RAM-Speichers für ihre Daten reservieren wollen. Tun diese Programme das nicht, erhalten Sie eventuell Fast-RAM. Dann erscheint unter Umständen kein Bild am Bildschirm oder es erklingen nicht die gewünschten Töne. Vielleicht stürzt das Programm aber auch einfach ab.

Für andere Zwecke kann es günstiger sein, Fast-Mem anzufordern. Fast-Mem ist der RAM-Speicher außerhalb des Chip-RAM-Bereichs. Da die Spezial-Chips auf diesen Speicherbereich nicht zugreifen können, braucht die CPU auch nicht verlangsamt werden, wenn ein anderer Chip, zum Beispiel der Blitter, den Speicher für sich allein haben möchte. Der Blitter arbeitet mit dem Chip-RAM, die CPU mit dem Fast-Mem-Bereich des RAM-Speichers und beide können mit voller Geschwindigkeit laufen. Daher hat der Fast-Mem-Bereich seinen Namen (engl. *fast memory* = schneller Speicher).

Beim Amiga 500 mit der Grundausstattung von 512 Kbyte RAM spielt die Unterscheidung zwischen Chip-RAM und Fast-Mem keine Rolle. Hier ist nur Chip-RAM vorhanden und kein Programm hat deshalb eine echte Wahl, welcher Speichertyp für welchen Zweck verwendet wird. Dasselbe gilt auch für Amiga-Modelle mit 1 Mbyte RAM und den neuen Versionen der Spezial-Chips (ECS), die 1 Mbyte Chip-RAM kontrollieren können. Bei Amiga-Modellen mit den alten Chips und mehr als 512 Kbyte RAM sowie bei Amiga-Modellen mit dem ECS und mehr als einem Mbyte RAM, können Programme Fehler bei ihren Speicheranforderungen machen.

Einige alte Programme aus der Zeit, als es nur den Amiga 1000 gab, machen solche Fehler. Und selbst neue Programme fordern nicht immer Fast-Mem an, wenn es möglich wäre und arbeiten deshalb nicht so schnell wie möglich. Um diese Programmfehler zu umgehen und die Arbeitsgeschwindigkeit neuer Programme zu optimieren, gibt es die Programme beziehungsweise Shell-Befehle *NoFastMem* und *FastMemFirst*.

*NoFastMem* belegt den gesamten RAM-Speicher außerhalb des Chip-RAM-Bereichs. Programme können deshalb nicht fälschlicherweise Fast-Mem für Zwecke verwenden, in

denen eigentlich Chip-RAM eingesetzt werden sollte (es gibt ja keinen freien Fast-Mem-Speicher). Das Programm *FastMemFirst* (das bei jedem Neustart automatisch ausgeführt wird), sorgt hingegen dafür, daß ein Programm, das einen Speicherbereich von der Systemsoftware anfordert, immer Fast-Mem erhält, wenn es nicht ausdrücklich Chip-RAM verlangt. Manche Programme laufen dann deutlich schneller, da sie sich die Benutzung des RAMs nicht mehr mit den Spezial-Chips teilen müssen.

### 11.5.7 Begriffsver- und -entwirrungen

Vielleicht ging es Ihnen in diesem Kapitel wie mir, als ich das erste Mal etwas über die Spezial-Chips des Amiga las und Sie leiden nun unter extremer Begriffsverwirrung. Ich hoffe, diese aber zum Abschluß dieses Kapitels noch etwas »entwirren« zu können. Vor allem möchte ich zwei Begriffe klären, die immer wieder verwendet werden, »Chip« und »Prozessor«.

Vielen dürfte im Moment nicht ganz klar sein, was denn nun ein Chip ist und wo die Koprozessoren des Amiga liegen. In der Berichterstattung über den Amiga in der Computer-Presse geht es meist auch wild durcheinander. Da ist von drei oder vier Custom-Chips die Rede: von Denise, Agnus (oder Agnes), von Paula, dem Blitter und dem Copper. Manchmal wurden der Blitter und der Copper als Chips bezeichnet und Denise und Agnus als Koprozessoren, und manchmal war es umgekehrt. In Wirklichkeit ist aber (hoffentlich) alles ganz einfach.

Paula, Denise und Agnus sind drei Chips im Innern des Amiga. Sie arbeiten aber so eng zusammen, daß sie eigentlich ein großes elektronisches Bauteil bilden und man sie deshalb am besten auf einem Chip zusammenfassen müßte. Im Moment ist die Chip-Technologie aber noch nicht so weit, daß man einigermaßen preiswert einen so großen Chip bauen könnte. Aus rein praktischen Erwägungen sind die für den Amiga benötigten grafischen und akustischen Funktionen deshalb auf drei getrennte Bausteine (Chips) verteilt worden.

Innerhalb der großen Funktionseinheit des Superchips, den Denise, Paula und Agnus zusammen bilden, existieren einzelne Untereinheiten, die sich um ganz spezielle Funktionen kümmern. Die Ansteuerung von externen Geräten, die Erzeugung eines Videobildes und die Manipulation einzelner Bits im Speicher sind zum Beispiel solche Funktionen, die von jeweils einer Untereinheit innerhalb des Superchips erledigt werden. Diese Untereinheiten nennen wir Prozessoren. Ein Prozessor ist eine logische Funktionseinheit, während ein Chip eine physikalische, greifbare Einheit ist (eben ein kleiner, schwarzer Klotz mit vielen Metallbeinchen).

Ein Prozessor kann komplett innerhalb eines der drei Chips liegen, kann sich aber auch aus Komponenten zusammensetzen, die auf mehrere Chips verteilt sind. Den Speicherzugriff für alle Aufgaben der verschiedenen Prozessoren erledigt zum Beispiel immer Agnus. Keine der anderen Funktionseinheiten in den anderen beiden Chips ist deshalb komplett ohne den Adreßgenerator in Agnus. Deshalb kann man eigentlich von keinem der Prozessoren, die oben besprochen wurden, sagen, daß er in einem Chip liegt.

Nur der Einfachheit halber spricht man oft davon, daß ein Prozessor in dem Chip liegt, der die »wesentlichen« Teile des Prozessors enthält. Denise wird deshalb nicht nur als Video-Chip, sondern manchmal auch als Video-Prozessor bezeichnet.

Andersherum können aber theoretisch durchaus mehrere Prozessoren innerhalb eines Chips liegen. Dies ist zum Beispiel beim Blitter und beim Copper der Fall, Prozessoren die sich im wesentlichen komplett in Agnus befinden. Wie Sie sehen, haben Chips und Prozessoren zwar viel miteinander zu tun, sind aber nicht identisch. Es sind Begriffe, die sich auf sehr unterschiedliche Eigenschaften beziehen. (Auch ein Mikroprozessor muß nicht auf einem Mikrochip liegen und ein Chip, den man als Mikroprozessor bezeichnet, kann mehrere verschiedene Prozessoren enthalten.)

Damit ist die Aufzählung der pingeligen Hardware-Details dann aber auch endgültig abgeschlossen. Ich hoffe, dem Leser in diesem Kapitel zumindest einen kleinen Eindruck von der Komplexität und Vielseitigkeit der Amiga-Hardware vermittelt zu haben, ohne dabei allzuviel Vorwissen vorausgesetzt zu haben.

# 12 Die Programmierung des Amiga 500

In den vorangegangenen Kapiteln haben Sie ja schon einen ungefähren Eindruck davon bekommen, welche Möglichkeiten die Hardware des Amiga bietet. Falls der eine oder andere unter den Lesern dadurch Appetit bekommen hat, auch einmal selbst ein wenig mit diesen Möglichkeiten »herumzuspielen«, kann ihm vielleicht dieses und das nächste Kapitel helfen. Ab jetzt geht es nämlich um die Programmierung des Amiga 500. Hierzu werden Sie in diesem Kapitel zunächst einige allgemeine Erläuterungen zum Thema Programmierung finden. Im nächsten Kapitel wird dann stichwortartig beschrieben, was die Programmiersprache zu bieten hat, die jedem Amiga beiliegt: Amiga-Basic.

Amiga-Programmierung spielt sich aber nie »im leeren Raum« ab. Alle Amiga-Programme nutzen intensiv die Systemsoftware, die sich auf der Workbench-Diskette befindet. Die »Kunst der Amiga-Programmierung« besteht vor allem auch darin, diese »Bibliotheken« von Funktionen, die einem viel Arbeit abnehmen können, richtig zu nutzen.

## 12.1 Architektur der Amiga-Software

Im Zusammenhang mit der Software eines Computers von »Architektur« zu reden, mag vielleicht etwas hochtrabend klingen. In Wirklichkeit ist die Entstehung eines so komplexen Gebildes, wie es die Systemsoftware des Amiga darstellt, ein mindestens ebenso schwieriger und komplizierter Vorgang wie der Bau eines Hauses. Systemsoftware muß deshalb sehr sorgfältig geplant werden und verlangt ein reibungsloses Zusammenspielen vieler Beteiligter. Auch die Teile des entstehenden Gebäudes müssen nahtlos zusammenpassen und sich gegenseitig in genau berechneter Weise stützen.

Eine der größten Stärken des Amiga liegt in diesem reibungslosen Zusammenspiel der einzelnen Komponenten der Systemsoftware mit anderen Software-Komponenten und mit den Hardware-Komponenten. So arbeiteten zum Beispiel von Anfang an die beiden Teams, die die Hardware und die Software entwickelten, sehr eng zusammen und beeinflußten sich immer wieder gegenseitig. Dies beschleunigte nicht gerade die Entwicklung, garantiert jetzt aber ein recht effizientes Softwaregebäude mit ideal aufeinander abgestimmten Bestandteilen.

Bei der Programmierung des Amiga muß diese Systemsoftware möglichst optimal genutzt werden. Diese Systemsoftware stellt deshalb (zum Teile) eine »Software-Bibliothek« dar, in der Programme sich die Funktionen aussuchen können, die sie benötigen – und somit nicht noch einmal selbst realisieren brauchen. Sie als Programmierer brauchen also nicht »das Rad ein zweites Mal erfinden«.

### 12.1.1 Grundriß der Amiga-Systemsoftware

Damit Sie vielleicht den folgenden Erläuterungen ein wenig besser folgen können, möchte ich Ihnen zunächst einmal den Grundriß der Amiga-Systemsoftware vorstellen. Sie haben dann schon einmal einen ersten Überblick über dieses Gebäude. Die folgende Abbildung zeigt in schematischer Weise, wie die verschiedenen Komponenten dieser Systemsoftware aufeinander aufbauen.

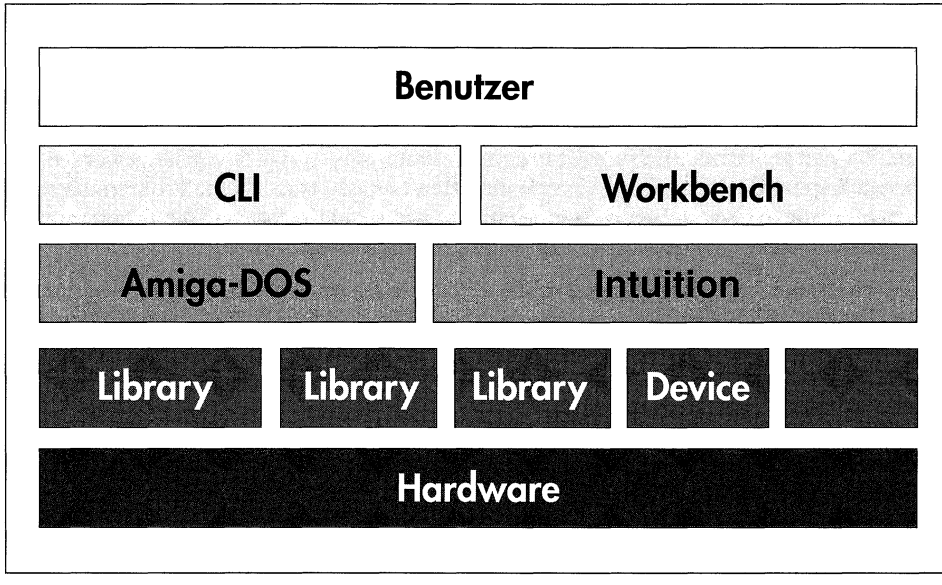



***Bild 12.1:*** *Grundriß der Amiga-Systemsoftware*

»Ganz unten« in diesem Bild liegt die Hardware. Sie bildet praktisch das Fundament, auf dem die Software aufgebaut ist. Anders als bei simpleren Computern ist es bei Programmen für den Amiga unüblich, daß Anwendungsprogramme direkt auf diese Hardware zurückgreifen. Das ist zwar möglich, denn die Hardware-Adressen und die Bedeutung der Register der Customchips sind gut dokumentiert. Es ist aber viel komplizierter als zum Beispiel auf einem Commodore 64 und auch gefährlicher. Der Amiga ist nämlich eine Multitasking-Maschine und wenn ein Programm nach seinem Willen die Custom-Chips manipuliert, versucht ein anderes vielleicht gleichzeitig etwas ganz anderes damit anzustellen, ohne vom ersten Programm etwas zu wissen. Das bedeutet Chaos (und abstürzende Programme). Lediglich einige Spielprogramme »greifen sich« gnadenlos den kompletten Amiga und

manipulieren direkt die Hardware. Solche Programme vertragen sich überhaupt nicht mit anderen Programmen. Um sie zu nutzen, muß man den Amiga mit einer speziellen Diskett neu starten und nach Beendigung des Spiels noch einmal einen Neustart mit einer »normalen« Workbenchdiskette durchführen, wenn man wieder andere Programme benutzen möchte.

Verwendet ein Programm jedoch die Amiga-Systemsoftware, um die Amiga-Hardware zu manipulieren, kann nichts passieren. Die Systemsoftware kontrolliert den Zugriff auf die Hardware, spielt den Schiedsrichter zwischen den konkurrierenden Programmen und paßt auf, daß ein Programm nicht einem anderen in die Quere kommt. Das kann aber nur funktionieren, wenn diese Programme die Systemsoftware nicht umgehen und hinter ihrem Rücken Unsinn anstellen – wie zum Beispiel direkt auf die Hardware zurückzugreifen.

Die Systemsoftware des Amiga besitzt einen schalenförmigen Aufbau, in dessen Kern die Hardware liegt. Direkt über der Hardware liegen *Exec* (siehe unten) und eine Reihe von Bibliotheken und sogenannten »Geräte-Treibern«. Diese Schicht der Software hat direkten Zugriff auf die Hardware des Amiga. Die nächsthöhere Schicht, die aus *Intuition* und Amiga-DOS besteht, benutzt die darunterliegende Schicht (*Exec* und die Bibliotheken), um die Hardware zu manipulieren.

Darüber kommen dann die Programme, mit denen Sie als Anwender direkt zu tun haben. Dazu gehören sowohl das CLI, die Shell und die Workbench, wie natürlich auch alle anderen Anwendungsprogramme (Malprogramme, Textverarbeitungsprogramme und so weiter). Diese Programme benutzen vor allem Intuition, um mit Ihnen, dem Anwender, zu kommunizieren und Amiga-DOS, um mit der Hardware des Amiga umzugehen. Sie greifen zum Teil aber auch noch auf Exec und die Bibliotheken zurück. Über der Schicht von Workbench und Shell kommt dann keine weitere Software mehr, sondern direkt der menschliche Benutzer des Amiga (also Sie selbst).

### 12.1.2 Exec

Die wichtigsten Komponenten der Hardware haben Sie ja in den vergangenen Kapiteln schon kennengelernt. Der wichtigste Bestandteil der Amiga-Systemsoftware ist *Exec*. *Exec* ist ein Paket von kleinen, sehr effizienten Routinen im ROM-Speicher des Amiga. Diese Routinen werden von allen anderen Komponenten der Software andauernd benötigt. Sie werden für eine Vielzahl von sehr elementaren Aufgaben bei der Verwaltung der verschiedenen »System-Ressourcen« des Amiga benötigt.

#### Listen

*Exec* bietet unter anderem Hilfsroutinen für die Verwaltung von Listen an. Solche Listen werden von fast jedem Programm und auch von anderen Teilen der Systemsoftware andauernd benötigt. In ihnen können beliebige Objekte, zum Beispiel Namen, Dateien, Programme, Fenster und so weiter gesammelt und verwaltet werden.

Da die Listenverwaltung sich innerhalb von *Exec* befindet, ist sie sehr schnell (in Maschinencode geschrieben) und steht jedem zur Verfügung, der sie benötigt. (*Exec* befindet sich ja immer im Speicher.) Andere Programme brauchen deshalb keine eigenen Pakete zur Listenverwaltung zu beinhalten und können dadurch kompakter werden.

## Tasks

Eine besonders wichtige Sorte der »Objekte«, die *Exec* in Listen verwaltet, sind die Tasks, die verschiedenen Programme beziehungsweise Prozesse im Amiga, die scheinbar alle gleichzeitig ablaufen. *Exec* erzeugt und verwaltet solche Tasks, teilt ihnen Speicher und Rechenzeit auf der CPU zu und zerstört sie auch wieder, wenn sie ihre Aufgabe erfüllt haben.

Die Verteilung der CPU-Zeit (und anderer Ressourcen) auf die verschiedenen darum konkurrierenden Tasks ist eine der Hauptaufgaben von *Exec*. Auch hierbei zählt vor allem die Geschwindigkeit. Besonders der Wechsel von einem Task zu einem anderen muß sehr schnell vor sich gehen, da solche Wechsel jede Sekunde zigmal vorkommen können.

## Nachrichten

Ebenfalls zur Task-Verwaltung gehört im weiteren Sinn auch der Nachrichten-Mechanismus von *Exec*. Er dient zu einer reibungslosen Kommunikation zwischen unterschiedlichen Tasks. Oft ist es nämlich nötig, daß verschiedene Tasks miteinander kommunizieren, um zum Beispiel Daten auszutauschen oder um sich mitzuteilen, wann wer mit welcher Aufgabe fertig ist. Dies ist nicht ganz so einfach, da kein Task vom anderen weiß, in welchem Zustand sich dieser gerade befindet. Ein Kommunikationsversuch zum falschen Zeitpunkt kann aber dazu führen, daß Daten verlorengehen oder den empfangenden Task bei einer wichtigen Aufgabe stören.

Der Nachrichten-Mechanismus von *Exec* umgeht diese Schwierigkeiten und sorgt dafür, daß Tasks nur dann Meldungen empfangen, wenn sie dazu bereit sind. *Exec* weiß natürlich als einziger immer ganz genau, ob ein Task gerade bereit ist, eine Nachricht zu empfangen oder nicht, da es die Tasks ja kontrolliert. Kann ein Task eine Nachricht nicht sofort empfangen, legt *Exec* sie erst einmal in eine Warteschlange (eine Liste; siehe oben), aus der sie der Task entnehmen kann, sobald er Zeit und Lust dazu hat. Keine Nachricht geht bei diesem Mechanismus verloren.

## Speicherverwaltung

*Exec* kümmert sich zusätzlich auch noch um die Speicherverwaltung im Amiga. Es ist naheliegend, einem übergeordneten Paket wie *Exec* diese Aufgabe zu überlassen. Der Speicher ist schließlich eine sehr »globale« Ressource des Amiga, die von allen Programmen (Tasks) benötigt wird. Würden die verschiedenen Tasks gleichzeitig und unkontrolliert im Speicher herumfuhrwerken, wäre ein Chaos unvermeidlich.

Wann immer ein Programm mehr Speicherplatz benötigt (zum Beispiel, weil ein Benutzer eine neue Zeile in einem Textverarbeitungsprogramm eingegeben hat), kann es zusätzlichen Speicherplatz bei *Exec* anfordern. *Exec* sucht dann nach einem Stück im Speicher, das die passende Größe hat und noch frei ist, reicht dieses dem anfragenden Task und markiert es dann als belegt. Währenddessen werden alle anderen Programme angehalten, damit sie nicht zur selben Zeit denselben Speicher anfordern können. *Exec* kann auf Wunsch die Suche nach einem freien Speicherblock auch auf eine bestimmte Sorte von Speicher, Fast-Mem oder Chip-RAM, begrenzen. Das ist sehr wichtig für bestimmte Arten von Daten, die unbedingt im Chip-RAM liegen müssen (siehe auch Kapitel 11).

»Brave« Programme geben einen einmal reservierten Speicherblock natürlich auch sofort, wenn sie ihn nicht mehr benötigen, an *Exec* zurück. Dieser freigewordene Speicher kann dann anderen Programmen zugeteilt werden, die ihn vielleicht dringend brauchen.

### 12.1.3 Bibliotheken

Die wichtigste Aufgabe von *Exec* ist neben den Tasks aber wahrscheinlich die Verwaltung von Programmbibliotheken. Diese können in ähnlicher Weise von vielen Programmen gleichzeitig genutzt werden wie *Exec* selbst. Sie brauchen sich aber nicht in der Kickstart-Software zu befinden. Bibliotheken sind nichts anderes als besondere Dateien, die die Routinen einer Programmsammlung (Programm-Bibliothek) enthalten. Solche Dateien können jederzeit auf die Workbench-Diskette kopiert und danach von allen Programmen genutzt werden.

*Exec* sorgt dafür, daß Programme, die eine Bibliothek nutzen wollen, nicht zu wissen brauchen, wie diese Datei aussieht, wo sie liegt und ob sie bereits von einem anderen Programm benutzt wird oder neu in den Speicher geladen werden muß. Ein Programm braucht eine Bibliothek nur mit einem entsprechenden Exec-Aufruf und dem Namen der Bibliothek zu »öffnen« und kann danach die Routinen der Bibliothek benutzen, als wären es Bestandteile des Programms selbst.

### 12.1.4 Geräte

Bei der Beschreibung der Shell haben Sie ja gesehen, daß die verschiedenen Ein- und Ausgabegeräte, die man am Amiga anschließen kann, behandelt werden können wie Dateien. Für diese Einheitlichkeit der Behandlung auch unterschiedlichster Geräte sorgen die Geräte-Bibliotheken (*devices*).

Für jedes angeschlossene Gerät richtet *Exec* nämlich zusammen mit dem sogenannten Treiber des Geräts einen separaten Task ein. Um Daten zu einem Gerät zu schicken oder welche von diesem zu holen, schickt *Exec* dem entsprechenden Task eine Nachricht, deren Format unabhängig von dem Gerät selbst ist. Diese Nachricht enthält nur die Art der Anforderung an das Gerät (also zum Beispiel, ob Lesen oder Schreiben eines Datenblocks gewünscht wird). Welche Disketten daraufhin gedreht oder welche Schalter umgelegt werden müssen, um diese Anforderung zu erfüllen, ist *Exec* egal. Der Geräte-Task ist alleine zuständig für die Beantwortung der Nachricht und somit für die exakte Ansteuerung der Hardware zu diesem Zweck.

Wenn ein neues Gerät an den Amiga angeschlossen wird, benötigt *Exec* nur einen passenden Treiber dazu. Dieser besteht einfach aus einer Datei, die der Geräte-Hersteller üblicherweise mitliefert und die in die Expansion-Schublade auf der Workbench-Diskette gelegt werden muß. Und schon können auch Programme Daten auf einem Gerät ausgeben, bei deren Entstehung es dieses Gerät vielleicht noch gar nicht gab. (In dieser Hinsicht sind Treiber durchaus ähnlich wie Bibliotheken zu verwenden.)

### 12.1.5 Die Grafik-Bibliothek

Eine dieser von *Exec* verwalteten Bibliotheken, die von fast allen Programmen genutzt wird, ist die Grafik-Bibliothek (*Graphics Support Library*). Mit ihr hat direkt oder indirekt jedes Programm etwas zu tun, das irgendwelche Ausgaben auf dem Bildschirm zeigen will – egal ob es sich um Texte oder Grafiken handelt.

Die Grafik-Bibliothek dient vor allem dazu, die Komplexität der Amiga-Grafik-Hardware vor dem Programm zu verbergen. Denn, obwohl die Fähigkeiten der Grafik-Hardware beachtlich sind, verlangen sie doch auch die Erledigung einiger Vorarbeiten und Haushaltungsaufgaben, bevor auch nur ein Punkt oder kurzer Strich auf dem Bildschirm erscheint.

Durch die Routinen der Grafik-Bibliothek sieht der Bildschirm auch für ein Programm so simpel aus wie eine Malfläche, über die ein Pinsel oder Stift bewegt wird. Nur die Größe dieser Fläche und die Anzahl der Farben in der Palette müssen vor der ersten Zeichenoperation festgelegt werden. Um danach zum Beispiel einen Strich zu zeichnen, wählt man auch als Programmierer zunächst eine Farbe aus. Der Zeichenstift kann dann durch Aufruf der entsprechenden Routinen zu jedem beliebigen Punkt der Fläche bewegt werden. Bei jeder solchen Bewegung des Stiftes kann man wählen, ob der Stift über der Zeichenfläche »schwebt« und somit nur bewegt wird, oder ob er »abgesenkt« wird und Spuren – also eine Linie – hinterläßt.

Andere Grafik-Routinen sind fähig, Rechtecke zu zeichnen und beliebige Flächen, die durch Punkte einer anderen Farbe begrenzt sein müssen, mit einer beliebigen neuen Farbe zu füllen. Letzteres funktioniert wie der Farbeimer in DeluxePaint (und ist auch genauso empfindlich gegenüber »Lecks« in der Umrandung).

Und schließlich ist die Grafik-Bibliothek auch noch für das »Malen« von Text auf dem Bildschirm zuständig. Dies ist bei bitmapped Grafik ja nicht so einfach wie bei Computern, die einen reinen Text-Modus besitzen. (Siehe auch Kapitel 11.) Erschwerend kommt noch hinzu, daß der Amiga viele verschiedene Schriftarten, Schriftstile und -größen kennt. Bei Verwendung der entsprechenden Routinen aus der Grafik-Bibliothek ist ein Text aber so einfach wie ein Strich in der Zeichenebene zu plazieren:

Zunächst wird die gewünschte Schriftart mit ihrem Namen ausgewählt und die Größe der Buchstaben sowie der Textstil (fett, unterstrichen, kursiv oder eine Kombination davon) festgelegt. Dann wird der Stift zu der Position bewegt, wo der Text beginnen soll und eine Routine der Grafik-Bibliothek aufgerufen, der der Text übergeben wird. Diese zeichnet ihn dann selbständig auf den Bildschirm, wobei für die Buchstaben die aktuelle Farbe des Zeichenstiftes verwendet wird.

### 12.1.6 Intuition

Auch *Intuition* ist eine Bibliothek – eine sehr wichtige allerdings. Was Intuition so alles leistet, haben Sie – ohne zu wissen, daß Intuition dafür verantwortlich ist – schon in den Kapiteln 2 bis 5 erfahren. *Intuition* ist der Teil der Systemsoftware, mit dem ein Programm wahrscheinlich am meisten kommuniziert. *Exec* und die Grafik-Bibliothek werden zwar andauernd benötigt, ein Programm merkt davon aber im allgemeinen recht wenig und hat auch nicht direkt mit *Exec* zu tun. Vieles hingegen »läuft« über Intuition-Routinen.

*Intuition* enthält zum Beispiel Routinen zum Öffnen und Schließen von Screens, Fenstern und Requestern. Auch alle Gadgets und Menüs werden von *Intuition* gezeichnet und bearbeitet. Die Routinen in *Intuition* achten auch darauf, daß alle Ausgaben eines Programms in dem ihm zugeteilten Fenster oder Screen bleiben. Eine lange Linie, die von einem Programm gezeichnet wird, stoppt *Intuition* zum Beispiel automatisch am Fensterrand. Durch *Intuition* wird so die (pseudo-) parallele Arbeit mit mehreren Programmen in

mehreren Fenstern erst möglich. *Intuition* sorgt dafür, daß sich die Ein- und Ausgaben gleichzeitig laufender Programme in säuberlich begrenzten Gebieten auf dem Bildschirm abspielen und sich deshalb nicht gegenseitig stören.

*Intuition* nimmt einem Programm zusätzlich auch noch sehr viel Arbeit ab bei der Kommunikation mit dem Anwender des Programms. Alle Aktionen des Anwenders – vor allem Mausklicks und Tastendrücke – laufen nämlich zunächst einmal durch *Intuition*, wo sie einer Vorverarbeitung unterzogen werden. Mausklicks in ein Gadget, die Titelleiste eines Screens oder Fensters und so weiter, bearbeitet Intuition zunächst einmal selbst. Vom Bewegen und Verkleinern eines Fensters, vom Herunterklappen eines Menüs merkt ein Programm recht wenig. Und dies ist auch gut so, weil die damit verbundenen Operationen recht komplex sind (obwohl alles so mühelos aussieht) und das Programm nur von der Erledigung seiner eigentlichen Aufgaben abhalten würden.

Erst, wenn eine Reaktion des Programms nötig wird – zum Beispiel wenn der Anwender einen Menübefehl ausgewählt oder eine Taste gedrückt hat – wird die entsprechende Aktion in Form einer Exec-Nachricht »verpackt« und an den Programm-Task des aktiven Fensters geschickt. Diese Nachricht enthält zum Beispiel die Nummer des ausgewählten Menü-Befehls oder den Buchstaben, der einer soeben gedrückten Taste entspricht. Was ein Programm mit dieser Nachricht macht, ist dann seine Sache. Es kann sie ignorieren oder auf bestimmte Art und Weise darauf reagieren.

### 12.1.7 Amiga-DOS

Ein weiterer wichtiger Bestandteil der Amiga-Systemsoftware ist Amiga-DOS. Die Bezeichnung »Amiga-DOS« ist eine Abkürzung für *Amiga **D**isk **O**perating **S**ystem*. Das sagt auch schon aus, wofür Amiga-DOS vor allem zuständig ist. Zusammen mit den oben beschriebenen Gerätetreibern (devices) sorgt Amiga-DOS dafür, daß sich die Programme beim Zugriff auf die Amiga-Hardware (zum Beispiel die Diskettenlaufwerke) nicht um Details kümmern brauchen. Für Programme sehen Dateien deshalb aus wie ein kontinuierlicher Strom von Daten. Nur Amiga-DOS braucht zu wissen, wie diese Daten auf der Diskette verteilt sind (sie können wild durcheinanderliegen).

Amiga-DOS kann neue Dateien und Directories auf Disketten und Festplatten anlegen, sie wiederfinden, wenn man den (Pfad-) Namen kennt, sie wieder löschen und umbenennen. Und schließlich sorgt Amiga-DOS – wieder zusammen mit den Gerätetreibern – dafür, daß fast jedes Ein-/Ausgabe-Gerät wie eine Datei behandelt werden kann.

## 12.2 Grundlagen der Programmierung

Zur Programmierung des Amiga benötigt man, wie für jeden anderen Computer auch, eine Programmiersprache. In Wirklichkeit versteht der Amiga nur eine (Programmier-) Sprache, den sogenannten Maschinencode. Dies ist eine Folge von Bits, die zu »Wörtern« von jeweils 16 Bit zusammengefaßt werden. Da ein Mensch normalerweise nicht in Bit-Folgen denkt und redet, spielt sich die Kommunikation mit dem Computer zwangsläufig etwas schleppend ab, wenn nicht zusätzliche Hilfsmittel eingesetzt werden.

### 12.2.1 Programmiersprachen

Ein solches Hilfsmittel stellen die sogenannten »höheren Programmiersprachen« dar – höher deshalb, weil man sich nicht mehr auf das »niedere« Niveau der Maschine hinabbegeben muß. Die wesentliche Idee bei allen höheren Programmiersprachen ist es, ein Programm zunächst einmal in einer einigermaßen lesbaren Sprache zu formulieren, die aus »richtigen« Wörtern besteht, welche aus »richtigen« Buchstaben gebildet werden. Wie nahe die so entstehende Sprache der menschlichen Sprache kommt, ist verschieden. Es gibt sehr primitive Programmiersprachen, die wohl nur von Fachleuten verstanden werden und sehr »hohe« Programmiersprachen, in denen sich Programme fast wie Gebrauchsanweisungen aus dem vielzitierten »richtigen Leben« lesen. Falls Sie bisher noch keine Programmiersprache kennen, lassen Sie sich also nicht vom Getue der Programmierer abschrecken, eine zu erlernen. Alle Programmiersprachen sind viel simpler und leichter zu erlernen als auch die einfachste menschliche Sprache.

Was Programmiersprachen bei all ihren Unterschieden gemeinsam haben, ist, daß immer ein Verfahren definiert ist, wie man von einer Anweisung in der höheren Programmiersprache zur äquivalenten (gleichbedeutenden) Bitfolge im Maschinencode kommt. Äquivalent heißt in diesem Fall, daß die Bitfolge das tut, was der Programmierer gemeint hat, als er das Programm in der höheren Sprache aufgeschrieben hat. (Eines der Hauptprobleme bei der Programmierung ist, daß sich Programmierer und Computer meist nicht so ganz darüber einig werden können, was der Programmierer denn nun eigentlich »gemeint« hat!)

Das Verfahren, das ein Programm in Maschinencode umwandelt, ist relativ (!) simpel und gradlinig. Es ist zumindest so simpel, daß man es wiederum einem Computer-Programm überlassen kann. Es gibt zwei große Gruppen solcher Programme, die zwei grundsätzliche Strategien repräsentieren, mit denen man diesen Übersetzungsprozeß angehen kann: Interpreter und Compiler.

Eine Ursache für viel Verwirrung bei Programmieranfängern ist oft, daß diese Programme, die andere Programme in Maschinencode umsetzen, ebenfalls Programmiersprache genannt werden. Dies ist eigentlich falsch! Genau wie ein Dolmetscher für Französisch nicht die Sprache Französisch ist, ist auch ein Interpreter für Basic nicht die Sprache Basic. Behalten Sie das immer im Auge, wenn Ihnen jemand das nächste Mal erzählen will, diese oder jene Computer-Sprache sei gut oder schlecht. Meist will er damit in Wirklichkeit sagen, daß der Compiler oder Interpreter gut oder schlecht ist.

### 12.2.2 Interpreter

Ein Interpreter ist die einfachste Möglichkeit, eine Umsetzung von höherer Programmiersprache zum Maschinencode zu realisieren. Er enthält üblicherweise für jede Anweisung (jedes Wort) der höheren Programmiersprache ein kleines Stück Maschinencode, das dieser Anweisung entspricht (ihr Äquivalent ist). Um ein Programm auszuführen, liest ein Interpreter es Zeile für Zeile, sucht die den Anweisungen entsprechenden Maschinencode-Stückchen heraus und übergibt diese dem Computer zur Ausführung. Diesen Vorgang nennt man dann sinnigerweise auch »Interpretieren« eines Programms.

Wenn man einen Vergleich mit menschlichen Sprachen ziehen würde, könnte man sagen, daß ein Interpreter einem Dolmetscher entspricht, der ein fremdsprachiges Buch vorliest und dabei simultan übersetzt.

### 12.2.3 Compiler

Ein Compiler hingegen nimmt den Programmtext (in der höheren Programmiersprache) und erzeugt daraus ein äquivalentes Programm in Maschinencode. Der gesamte Übersetzungs-Vorgang heißt »Kompilieren« oder auch einfach nur »Übersetzen«. Dabei werden raffinierte Techniken angewandt, um den Maschinencode möglichst klein und schnell zu machen. Im allgemeinen produzieren diese Techniken nicht aus einer Programmanweisung immer ein und dasselbe Stückchen Maschinencode, sondern verwenden Wissen über die vorangehenden und nachfolgenden Anweisungen, um den erzeugten Code zu optimieren.

Wollte man wieder den Vergleich mit dem fremdsprachigen Buch bemühen, könnte man sagen, daß ein Compiler einem Dolmetscher entspricht, der eine schriftliche Übersetzung des Buches anfertigt, die man danach selbst lesen kann, ohne dazu den Dolmetscher noch zu benötigen.

### 12.2.4 Vergleich von Compiler und Interpreter

Die Vorteile des Compiler-Prinzips liegen auf der Hand. Der erzeugte Maschinencode ist selbständig und benötigt kein zusätzliches Programm (den Interpreter) zur Ausführung. Zudem sind kompilierte Programme üblicherweise fast immer schneller, da kein Interpreter mühselig Zeile für Zeile den Programmtext durchlaufen, analysieren und interpretieren muß.

Compiler haben aber auch gravierende Nachteile, weshalb sie sich bisher trotz ihrer Vorteile noch nicht generell für alle Sprachen durchgesetzt haben. Der Vorgang des Kompilierens eines Programms dauert zum Beispiel oft sehr lange. Das dabei entstehende Programm ist zwar schnell, dafür muß man zunächst aber erst einmal eine Wartezeit in Kauf nehmen. Dies ist besonders während der Programmentwicklung sehr lästig. Nach jeder kleinen Änderung muß das Programm neu übersetzt werden und man muß oft mehrere Minuten warten, bis die Änderungen am laufenden Programm ausgetestet werden können. Bei Verwendung eines Interpreters ist ein Programm auch nach einer Änderung sofort wieder lauffähig.

Compiler sind zudem meist schwieriger zu bedienen als Interpreter. Während es bei Interpretern in der Regel ausreicht, einen Start-Befehl (zum Beispiel RUN) zu geben, um ein Programm mit der Arbeit beginnen zu lassen, erfordern Compiler mehrere Arbeitsschritte bis zu diesem Punkt (bei denen man natürlich auch jedesmal Fehler machen kann). Interpreter sind deshalb auch heute noch unschlagbar, wenn es darum geht, eine Programmiersprache zu erlernen oder ein Programm wirklich interaktiv zu entwickeln und auszutesten.

## 12.3 Programmiersprachen für den Amiga

Für den Amiga 1000 (das Vorgängermodell des Amiga 500) gab es schon kurze Zeit nach der Markteinführung eine Vielzahl von Compilern und Interpretern – deren Zahl ist im Laufe der Zeit natürlich noch angewachsen. Für den Amiga sind deshalb inzwischen alle weitverbreiteten Computersprachen zu haben. Kein Programmierer wird also durch fehlende Werkzeuge davon abgehalten, sich mit den phantastischen Möglichkeiten des Amiga näher zu beschäftigen.

### 12.3.1 Assembler und Maschinencode

Die Programmiersprache, die dem Maschinencode noch am nächsten kommt, ist die Assembler-Sprache. Bei einem Assembler entspricht jede Zeile des Programms ziemlich genau einer Instruktion (einem Wort) im Maschinencode. Der Compiler (in diesem Fall ebenfalls »Assembler« genannt) hat es also mit dieser Sprache recht leicht.

Mit einem guten Assembler kann man wirklich alle Möglichkeiten nutzen, die ein Computer zu bieten hat, muß sich dabei aber auf ein recht niedriges Niveau der Programmierung hinabbegeben. Selbst einfache Dinge – wie das Ausgeben eines kurzen Textes auf dem Bildschirm – erfordern in Assembler-Sprache oft nicht-triviale, größere Programme.

Ein weiteres Problem der Assembler-Sprache ist, daß es sehr leicht ist, beliebig viele logische Fehler in ein Programm einzubauen; der Assembler merkt nichts davon. Höhere Programmiersprachen schützen in stärkerem Maße vor den eigenen Fehlern und nehmen einem damit die lästige Arbeit der Fehlersuche zumindest zum Teil ab.

Nur wenige Programme werden heute noch in Assembler-Sprache entwickelt, da es inzwischen andere Programmiersprachen gibt, mit denen die meisten Programmieraufgaben genauso gut erledigt werden können wie in Assembler-Sprache. Nur wenn Geschwindigkeit und Kompaktheit eine extrem große Rolle spielen (oder der Programmierer einfach Spaß am »Bit-Popeln« hat) greift man heute noch auf die Assembler-Sprache zurück.

Ein Assembler gehört zum Standard-Lieferumfang des Entwickler-Paketes, das Software-Entwickler bei Commodore erwerben können. Beim Amiga für »normale Sterbliche« liegt kein Assembler bei. Er kann aber jederzeit mit den dazugehörigen Unterlagen auch separat erworben werden. Bei diesem Assembler handelt es sich ansonsten um einen recht komfortablen Vertreter seiner Gattung (für Fachleute: es ist ein vollausgebauter Makro-Assembler). Auch andere Firmen bieten inzwischen eine ganze Reihe verschiedener Assembler (meist mit »Monitor« oder »Debugger«) für den Amiga an. Diese sind dem Assembler aus dem Commodore-Entwicklerpaket zum Teil weit überlegen – sowohl was den Arbeitskomfort als auch die Arbeitsgeschwindigkeit angeht.

### 12.3.2 Basic

Basic dürfte heute die am weitesten verbreitete Computersprache auf der ganzen Welt sein. Sie gehört standardmäßig zum Lieferumfang vieler Computer, so auch des Amiga, oder wird sogar gleich in die Computer eingebaut (in Form eines ROMs). Über Basic und speziell über Amiga-Basic erfahren Sie mehr im folgenden Kapitel, das sich der Sprache und dem zugehörigen Interpreter ausführlich widmet. Diese Sprache liegt dem Amiga beim Kauf gleich bei. Wem sie nicht gefällt, der bekommt inzwischen einige Alternativen dazu auf dem Markt angeboten, die verschiedene Vorteile (und Nachteile) gegenüber Amiga-Basic haben.

### 12.3.3 C

Eine Sprache, die sich in den letzten Jahren gerade auf Mikrocomputern wachsender Beliebtheit erfreut, ist auch C. Sie stellt einen – nach Meinung vieler Fachleute sehr gelungenen – Kompromiß zwischen Maschinennähe (à la Assembler) und Programmier-

komfort dar. C wird fast nur im Zusammenhang mit Compilern verwendet und ist dementsprechend auch nicht gerade eine Sprache für Anfänger.

Ein zusätzlicher Nachteil von C aus der Sicht eines Anfängers ist, daß es, ähnlich wie beim Assembler, wenig Schutz vor den eigenen Fehlern gibt. Man kann viele Fehler in ein Programm einbauen, die so «dumm« sind, daß sie ein derart aufwendiges Programm wie es ein guter C-Compiler darstellt, eigentlich finden könnte – nur: er sucht gar nicht danach!

Dafür erzeugen C-Compiler aber meist sehr »guten« Maschinencode, der recht nahe an die Geschwindigkeit und die Kompaktheit herankommt, die man erreichen könnte, wenn man dasselbe Programm mit einem Assembler erstellen würde – was ungleich schwieriger und langwieriger wäre. Zudem sind die C-Compiler vieler Hersteller so raffiniert konstruiert, daß man sie sehr leicht an einen neuen Computer anpassen kann. Dies gilt selbst dann, wenn dieser einen ganz anderen Maschinencode versteht. Es gibt deshalb für neue Computer meist sehr schnell C-Compiler, während Compiler und Interpreter für alle anderen Sprachen oft lange auf sich warten lassen.

Genau das war auch beim Amiga der Fall. Für den Amiga (mit zweitem Diskettenlaufwerk oder Festplatte) gibt es inzwischen mehrere C-Compiler, die alle sehr gut sind. Einen dieser Compiler gab es sogar schon, bevor der Amiga überhaupt auf dem Markt war.

### 12.3.4 Pascal und Modula

Ebenfalls für die Ausbildung – in diesem Fall aber die Ausbildung von Informatikern an der ETH Zürich – wurde die Programmiersprache Pascal konzipiert. Pascal ist nicht ganz so leicht zu erlernen wie zum Beispiel Basic, zwingt den Programmierer aber mehr oder weniger zu strukturierten und leserlichen Programmen. In Pascal sind zudem diverse Konzepte »eingebaut«, durch die viele Fehler, die die Compiler anderer Sprachen einfach »durchgehen lassen«, schon bei der Übersetzung des Programms erkannt werden können.

Jedem, der nicht nur »schnell mal eben« eine Programmiersprache, sondern das »Programmieren an sich«, in fundierter Weise lernen möchte, kann Pascal nur wärmstens empfohlen werden. Wer mit Pascal das Programmieren gelernt hat, hat meist auch nur wenig Schwierigkeiten, eine weitere Sprache zu lernen – was man vom umgekehrten Weg leider nicht behaupten kann.

Auch für den Amiga gibt es einen Pascal-Compiler (MCC-Pascal). Dieser entspricht hundertprozentig dem internationalen Sprach-Standard, wie ihn die ISO (International Standards Organisation) definiert hat und hat den zusätzlichen Vorteil, Programmstücke in Pascal, C und Assembler-Sprache innerhalb eines Programms kombinieren zu können. Er bietet aber leider nur sehr umständlichen Zugriff auf die speziellen Fähigkeiten des Amiga.

Pascal war aber nur der Anfang einer ganzen Sprachfamilie. Nach einer gewissen Zeit der Verwendung von Pascal stellt sich nämlich heraus, daß es bei allen Vorteilen dieser Sprache ihr doch an einigen wichtigen Fähigkeiten für bestimmte Gruppen von Programmen mangelte. So wird zum Beispiel die Entwicklung großer Programmprojekte durch ganze Gruppen von Programmierern kaum unterstützt und es gibt kaum Möglichkeiten, auf die Hardware des verwendeten Computers zurückzugreifen.

Die Entwickler von Pascal nahmen dies zum Anlaß, eine neue Sprache zu entwerfen, die zwar nicht als Ersatz, aber auf vielen Gebieten als Ergänzung zu Pascal gesehen werden

kann: Modula (beziehungsweise Modula-2). Sie beseitigt weitgehend alle als schwerwiegend empfundenen Mängel von Pascal und hat dabei aber genügend Ähnlichkeiten mit Pascal, um vom Programmierer nur minimale Umstellungen zu verlangen. Auch für den Amiga können interessierte Programmierer schon verschiedene Modula-2-Systeme erwerben. Auch hier – wie bei allen anderen Sprachen – steht man also vor der Qual der Wahl.

# 13 Amiga-Basic

Die Programmiersprache des Amiga heißt Amiga-Basic. Das zugehörige Werkzeug befindet sich auf der Extras-Diskette (sie trägt auf der Workbench den Namen *ExtrasD1.3*), die jedem Amiga beim Kauf beiliegt. Amiga-Basic gibt so jedem Besitzer des Amiga die Gelegenheit, eigene Programme zu entwickeln oder auch nur ein wenig mit den Möglichkeiten der Computerprogrammierung herumzuspielen. Sie werden Amiga-Basic in diesem Kapitel etwas näher kennenlernen.

Der Leser möge mir aber verzeihen, wenn er hier keine ausführliche Einführung in die Computer-Programmierung findet, ja noch nicht einmal eine Anleitung zur Programmierung des Amiga. Programmierung ist eine Sache, die man nicht in einem Buchkapitel abhandeln kann. Und Amiga-Basic ist eine so umfangreiche und vielseitige Sprache, daß selbst ein Buch vom Umfang des hier vorliegenden sich wahrscheinlich noch recht knapp fassen müßte, wenn es Amiga-Basic vollständig beschreiben wollte.

Die folgenden Abschnitte geben deshalb nur einen kurzen Überblick über die Möglichkeiten der Sprache Basic und des speziellen Dialekts Amiga-Basic. Sie werden so unter anderem erfahren, welche Möglichkeiten des Amiga Sie in eigenen Programmen nutzen können (fast alle!) und wie Amiga-Basic gegenüber anderen Dialekten der Sprache Basic auf anderen Computern einzuordnen ist.

Die letzten Abschnitte dieses Kapitels stellen schließlich einige Beispielprogramme vor, mit denen Sie versuchen können, mit Amiga-Basic »etwas warm zu werden«. Diese Programme leisten alle nichts Weltbewegendes, sondern dienen wirklich nur zu Demonstrationszwecken. Falls Sie wirklich eigene Programme für den Amiga entwickeln wollen, so erhalten Sie Informationen dazu zusammen mit dem Amiga selbst im relativ ausführlichen Handbuch für Amiga-Basic. Falls Sie noch keinerlei Erfahrung mit der Programmierung von Computern haben, wird dieses Handbuch allerdings nicht ausreichen. Für die Einführung in die Sprache Basic existieren aber bereits einige sehr gute Lehrbücher, die Ihnen in diesem Fall helfen sollten. Auch speziell zu Amiga-Basic gibt es eine Reihe von Lehrbüchen und Tip-Sammlungen. Viele Volkshochschulen bieten zudem inzwischen Basic-Kurse an. Diese sind zwar nicht speziell für den Amiga ausgelegt, wenn Sie aber erst einmal ein Basic kennengelernt haben, werden Ihnen die anderen Dialekte keine allzu großen Schwierigkeiten mehr bereiten. Es könnte sogar sinnvoll sein, zunächst eine ganz andere Programmier-

sprache, wie zum Beispiel Pascal, zu erlernen. Die Unterschiede zwischen den verschiedenen Computer-Sprachen zu bewältigen, ist bei weitem nicht so schwer wie ein Wechsel von Deutsch nach Französisch.

## 13.1 Basic

Basic ist mehr oder weniger die Computersprache, die für die meisten Leute untrennbar mit dem Mikrocomputer verbunden ist. Sie hatte aber schon eine lange Geschichte hinter sich, bevor die erste Zeile eines Basic-Programms auf einem Mikrocomputer ausgeführt wurde.

Das Wort Basic ist eine Abkürzung für *Beginners All Purpose Symbolic Instruction Code* (Allzweck-Instructionscode für Anfänger) und deutet damit schon auf den Zweck hin, zu dem die Sprache ursprünglich am Dartmouth-College in den USA entwickelt wurde: der Ausbildung von Anfängern in der Computer-Programmierung (in einem Instructionscode, wie man damals noch sagte). Auch heute noch wird Basic vielfach für diese Zwecke eingesetzt. Nach wie vor ist einer der Hauptvorteile dieser Sprache nämlich die schnelle Erlernbarkeit. Das führt dazu, daß jeder Anfänger relativ schnell eigene Programme schreiben und so seine ersten Erfolgserlebnisse am Computer haben kann.

Die ersten Basic-Realisierungen für Mikrocomputer waren aber (aus Gründen des damals noch sehr knappen Speicherplatzes) notwendigerweise sehr beschränkt. Basic-Dialekte auf Mikrocomputern waren lange Zeit recht primitive Sprachen, die wirklich nur die allernötigsten Hilfsmittel der Programmierung zur Verfügung stellten. Das führte in der Folgezeit dazu, daß Basic in Kreisen der Computer-Wissenschaft (in Deutschland heißt sie Informatik) einen sehr schlechten Ruf bekam. Mit diesen primitiven Basic-Formen war es nämlich sehr leicht, Programme zu schreiben, die so unübersichtlich waren, daß sie nach kurzer Zeit niemand mehr verstehen und vor allem ändern konnte. Gerade das Verstehen und Ändern alter Programme ist aber eine der Hauptaufgaben vieler Programmierer und kostet die Programm-Hersteller auf Dauer mehr als das Erstellen einer ersten Programmversion.

Einer der Vorwürfe, die man Basic oft machte, war zum Beispiel, daß die Sprache zu sogenanntem »Spaghetti-Code« verführt. Viele Programme in den frühen Basic-Versionen waren nämlich lange Ketten von unstrukturierten Programmzeilen, zwischen denen das Programm bei der späteren Ausführung hin- und herhüpfen mußte. Wenn man diese Programmsprünge auf Papier mit einem Bleistift nachzuvollziehen versuchte, ergab dies ein wildes Durcheinander von Linien – eben den berühmten Spaghetti-Code.

Zudem kannten frühe Basic-Versionen keine Namen für Teilprogramme (Subroutinen oder Prozeduren), sondern nur Nummern. Stand in einem anderen Programmteil nun der Aufruf einer solchen Prozedur, mußte man erst die Bedeutung dieser Nummer nachschlagen, wenn man wissen wollte, was das Programm in diesem Teil tun sollte. (Oft war diese Bedeutung sogar überhaupt nirgends dokumentiert und nur der Programmierer allein wußte, was das entsprechende Unterprogramm tun sollte.) In anderen Programmiersprachen war es zu dieser Zeit längst üblich, Unterprogrammen sinnvolle Namen zu geben und sie auch mit diesen Namen aufzurufen. Statt GOSUB 5390 in Basic konnte man schreiben *Zeige_Daten_Von(Müller).*

Im Laufe der Zeit wurde Basic aber auch auf Mikrocomputern immer »erwachsener», was nicht zuletzt damit zusammenhing, daß der Speicher immer billiger wurde, und die Basic-

Programme größer wurden und so auch mehr Möglichkeiten enthalten konnten. Das erste Basic auf einem Mikrocomputer war zum Beispiel gerade 4 Kbyte groß; Amiga-Basic hingegen ist immerhin schon 80 Kbyte groß und es ist nicht einmal das größte Basic auf Mikrocomputern.

Viele heutige Basic-Dialekte (wie zum Beispiel Amiga-Basic) enthalten alle wesentlichen Merkmale einer modernen Programmiersprache. Sie erlauben mühelos die sogenannte strukturierte Programmierung, mit deren Hilfe Programme so übersichtlich geschrieben werden können, daß man sie auch noch Jahre, nachdem sie geschrieben wurden, verstehen und ändern kann. Unterprogramme und Programmvariablen können sinnvolle Namen erhalten und eine Vielzahl wichtiger mathematischer und anderer Funktionen sind fest eingebaut und brauchen nicht erst mühselig von jedem Programmierer neu geschrieben zu werden.

Basic hat im Laufe der Jahre viele gute und erprobte Fähigkeiten anderer Programmiersprachen aufgenommen und läßt auch hinsichtlich des Komforts bei der Programmentwicklung und Fehlersuche heute nicht mehr viele Wünsche offen. Im nächsten Abschnitt wollen wir uns den Dialekt »Amiga-Basic» einmal genauer betrachten, um zu sehen, wie er im Vergleich abschneidet.

## 13.2 Der Basic-Dialekt »Amiga-Basic«

Das Basic des Amiga stammt von der Firma Microsoft®. Dieses Softwarehaus ist bezüglich der Sprache Basic praktisch der »Hauslieferant» für fast alle Hersteller von Mikrocomputern. Das erste Basic für einen der ersten Mikrocomputer stammte von Microsoft, und bis heute gibt es kaum einen Mikrocomputer, für den es kein Microsoft-Basic (MS-Basic) gibt. Es ist meist die erste Programmiersprache, die für jeden neuen Computer auf den Markt kommt.

### 13.2.1 Microsoft-Basic

Diese weite Verbreitung von MS-Basic ist ein enormer Vorteil für den Programmierer, der diese Sprache benutzt. Alle Basic-Versionen der Firma Microsoft sind sich nämlich (jeweils in der neuesten Version) weitgehend ähnlich. Wenn man Microsoft-Basic auf einem Computer kennt und dann auf einen anderen überwechselt, gibt es deshalb sehr wenig Umstellungsschwierigkeiten. (Dies betrifft natürlich nicht die speziellen Fähigkeiten der verschiedenen Computer, die natürlich in den verschiedenen Basics auch unterschiedlich unterstützt werden müssen. Viele andere Computer haben nun einmal keine Maus, Fenster und hochauflösende Farbgrafik wie der Amiga.) Viele Programme können deshalb mit sehr wenigen Änderungen direkt von einem Computer zum anderen übernommen werden, so daß man besonders bei neuen Computern gleich zu Anfang schon etwas Software zur Verfügung hat.

Auch wenn Sie ein gutes Lehrbuch für Amiga-Basic suchen, hilft Ihnen die weite Verbreitung von MS-Basic. Es dauert nämlich naturgemäß einige Zeit, bis gute Programmierbücher für einen neuen Computer auf den Markt kommen. Falls Sie aber ein gutes Basic-Buch für den IBM-PC® (PC-Basic® oder MS-Basic®) oder noch besser den Apple® Macintosh®

finden, zögern Sie nicht, zuzugreifen! Alle »wesentlichen« Eigenschaften des Basic für den IBM-PC und den Amiga sind gleich. Die Ähnlichkeiten bezüglich der Verwaltung von Menüs, Maus und Fenstern zwischen Amiga-Basic und MS-Basic für den Apple Macintosh sind sogar noch größer.

### 13.2.2 Moderne Features von Microsoft-Basic

Einige der wichtigsten Eigenschaften des MS-Basic für den Amiga sollen nun stichwortartig aufgelistet werden. Es gibt sie fast alle in jeder neueren Version von MS-Basic und sie sind deshalb nicht Amiga-spezifisch. Es sind aber Möglichkeiten, die durchaus nicht in jedem Basic zu finden sind, und die deutliche Fortschritte gegenüber den primitiven früheren Basic-Versionen (siehe oben) darstellen.

- Optionale Zeilennummern: Alle Zeilen eines Programms können, müssen aber nicht, mit Zeilennummern versehen werden. MS-Basic benötigt intern diese Zeilennummern nur noch für die Programmzeilen, die Ziel eines Programmsprungs (GOTO) sind. Nur die wenigsten Zeilen eines Programms sind aber normalerweise Spungziele. Diese Zeilen können mit einer Nummer versehen werden. Noch besser ist es aber, sie mit einer lesbaren Marke zu versehen (siehe unten).
- Programm-Marken: Wichtige Stellen eines Programms, zum Beispiel die Ziele von Sprunganweisungen, können mit einem lesbaren Namen, einer sogenannten Marke (engl. Label), versehen werden. Diese wichtigen Stellen werden so deutlich hervorgehoben und das Programm in überschaubare Abschnitte gegliedert.
- Anweisungs-Blocks: Eine beliebig lange Folge von Zeilen eines Programms kann zu einem Block zusammengefaßt werden. Solche Blöcke finden zum Beispiel bei bedingten Anweisungen (IF-THEN-ELSE-ENDIF), Programmschleifen (FOR-NEXT, WHILE-WEND) oder Unterprogrammen (SUB-END-SUB) Anwendung. Die Verwendung von unübersichtlichen Sprüngen (GOTO-Anweisungen) für diese Zwecke ist nicht mehr nötig.
- Lange Namen: Alle Bezeichnungen für Variablen, Marken, Unterprogramme und so weiter können lange Namen sein (denen man wirklich etwas entnehmen kann) und so jedes Programm wesentlich lesbarer machen. Alle diese Namen können bis zu 40 Buchstaben lang werden, und alle 40 Buchstaben werden auch zur Unterscheidung der verschiedenen Namen eingesetzt.
- Reiche Auswahl an Datentypen: Für Variablen und die Ergebnisse von Funktionen stehen insgesamt 5 verschiedene Datentypen zur Verfügung. Amiga-Basic kennt zwei Genauigkeitsstufen für ganze Zahlen (16 Bit und 32 Bit), zwei Genauigkeitsstufen für Fließpunktzahlen (32 Bit und 64 Bit) und Strings (Texte) die bis zu 32 768 Buchstaben enthalten können. Arrays (Felder bzw. Matrizen) können bis zu 255 Dimensionen haben und in jeder Dimension bis zu 32 768 Elemente enthalten.
- Subprogramme: Neben den auch aus anderen Basic-Dialekten bekannten Subroutinen gibt es auch echte »Subprogramme« oder »Prozeduren«, die die Übersichtlichkeit großer Programme wesentlich verbessern und ihre Fehleranfälligkeit verringern können. Subprogramme haben stets einen (hoffentlich aussagekräftigen) Namen und können Parameter besitzen, die beim Aufruf mit einem Wert belegt werden. Weiterhin können Subprogramme eigene (lokale) Variablen besitzen, die von denen des Hauptprogramms ver-

schieden sind, selbst wenn sie genauso heißen. Subprogramme können sich sogar selbst aufrufen, was gar nicht so selten oder überflüssig ist, wie man vielleicht meint (der Informatiker nennt solche Subprogramme dann »rekursiv«).

- Programm-Chaining: Jedes Basic-Programm kann andere Basic-Programme aufrufen und diesen sogar (über die COMMON-Anweisung) Daten übergeben. Dies nennt man »Programm-Verkettung« oder »-Chaining«. Auf diese Weise kann man sehr große Programme entwickeln, von denen aber immer nur die gerade benötigten Teile im Speicher sind. So bleibt mehr Platz für Daten frei.
- Komfortable Dateiverwaltung: MS-Basic besitzt reichhaltige Möglichkeiten für die Verarbeitung von Dateien, die auf der Diskette oder einer Festplatte gespeichert sind. Solche Dateien können in Datensätze gleicher Struktur aufgeteilt werden, bei denen jeder Datensatz aus einer Reihe von Feldern mit aussagekräftigen Namen besteht. Dateien können sequentiell (also Datensatz für Datensatz) oder auch mit wahlfreiem Zugriff (engl. *random access*) auf jeden beliebigen Datensatz bearbeitet werden.

## 13.3 Amiga-Basic und der Amiga 500

Obwohl Amiga-Basic sehr ähnlich zu den anderen MS-Basic-Dialekten auf anderen Computern ist, hat Microsoft es aber doch auch sehr sorgfältig an die speziellen Fähigkeiten des Amiga angepaßt. Fast jede dieser Fähigkeiten, die den Amiga ja erst zu einem so außergewöhnlichen Computer machen, kann von Basic aus genutzt werden. Es ist also problemlos möglich, eigene Programme zu schreiben, die wie »richtige Amiga-Programme» aussehen; mit eigenen Screens, Fenstern, Menüs und Maus-Bedienung. Zugleich werden diese Möglichkeiten aber auch für die Programmentwicklung selbst genutzt und machen dem Programmierer so das Leben etwas leichter.

### 13.3.1 Die Bedienung von Amiga-Basic

Amiga-Basic ist, wie die meisten Basic-Systeme, ein sogenannter Interpreter. Dies bedeutet, daß für die Ausführung eines Programms immer ein kleines Programm (eben der Interpreter) benötigt wird, das Zeile für Zeile durch den Programmtext läuft, jeweils eine Anweisung liest und sie dann ausführt. Dies ist ein verhältnismäßig (für einen Computer) langsamer Prozeß. Schnellere Programme erhält man meist durch einen Compiler, der einen Programmtext nimmt und in einen Code übersetzt, der direkt von der CPU des Computers ausgeführt werden kann, den sogenannten Maschinen-Code. Auch für Amiga-Basic wird sicherlich über kurz oder lang ein solcher Compiler auf den Markt kommen. Für andere Basic-Varianten gibt es solche Compiler bereits.

Die Arbeit mit einem Interpreter hat aber unschätzbare Vorteile für den Bedienungskomfort. Während die oben beschriebene Übersetzung eines Programms in Maschinen-Code ein langwieriger und vor allem für Anfänger fehleranfälliger Vorgang ist, braucht man ein Amiga-Basic-Programm einfach nur einzutippen und kann es sofort starten. Auch jede Änderung kann sofort – ohne jede Wartezeit – ausprobiert werden.

Weiterhin arbeitet man mit Amiga-Basic in zwei verschiedenen Fenstern. In einem, dem sogenannten Output-Fenster, spielen sich die Ein- und Ausgaben von Programmen ab. Sie

können hier auch über die Tastatur Befehle an das Basic-System erteilen. Das zweite Fenster, das sogenannte »Listing-Fenster«, enthält den Text des gerade bearbeiteten Programms. Basic-Programme können darüber hinaus beliebig viele zusätzliche Fenster öffnen, für deren Inhalt sie dann auch selbst zuständig sind. Aber auch die beiden ersten Fenster müssen nicht immer sichtbar sein. Wenn ein Programm fehlerfrei läuft, wird man üblicherweise das Listing-Fenster schließen und Programme, die selbst eigene Fenster öffnen, werden oft auch das erste Output-Fenster schließen und alle Ein- und Ausgaben in den eigenen Fenstern erledigen.

Auch die Eingabe und das Ändern von Programmen im Listing-Fenster ist sehr komfortabel. Wer Basic-Interpreter auf anderen Computern kennt, wird staunen. Zur Bearbeitung des Programmtextes im Listing-Fenster stehen Ihnen ähnliche Möglichkeiten wie im *Notepad* zur Verfügung. Mehr dazu aber weiter unten.

Tauchen während des Programmablaufs Fehler auf, so stoppt Basic, gibt eine entsprechende Fehlermeldung aus und zeigt Ihnen die Fehlerstelle auch durch einen dünnen Rahmen um die entsprechende Stelle im Programmtext. War das Listing-Fenster geschlossen, wird es hierzu zuvor geöffnet. Sie können den Fehler nun beseitigen und das Programm noch einmal laufen lassen.

Basic selbst findet aber natürlich nur recht grobe Fehler, zum Beispiel Tippfehler. Die meisten logischen Fehler müssen Sie auf andere Weise herausfinden, wofür es aber eine ganze Reihe von Hilfen gibt.

Mit dem Befehl *Trace On* haben Sie zum Beispiel eine genaue Kontrolle über den Ablauf eines Programms. Nachdem *Trace On* ausgewählt wurde, zeigt Amiga-Basic während der Ausführung eines Programms kontinuierlich an, in welcher Zeile sich das Programm gerade befindet. Hierzu wird im Listing-Fenster die gerade ausgeführte Zeile orange umrahmt. Und mit dem Befehl Step schließlich können Sie Schritt für Schritt (Zeile für Zeile) durch ein Programm gehen. Nach der Ausführung jeder Anweisung des Programms hält Amiga-Basic automatisch an.

### 13.3.2 Unterstützung der Amiga-Fähigkeiten

Amiga-Basic sieht eine ganze Reihe von Funktionen vor, die es erlauben, von einem Basic-Programm aus alle wichtigen Elemente der Amiga-Benutzerschnittstelle Intuition zu verwenden. Dies ist natürlich wichtig für jeden, der Programme schreiben möchte, die wie »echte Amiga-Programme aussehen«, also sich von ihrer Bedienung her nicht von professionellen Programmen unterscheiden.

Im folgenden finden Sie einen kleinen Überblick über diese Möglichkeiten. Er ist recht stichwortartig gehalten und soll Ihnen nur einen Eindruck von den Möglichkeiten verschaffen, die Amiga-Basic bietet. Detaillierte Erläuterungen der einzelnen Befehle finden sich im Kapitel 8 des Amiga-Basic-Handbuchs.

#### Grafik

Der einfachste Grafik-Befehl von Amiga-Basic ist der LINE-Befehl. Mit ihm können Linien zwischen zwei Punkten in einer wählbaren Farbe gezogen werden. Alternativ kann er auch dazu verwendet werden, Rechtecke zu zeichnen, die auf Wunsch gleich mit einer bestimm-

ten Farbe gefüllt werden. Der Befehl CIRCLE kann Kreise, Ellipsen sowie Kreisbögen mit einem bestimmten Radius um einen festgelegten Mittelpunkt herum zeichnen. Auch diese können auf Wunsch gleich gefüllt werden. Komplexere Formen, Polygone (Vielecke), können mit dem AREA-Befehl Punkt für Punkt definiert und mit AREAFILL dann gezeichnet werden.

Noch komplexere Formen als die oben aufgeführten muß man aus diesen »Elementen« zusammensetzen. Sobald eine Fläche der gewünschten Form aber erst einmal einen Rahmen hat, kann sie mit Befehl PAINT jederzeit mit einer beliebigen Farbe gefüllt werden. Dies wirkt genauso wie die Anwendung des Farbeimers in *Deluxe Paint* oder anderen Malprogrammen.

Bei den Farben, die für eine Basic-Grafik verwendet werden können, ist man natürlich nicht auf die vier »Workbench-Farben» beschränkt. Mit dem Befehl PALETTE kann man diese jederzeit ändern. Über PALETTE kann man jedes der Farbregister im Video-Chip Denise mit neuen Rot-, Grün- und Blau-Werten füllen. (Siehe auch Kapitel 11.) Wem vier Farben nicht genügen, der kann sich mit SCREEN-Anweisung (siehe unten) einen eigenen Screen erzeugen, der bis zu 32 Farben enthalten kann.

### Fenster und Screens

»Normalerweise« gehen alle Ausgaben eines Programms in das eine Output-Fenster, das gleich am Anfang, wenn man Amiga-Basic startet, erscheint. Auch die eben beschriebenen Grafikbefehle arbeiten zunächst in diesem Fenster. Wem ein Fenster aber nicht genügt oder das Standard-Output-Fenster nicht gefällt, der kann sich mit dem Befehl WINDOW ein neues erzeugen, dessen Titel, anfängliche Größe und sonstige Eigenschaften völlig frei gewählt werden können. Auch Fenster, die von Programmen erzeugt wurden, können als Output-Fenster verwendet werden, indem man sie mit der WINDOW OUTPUT-Anweisung zum aktuellen Output-Fenster macht. Man kann zum Beispiel PRINT-Befehle in diese Fenster schicken und in ihnen Grafiken zeichnen. Von einem Programm erzeugte Fenster, aber auch die beiden Standard-Fenster, können jederzeit auch von einem Programm aus mit WINDOW CLOSE wieder geschlossen werden.

Für anspruchsvolle Grafiken genügt es aber kaum, ein spezielles Fenster zu öffnen. In Basic steht einem ja »normalerweise» nur ein Screen mit vier Farben zur Verfügung. Braucht man mehr Farben, eine andere Bildschirmauflösung oder möchte man gar den HAM-Modus des Denise-Chips einmal ausprobieren, kann man mit SCREEN einen neuen virtuellen Bildschirm erzeugen und ihn mit SCREEN CLOSE wieder schließen, sobald er nicht mehr gebraucht wird.

### Menüs

Die Pull-down-Menüs sind eine der wichtigsten Komponenten der Amiga-Benutzerschnittstelle Intuition. Auch von Amiga-Basic aus können deshalb neue Menüs und Menü-Punkte erzeugt werden. Die Basic-Anweisung MENU tut genau dies. Amiga-Basic kann allerdings nur reine Text-Menüs erzeugen. Grafische Menüs, wie sie in vielen Mal- und Zeichenprogrammen vorkommen, sind nicht möglich.

Mit einer zweiten Form des MENU-Befehls kann ein Programm jederzeit feststellen, aus welchem Menü der Benutzer zuletzt welchen Befehl ausgewählt hat. Sie werden in den folgenden Abschnitten noch anhand einiger Beispiele sehen, wie man das macht. Zum

Feststellen der Tatsache, ob der Benutzer überhaupt eine Menü-Auswahl getätigt hat, wird ein anderer Mechanismus verwendet, der weiter unten noch erläutert wird.

### Die Maus

Einen Teil der Mausbewegungen und Mausklicks überwacht und interpretiert ja bereits die Amiga-Benutzerschnittstelle *Intuition* (zum Beispiel das Bewegen von Fenstern und Herunterholen von Pull-down-Menüs mit der Maus). Mausklicks innerhalb eines Fensters müssen aber normalerweise von dem für dieses Fenster zuständigen Programm behandelt werden. Ein Basic-Programm verwendet dafür üblicherweise die Funktion MOUSE. Mit dieser kann man feststellen, ob die linke Maustaste im Moment gedrückt ist oder nicht, über welchem Punkt sie gedrückt und wieder losgelassen wurde und ob es ein Doppelklick war oder nicht.

Die rechte Menütaste der Maus kann von Basic aus nicht abgeprüft werden. Sie kann nur indirekt über die MENU-Funktion von einem Basic-Programm benutzt werden.

### BOBs und Sprites

Eine der faszinierendsten Möglichkeiten, die der Amiga grafisch bietet, sind ja bewegte Objekte. Diese gibt es dabei in zwei »Geschmacksrichtungen«: die in Hardware erzeugten Sprites, die schnell sind, aber diversen Einschränkungen unterliegen, oder die etwas langsameren, aber wesentlich vielseitigeren BOBs, die von der Systemsoftware des Amiga und dem Blitter ermöglicht werden. (Siehe hierzu auch Kapitel 11.) In Amiga-Basic werden beide Typen einfach Objekte genannt und immer mit demselben Satz von Befehlen behandelt.

Für die Definition des Aussehens von Objekten befindet sich ein separates Programm namens *ObjEdit* auf der Extras-Diskette. Es ist übrigens in Basic geschrieben und demonstriert so recht eindrucksvoll die Möglichkeit, sinnvolle Amiga-Programme in Basic zu entwickeln.

Die mit diesem Programm entwickelte Form eines Objektes kann dann mit OBJECT.SHAPE für die Erzeugung eines neuen Sprites oder BOBs verwendet werden. Ein solches Objekt kann man dann mit OBJECT.X und OBJECT.Y an einem bestimmten Punkt am Bildschirm zeigen und ihm mit OBJECT.VX und OBJECT.VY eine Geschwindigkeit geben, mit der es sich (ohne weiteres Zutun des Programms!) über den Bildschirm bewegt. Mit OBJECT.AX und OBJECT.AY bekommt es dann sogar noch eine Beschleunigung, wodurch die Geschwindigkeit des Objekts sich mit der Zeit automatisch ändert. Treffen zwei Objekte aufeinander oder versuchen sie einen »zulässigen» Bereich zu verlassen, kann sich ein Programm davon informieren lassen (siehe auch den nächsten Abschnitt) und mit dem Befehl COLLISION feststellen, welches Objekt auf welches Hindernis gestoßen ist.

Mit OBJECT.PRIORITY kann schließlich auch noch festgelegt werden, welches Objekt »vor« welchem anderen erscheint, wenn zwei oder mehr Objekte an ein und derselben Stelle am Bildschirm stehen. 3-D-Effekte sind so auch für ein Basic-Programm kein Problem.

### Abfangen von Ereignissen

Auf viele Ereignisse muß ein Programm sehr schnell reagieren können. Wenn zwei Objekte zusammenstoßen oder die linke Maus-Taste niedergedrückt wird, muß am besten sofort eine

Reaktion erfolgen. Dies ist aber, wenn es sehr viele solche Ereignisse gibt, die ja alle regelmäßig und möglichst schnell abgeprüft werden müssen, schwer zu programmieren.

Amiga-Basic bietet deshalb mit dem ON...GOSUB-Befehl (die drei Pünktchen deuten ein anderes Wort an, das hier eingesetzt werden muß) eine Möglichkeit, ein bestimmtes Ereignis mit einer bestimmten Subroutine fest zu verknüpfen. Dies nennt man dann »Abfangen des Ereignisses« (engl. *event trapping*). Immer wenn dann ein solches Ereignis, zum Beispiel ein Mausklick, eintritt, ruft Amiga-Basic sofort diese Subroutine auf, die in der entsprechenden ON...GOSUB-Anweisung genannt wurde – egal welcher andere Programmteil gerade aktiv ist. Wenn die Subroutine mit RETURN beendet wird, springt das Programm automatisch wieder an die Stelle zurück, an der es unterbrochen wurde. Eine sehr schnelle Reaktion auf bestimmte Ereignisse kann so stets gewährleistet werden.

Die Ereignisse, die auf diese Weise abgefangen werden können, sind:

- Die Auswahl eines Menü-Punktes (ON MENU)
- Ein Mausklick (ON MOUSE)
- Das Ablaufen einer bestimmten Zeitspanne (ON TIMER)
- Die Kollision zweier Objekte (ON COLLISION)
- Der Versuch des Benutzers, ein Programm zu stoppen (ON BREAK)
- Ein Programmfehler (ON ERROR)

Die Subroutine, die auf ein bestimmtes Ereignis hin von Amiga-Basic aufgerufen wird, kann jederzeit mit einem erneuten Aufruf von ON...GOSUB wieder geändert werden. Je nach Programmsituation können also andere Programmteile ein bestimmtes Ereignis bearbeiten.

### Musik- und Sprachausgabe

Und last-but-not-least können auch die musikalischen und sprachlichen Fähigkeiten des Amiga von Basic aus genutzt werden. Der Befehl SOUND gibt eine Note einer bestimmten Frequenz mit einer bestimmten Dauer und Lautstärke auf einem der vier Klangkanäle aus. Für jeden Klangkanal kann zuvor eine Klangform mit dem Befehl WAVE festgelegt werden.

Die Befehle TRANSLATE und SAY schließlich dienen der Sprach-Synthese. TRANSLATE wandelt einen (englischen) Text in die Phonemschreibweise um und SAY gibt einen Text in Phonemschreibweise aus. Um wirklich gute Ergebnisse (oder deutsche Sprache) zu erzielen, wird man die Phonemschreibweise eines Wortes im allgemeinen nicht durch TRANSLATE ermitteln lassen, sondern mit einem guten Wörterbuch feststellen.

## 13.4 Umgang mit Amiga-Basic

In diesem und den folgenden Abschnitten werden Sie Amiga-Basic etwas näher kennenlernen – sofern Sie Lust dazu verspüren. Dazu werde ich zunächst den Umgang mit dem Amiga-Basic-Interpreter kurz vorstellen und dann einige kleine Beispielprogramme vorstellen.

### 13.4.1 Amiga-Basic starten

Amiga-Basic ist ein Werkzeug wie alle anderen Workbench-Werkzeuge, die Sie schon kennengelernt haben. Das entsprechende Piktogramm befindet sich auf der Extras-Diskette. Sie können Amiga-Basic starten, indem Sie einfach einen Doppelklick auf dieses Piktogramm machen.

Danach erscheinen zwei Fenster am Bildschirm. Sie heißen Basic und LIST. Basic ist das Output-Fenster und LIST das Listing-Fenster. Wenn Sie nun die Menütaste drücken und in der Titelleiste des Bildschirms hin- und herfahren, können Sie auch die Menüs von Amiga-Basic sehen. Verlassen können Sie Amiga-Basic übrigens mit dem Befehl Quit im Menü Project – aber darauf wären Sie als erfahrener Amiga-Besitzer sicherlich auch selbst gekommen.

### 13.4.2 Programme eingeben und editieren

Das Listing-Fenster dient zur Eingabe und zur Änderung von Basic-Programmen. (Basic-Anweisungen, die Sie im Output -Fenster eingeben, werden hingegen sofort ausgeführt, wenn Sie [Return] drücken und können nicht wieder geändert werden.) Das Listing-Fenster verhält sich fast genauso wie das Fenster des Programms *Notepad*, in dem Sie ja auch Texte editieren können. Ein blinkender Strich, die Schreibmarke, markiert immer die Stelle, an der neu eingegebene Buchstaben erscheinen. Tippen Sie einfach einmal ein paar Zeilen ein. Diese müßten daraufhin im Listing-Fenster erscheinen. Mit der [Backspace]-Taste können Sie dann wie üblich den Buchstaben links von der Schreibmarke und mit [Del] den rechts von der Schreibmarke löschen.

Die Schreibmarke (der blinkende Strich) kann mit der Maus oder mit den vier Cursor-Tasten beliebig im ganzen Text eines Programms verschoben werden. Ein Verschieben des Fensters über ein großes Programm geschieht jedoch nicht mit Rollbalken, wie es in anderen Programmen üblich ist, sondern ebenfalls mit den Cursor-Tasten (indem man die [Shift]-Taste festhält und dann auf die Cursor-Taste drückt, deren Aufdruck in die Richtung weist, in die man das Fenster verschieben möchte).

Wenn Sie eine beliebige Stelle im Text mit der linken Maustaste anklicken und bei gedrückter Taste den Mauszeiger zu einer anderen Stelle ziehen, werden alle Buchstaben zwischen diesen beiden Punkten hervorgehoben. (Einzelne Wörter können auch durch einen Doppelklick über einer beliebigen Stelle des Wortes hervorgehoben werden.) Ein so ausgewählter Bereich kann mit den Menü-Befehlen aus dem Menü *Edit* dann bearbeitet werden. Der Befehl *Cut* entfernt ihn aus dem Text, legt ihn aber zugleich in eine »Zwischenablage«. Der Befehl *Copy* kopiert den hervorgehobenen Text nur in die Zwischenablage. Und der Befehl *Paste* fügt schließlich den Text aus der Zwischenablage vor der Schreibmarke in den Programmtext ein. Diese drei Befehle verhalten sich sehr ähnlich wie die gleichnamigen Befehle im *Notepad*. Mit Hilfe des *Edit*-Menüs kann man sehr leicht auch größere Umstellungen im Programm vornehmen.

Ein großer Vorteil von Amiga-Basic gegenüber primitiveren Versionen dieser Sprache sind die Möglichkeiten zur Formatierung eingegebener Programme. So kann das Innere von Schleifen oder IF-THEN-ELSE-Anweisungen nach rechts eingerückt werden, wodurch solche Blöcke natürlich gleich viel mehr ins Auge stechen. Die [Tab]-Taste ist ideal für solche Einrückungen. Sie fügt jeweils 3 Leerzeichen bei der Schreibmarke ein.

Neben diesen Formatierungen, für die man selbst sorgen muß, gibt es aber auch automatische Formatierungen, die Amiga-Basic selbst durchführt. Wenn Sie ein Basic-Schlüsselwort, wie zum Beispiel IF, FOR oder GOTO eingegeben haben und in eine neue Zeile gehen, schreibt Basic diese Schlüsselwörter automatisch groß, egal, wie Sie sie eingegeben haben. Dies erhöht natürlich ebenfalls die Lesbarkeit des Programms und hilft Ihnen zugleich, Tippfehler zu entdecken. Wollten Sie nämlich ein Schlüsselwort eingeben und es erscheint nicht in Großbuchstaben, muß sich darin ein Tippfehler befinden. Verwenden Sie aber einen Variablen- oder Subprogramm-Namen und dieser erscheint plötzlich in Großbuchstaben, obwohl Sie ihn klein eingegeben haben, so gibt es ein Schlüsselwort gleichen Namens, woraufhin Sie sich einen anderen Namen ausdenken sollten. Sie werden diese Effekte gleich beobachten können. Geben Sie dazu einfach das folgende kurze Programm im Listing-Fenster ein (löschen Sie dazu eventuell die Spuren Ihrer ersten Tipp-Versuche):

```
Start:
print "Hallo Ihr da!"
goto start
```

Wenn Sie dieses Programm eingegeben haben ([Return] am Zeilenende nicht vergessen!), sollte es so aussehen wie das folgende Programm:

```
start:
PRINT "Hallo Ihr da!"
GOTO start
```

Amiga-Basic hat die beiden Schlüsselwörter *print* und *goto* automatisch groß geschrieben und auch dafür gesorgt, daß die beiden Vorkommnisse des Wortes *start* identisch geschrieben werden.

### 13.4.3 Programme starten und wieder anhalten

Sie können dieses winzige Programm nun starten, indem Sie den Befehl *Start* aus dem Menü *Run* wählen. Das Output-Fenster kommt daraufhin nach vorn und eine endlose Kette von »Hallo Ihr da!« wird darin ausgegeben. Das Programm hat kein »natürliches« Ende, da der Sprung zurück nach *start* in der letzten Zeile eine Endlosschleife schließt. Sie müssen es deshalb »von außen« anhalten. Wählen Sie hierzu den Befehl *Stop* aus dem Menü *Run*. Das Programm hält dann an und das Listing-Fenster kommt automatisch wieder nach vorn.

### 13.4.4 Fehler in Programmen suchen

Bauen Sie nun einmal einen kleinen Fehler in das Programm ein, um zu sehen, wie Amiga-Basic darauf reagiert. Fügen Sie dazu vor *GOTO start* in eine neue Zeile den Befehl *GOTO Hallo* ein. Das Programm müßte dann aussehen wie folgt:

```
start:
PRINT "Hallo Ihr da!"
GOTO Hallo
GOTO start
```

Wenn Sie nun dieses Programm starten, ertönt sofort ein kleiner Piepston und ein schmales Fenster erscheint am oberen Bildschirmrand, das Ihnen die Fehlermeldung *Undefined label*

(nicht definierte Sprungmarke) zeigt. Sie können dieses Fenster mit einem Klick auf den *OK*-Knopf darin schließen. Unterdessen hat Amiga-Basic aber auch schon das Listing-Fenster nach vorne gebracht und die Zeile mit dem Fehler darin markiert. Der Fehler selbst ist Ihnen ja wahrscheinlich klar, oder? Sie können mit der GOTO-Anweisung nicht zu einer Sprungmarke springen, wenn diese Sprungmarke nirgends definiert wurde. Die Sprungmarke *start* wurde ja definiert, indem Sie den Namen start mit einem Doppelpunkt dahinter an den Anfang einer Zeile geschrieben haben. Löschen Sie deshalb nun den fehlerhaften Befehl *GOTO Hallo* wieder und starten Sie das Programm erneut. Nun müßte wieder die endlose Folge von »Hallo Ihr da!«-Zeilen erscheinen.

Rufen Sie nun einmal den Befehl *Trace On* aus dem Menü *Run* auf. Das Programm wird dann merklich langsamer. Ansonsten passiert aber scheinbar nichts. Daß nichts passiert liegt daran, daß das Listing-Fenster vom Output-Fenster verdeckt wird. Holen Sie es deshalb nun – bei laufendem Programm – mit dem Befehl *Show List* im Menü Windows nach vorn. Nun können Sie verfolgen, was geschieht, wenn Sie *Trace On* aufrufen. Während des Ablaufs des Programms wird der gerade ausgeführte Befehl immer mit einem Rahmen versehen. Sie können damit sehr gut verfolgen, ob das Programm wirklich die »Wege geht«, die Sie geplant haben. In diesem simplen Programm ist das natürlich keine große Hilfe. In einem großen komplexen Programm kann es entscheidend dazu beitragen, logische Fehler zu finden.

Sie können die Markierung der aktiven Zeile jederzeit wieder abschalten, indem Sie *Trace Off* aus dem Menü *Run* wählen. Das Programm arbeitet dann wieder mit gewohnter Geschwindigkeit. Wenn Ihnen aber selbst die Geschwindigkeit bei *Trace On* noch zu groß ist, haben Sie noch eine andere Möglichkeit, den Programmfluß zu kontrollieren. Halten Sie dazu das Programm zunächst einmal mit dem Befehl *Stop* aus dem Menü *Run* an und wählen Sie dann den Befehl *Step* aus dem Menü *Run*. Nun wird auch wieder der aktuelle Befehl dünn umrahmt, das Programm läuft aber nicht weiter. Erst, wenn Sie wieder Step aufrufen, macht es den nächsten Arbeitsschritt. Dazu einen Menübefehl aufrufen zu müssen, ist natürlich umständlich. Die Tastenkombinationen [*A*]+[T], das Tastaturäquivalent von *Step*, ist da schon handlicher. Halten Sie dazu die Amiga-Taste fest und drücken Sie immer dann auf [T], wenn Sie wollen, daß das Programm einen Schritt weitergeht. Das ist praktisch, nicht wahr?

Wenn es Ihnen aber auf den Geist geht, für jeden Programmschritt auf eine Taste drücken zu müssen, können Sie den Befehl *Continue* aus dem Menü *Run* wählen. Das Programm läuft dann normal weiter. Der letzte Befehl im *Run*-Menü, *Suspend*, wirkt ähnlich wie *Stop*. Er hält also das laufende Programm an. Dieser Stop ist aber nicht endgültig; das Programm wird nur so lange unterbrochen, bis auf eine beliebige Taste gedrückt wird.

### 13.4.5 Programme abspeichern

Für Amiga-Basic gilt dasselbe wie für die meisten anderen Programme: Neueingaben oder Änderungen (an einem Programm) werden zunächst einmal nur in einem Puffer im RAM-Speicher ausgeführt. Wenn Sie die Änderungen auch nach dem Verlassen des Programms erhalten wollen, müssen Sie sie auf Diskette oder Festplatte sichern. Zum Sichern eines Basic-Programms gibt es die zwei Befehle *Save* und *Save As* im Run-Menü.

Wenn Sie ein neues Programm eingeben und dann Save aufrufen, erscheint ein Requester, der Sie nach dem Namen des Programms fragt. Klicken Sie dann einmal in das Textgadget

unter der Zeile *Save program as:* und geben Sie den Namen ein, unter dem das Programm gesichert werden soll. Wenn es in einer bestimmten Schublade abgelegt werden soll, müssen Sie hier den vollen Pfadnamen eintragen (also zum Beispiel *Extras1.3D:BasicDemos/test*, um das Programm auf der Diskette *Extras1.3D* in der Schublade *BasicDemos* unter dem Namen *test* abzuspeichern). Sobald Sie auf [Return] drücken oder den *OK*-Knopf anklicken, wird das Programm dann unter dem angegebenen Namen gesichert. Wenn Sie später – nach Änderungen – das Programm erneut sichern wollen, genügt es, den Befehl *Save* auszuwählen. Ein Name wird dann nicht noch einmal abgefragt.

Wollen Sie das Programm jedoch unter einem anderen Namen abspeichern, so können Sie den Befehl *Save As* verwenden. Er verhält sich genauso wie der erste *Save*-Befehl, erlaubt es Ihnen aber, auch ein Programm, das schon einen Namen hat, noch einmal unter einem anderen zu sichern.

### 13.4.6 Programme einlesen

Das Einlesen eines Basic-Programms geht ähnlich vonstatten wie das Abspeichern. Wählen Sie dazu den Befehl *Open* aus dem Menü *Run*. Wenn Sie das aktuelle Programm zuvor noch nicht gesichert hatten, erscheint nun eine Warnung (*Current program is not saved, Do you want to save it, before proceeding?*) in Form eines Requesters mit drei Knöpfen. Klicken Sie in diesem Requester auf den Knopf *Yes*, wird das aktuelle Programm gesichert, bevor das neue geladen wird. Klicken Sie auf *No*, werden die Änderungen am aktuellen Programm verworfen. Und wenn Sie auf *Cancel* klicken, wird der Befehl abgebrochen.

Wenn Sie *Yes* oder *No* angeklickt haben oder kein Programm zu sichern war, erscheint dann ein Requester, in dem Sie den Namen des zu ladenden Programms eingeben müssen. Klicken Sie dazu das Textgadget unter der Zeile *Name of program to load:* an und tippen Sie den Pfadnamen des gewünschten Programms ein. Sobald Sie auf [Return] drücken oder den *OK*-Knopf anklicken, wird dieses Programm geladen. Falls es kein Programm dieses Namens gibt, erscheint eine Fehlermeldung (*File not found*).

Sowohl bei *Save* wie auch bei *Open* können Sie sich die Angabe des vollen Pfadnamens sparen, wenn das Programm, das abgespeichert oder geladen werden soll, im aktuellen Directory liegt. Beim Start entspricht das aktuelle Directory der Schublade, in der Sie den Doppelklick gemacht haben, um Amiga-Basic zu starten. Das ist die Schublade mit dem Piktogramm Amiga-Basic, oder die mit dem Piktogramm eines Programms, wenn Sie Amiga-Basic durch einen Doppelklick auf ein Basic-Programm gestartet haben. Sie können aber auch innerhalb von Amiga-Basic das aktuelle Directory ändern. Klicken Sie dazu einmal (während kein Programm läuft) in das Output-Fenster und geben Sie den Befehl *chdir* und dahinter den Namen des gewünschten Directory in Anführungsstrichen ein. Der folgende Befehl macht zum Beispiel das Directory *BasicDemos* auf der Diskette *Extras1.3D* zum aktuellen Directory:

```
chdir "extras1.3d:basicdemos"
```

Programme aus diesem Directory (dieser Schublade) können Sie nun laden, indem Sie nur ihren Namen (und nicht den vollständigen Pfadnamen) in den entsprechenden Requester eintragen und [Return] drücken. Welche Programme in diesem Directory zur Wahl stehen, können Sie feststellen, indem Sie im Output-Fenster den folgenden Befehl eingeben:

```
files
```

Daraufhin erscheint eine Liste der Dateien in diesem Directory. Wenn Sie Lust darauf verspüren, ein richtig großes Programm kennenzulernen, können Sie ja nun einmal (mit *Open*) ein paar dieser Programme aus der Schublade *BasicDemos* der Extras-Diskette laden, laufen lassen und sich im Listing-Fenster anschauen. Sehr interessant sind zum Beispiel die Programme *Dreier*, *Music* und *Screen*.

## 13.5 Grafik mit Amiga-Basic

Nach diesem kleinen Schnellkurs über den Umgang mit Amiga-Basic sollten Sie nun – sofern Sie Lust dazu verspüren – aber auch anhand einiger kleiner Beispielprogramme ein paar Basic-Befehle ausprobieren. Den Anfang machen in diesem Kapitel die Grafik-Befehle.

### 13.5.1 Pixel

Der einfachste Grafik-Befehl ist PSET (*Pixel Set*). PSET färbt genau einen Bildschirmpunkt (ein Pixel) auf dem Bildschirm ein. Er benötigt zwei Parameter, eine Koordinatenangabe und eine Zahl, die besagt, welche Farbe dieser Punkt bekommen soll. Koordinatenangaben sind dabei immer zwei Zahlen (eine X- und eine Y-Koordinate), die von einem Komma getrennt in runden Klammern stehen. PSET (100,100),3 färbt im Output-Fenster zum Beispiel den Punkt an der Position (100,100) mit der Farbe 3 (das ist Orange, falls Sie die Workbench-Farben nicht verändert haben). Das folgende kleine Programm füllt das ganze Output-Fenster mit einer zufällig verteilten Wolke solcher farbiger Pixel an.

```
start:
x = RND * 630
y = RND * 200
c = RND * 3 + 0.4
PSET (x,y),c
GOTO start
```

Der Aufbau des Programms ist denkbar simpel. Es verwendet vor allem die Funktion RND, die eine Zufallszahl zwischen 0 und 1 ergibt. Durch Multiplikation wird dieser Bereich zwischen 0 und 1 dann auf die Bereiche von 0 bis 630, von 0 bis 200 und von 0,4 bis 3,4 »gestreckt«. Diese gestreckten Zufallszahlen werden dann Variablen zugewiesen und schließlich in der PSET-Anweisung dazu verwendet, um einen Punkt im Output-Fenster an einer zufälligen Position mit einer zufälligen Farbe zu versehen.

Die Farbe *c* liegt dazu in einem Bereich zwischen 0,4 und 3,4. Diese krummen Zahlen rundet der PSET-Befehl selbst auf die vier Zahlen 0, 1, 2 und 3. Diese entsprechen den vier Farben, die auf dem Workbench-Screen zur Verfügung stehen. Die Farbwahl und das Setzen des Punktes ist in eine Endlos-Schleife gepackt. Diese müssen Sie mit dem Befehl *Stop* abbrechen, wenn Sie genug Punkte vor Augen haben. Das folgende Programm leistet übrigens genau dasselbe, benutzt dazu aber statt des altmodischen GOTOs die elegante WHILE-Schleife.

```
WHILE 1=1
  x = RND * 630
  y = RND * 200
  c = RND * 3 + 0.4
  PSET (x,y),c
WEND
```

Diese WHILE-Schleife zwischen WHILE und WEND läuft so lange, wie die Bedingung hinter WHILE zutrifft (genauer gesagt läuft sie so lange, wie dieser Ausdruck ein Ergebnis ungleich Null ergibt). Und die Bedingung 1=1 trifft immer zu. Dies ist noch ein recht triviales Beispiel. Aber Sie sehen vielleicht schon, daß WHILE-Schleifen zusammen mit der Einrückung ein Programm übersichtlicher machen können.

### 13.5.2 Linien

Eine ganz einfache und sehr interessante Abwandlung des Programms aus dem letzten Abschnitt macht die LINE-Anweisung möglich. Sie arbeitet sehr ähnlich wie die PSET-Anweisung, benötigt vor der Angabe einer Farbe allerdings zwei Koordinaten, die einen Anfangs- und einen Endpunkt bestimmen, zwischen denen dann eine dünne Linie in der angegebenen Farbe gezogen wird. Das folgende Programm enthält die nötigen Änderungen, um statt Punkte nun farbige Linien auf dem Bildschirm erscheinen zu lassen.

```
WHILE INKEY$=""
  x1 = RND * 630
  x2 = RND * 630
  y1 = RND * 200
  y2 = RND * 200
  c = RND * 3 + 0.4
  LINE (x1,y1)-(x2,y2),c
WEND
```

(Beachten Sie bitte, daß die beiden Koordinatenangaben in der LINE-Anweisung nicht durch ein Komma, sondern durch einen Bindestrich »–« voneinander getrennt sind!) Diese Programme brauchen Sie nicht mit *Stop* abzubrechen. Es genügt, einmal in das Output-Fenster zu klicken und dann auf eine beliebige Taste zu drücken. Diesen Buchstaben liefert die INKEY$-Funktion dann als Ergebnis zurück. INKEY$ ist dann nicht mehr gleich "" (dem leeren String) und die WHILE-Schleife bricht ab. Ansonsten gibt es an diesem Programm nicht viel Neues. Statt einer Koordinatenangabe erzeugen die Zufallsanweisungen hier nun zwei, da die LINE-Anweisung zwei davon braucht.

### 13.5.3 Mondrian

Hinter der LINE-Anweisung verbirgt sich jedoch mehr, als auf den ersten Blick ersichtlich ist. Sie haben bislang nur die einfachste Form mit zwei Koordinaten und einer Farbangabe kennengelernt. Dahinter kann jedoch noch ein Parameter kommen. Dieser kann *b* oder *bf* lauten und ändert die Wirkung der LINE-Anweisung total. Statt eine Linie zu ziehen, zeichnet diese Anweisung dann ein Rechteck, dessen obere linke und untere rechte Ecke die

beiden Koordinaten angeben. Verwenden Sie den Parameter *b* hinter der Farbangabe, wird dieses Rechteck nur mit einer dünnen Linie gezogen. Verwenden Sie jedoch *bf*, wird das Rechteck mit der angegebenen Farbe gefüllt. Die folgenden beiden Programme benutzen diese Optionen der LINE-Anweisung und erzeugen damit Bilder, die etwas an die Werke des holländischen Künstlers Mondrian erinnern:

```
WHILE INKEY$=""
  x1 = RND * 630
  x2 = RND * 630
  y1 = RND * 200
  y2 = RND * 200
  c = RND * 3 + 0.4
  LINE (x1,y1)-(x2,y2),c, b
WEND
WHILE INKEY$=""
  x1 = RND * 630
  x2 = RND * 630
  y1 = RND * 200
  y2 = RND * 200
  c = RND * 3 + 0.4
  LINE (x1,y1)-(x2,y2),c, bf
WEND
```

Falls Ihnen die dabei auftauchenden Rechtecke zu groß sind, können Sie eine weitere Möglichkeit der LINE-Anweisung verwenden, um deren Abmessungen zu begrenzen. Wenn Sie nämlich vor das zweite Koordinatenpaar (x2,y2) das Schlüsselwort STEP schreiben, so geben *x2* und *y2* nicht mehr die Position des zweiten Eckpunkts, sondern die Abstände des zweiten Eckpunkts vom ersten an. Wenn Sie dann noch dafür sorgen, daß *x2* und *y2* klein bleiben, bleiben auch die dabei entstehenden Rechtecke klein, wie das folgende Programm demonstriert:

```
WHILE INKEY$=""
  x1 = RND * 630
  x2 = RND * 60
  y1 = RND * 200
  y2 = RND * 30
  c = RND * 3 + 0.4
  LINE (x1,y1)-STEP (x2,y2),c, bf
WEND
```

### 13.5.4 Eigene Fenster und Screens

Obwohl die bisherigen Beispielprogramme schon recht nette Bildchen zustande brachten, fehlt ihnen noch der besondere Pfiff. Das liegt einfach daran, daß die Programme zu wenig Farben verwenden. Der Workbench-Screen bietet aber nur vier Farben und das reicht einfach nicht für spektakuläre Farbgrafiken. Deshalb bleibt uns nichts anderes übrig, als einen neuen Screen zu erzeugen und darin ein neues Output-Fenster zu öffnen. Dazu werden

die Anweisungen SCREEN und WINDOW benötigt, die beide recht kompliziert sind und viele Parameter benötigen. Zunächst aber das Programm selbst (und danach die Erläuterungen):

```
SCREEN 1, 320, 256, 5, 1
WINDOW 3, "Mondrian", (0,0)-(310,240),2, 1
WINDOW OUTPUT 3
WHILE INKEY$=""
  x1 = RND * 310
  x2 = RND * 310
  y1 = RND * 230
  y2 = RND * 230
  c = RND * 31 + 0.4
  LINE (x1,y1)-(x2,y2), c, bf
WEND
WINDOW CLOSE  3
WINDOW OUTPUT 1
SCREEN CLOSE  1
```

Am Anfang des Programms wird ein neuer Screen erzeugt. Die Parameter der SCREEN-Anweisung (in dieser Reihenfolge) bedeuten: Nummer des Screens, Breite, Höhe, Tiefe und Modus. Die Nummer wird nur dazu benötigt, wenn man diesen Screen später ansprechen will. Die Bedeutung von Breite und Höhe dürfte klar sein. Die Tiefe bestimmt die Anzahl der möglichen Farben, wie es in Kapitel 12 beschrieben wurde. In diesem Fall ist die Tiefe gleich 5 und es stehen damit 32 Farben zur Verfügung. Der Modus-Parameter bestimmt den Grafik-Modus des Screens. Ein Modus von 1 bedeutet einen Screen niedriger horizontaler Auflösung (320 Punkte) ohne Interlace.

Die zweite Anweisung erzeugt ein neues Fenster. Die Parameter der WINDOW-Anweisung (in dieser Reihenfolge) bedeuten: Fenster-Nummer, Titel, obere linke Ecke, untere rechte Ecke, Fenstertyp und Screen. Die Fenster-Nummer dient wie beim Screen zur Identifikation des Fensters und der Titel ist ein Text, der in der Titelleiste erscheint. Der Fenstertyp-Parameter bestimmt, über welche Eigenschaften das Fenster verfügt. Ein Wert von 2 besagt, daß man dieses Fenster verschieben kann. (Mehr zu den anderen möglichen Werten von Modus finden sie im Amiga-Basic-Handbuch.) Der letzte Parameter, Screen, bestimmt, in welchem Screen das Fenster auftaucht. In diesem Fall ist es der eben neu erzeugte Screen Nummer 1.

Mit WINDOW OUTPUT 3 wird dann das eben erzeugte Fenster Nummer 3 zum aktuellen Output-Fenster, in das Print-Befehle und Grafik-Anweisungen gehen. Hierdurch kommt auch der neue Screen, in dem das Fenster liegt, nach oben. Die darauffolgende Schleife kennen Sie ja bereits aus dem Programm *Mondrian* im vorigen Abschnitt. Sie können sie wie üblich durch Betätigung einer beliebigen Taste abbrechen.

Ist die WHILE-Schleife beendet, beginnt das Programm, ordnungsgemäß »aufzuräumen«. Hierzu wird das neue Fenster geschlossen (WINDOW CLOSE 3), dann das normale Fenster auf dem Workbench-Screen wieder zum Output-Fenster gemacht und schließlich auch der Screen mit den 32 Farben wieder gelöscht. Das ist alles nicht unbedingt nötig – aber guter Programmierstil.

# 13.6 Ereignisreiche Programme

Die bislang vorgestellten Beispielprogramme haben – bei aller Farbenpracht – aber nur etwas auf dem Bildschirm gemalt und nicht auf Ihre Eingaben reagiert. Das soll nun anders werden. Und wie sich das für den Amiga gehört, werden wir uns im folgenden nicht mit trivialen Anweisungen wie READ beschäftigen, die es in anderen Basics für andere Computer auch gibt, sondern um die Eingabe mit der Maus und über Menüs.

## 13.6.1 Das einfachste Malprogramm der Welt

Damit Sie ein Gefühl für die Benutzung der Maus von einem Basic-Programm aus bekommen, möchte ich Ihnen nun das einfachste Malprogramm der Welt vorstellen. Sie können damit – zusammen mit etwas Geschick – ohne weiteres kleine Bilder im Outputfenster malen. Trotzdem umfaßt es nur drei Zeilen:

```
WHILE INKEY$=""
  IF MOUSE(0) <> 0 THEN PSET(MOUSE(1),MOUSE(2)),1
WEND
```

Das Programm ist sehr einfach aufgebaut und verwendet fast nur die Funktion MOUSE, die Informationen über den Zustand der Maus liefert. Zunächst wird MOUSE(0) aufgerufen. MOUSE(0) liefert als Ergebnis 0, wenn die linke Maustaste nicht gedrückt ist. Ist die Maustaste gedrückt, oder war sie seit dem letzten Aufruf von MOUSE(0) einmal gedrückt, ergibt MOUSE(0) verschiedene andere Ergebnisse (über die Sie im Amiga-Basic-Handbuch mehr erfahren können). In diesem Fall wird die Position der Maus erfragt und ein weißer Punkt (Farbe Nummer 1) dorthin gesetzt. MOUSE(1) ergibt dazu die horizontale (X-) Position der Maus und MOUSE(2) die vertikale (Y-) Position.

Solange die Maustaste nicht gedrückt ist, tut das Programm nichts – es wartet. Dabei durchläuft es allerdings andauernd die WHILE-Schleife. Das ist eine sehr uneffiziente Nutzung des Computers, der in dieser Wartezeit eigentlich schon wieder andere Aufgaben erledigen könnte. Effizienter wäre es, wenn das Programm ständig irgendwelche wichtigen (und langwierigen) Aufgaben erledigen könnte und sich nur bei Bedarf um die Maus kümmern würde. Dieser »Bedarf« läßt sich relativ leicht daran feststellen, daß die Maustaste gedrückt wird. Es wäre nun aber sehr lästig für ein Programm, wenn es andauernd überprüfen müßte, ob gerade die Maustaste gedrückt ist, während es sich eigentlich um andere Dinge kümmert.

## 13.6.2 Die ON MOUSE GOSUB-Anweisung

Um solche pausenlosen Abfragen zu vermeiden, wurde die ON...GOSUB-Anweisung geschaffen. Im folgenden Programm kommt zum Beispiel eine Zeile vor, die »ON MOUSE GOSUB mousehandler« lautet. Diese Anweisung bedeutet, daß das Programm immer, wenn die linke Maustaste gedrückt wird, automatisch zur Subroutine *mousehandler* verzweigen soll. Dabei spielt es keine Rolle, was dieses Programm in dem Moment tut, wenn die Maustaste gedrückt wird. Amiga-Basic sichert dann den Zustand des Programms in einem dafür vorgesehenen Puffer und springt zu dieser Subroutine. Ist diese beendet, geht es im Programm automatisch dort wieder weiter, wo es unterbrochen wurde. Eine solche ON MOUSE GOSUB-Anweisung muß allerdings noch durch eine MOUSE ON-Anweisung

»aktiviert« werden, sonst tut sie gar nichts. Genauso kann man die ON MOUSE GOSUB-Anweisung auch später wieder deaktivieren, indem man die Anweisung MOUSE OFF ausführt.

Nun aber genug der Vorrede. Das folgende Programm demonstriert eine sehr simple Anwendung der ON MOUSE GOSUB-Anweisung. Im Hauptprogramm wird dazu in einer Schleife immer wieder derselbe Text ausgegeben. Sobald Sie aber auf die Maustaste drücken, verzweigt das Programm dank der ON MOUSE GOSUB-Anweisung zur Subroutine *mousehandler*.

```
ON MOUSE GOSUB mousehandler
MOUSE ON
WHILE INKEY$=""
  PRINT "Hallo Ihr da!"
WEND
END
mousehandler:
  PRINT "Und das war ein Mausklick <<<<<<———"
RETURN
```

Probieren Sie das Programm am besten einmal aus und schalten Sie dabei die Trace-Anzeige mit dem Befehl *Trace On* ein. Sie können dann gut verfolgen, wie immer dann, wenn Sie die Maustaste drücken, nach *mousehandler* verzweigt wird. Beachten Sie bitte, daß Sie auch dieses Programm wieder mit dem Betätigen einer beliebigen Taste verlassen können. Hinter WEND stößt das Programm dann auf die END-Anweisung, die das Programm stoppt (sonst würde es fälschlicherweise in die Subroutine *mousehandler* hineinlaufen).

Das folgende Programm zeigt schon eine etwas komplexere Anwendung der ON MOUSE GOSUB-Anweisung, die demonstriert, was Sie unter Umständen beachten müssen, wenn Sie auf einen Mausklick nicht nur mit einer simplen Meldung reagieren wollen.

```
WINDOW 2, "Maler", (40,40)-(500,180)
WINDOW OUTPUT 1
ON MOUSE GOSUB mousehandler
MOUSE ON
WHILE INKEY$=""
  PRINT "Hallo Ihr da!"
WEND
END
mousehandler:
  WINDOW OUTPUT 2
  WHILE MOUSE(0) <> 0
    PSET(MOUSE(1),MOUSE(2)),1
  WEND
  WINDOW OUTPUT 1
RETURN
```

Zu Anfang dieses Programms wird ein neues Fenster namens »Maler« geöffnet. Wegen der Anweisung WINDOW OUTPUT 1 gehen die Ausgaben in der folgenden WHILE-Schleife aber nicht in dieses, sondern in das normale Output-Fenster von Amiga-Basic. Wie schon im ersten Programm zu diesem Thema, wird bei einem Mausklick wieder nach *mousehand-*

*ler* verzweigt. Diesmal verbirgt sich aber hinter dieser Subroutine das schon bekannte kleine Mal-Programm. Solange die Maustaste gedrückt ist, stoppt deshalb die Ausgabe der Texte und Sie können im Fenster »Maler« malen. Dazu muß *mousehandler* aber zunächst dieses Fenster mit WINDOW OUTPUT 2 zum Output-Fenster machen. Vor Abschluß der Subroutine, wenn die Maustaste also losgelassen wurde, wird der alte Zustand mit WINDOW OUTPUT 1 wiederhergestellt.

Dieses Programm führt schon recht deutlich die Möglichkeiten (und Probleme) der ON MOUSE GOSUB-Anweisung vor. Hier laufen eigentlich zwei Programme in zwei Fenstern nebeneinander her. Das eine läuft nach dem Programmstart zunächst einmal an, wird aber bei einem Mausklick sofort von dem anderen abgelöst. Auf ähnliche Weise können Programme auch auf andere Ereignisse reagieren. Hierfür sorgen Anweisungen wie zum Beispiel ON MENU, ON TIMER und ON COLLISION, die auf Menüauswahl, auf den Ablauf einer gewissen Zeitspanne und auf den Zusammenstoß zweier Sprites reagieren.

### 13.6.3 Eigene Menüs in Basic-Programmen verwenden

Die ON-MENU-Anweisung können Sie nun auch sofort ausprobieren. Sie dient der Reaktion auf eine Menüauswahl, die vom Anwender des Programms vorgenommen wurde. Menüs werden von Basic-Programmen fast genauso wie Mausklicks gehandhabt. Hauptunterschied zur Mausbehandlung ist, daß Menüs zunächst einmal erzeugt werden müssen. Hierzu dient die MENU-Anweisung. Sie hat drei Parameter: die Nummer des zu erzeugenden Menüs, die Nummer des zu erzeugenden Menübefehls, eine Zahl (0 oder 1), die besagt, ob dieser Menübefehl wählbar sein soll und den Text des Menübefehls selbst. Ist die Nummer des Menübefehls gleich Null, ist damit der Menütitel gemeint, es wird also ein neues Menü erzeugt.

Sie können diese Anweisung aber nicht nur zum Erzeugen von Menüs und Menübefehlen verwenden. Rufen Sie MENU mit Parametern auf, die ein schon existierendes Menü oder einen Menübefehl betreffen, so können Sie dieses Menü beziehungsweise diesen Menübefehl ändern. Menübefehle können so nachträglich ihren Text ändern, auswählbar oder unauswählbar werden und so weiter.

Das folgende Programm zeigt eine sehr triviale Anwendung eigener Menüs. Es erzeugt zwei Menüs und ermöglicht die Auswahl aus diesen mit der ON-MENU-GOSUB-Anweisung. Jede Menüauswahl wird durch Ausgabe eines entsprechenden Textes bestätigt. Der Befehl *Beenden* im Menü *Project* schließlich tut genau das, was sein Name sagt: er stoppt das Programm.

```
MENU 1, 0, 1, "Project  "
MENU 1, 2, 1, "Beenden  "
MENU 2, 0, 1, "Getränke "
MENU 2, 1, 1, "Cola "
MENU 2, 2, 1, "Aqua Rülps "
MENU 2, 3, 1, "O-Saft "
MENU 2, 4, 0, "————"
MENU 2, 5, 1, "Milch "
ON MENU GOSUB MenuHandler
MENU ON
```

```
ende = 0
WHILE ende = 0
WEND
MENU RESET
END
MenuHandler:
dasMenu      =MENU(0)
derBefehl  =MENU(1)
IF      dasMenu = 1 THEN
  IF derBefehl = 1 THEN
    ende = 1
  END IF
ELSEIF dasMenu = 2 THEN
  IF      derBefehl = 1 THEN
    PRINT "Cola"
  ELSEIF derBefehl = 2 THEN
    PRINT "Aqua Rülps"
  ELSEIF derBefehl = 3 THEN
    PRINT "O-Saft"
  ELSEIF derBefehl = 5 THEN
    PRINT "Milch"
  END IF
END IF
RETURN
```

Am Anfang dieses Programms werden die Menüs zunächst aufgebaut, dann werden sämtliche Menüauswahlen mit ON MENU... und MENU ON an die Subroutine MenuHandler geschickt. Das Programm geht dann in eine Leerschleife, die erst dann verlassen wird, wenn die Variable end ungleich Null ist. Sobald das der Fall ist, wird die normale Menüleiste von Amiga-Basic mit MENU RESET wiederhergestellt und das Programm mit END gestoppt. Falls es Ihnen einmal passieren sollte, daß das Programm (vielleicht wegen eines Tippfehlers) abbricht, können Sie die Anweisung MENU RESET auch direkt im Output-Fenster eingeben. Diese Anweisung ist nötig, damit die neu erzeugten Menüs verschwinden und die Basic-Menüs wieder auftauchen.

Nach der bisherigen Beschreibung tut das Programm in der Leerschleife (WHILE ende = 0 ... WEND) scheinbar gar nichts. Das täuscht aber, denn immer, wenn Sie einen Menübefehl auswählen, wird die Subroutine MenuHandler aufgerufen. Diese stellt dann zunächst fest, welcher Befehl in welchem Menü ausgewählt wurde. Dazu wird die Funktion MENU (nicht zu verwechseln mit der Anweisung MENU) aufgerufen. MENU(0) ergibt die Nummer des gewählten Menüs und MENU(1) die Nummer des gewählten Befehls in diesem Menü. Diese Nummern sind dabei dieselben Nummern, die bei der Erzeugung dieses Menüs (mit der MENU-Anweisung) verwendet wurden. Mit den beiden Nummern, die in den Variablen *derBefehl* und *dasMenu* gespeichert werden, geht es dann in eine große Fallunterscheidung hinein. Diese Fallunterscheidung besteht aus einer Reihe von ineinander verschachtelten IF-ELSEIF-ENDIF-Anweisungen. Sie lernen damit auch gleich die interessante Möglichkeit kennen, eine IF-Anweisung auf mehrere Zeilen zu verteilen, die auch nicht jedes Basic zu bieten hat. Ich habe versucht, die Schachtelung durch Einrücken der Zeilen so deutlich zu machen, daß Sie eigentlich sofort verstehen müßten, wie eine solche verschachtelte IF-ELSEIF-ENDIF-Anweisung funktioniert.

Die äußere IF-Anweisung unterscheidet nur zwei Fälle, da es nur zwei Menüs gibt. Sie endet – wie jede mehrzeilige IF-Anweisung – in der letzten Programmzeile mit einem END IF. Sie besitzt zwei »Zweige«, die den beiden Menüs entsprechen, und jeder dieser Zweige enthält wieder eine IF-ELSEIF-ENDIF-Anweisung, die nach den einzelnen Befehlen im jeweiligen Menü unterscheidet. Zu jedem Menübefehl wird dann eine PRINT-Anweisung aufgerufen, die meldet, welche Wahl getroffen wurde. Nur, wenn *derBefehl* und *dasMenu* beide gleich 1 sind (der Befehl Beenden wurde gewählt) wird die Variable *ende* auf 1 gesetzt. Nach Beendigung der Subroutine MenuHandler trifft die Bedingung der WHILE-Schleife dann nicht mehr zu und das Programm wird beendet.

So, das war eigentlich schon alles, was Sie wissen müssen, um in Ihren eigenen Programmen Menüs zu verwenden. Vergessen Sie nur hinter ON MENU GOSUB nie die MENU ON-Anweisung (sonst funktioniert ON MENU GOSUB nicht) und gewöhnen Sie sich es an, am Ende Ihrer Programme immer mit MENU RESET die normale Basic-Menüleiste wiederherzustellen!

## 13.7 Klangerzeugung mit Amiga-Basic

Nachdem Sie die grafischen Möglichkeiten von Amiga-Basic jetzt schon (andeutungsweise) kennengelernt haben, möchte ich Ihnen aber auch die wichtigsten Basic-Befehle vorstellen, die Sie zur Klangerzeugung benötigen. Zunächst geht es dabei in diesem Abschnitt um die Erzeugung beliebiger »Töne« und dann um ganz spezielle Töne: die synthetische Sprache.

### 13.7.1 Ein paar schräge Töne

Die Klangerzeugung mit Amiga-Basic kreist um einen einzigen Befehl: SOUND. Mit diesem Befehl können Sie einen Ton bestimmter Tonhöhe, Lautstärke und Dauer aus einem der vier Tonkanäle, über die der Amiga 500 verfügt, ausgeben lassen. Dazu benötigt SOUND vier Parameter, Frequenz, Dauer, Lautstärke und Kanal. Die Tonhöhe (Frequenz) wird in Hz (Hertz) gemessen und die Dauer des Klangs in Takten (wobei 18,2 Takte laut Handbuch eine Sekunde dauern). Die Frequenz darf zwischen 20 und 15 000 liegen und die Dauer zwischen 0 und 77. Die Lautstärke muß als Zahl zwischen 0 (stumm) und 255 (maximale Lautstärke) angegeben werden, und als Kanal stehen die Zahlen 0 bis 3 zur Verfügung. Lautstärke und Kanal sind dabei optional, das heißt, Sie können sie fortfallen lassen und Amiga-Basic verwendet dann Standardwerte (maximale Lautstärke und Kanal Nummer 0).

Geben Sie testweise einmal die folgende SOUND-Anweisung im Output-Fenster ein und drücken Sie die [Return]-Taste:

```
SOUND 440, 10
```

Gleichzeitig mit der *Ok*-Meldung müßte dann ein Ton erklingen (es ist der Kammerton A mit einer Frequenz von 440 Hz). Damit Sie etwas hören, müssen Sie allerdings die beiden Sound-Ausgänge an der Rückseite des Amiga 500 mit einem Paar Lautsprecher oder über ein spezielles Kabel mit Ihrem Monitor verbunden haben.

Wenn Sie Lust haben, können Sie nun auch einmal andere Werte für die Parameter Frequenz und Tondauer ausprobieren. Achten Sie dabei auch darauf, daß der einmal »angestoßene« Ton auch dann noch »nachklingt«, wenn die Ok-Meldung bereits erschienen ist und Sie schon den nächsten Befehl eingeben können. (Am auffälligsten ist das natürlich, wenn die Tondauer länger wird.) Dies läßt eine erste Ahnung davon aufkommen, daß die Hardware (und Systemsoftware) einen Ton selbständig erzeugt und das Basic-Programm dazu nicht mehr benötigt wird, wenn es einmal den Ton angestoßen hat. Um das noch etwas deutlicher zu zeigen, geben Sie bitte einmal das folgende Programm ein:

```
PRINT "Start !"
FOR i = 1 TO 10
  SOUND 4000, 10, 255
  SOUND 200, 10, 128
NEXT i
PRINT "Fertig !"
```

Dieses Programm läßt zwanzig Töne erklingen. Ein hoher leiserer und ein tieferer lauter Ton wechseln sich dabei immer ab. (Wie gut sie klingen, hängt ein bißchen von dem Lautsprecher oder Monitor ab, den Sie angeschlossen haben.) Erstaunlich an diesem Programm ist aber vor allem, daß ein Großteil der Töne erst erklingt, wenn das Programm längst beendet ist. Es ist nämlich so, daß nicht einmal zum »Anstoßen« eines Tons ein Programm laufen muß. Jeder SOUND-Befehl wird zunächst einmal in einer Warteschlange gespeichert. Für jeden der vier Tonkanäle gibt es eine eigene Warteschlange für diesen Zweck. Sobald ein Tonkanal nichts mehr zu tun hat (also der vorige Ton verklungen ist) überprüft die Systemsoftware des Amiga, ob die zugehörige Warteschlange leer ist. Ist sie das nicht, wird der älteste SOUND-Befehl, der schon am längsten wartet, aus der Warteschlange genommen und der entsprechende Ton erklingt. Ist dieser verklungen, ist der nächste auf diesen Kanal wartende SOUND-Befehl an der Reihe. Das alles läuft völlig unabhängig davon ab, was das laufende Basic-Programm gerade tut. Es muß – wie oben demonstriert – noch nicht einmal ein aktives Basic-Programm geben.

Das folgende Programm demonstriert diese Eigenschaft der Sounderzeugung auf dem Amiga noch eindrucksvoller. Es benutzt dazu die Befehle SOUND WAIT, der verhindert, daß SOUND-Befehle aus einer Warteschlange genommen werden, und SOUND RESUME, der die Soundmaschinerie wieder in Gang setzt.

```
SOUND WAIT
PRINT "Start !"
FOR i = 1 TO 5
  SOUND 4000, 10, 255
  SOUND 200, 10, 128
NEXT i
SOUND RESUME
PRINT "Fertig !"
```

Die vier Warteschlangen können leider aber nicht beliebig lang werden. Sie können also nicht SOUND WAIT aufrufen, ein beliebig langes Musikstück in die vier Warteschlangen legen und es danach mit SOUND RESUME erklingen lassen, während Sie längst an einem ganz anderen Programm arbeiten. Falls die Warteschlange überläuft, erscheint ein Alert, der Ihnen meldet, daß kein Speicher mehr für diesen Zweck da ist (*Out of heap space. Press left*

*mouse button, to continue*). Um diesen Fehler zu vermeiden, wurde übrigens in dem obigen Programm die obere Grenze der FOR-Schleife auf 5 heruntergesetzt.

Apropos Musik: Ich habe Ihnen in diesem Kapitel ja nur gezeigt, wie man recht schräge Töne erzeugen kann. Zu mehr reicht mein musikalisches Talent nicht aus. Wenn Sie einmal sehen wollen, wie man richtige Musik mit dem Amiga erzeugen kann, schauen Sie sich doch einfach einmal das Beispielprogramm *Music* in der Schublade *BasicDemos* auf der Extras-Diskette an. Es zeigt auf eindrucksvolle Weise, wie man ein Programm eine komplizierte mehrstimmige Melodie spielen lassen kann und dabei gleichzeitig noch eine hübsche Grafik auf den Bildschirm bringt.

### 13.7.2 Sprachsynthese

Nun möchte ich Ihnen aber auch noch den zweiten wichtigen Aspekt der Sounderzeugung auf dem Amiga vorstellen, die Erzeugung synthetischer Sprache (Sprachsynthese). Dies ist von Amiga-Basic aus so lächerlich einfach, daß ich mich fast schäme, das folgende Programm vorzustellen. Es kann fast genausoviel wie das Werkzeug *Say*, das in Kapitel 5 vorgestellt wurde. Um dieses Programm auszuprobieren, dürfen Sie aber auf keinen Fall die Sprachbibliotheken von Ihrer Workbench-Diskette entfernt haben, wie es im Kapitel 8 beschrieben wurde!

```
Sag$ = "X"
WHILE Sag$<>""
  LINE INPUT "Was soll ich sagen? ";Sag$
  Phonem$ = TRANSLATE$(Sag$)
  PRINT "Phoneme  : ";Phonem$
  SAY(Phonem$)
WEND
```

Das Programm ist in eine große WHILE-Schleife eingebettet, die abbricht, sobald die Variable *Sag$* leer ist. (Deshalb wird *Sag$* zu Anfang auf »X« gesetzt). In der WHILE-Schleife wird jeweils eine ganze Zeile (bis [Return]) über die Tastatur eingelesen und in *Sag$* gespeichert. Dieser Text wird dann mit der Funktion TRANSLATE$ in Phoneme umgewandelt. Und eine Liste von Phonemen kann man mit SAY aussprechen lassen – was das Programm dann auch tut. (Der Zusammenhang zwischen Sprache und Phonemen ist bereits in Kapitel 11 erläutert worden.) Das klingt wahrscheinlich nicht besonders schön. Wie Sie ja wissen, versucht der Amiga jeden Text zunächst einmal auszusprechen, als wären es englische Wörter.

Sie können das nur umgehen, indem Sie die SAY-Funktion direkt ohne Umweg über TRANSLATE verwenden und die Phoneme, die ausgesprochen werden sollen, selbst (zum Beispiel anhand eines Wörterbuches) ermitteln. Eine andere Lösung ist es, selbst ein Programm zu entwickeln, das deutsche Sprache in die richtigen Phoneme umwandelt. Das ist allerdings nicht ganz so einfach. Glücklicherweise hat Ihnen aber schon jemand diese Arbeit abgenommen. Auf der Extras-Diskette befindet sich in der Schublade *BasicDemos* ein Programm namens *SpeechD*, das unter anderem auch deutsche Texte (einigermaßen) gut aussprechen kann. Dieses Programm kann auch sonst einiges mehr als das obige Mini-Sprachprogramm. Es erlaubt Ihnen unter anderem auch, Sprechgeschwindigkeit und Tonhöhe einzustellen. Wenn Sie also wissen wollen, wie man so etwas in einem Basic-Programm macht, schauen Sie sich *SpeechD* ruhig einmal im Listing-Fenster an.

## 13.8 Libraries: Amiga-Basic ohne Grenzen

Obwohl Amiga-Basic bereits eine Vielzahl von Befehlen enthält, kann es doch immer wieder vorkommen, daß zusätzliche Fähigkeiten benötigt werden, die in Basic nicht realisierbar sind. Von Basic aus können zum Beispiel keine Requester und Gadgets erzeugt und benutzt werden. Fortgeschrittene Programmierer haben aber dennoch die Möglichkeit, ohne allzu große Mühe auch diese Einschränkungen zu umgehen. Diese Möglichkeiten sind aber »mit Vorsicht zu genießen«, da einige Schritte dabei recht fehleranfällig sind.

Funktionen, die diese fehlenden Fähigkeiten realisieren, befinden sich nämlich zum großen Teil bereits in den Bibliotheks-Dateien, aus denen die Systemsoftware des Amiga zum Teil besteht. Und Amiga-Basic besitzt eine recht elegante Konstruktion, mit der es möglich ist, solche Funktionen auch in einem Basic-Programm zu verwenden, als wären sie in Basic geschrieben und würden sich im selben Programm befinden. Mit dem LIBRARY-Befehl können eine Reihe solcher Bibliotheks-Dateien angegeben werden, die Amiga-Basic durchsucht, um diese Befehle zu finden. Wird danach in einem Programm ein Befehl verwendet, der nicht im Programm selbst als Subprogramm definiert ist, versucht Amiga-Basic, ihn in einer Bibliothek (engl. *library*) zu finden.

Solche Bibliotheken müssen aber nicht zur Amiga-Systemsoftware gehören. Auch eigene Funktionspakete, die man zum Beispiel aus Geschwindigkeitsgründen in Maschinensprache geschrieben hat, würde man typischerweise in Bibliotheken ablegen. Für Basic spielt dies keine Rolle. Die einzelnen Funktionen müssen sich nur an gewisse Aufruf-Konventionen halten, die zusammen mit den Funktionsnamen in einer von der eigentlichen Bibliothek getrennten Datei verzeichnet sein müssen. Diese Datei trägt immer denselben Namen wie die Bibliotheks-Datei selbst, aber mit einem zusätzlichen ».bmap« dahinter.

Die Beschreibungen der Bibliotheksfunktionen kann man zunächst in recht lesbarer Form erstellen. Das Format dafür ist im Anhang des Amiga-Basic-Handbuchs erläutert. Es handelt sich dabei um die sogenannten FD-Dateien. Ein besonderes Programm namens *ConvertFd* auf der Extras-Diskette wandelt diese Beschreibung aus der FD-Datei dann in das kompakte ».bmap»-Format um, das der LIBRARY-Befehl verlangt. Zu den wichtigsten Bibliotheken befinden sich aber schon bmap-Dateien in der Schublade *BasicDemos* auf der Extras-Diskette. Sie werden also wahrscheinlich nie in die Verlegenheit kommen, *ConvertFd.bas* zu brauchen. Falls doch, finden Sie alle nötigen FD-Dateien in der Schublade FD1.3 auf der Extras-Diskette.

Um die in den Systembibliotheken versammelten Funktionen aber sinnvoll nutzen zu können, benötigen Sie weit mehr Wissen über die Programmierung des Amiga, als ich Ihnen in diesem Kapitel verschaffen konnte. Ich erspare mir deshalb ein Beispielprogramm, das LIBRARY-Funktionen verwendet.

Und das war es dann auch schon! Ich hoffe, Sie haben in diesem Kapitel gesehen, daß Amiga-Basic nicht nur ein vielseitiger und komfortabler Dialekt der Programmiersprache Basic, sondern auch grenzenlos erweiterbar ist. Auch ohne diese Erweiterungen sind aber schon mit wenigen Anweisungen interessante Programme möglich, wie Sie anhand der oben vorgestellten Beispielprogramme sehen konnten. Amiga-Basic ist deshalb vielleicht die einzige Sprache, die Sie jemals benötigen werden. Fairerweise muß man jedoch dazu sagen, daß es viele Programmierprobleme gibt, für die man vielleicht doch besser eine andere Programmiersprache einsetzen sollte – aber eben nicht unbedingt muß.

# Anhang A Fehlercodes und ihre Bedeutung

Sowohl im Shell-Fenster wie auch auf der Workbench werden Ihnen gelegentlich Fehlermeldungen gezeigt. Bei Shell-Befehlen werden diese Meldungen im Shell-Fenster ausgegeben. Andere Programme zeigen Ihnen dazu vielleicht einen Requester. Die Workbench beschränkt sich oft auf einen Piepston und eine knappe Meldung in der Titelleiste des Workbench-Screens. Diese Meldung können Sie sich übrigens noch einmal anzeigen lassen, indem Sie den Befehl *Last Error* aufrufen. Alle diese Fehlermeldungen haben aber eine gemeinsame Eigenschaft: sie sind recht knapp und oft unverständlich. Deshalb folgt an dieser Stelle nun eine etwas ausführlichere Beschreibung der Fehler, die bei der Bedienung des Amiga auftreten können.

Jede dieser Fehlerbeschreibungen beginnt mit einer Nummer und einer knappen englischen Meldung. Dies sind die beiden Komponenten einer Fehlermeldung, die Sie auch sonst geboten bekommen. (Der Fehlercode – die Nummer – dient zur eindeutigen Identifizierung des Fehlers.) Dann folgt eine kurze Übersetzung der Fehlermeldung und eine etwas längere Erläuterung der möglichen Fehlerursache. Zu jedem Fehler finden Sie dann auch noch einen oder mehrere Vorschläge, wie Sie das Problem eventuell beheben könnten.

Diese Liste ist übrigens nicht vollständig. Unter gewissen Umständen können auch Fehler auftreten, die hier nicht verzeichnet sind. In diesen Fällen treten die Fehler aber aufgrund interner Probleme in Programmen auf, auf die Sie keinerlei Einfluß haben. Eine Erläuterung solcher Fehlermeldungen würde Ihnen also wenig helfen!

***103:*** *insufficient free store*
Freier RAM-Speicher reicht nicht aus.

**Erläuterung:** Sie haben versucht, etwas zu tun, wofür RAM-Speicher benötigt wird – zum Beispiel ein Programm gestartet. Es ist aber nicht mehr genügend Speicher frei.

**Gegenmaßnahmen:** Sie müssen versuchen Speicher freizumachen. Beenden Sie dazu zunächst alle Programme, die Sie im Augenblick nicht unbedingt benötigen und schließen Sie möglichst viele Fenster. Wenn das immer noch nicht hilft, versuchen Sie es nach einem Neustart noch einmal. Wenn das auch nichts hilft, sollten Sie die Anschaffung einer Speichererweiterung ernsthaft ins Auge fassen.

***120:*** *argument line invalid or too long*
Kommandozeile ist zu lang oder enthält fehlerhafte Teile.

**Erläuterung:** Sie haben versucht, einen Shell-Befehl einzugeben, der zu lang ist oder ein falsches Format für die Parameter verwendet.

**Gegenmaßnahmen:** Es sollte eigentlich nie nötig sein, einen Befehl einzugeben, der mit allen Parametern länger wird als eine Zeile. Falls Sie sehr lange Pfadnamen als Parameter verwenden müssen, sollten Sie das aktuelle Directory wechseln.

***121:*** *file is not an object module*
Datei ist keine Programmdatei.

**Erläuterung:** Sie haben versucht, eine Datei wie ein Werkzeug oder einen Shell-Befehl zu starten, aber diese Datei hat nicht das korrekte Format einer Programmdatei.

**Gegenmaßnahmen:** Prüfen Sie – falls dieser Fehler nach einem Shell-Befehl auftaucht – ob Sie sich nicht verschrieben haben. Schlagen Sie im Zweifelsfall den Befehl im Kapitel 7 nach. Beachten Sie bitte auch, daß Sie Kommandofolgen entweder über EXECUTE aufrufen oder mit dem PROTECT-Befehl das S-Flag setzen müssen. Auf der Workbench kann dieser Fehler eigentlich nur dann vorkommen, wenn Sie Info-Dateien mit dem *IconEd* oder im Shell-Fenster manipuliert haben und dabei einer Datei ein Werkzeug-Piktogramm verschafft haben, obwohl die Datei kein Werkzeug ist. Falls Sie alle diese Ursachen ausschließen können, kann es auch noch sein, daß die Programmdatei defekt ist. In diesem Fall sollten Sie sie löschen und durch eine Sicherungskopie ersetzen.

***202:*** *object in use*
Objekt in Gebrauch.

**Erläuterung:** Sie haben versucht, eine Datei oder ein Directory zu manipulieren, zum Beispiel zu löschen, obwohl diese Datei oder das Directory von einem anderen Programm benötigt wird.

**Gegenmaßnahmen:** Eine häufige Ursache für diesen Fehler ist der Versuch, eine Datei oder ein Directory zu löschen, die (das) in Gebrauch ist. Ein Beispiel hierfür wäre eine Kommandofolge, die versucht, sich mit DELETE selbst zu löschen. Das geht natürlich nicht. Wollen Sie diese Datei wirklich löschen, sollten Sie dieses andere Programm beenden und es dann noch einmal versuchen.

***203:*** *object already exists*
Eine Datei oder ein Directory dieses Namens existieren schon.

**Erläuterung:** Sie können beim Umbenennen eines Piktogramms auf der Workbench oder einer Datei im Shell-Fenster nicht den Namen einer schon existierenden Datei oder eines schon existierenden Directory verwenden. Auf der Workbench kann es sogar vorkommen, daß ein Umbenennen auch dann nicht möglich ist, wenn in derselben Schublade kein Piktogramm gleichen Namens existiert. In diesem Fall gibt es aber vielleicht eine unsichtbare Datei (ohne Piktogramm), die diesen Namenskonflikt verursacht.

**Gegenmaßnahmen:** Wenn eine umbenannte Datei die alte Datei gleichen Namens ersetzen soll, müssen Sie sie vorher löschen. Ist das nicht möglich, müssen Sie einen anderen Namen verwenden.

***204:*** *directory not found*
Directory nicht vorhanden.

**Erläuterung:** In einem Namen oder Pfadnamen, den Sie eingegeben haben, kommt ein Directory vor, das es nicht gibt. Es kann auch sein, daß Sie einen Doppelklick auf ein Schubladen-Piktogramm gemacht haben, zu dem es kein Directory gibt. Das kann leicht vorkommen, wenn Sie mit dem Piktogramm-Editor *IconEd* herumspielen.

**Gegenmaßnahmen:** Kontrollieren Sie die Schreibweise des eventuell eingegebenen Pfadnamens. In langen Pfadnamen verschreibt man sich leicht. Tritt diese Meldung beim Öffnen eines Schubladen-Piktogramms auf, sollten Sie diese Schublade mit *Discard* löschen (sie ist sowieso leer).

***205:*** *object not found*
Datei nicht vorhanden.

**Erläuterung:** Entweder Sie haben den Namen eines Programms eingegeben und AmigaDOS kann diese Datei nicht finden, oder Sie haben versucht, einen Doppelklick auf ein Projekt-Piktogramm zu machen und die Workbench kann das zugehörige Werkzeug nicht finden.

**Gegenmaßnahmen:** Wenn dieser Fehler im Shell-Fenster auftritt, sollten Sie wie üblich die Schreibweise des Befehls und aller Parameter im letzten Kommando überprüfen. Besonders in langen Pfadnamen verschreibt man sich leicht. Passiert dieser Fehler jedoch auf der Workbench, so befindet sich das Werkzeug zu einem Projekt nicht dort, wo es die Workbench erwartet. Öffnen Sie dann (mit dem Befehl *Info*) das Info-Fenster dieses Projekts und tragen Sie dort den vollen Pfadnamen des zugehörigen Werkzeugs ein. Falls die Workbench das Werkzeug nur deshalb nicht finden konnte, weil die entsprechende Diskette nicht eingelegt war, kann es auch reichen, die Diskette in ein Laufwerk einzulegen und das Projekt noch einmal zu öffnen.

***210:*** *invalid stream component name*
Ein Teil eines (Pfad-) Namens ist nicht korrekt.

**Erläuterung:** Sie haben – zum Beispiel beim Umbenennen oder Kopieren einer Datei – einen Namen eingegeben, der nicht zulässig ist.

**Gegenmaßnahmen:** Verwenden Sie in Namen, die Sie Dateien, Directories und Piktogrammen geben, möglichst keine Sonderzeichen. Vor allem auf »:« und »/« sollten Sie verzichten. Zudem dürfen solche Namen maximal 30 Zeichen lang sein. Pfadnamen können natürlich länger werden – nicht aber die einzelnen Komponenten eines Pfadnamens zwischen den Schrägstrichen und Doppelpunkten.

*212: object not of required type*
Objekt besitzt nicht den richtigen Typ.

**Erläuterung:** Sie haben bei einem Shell-Befehl den Namen eines Directory angegeben, wo ein Dateiname erwartet wurde, oder umgekehrt.

**Gegenmaßnahmen:** Schlagen Sie in Kapitel 7 die genaue Funktion des jeweiligen Befehls nach. Sie können als Argument von TYPE zum Beispiel kein Directory angeben. Vielleicht haben Sie sich aber auch nur verschrieben oder eine Komponente eines langen Pfadnamens vergessen.

*213: disk not validated*
Diskette konnte nicht validiert werden.

**Erläuterung:** Amiga-DOS versucht nach jedem Einlegen einer Diskette, diese zu »validieren«. Dabei wird überprüft, ob die internen Datenstrukturen, die Amiga-DOS benötigt, um die Daten auf der Diskette wiederzufinden, vorhanden und in Ordnung sind. Dieser Vorgang ist langwierig und deshalb kann die entsprechende Fehlermeldung eine ganze Weile nach dem Einlegen der Diskette auftauchen. Solange eine Diskette nicht validiert ist, können Sie übrigens auch keine neuen Dateien darauf anlegen (zum Beispiel durch Kopieren einer anderen Datei).

**Gegenmaßnahmen:** Wenn Sie diese Fehlermeldung erhalten, sieht es recht schlecht für die jeweilige Diskette aus. Machen Sie eine Kopie davon und versuchen Sie diese Kopie mit DISKDOCTOR (siehe Kapitel 7) wiederherzustellen. Kopieren Sie dann alle intakten Dateien auf eine neue leere Diskette und formatieren Sie die defekte Diskette neu.

*214: disk write-protected*
Diskette ist schreibgeschützt.

**Erläuterung:** Sie haben versucht, eine Schreiboperation mit einer Diskette durchzuführen, bei der der Schreibschutz aktiviert ist. Beachten Sie bitte, daß Schreiboperationen oft auch dann nötig sind, wenn Sie scheinbar keine entsprechende Aktion durchgeführt haben. Wenn Sie zum Beispiel im Hauptfenster von *Preferences* auf *SAVE* klicken oder einen Schnappschuß machen, müssen Daten auf die Diskette geschrieben werden.

**Gegenmaßnahmen:** Nehmen Sie die Diskette aus dem Laufwerk, entfernen Sie den Schreibschutz und wiederholen Sie die Aktion, die zu dem Fehler führte.

*215: rename across devices attempted*
Umbenennung von einem Gerät zum anderen versucht.

**Erläuterung:** Sie haben versucht, eine Datei »auf eine andere Diskette umzubenennen«. Das geht nicht. Der neue Name einer Datei kann zwar in einem anderen Directory liegen, aber nicht auf einer anderen Diskette.

**Gegenmaßnahmen:** Kopieren Sie zuerst die Datei auf die andere Diskette und löschen Sie dann das Original.

***216:*** *directory not empty*
Directory ist nicht leer.

**Erläuterung:** Sie haben im Shell-Fenster versucht, ein Directory, das noch andere Directories oder Dateien enthält, mit dem DELETE-Befehl zu löschen. Directories können aber nur dann gelöscht werden, wenn sie leer sind.

**Gegenmaßnahmen:** Leeren Sie das Directory, bevor Sie es löschen oder verwenden Sie die ALL-Option des DELETE-Befehls (siehe Kapitel 7).

***218:*** *device not mounted*
Gerät oder Diskette stehen im Moment nicht zur Verfügung.

**Erläuterung:** Ein Programm hat versucht, eine Diskette anzusprechen, die im Moment in keinem Laufwerk liegt (aber früher schon einmal eingelegt worden war).

**Gegenmaßnahmen:** Legen Sie die gewünschte Diskette ein und versuchen Sie dann noch einmal, die Arbeitsschritte durchzuführen, die zu dem Fehler führten.

***220:*** *comment too big*
Kommentar ist zu lang.

**Erläuterung:** Amiga-DOS verwaltet zu jeder Datei einen Kommentar, in dem Sie zusätzliche Informationen unterbringen können, die nicht in den Dateinamen passen. Dieser Kommentar darf maximal 79 Zeichen lang sein und kann im Info-Fenster oder mit dem Befehl FILENOTE verändert werden.

**Gegenmaßnahmen:** Versuchen Sie, sich kürzer zu fassen und einen Kommentar mit weniger als 80 Zeichen zu formulieren.

***221:*** *disk full*
Diskette ist voll.

**Erläuterung:** Eine Diskette kann eine bestimmte Maximalmenge an Daten aufnehmen (etwa 900 000 Zeichen beziehungsweise 880 Kbyte). Die Fehlermeldung *disk full* erscheint, wenn bei einer Schreiboperation versucht wird, diese Grenze zu überschreiten. Das kann beim Kopieren einer Datei passieren oder auch beim Abspeichern einer Datei nach Änderungen.

**Gegenmaßnahmen:** Schaffen Sie Platz auf der Diskette. Leeren Sie dazu zuerst den Papierkorb (mit dem Workbench-Befehl *Empty Trash*) und löschen Sie dann nicht mehr benötigte Dateien und Directories (Schubladen). Wenn das alles nichts hilft, müssen Sie eine neue leere Diskette verwenden, um die Daten abzuspeichern, die die Fehlermeldung verursacht haben.

*222: file is protected from deletion*
Datei ist vor dem Löschen geschützt.

**Erläuterung:** Sie haben versucht, eine Datei zu löschen, deren Löschschutz aktiviert ist.

**Gegenmaßnahmen:** Wenn Sie die so geschützte Datei wirklich löschen wollen, brauchen Sie diesen Löschschutz nur zu entfernen und dann die Löschoperation zu wiederholen. Den Löschschutz einer Datei können Sie im Info-Fenster dieser Datei oder mit dem Shell-Befehl PROTECT (siehe Kapitel 8) entfernen.

*223: file is write protected*
Datei ist vor dem Überschreiben geschützt.

**Erläuterung:** Sie haben versucht, eine Datei zu überschreiben, deren Überschreibungsschutz aktiviert ist.

**Gegenmaßnahmen:** Wenn Sie die so geschützte Datei wirklich überschreiben wollen, brauchen Sie den Überschreibungsschutz nur zu entfernen und dann die Operation zu wiederholen, die zu der Fehlermeldung führte. Den Überschreibungsschutz einer Datei können Sie mit dem Shell-Befehl PROTECT (siehe Kapitel 8) entfernen.

*225: not a valid DOS disk*
Keine Amiga-DOS-Diskette.

**Erläuterung:** Es gibt viele Computer, die dieselbe Diskettenform verwenden wie der Amiga (zum Beispiel der Atari ST, der Apple Macintosh und eine Reihe tragbarer Computer). Alle diese Computer verwenden verschiedene Formate, um die Daten auf Diskette zu schreiben, die der Amiga »nicht versteht«. Es kann aber auch sein, daß Sie versehentlich eine sogenannte Kickstart-Diskette des Amiga 1000 eingelegt haben. Auch diese Diskette können Sie auf der Workbench und im Shell-Fenster nicht bearbeiten.

**Gegenmaßnahmen:** Sie brauchen die Diskette nur wieder aus dem Laufwerk zu nehmen, wenn dieser Befehl auftaucht. Wollen Sie auf dieser Diskette jedoch Daten abspeichern, müssen Sie sie zunächst initialisieren beziehungsweise formatieren. (Dabei gehen die darauf schon vorhandenen Daten aber verloren.)

*226: no disk in drive*
Keine Diskette im Laufwerk.

**Erläuterung:** Ein Programm hat versucht, auf ein Laufwerk zuzugreifen (zum Beispiel beim Kopieren einer Diskette), und es befindet sich keine Diskette in diesem Laufwerk. Dieser Fehler tritt aber sehr selten auf. Die meisten Programme zeigen in solchen Fällen einen Requester, der Sie auffordert, die Diskette einzulegen.

**Gegenmaßnahmen:** Legen Sie die gewünschte Diskette in das Laufwerk.

# Anhang B ANSI-Escape-Sequenzen

Die Textausgabe in einem Shell-Fenster kann beeinflußt werden, indem man eine Reihe spezieller Buchstabenkombinationen in diesem Fenster ausgibt. Auf diese Art und Weise kann unter anderem die Schrift- und die Hintergrundfarbe des Fensterinhalts geändert werden. Man kann aber auch auf Fettdruck oder Kursivschrift umschalten.

Die Buchstabenkombinationen beginnen alle mit dem ASCII-Zeichen *Escape*, das normalerweise mit der [Esc]-Taste oben links auf der Tastatur erzeugt wird. In Texten, die mit dem Befehl ECHO ausgegeben werden, kann diese Taste aber nicht erzeugt werden. Sie müssen statt dessen dort die Buchstabenkombination »*E« statt des echten Escape-Zeichens verwenden.

Die folgende Liste enthält die wichtigsten Escape-Sequenzen, die Sie in einem Shell-Fenster verwenden können. In dieser Liste wird das Escape-Zeichen durch die drei Buchstaben *Esc* in kursiver Schrift angedeutet.

| | |
|---|---|
| *Esc*c | Fensterinhalt löschen und Fenster in »Normalzustand« zurücksetzen. |
| *Esc*[0m | Normale Zeichendarstellung einschalten. |
| *Esc*[1m | Fette Schrift einschalten (für alle folgenden Zeichen). |
| *Esc*[22m | Fette Schrift abschalten (für alle folgenden Zeichen). |
| *Esc*[3m | Kursive Schrift einschalten (für alle folgenden Zeichen). |
| *Esc*[23m | Kursive Schrift abschalten (für alle folgenden Zeichen). |
| *Esc*[4m | Unterstreichung einschalten (für alle folgenden Zeichen). |
| *Esc*[24m | Unterstreichung abschalten (für alle folgenden Zeichen). |
| *Esc*[2m | Schwarze Schrift einschalten (für alle folgenden Zeichen). |
| *Esc*[30m | Schriftfarbe Blau wählen (für alle folgenden Zeichen). |
| *Esc*[31m | Schriftfarbe Weiß wählen (für alle folgenden Zeichen). |

| | |
|---|---|
| *Esc*[32m | Schriftfarbe Schwarz wählen (für alle folgenden Zeichen). |
| *Esc*[33m | Schriftfarbe Orange wählen (für alle folgenden Zeichen). |
| *Esc*[40m | Hintergrundfarbe Blau wählen. |
| *Esc*[41m | Hintergrundfarbe Weiß wählen. |
| *Esc*[42m | Hintergrundfarbe Schwarz wählen. |
| *Esc*[43m | Hintergrundfarbe Orange wählen. |
| *Esc*[7m | Inverse Schrift einschalten. |
| *Esc*[8m | Schrift unsichtbar machen (Blau). |
| *Esc*[nu | Fensterbreite auf <n> Zeichen setzen. |
| *Esc*[nt | Fensterhöhe auf <n> Zeilen setzen. |
| *Esc*[nx | Abstand zwischen dem linken Fensterrand und dem »Innern« des Fensters auf <n> Pixel ändern. |
| *Esc*[ny | Abstand zwischen dem oberen Fensterrand und dem »Innern« des Fensters auf <n> Pixel ändern. |

Bei der direkten Eingabe einer solchen Escape-Sequenz müssen Sie *Esc* durch die Betätigung der [Esc]-Taste ersetzen. In Texten, die als ECHO-Parameter dienen, entspricht *Esc* der Buchstabenkombination »*E«. Um die Escape-Sequenz »*Esc*[1m« zu aktivieren (die den Fettdruck einschaltet), müßten Sie also den folgenden ECHO-Befehl verwenden:

```
ECHO "*E[1m"
```

Vorsicht ist mit den Befehlen geboten, die die Eigenschaften des Shell-Fensters ändern. Wenn Sie mit einer Escape-Sequenz die »logische« Fensterbreite und Fensterhöhe beispielsweise größer machen, als das wirkliche Shell-Fenster hoch und breit ist, kann es sein, daß Teile von Ausgaben verloren gehen. Dagegen kann man das logische Fenster bedenkenlos kleiner machen, als das wirkliche Fenster. Der Zeilenumbruch beziehungsweise das Rollen von Textausgaben beginnt dann lediglich eher.

Auch der linke und obere Rand sollten nur in Ausnahmefällen geändert werden. Der normale Abstand zwischen der Oberkante des Fensterinnern und der echten Oberkante des Fensters (der Oberkante der Titelzeile) beträgt 10 Pixel. Wird dieser Abstand unterschritten, wird die Titelleiste bei der Textausgabe unter Umständen zerstört. Der normale Abstand zwischen dem linken Rand des Fensterinnern und dem linken Rand des Fensterrahmens beträgt 2 Pixel. Wird dieser Abstand unterschritten, wird die Linie am linken Fensterrand bei der Textausgabe unter Umständen zerstört. Die Befehlsfolge *Esc*[0x*Esc*[0y führt deshalb zu einem Fenster völlig ohne linken Rand und ohne Titelzeile. Geben Sie danach einfach einen DIR-Befehl ein, um sich davon zu überzeugen.

Sollten Sie die Eigenschaften des Fenster so weit geändert haben, daß Sie kaum noch etwas erkennen können, sollten Sie die Tastenfolge [Esc] [c] drücken und dann die [Return]-Taste betätigen. Das Fenster wird dann in seinen »Urzustand« zurückversetzt.

# Stichwortverzeichnis

**A**

A 1081 396
A 1084 396f.
A 501 29, 255, 300, 395, 404
A 520 30, 397, 407
Abfangen eines Ereignisses 506, 516
ADDBUFFERS 243
Adreßgenerator 481
ADSR-Kurve 471
Agnus 481, 482
Akzent 185
Alarmzeit
– einstellen 104
ALIAS 243, 244, 369
Amiga 95
Amiga 1000 40f.
Amiga 2000 41
Amiga 500 41
–, Aufbau des 24ff.
–, Fähigkeiten des 33ff.
–, Geschichte des 39ff.
–, mitgelieferte Software 32ff.
–, Software des 37ff.
–, Zubehör für den 28ff.
Amiga-Architektur 484
Amiga-Basic 32, 173, 487, 499f., 503ff.
–, Bedienung von 503f.
–, Menüs in 518
–, Sprachsynthese mit 522
– starten 508
–, Umgang mit 507
Amiga-DOS 192, 202, 224, 226, 493
Amiga-Grafik 24, 47
Amiga-Software 487
Amiga-Systemsoftware 488f., 493
Animation 34
AnimObject 466
Anklicken 52
ANSI-Sequenz 262f., 529
Anweisungs-Block 502
Anwendungsprogramm 489
AREA-Anweisung 504
AREAFILL-Anweisung 504
Argument 198
ASK 244
Assembler 496
ASSIGN 229f., 245, 246, 314
Atari 39
Auflösung 454
Ausbau
– des Amiga 500 393ff.
Ausdruckgröße 155f.
Ausgabe
– anhalten 200
– umleiten 224f.
Ausgabemuster 281
Auswählen eines Piktogramms 68, 94
Autobootfähigkeit 419
AUX 312
AVAIL 246f.

**B**

Back-Gadget 58, 135
Basic 173, 496, 499f.
Basic-Dialekt 500f.
Baudrate 146
Befehl
– abbrechen 200f.

Befehl
– eingeben 195ff., 319ff.
– korrigieren 198ff., 320ff.
–, residenter 295f.
– suchen 307f.
– verzögern 306f.
Befehls-Menü 53
Beschleuniger 413
Bibliotheks-Dateien 523
Bildauflösung 454
Bildschirm 29, 141, 396ff.
– zentrieren 142
Bildschirm-Editor 322
Bildschirmauflösung 454
Bimmer 482
BINDDRIVERS 247
Bitkarte 450
Bitmapped-Grafik 34, 449ff.
Blitter 482f.
BOB 464ff., 482, 506
Bootblock 278f.
BREAK 248
Buchstabendichte 150

## C

C-Sprache 496
Cache-Technik 243
Calculator 104, 157
CD 219f., 248f.
Chaining 503
CHANGETASKPRI 249f.
Chip-RAM 168, 247, 269, 288, 484
chublade 76
CIRCLE-Anweisung 504
Clean up-Befehl 93
CLI 33, 38, 171f., 191ff., 287, 489
CLI-Fenster 287, 313
– öffnen 287, 366
CLI-Startup 366
Clock 100ff., 157, 250
ClockPtr 161f., 251
Cmd 166, 252f.
COMMENT-Feld 123
COMMON-Anweisung 503
Compiler 494f.
Composite-Monitor 29
Composite-Signal 396
Computer-Virus 279
Computergrafik 448f., 453
CON: 313
ConvertFd-Programm 523
Copper 483
COPY 210, 253ff.
Copyright-Meldung 45
CPU 26, 36, 476
Cursor 69, 106, 329
Custom-Chip 39, 476
Cut-Befehl 111

## D

Dartmouth-College 500
DATE 255f.
Datei 89, 201f.
– anzeigen 285f.
– ausdrucken 290
– ausgeben 285f., 306
– betrachten 211
– durchsuchen 299f.
– kopieren 209f., 253ff.
– löschen 211, 256f.
– sortieren 303f.
– suchen 299f.
– umbenennen 210f., 294f.
– verknüpfen 279
Datei-Kommentar 123, 270
Dateienbaum 202f., 220f.
Dateiname 201, 215
Dateiverwaltung 503
Datenträger umbenennen 293
Datentyp 502
Datenübertragung 146
Datum 255f., 300
Debugger 496
Defaultwert 350f.
DELETE 211, 223, 256f.
Denise 480f.
Deselektieren 52
Dialogfenster 70
digitales Prinzip 448
DIR 195, 206, 213f., 236, 257ff.
Directory 202ff., 213, 219ff., 229, 235, 248f., 257ff., 280ff.
–, aktuelles 204f., 219ff., 248f.
– anzeigen lassen 206f., 213f.
– erzeugen 222f., 283f.
– kopieren 253f.
– löschen 223, 256f.
– umbenennen 294f.
Directory-Hierarchie 203
Discard-Befehl 91f.
DISKCHANGE 259
DiskCopy 167, 208f., 259f.
DISKDOCTOR 261
Diskette 84, 88, 201, 205

Diskette
–, aktuelle 205
– formatieren 271f.
–, Info-Fenster einer 85f.
– kopieren 60ff., 86, 208f., 259f.
– löschen 87
– reparieren 261
–, startfähige 278
– umbenennen 293
–, unlesbare 528
– wiederherstellen 261
Disketten-Füllanzeige 85
Diskettenfenster 84
Diskettenformat 528
Diskettenlaufwerk 26, 61, 205, 402f.
–, externes 403
–, zusätzliches 28
Diskettenname 205
Diskettenwechsel 259
Diskettenpiktogramm 67f., 84
Dithering 154
DMA 478, 481
Doppelklick 100, 126, 143f.
DPAT 362
Druck 121, 166
Druckdatei 166, 252f.
Druckdichte 153
Drucker 30f., 121, 148ff., 166, 252, 311, 399
– initialisieren 169, 278
Druckeranpassung 148ff.
Druckeranpassungsfenster 148f.
Druckertreiber 149
Dual-Playfield-Modus 463f.
Duplicate-Befehl 61ff., 80

**E**

ECHO 262f., 529
ED 263, 322f.
EDIT 263
Editor 263, 284f., 322, 325
Ein-/Ausgabe-Umleitung 224f.
Eingabe
– umleiten 224f.
Eingabeaufforderung 291
ELSE 264, 353f.
emacs 325
Empty Trash-Befehl 83
ENDCLI 264
ENDIF 265, 353f.
Endlospapier 149
ENDSHELL 197, 232, 265
ENDSKIP 265f., 280, 303, 356
ERROR 355
Erste-Schritte-Diskette 32
Erweiterungsschnittstelle 28, 395, 404f.
Escape-Sequenz 529
ESDI-Kontroller 418
EVAL 266f., 360f.
Exec 489ff.
EXECUTE 267f., 344, 346f.
Expansion-Directory 247
Extras-Diskette 32, 149, 158, 172f.

**F**

FAIL 355
FAILAT 268, 355, 357
Fallunterscheidung 245
Farbdruck 151, 154
Farbkorrektur beim Drucken 154
Farbpalette 456ff.
Farbpalettenanimation 458f.
Fast-Fonts-Software 270
FastFileSystem 271, 420, 430
FastMem 168, 288, 247, 484f.
FastMemFirst 168, 268, 484
FAULT 269, 308
FBAS-Signal 396ff., 407
Fed 188
Fehler 269, 247, 484f.
– in Kommandofolgen 357f.
Fehlerabbruchlevel 268
Fehlercode 269, 525
–, Bedeutung eines 525
Fehlerlevel 268
Fehlermeldung 70f., 269, 308
Fenster 56ff., 77, 96, 492, 514
–, eines Werkzeugs 101ff.
–, verschieben 56f.
Fenster-Gadgets 58
Fernseher 397f.
Fernsehtuner 408
Festplatte 31, 84, 88f., 201, 402ff., 414ff.
–, Aufbau einer 414ff.
–, autobootfähige 404, 419ff., 433, 435
– formatieren 271f., 426, 430f.
– initialisieren 271f., 426
– installieren 424f.
– kopieren 441f.
– mit Software füllen 431f.
– sichern 441f.
– umbenennen 293
Festplatten-Controller 417f.
Festplatten-Organisation 440

Festplatten-Partition 421, 427
FF 269f.
FFS (siehe *FastFileSystem*)
FILENOTE 270f., 527
FixFonts 171
Flag 292
Flickerfixer 461
Font 171
Font-Menü 114
Format 167, 271f.
Formatieren
– einer Diskette 271
– einer Festplatte 271
Framegrabber 408
FreeMap 187
Front-Gadget 58, 135
Füllanzeige 84
Fülleinzeige 85
Funktionstasten 25

G

Gadget 58, 76
Gedächtnis
– der Shell 320f.
GenLock-Interface 409
Gerät 223, 229, 286, 309, 313f., 491
–, logisches 229ff., 245f., 313f.
Gerätename 223, 309
Gerätetreiber 247, 286
GETENV 267, 272f., 301f., 359
GOSUB 506
GOTO 510
Grafik
– mit Amiga-Basic 504, 512
Grafik-Bibliothek 491f.
Grafik-Fähigkeiten des Amiga 33
Grafik-Hardware 454ff., 480f., 492
Grafikauflösung 454, 457
Grafikdruck 150f., 152
Graphic Dump 162
GRAPHICDUMP 273
Gray Scale 151
Größen-Gadget 76

H

Halbbild 460
HAM-Modus 462
Handshaking 147
Hardware-Uhr 300
Hauptfenster
– von Preferences 138
Hauptprozessor 477
HF-Ausgang 30
HF-Signal 397
HiFi-Anlage 411
Hintergrund-Befehle 234
Hintergrund-Prozeß 296ff.
Hintergrund-Task 248f.
Hüllkurve 470f., 473

I

IconEd 174ff.
IconMerge 182ff.
ICONX 274f.
IF 275f., 353ff.
INFO 277
Info-Befehl 72, 81
Info-Fenster 71, 82
– einer Diskette 85f.
– einer Schublade 81f.
– eines Projektes 127f.
– eines Werkzeugs 122ff.
Initialisierung eines Druckers 169, 278
InitPrinter 169, 278
INSTALL 278f.
InstallPrinter 149, 157ff.
Interlace-Modus 142, 459ff.
Interpreter 494f., 503
Interrupt 479f.
Intuition 492f.

J

Jay Miner 39
JOIN 279
Jokerzeichen 215
Joystick 27

K

Keyboard 411, 475
KeyToy2000 184f.
Kickstart-ROM 302, 420
Klang 467
– digitalisieren 468
– erzeugen 468f., 472
Klangerzeugung 467
– mit Amiga-Basic 520
Klangform 467, 469
Kollisionserkennung 466
Kommandofolge 244, 264f., 267f., 274ff., 280, 282, 293, 301, 303, 315, 344ff.
– automatisch erzeugen 282, 361
– beenden 293
– erstellen 345
– mit Piktogramm 274f.

Kommentar 123, 270, 351, 527
Kommunikationsprogramm 189
Kontroller 417f., 422
Kopieren
– einer Datei 209f., 253ff.
– einer Diskette 60ff., 86, 259f.
– eines Directory 253ff.
– mit der RAM-Disk 94f.
– mit einem Diskettenlaufwerk 61
– mit zwei Diskettenlaufwerken 64
Koprozessor 477f.
Korrespondenzqualität 150

**L**

LAB 280, 303, 356
Label 502
Laserdrucker 400f.
Laufwerk 229f., 259, 313f.
–, aktuelles 249
–, logisches 229ff., 245f., 313f.
Laufwerksname 205f.
LFORMAT 361
Library 314, 523
LIBRARY 523
LINE 504, 513
LIST 207, 280f., 361f.
Listing-Fenster 503, 508
LOADWB 282, 364
LOCK 283
Lorraine 39
Löschen 123
– einer Datei 211
– einer Diskette 87
– einer Schublade 82
– eines Piktogramms 82, 91
Löschschutz 123

**M**

Macintosh 502
MAKEDIR 222f., 283f.
Makro 243
Margin 150
Marke (siehe *Sprungmarke*)
Maschinencode 494ff.
Matrixdrucker 399ff.
Maus 24, 26, 48ff., 95
Maustasten 51
Mausübersetzung 143
Mauszeiger 50f., 54, 145f., 251
– ändern 145f.
Mauszeiger-Farben 145
Mediencomputer 406
Mehrfarbdrucker 400f.
Meldung
– ausgeben 262
MEMACS 184, 284f., 323, 325f., 329, 345
– beenden 327
–, Bildschirmaufbau von 327
–, Geschichte von 325
– starten 326f.
MEMACS-Fenster 327
MEMACS-Screen 327
MENU-Anweisung 505, 518
Menü 52f.
– in Amiga-Basic 518
Menütaste 49, 52ff.
MENU-Funktion 506
Menübefehle 54f., 110
– der Workbench 97
Menüleiste 52f.
– eines Werkzeugs 102
Menüpunkt 54f.
Menütitel 52
Menüzeile 54
MergeMem 168, 285
MFM-Kontroller 418
Micro-Emacs 325
Microsoft-Basic 501f.
MIDI 411f., 475
Modem 188
Modifikationsdatum 301
Modula-2 497
Mondrian 513
Monitor 24, 29f., 398, 496
More 162ff., 285f.
MOUNT 286, 364f., 420, 426, 428f.
Mountlist 286, 421, 426ff.
Mountlist-Eintrag 428
MOUSE-Anweisung 506
MOUSE-Funktion 516
MS-Basic 501f.
Müllabfuhr 83
Mülleimer 82f.
Mülleimerpiktogramm 83
MultiSync-Monitor 398f., 461
Multitasking 38, 128ff., 232ff., 480
Muster 215ff., 241, 257

**N**

Nadeldrucker 400
Namensmuster 215
Near Letter Quality 150
Netzteil 24

Neustart 46f.
– von einer RAM-Disk 436
– von Festplatte 432ff.
NEWCLI 287, 313
NEWCON 287, 313
NEWSHELL 232f., 287f., 312f.
NIL: 310
NoFastMem 167f., 288f., 484
Notepad 106f., 110, 120, 157, 323f.
–, Parameter von 125f.
Notizblock (siehe *Notepad*)

## O

Öffnen
– eines Projektes 126f.
ON MOUSE-Anweisung 516f.
ON-MENU-Anweisung 518
Open-Befehl 55, 120, 126
Open-Requester 120
Optionale Zeilennummer 502
Output-Fenster 503, 508

## P

PAINT 505
PAL-Encoder 407
PAL-Modulator 407, 409
Palette 457f.
PALETTE-Anweisung 505
Palette-Programm 185f.
Papiergröße 149
PAR: 311
parallele Schnittstelle 27
Parameter 198, 214f., 239ff., 348f., 351
– eines Projektes 128
– einer Kommandofolge 347ff.
– erfragen 214f.
– optionaler 239, 349
– von Notepad 125f.
Partition 272, 421, 427f., 429, 433
Partitionsgröße 429
Pascal 497
Paste-Befehl 112
PATH 227f., 238, 289
Paula 472, 479f.
Pause in Kommandofolgen 306f.
PerfMon 187f.
Periode eines Klangs 471
Pfadname 203ff., 220f.
Phonem 475
Piktogramm 67f., 72, 76, 89, 174ff., 183, 235
– ändern 175ff.
Piktogramm
– auswählen 68, 94
– löschen 82, 91
– mit Kommentar versehen 123
– selektieren 94
– umbenennen 68ff.
– verschieben 68
Piktogramm-Editor 175
PIPE: 311
Pitch 150
Pixel 449, 455
Pixelgrößen 455
Platzhalter 216, 348
Playfield 481
Pozeß 305
Preferences 137ff., 290
Preferences-Hauptfenster 138
PRINT-Anweisung 505
PrintFiles 165, 290
Prinzip, digitales 448
Priorität 250
– eines Prozesses 249
Programm 99f., 129, 307, 495f.
– abspeichern 510
– auf Festplatte installieren 443f.
– eingeben 508
– einlesen 511
– starten 207f., 509
Programm-Chaining 503
Programm-Marke 502
Programmbibliothek 314, 491
Programmfehler
– suchen 509
Programmiersprache 494ff., 499, 501
Programmierung 487, 493, 499
Programmschleife 358
Project-Menü 118, 120, 122
Projekt 89, 91, 99ff.
–, Infofenster eines 127f.
–, Öffnen eines 126f.
–, Parameter eines 128
Prompt 195, 220, 291
Proportionen eines Ausdrucks 155f.
PROTECT 291f., 347
Prozedur 502
Prozedure 500
Prozeß 249
– abbrechen 248
PRT: 311
PSET 512
Pufferspeicher 243
Punktbefehl 348

## Q

QUIT 293, 356, 358

## R

RAD: 294, 302, 310, 371, 373
– löschen 295
RAM-Disk 94f., 294, 310, 369ff.
–, Kopieren mit der 94f.
– löschen 294
RAM-Speicher 29, 246f., 288, 404
Rastergrafik 448f.
RAW: 313
Rechnen
– im Shell-Fenster 266
Rechner 104
Register 477
RELABEL 293f.
REMRAD 294, 310
RENAME 210f., 294f.
Rename-Befehl 68ff.
Requester 70, 119
RESET 228
RESIDENT 295f., 364f.
Returncode 268, 276, 293, 355
RGB-Ausgang 397
RGB-Monitor 29f., 396, 398, 407
RGB-Signal 398
RLL-Kontroller 418
Rollbalken 76ff., 108
Rollkasten 77f.
ROM-Speicher 27, 192
RUN 234, 296ff.

## S

S-Flag 267, 347
Sampling 467f.
Sampling-Frequenz 467
Save-Befehl 119f.
Save-Requester 120
Say 159ff., 298, 474
SAY-Anweisung 507, 522
SCART-Eingang 398
Schleife 358
Schlüsselwort 214, 239ff.
–, optionales 239
Schnappschuß machen 93
Schnittstelle 27, 394f.
–, parallele 27, 311
–, serielle 27, 146f., 311
Schreib-Lese-Kopf 414
Schreibmarke 69, 106ff., 199, 329, 508
Schreibschutz 70, 283
Schreibschutz
–, einer Festplatte 283
Schrift 115ff., 171
Schriftart 115ff., 171, 314
Schriftausgabe 269
Schriftgröße 115
Schriftstil 115ff.
Schublade 72ff., 79ff., 235
– aufräumen 93
– bewegen 79
– duplizieren 80f.
– erzeugen 82
– Info-Fenster einer 81f.
– löschen 82
– öffnen 73
– optimal anordnen 440
– schließen 75
Schubladen-Piktogramm 235
Schubladenfenster 76, 84
Schubladenpiktogramm 84
Schutzflag 291f., 347
Screen 132f., 492, 514
– bewegen 133f.
SCREEN-Anweisung 505, 514f.
Screendump 162, 273
SCSI-Kontroller 418, 423
SCSI-Standard 418
SEARCH 299f.
Sektor 414
Selektion 52, 93
– eines Piktogramms 68
Selektionstaste 49, 51
SER 312
serielle Schnittstelle 27
– einstellen 146f.
SETCLOCK 300, 364
SETDATE 301
SETENV 267, 273, 301f., 359
SetMap 169ff., 364
SETPATCH 302
Shell 38, 172, 191, 231f., 489
– starten 194f.
– verlassen 197
Shell-Befehl 198, 214, 237, 243, 308ff., 525
–, Aufruf eines 238f.
– eingeben 321f.
–, Format eines 238f.
– korrigieren 321f.
– suchen 307f.
– verzögern 306f.

Shell-Fenster 196, 232, 237, 262, 287f., 313, 319, 529
– öffnen 231ff., 287f., 366
– schließen 264f.
Shell-Makro 243
Shell-Prompt 220, 291
Shell-Prozeß 264
Shell-Startup 287, 366f.
Shell-Task 305
Sicherungskopie 59, 441
– einer Festplatte 441f.
Sicherungsprogramm 442
Skalierung
– beim Ausdruck 155f.
SKIP 265f., 280, 303, 356, 358, 360
Smoothing 152
Snapshot-Befehl 93
Sonderzeichen 184f.
– in Kommandofolgen 352
SORT 303f.
SOUND-Anweisung 520f.
Sound-Chip 472
Sound-Digitizer 412
Sound-Fähigkeiten
– des Amiga 35
Sound-Hardware 473
Sound-Sampling 467
Spacing 150
Spaghetti-Code 500
SPAT 362
SPEAK 226, 312
Speichererweiterung 29, 168, 285, 395, 404f.
–, externe 405
Speicherplatz-Probleme 131
Speicherverwaltung 490
Spezial-Chip 36, 447, 476, 478
Sprachsynthese 36, 159, 161, 298, 312, 474f.
– in Amiga-Basic 522
Sprite 464f., 481, 506
Sprungmarke 280, 356, 510
Spur 414f.
ST-506-Standard 418
STACK 304
Stapelspeicher 304
– ändern 305
Start-Befehl 509
Starten
– des Amiga 500 43f.
Startup-Sequence 363, 367f., 376, 434f., 437f.
Startup-Sequence.hd 434f.
StartupII 365, 367ff.
STATUS 233, 297, 305
Steckkarte 394
Step-Befehl 510
Steuersequenz 262
Streamer 423, 442
Stromanschluß 27
Subprogramm 502, 523
Subroutine 500
Suchpfad 227f., 238, 289f.
– ändern 289f.
– anzeigen 289f.
Synthesizer 412, 474
System-Schublade 166f.
Systemprogramm 99
Systemsoftware 48, 192, 487ff., 492f.
Systemzeit 255, 300

## T

Taschenrechner 105, 157, 266
Task 248f., 305, 490
– abbrechen 248
Tastatur 25, 184
Tastatur-Maus 95
Tastaturbelegung 169ff.
Tastenäquivalent 110
Tastenwiederholung, automatische 143
Telekommunikationsprogramm 188, 190
Terminal 188ff.
Text
– abspeichern 328
– ausdrucken 290
– ausgeben 262, 306
– ausschneiden 111
– bearbeiten 263, 284, 322, 324, 330
– eingeben 330
– einlesen 328
– ersetzen 112f.
– gestalten 114f.
– kopieren 112
Text
– selektieren 109f.
– sortieren 303f.
– suchen 112f., 299f.
– umstellen 111
Textausgabe 269
Textblock in MEMACS 331
Texteditor 263, 284f., 322, 325
Texteingabe 107f.
Thermotransfer-Drucker 400
Tintenstrahl-Drucker 400

Titelzeile 53
– des Workbench-Bildschirms 46
– eines Fensters 56
Ton 467
– digitalisieren 468
– erzeugen 468f., 472
Tonerzeugungs-Kanal 473
TOOL-TYPES-Feld 123f.
Trace-Befehl 510
TRANSLATE 507, 522
Trashcan 82
Treshold 152
Tutorial 32
TYPE 211, 306
Typenraddrucker 401

**U**

Uhr 29, 100ff., 250f., 300
– einstellen 140f.
Uhrzeit 255f., 300
Umbenennen
–, einer Datei 210f.
Umgebungsvariable 272f., 276, 301f., 359, 361
Utilities-Schublade 157

**V**

Variable 502
Video-Digitizer 408
Videokamera 408
Videomonitor 30, 396
Videorecorder 30, 396, 407
Virus 279
Vollbild 460
Voreinstellungen 290

**W**

WAIT 306f.
WARN 355
Wechselplattenlaufwerk 442
Weckzeit 104
Werkzeug 89ff., 137, 238
–, Infofenster eines 122ff.
– starten 99ff.
WHICH 307f.
WHILE-Schleife 512f.
WHY 308
WINDOW-Anweisung 505, 514f.
Word Wrap 117
Workbench 33, 37f., 43, 47, 52, 55, 235, 282, 489
– auf Festplatte installieren 432
Workbench-Diskette 32, 44ff., 278f.
Workbench-Farben
– ändern 139
Workbench-Menü 53

**Z**

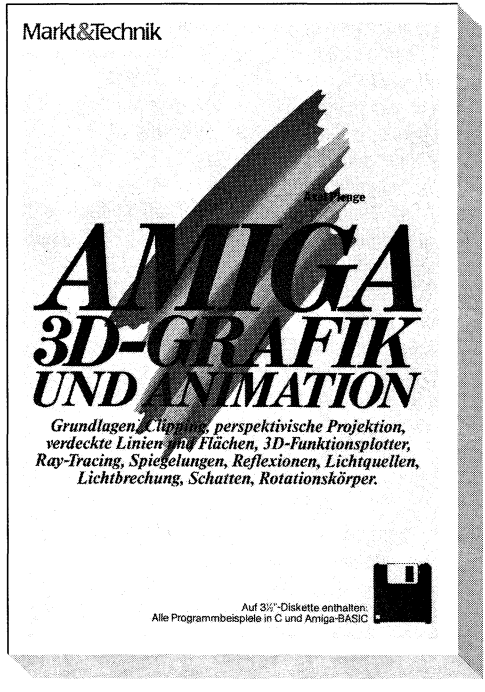
Zählschleife 267
Zeichensatz 115ff.
Zeichenstil 116
Zeilen-Editor 323
Zeileneditor 263
Zeilensprung 460
Zeilenumbruch 117
Zentrieren des Bildschirms 142
Zugriffszeit 422
Zwischenablage 112
Zylinder 415

3,5-Zoll-Disketten 402
5,25-Zoll-Disketten 402
68000 413
68020 413
68030 413

# Bücher zum Amiga




H. Knappe
**Fraktale Grafik auf dem Amiga**
Das Thema dieses Buches wird den meisten als ungewöhnlich erscheinen, denn es führt in die Grenzbereiche des heutigen Wissens der Mathematik und Technik.
Grundlegende Kenntnisse der Programmiersprache C und ihrer Anwendung auf dem Amiga werden vorausgesetzt. Für die nachträgliche Veränderung berechneter Bilder ist es sinnvoll, ein Malprogramm (z. B. Deluxe Paint) zu besitzen.
1988, 278 Seiten,
inkl. Diskette
Bestell-Nr. 90600
ISBN 3-89090-600-1
**DM 79,–**
(sFr 72,70/öS 616,–)




A. Plenge
**Amiga 3-D-Grafik und Animation**
Angefangen bei den einfachsten Problemstellungen lernen Sie, professionelle 3-D-Grafiken auf Ihrem Commodore Amiga zu planen, zu programmieren und darzustellen.
1988, 376 Seiten,
inkl. Diskette
Bestell-Nr. 90526
ISBN 3-89090-526-9
**DM 69,–**
(sFr 63,50/öS 538,–)




H. R. Henning
**Grafik mit Amiga-Basic**
Dieses Buch ist speziell der Grafik-Programmierung auf dem Amiga gewidmet. Der erste Teil stellt für den Anfänger alle bekannten Grafik-Befehle des Amiga-Basic vor. Mit Beginn des zweiten Teiles werden die Routinen des Betriebssystems zur Grafik Programmierung herangezogen. Damit werden die Möglichkeiten des Basic um ein Vielfaches erweitert, und es sind Geschwindigkeiten möglich die kaum vermuten lassen daß dabei ein Basic-Programm abläuft.
1989, ca. 300 Seiten,
inkl. Diskette
Bestell-Nr. 90669
ISBN 3-89090-669-9
**DM 59,–**
(sFr 54,30/öS 460,–)

Markt&Technik
Zeitschriften · Bücher
Software · Schulung

Markt&Technik-Produkte erhalten Sie bei Ihrem Buchhändler, in Computer-Fachgeschäften oder in den Fachabteilungen der Warenhäuser.

429/901

Markt&Technik Verlag AG, Buchverlag, Hans-Pinsel-Straße 2, 8013 Haar bei München, Telefon (089) 4613-(

Bitte schneiden Sie diesen Coupon aus, und schicken Sie ihn in einem Kuvert an:
Markt&Technik Verlag AG, Buchverlag, Hans-Pinsel-Straße 2, 8013 Haar

# Computerliteratur und Software vom Spezialisten

Vom Einsteigerbuch für den Heim- oder Personalcomputer-Neuling über professionelle Programmierhandbücher bis hin zum Elektronikbuch bieten wir Ihnen interessante und topaktuelle Titel für

• Apple-Computer • Atari-Computer • Commodore 64/128/16/116/Plus 4 • Schneider-Computer • IBM-PC, XT und Kompatible
sowie zu den Fachbereichen Programmiersprachen • Betriebssysteme (CP/M, MS-DOS, Unix, Z80) • Textverarbeitung • Datenbanksysteme • Tabellenkalkulation • Integrierte Software • Mikroprozessoren • Schulungen.
Außerdem finden Sie professionelle Spitzen-Programme in unserem preiswerten Software-Angebot für Amiga, Atari ST, Commodore 128, 128 D, 64, 16, für Schneider-Computer und für IBM-PCs und Kompatible!
Fordern Sie mit dem nebenstehenden Coupon unser neuestes Gesamtverzeichnis und unsere Programmservice-Übersichten an, mit hilfreichen Utilities, professionellen Anwendungen oder packenden Computerspielen!

Markt&Technik
Zeitschriften · Bücher
Software · Schulung

Markt&Technik Verlag AG, Buchverlag, Hans-Pinsel-Straße 2,
8013 Haar bei München, Telefon (089) 4613-0

709005

Adresse:

Bitte schicken Sie mir:
☐ Ihr neuestes Gesamtverzeichnis
☐ Eine Übersicht Ihres Programmservice-Angebotes aus der Zeitschrift _____

Name _____

☐ Außerdem interessiere ich mich für folgende/n Computer: _____

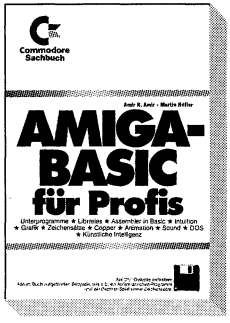
Straße _____

Ort _____

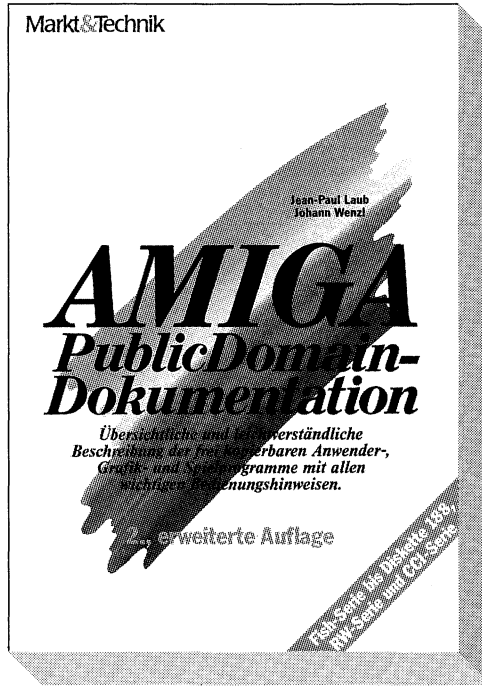
(PS: Wir speichern Ihre Daten und verpflichten uns zur Einhaltung des Bundesdatenschutzgesetzes)

Markt&Technik Verlag AG
– Unternehmensbereich Buchverlag –
Hans-Pinsel-Straße 2
D-8013 Haar bei München

# Bücher zum Amiga




A. Amir/M. Höfler
**Amiga-Basic für Profis**
Das vorliegende Buch gibt Ihnen die Möglichkeit, mit Amiga-Basic zur Leistungsgrenze Ihres Rechners vorzustoßen. Es umfaßt Gebiete wie die gesamte Intuition (mit Gadgets, Requestern, Menüs etc.), Grafik, 4096 Farben, Darstellung von dreidimensionalen Objekten, Fractals, Animation, Amiga-DOS, digitalisierter Sound, künstliche Intelligenz und viele andere. Eine detaillierte und leichtverständliche Erklärung der Themen und eine Vielzahl von Beispielprogrammen auf zwei mitgelieferten 3½"-Disketten zeichnen das umfangreiche Werk aus.
1989, 573 Seiten,
inkl. 2 Disketten
ISBN 3-89090-710-5
**DM 79,–**
(sFr 72,70/öS 616,–)




J.-P. Laub/J. Wenzl
**Amiga Public-Domain-Dokumentation**
Dieses Buch verschafft Ihnen einen detaillierten und aktuellen Überblick über die Vielfalt der Public-Domain-Dokumentation.
2. überarbeitete Auflage
1989, 295 Seiten
ISBN 3-89090-242-1
**ca. DM 49,–**
(sFr 45,10/öS 382,–)




F. Riemenschneider
**Amiga: Programmieren in Maschinensprache**
Durch diesen ausführlichen Assemblerkurs wir die Programmierung auc für Anfänger und Nur-Basic-Programmierer ein Kinderspiel. Der Clou dieses Buches besteht darin, daß das Betriebssystem des Amiga in ausführlichen Beispielen vorgestellt wird, wie sie bisher in Büchern nicht z finden waren. Die Beispiele sind so aufgebaut daß Sie große Teile unverändert übernehmei und eigenen Programme einen professionellen Anstrich geben können.
- Für Einsteiger und Profis.

1989, 469 Seiten,
inkl. Diskette
ISBN 3-89090-712-1
**DM 69,–**
(sFr 63,50/öS 538,–)

Markt&Technik
Zeitschriften · Bücher
Software · Schulung

4211/907

Markt&Technik-Produkte erhalten Sie bei Ihrem Buch- oder Computerfachhändler

# Amiga 500-Buch

**Der Autor:**

MARKUS BREUER, Jahrgang 1960, studierte Informatik an der Universität Dortmund. Hauptinteressengebiet sind seit dem Studium Computer mit hochentwickelten grafischen Benutzeroberflächen. Er verfaßte bereits eine Vielzahl von Artikeln, Büchern und Handbüchern zu den Themenkreisen Apple Macintosh, Commodore Amiga, moderne Programmiersprachen, Computergrafik, Multimedia-Anwendungen und Desktop Publishing und arbeitet seit 1989 als freier Autor und EDV-Berater.

Der Commodore Amiga 500 besitzt nicht nur phantastische Farbgrafik- und Sound-Fähigkeiten, sondern ist auch ungewöhnlich leicht zu bedienen. Grafische Symbole (Piktogramme), ein komfortables Menüsystem und die Maus machen das Auswendiglernen komplizierter Befehlsfolgen überflüssig. Die Bedienung des Amiga 500 ist fast so einfach wie die Auswahl eines Getränks auf einer Speisekarte – will aber trotzdem erlernt werden. Das vorliegende Buch will Ihnen dabei helfen. Es bietet zunächst eine Einführung, die Sie anhand vieler Abbildungen und Beispiele mit der Bedienung des Amiga 500 vertraut macht. Es beschränkt sich aber nicht auf eine bloße Einführung. Viele Tips und ein umfangreicher Handbuchteil helfen Ihnen, im täglichen Einsatz schnell und reibungslos mit dem Amiga 500 zu arbeiten. Daneben erhalten Sie auch eine ausführliche Beschreibung des Amiga 500: alles über Hardware, Software, Zubehör und Fähigkeiten.
Bei der Beschreibung der Software werden alle Funktionen der neuen Workbench 1.3 ausführlich behandelt. Dabei kommen auch das CLI und die neue Shell nicht zu kurz. Kenntnisse des CLIs oder der Shell werden für viele Arbeiten mit dem Amiga – speziell beim Einsatz von Festplatten – unbedingt benötigt und können vor allem dem versierten Anwender die Arbeit wesentlich erleichtern. Dieses Buch trägt diesem Umstand Rechnung: mit einer schrittweisen Einführung für den Shell-Anfänger, detaillierten Beschreibungen und Tips für den bereits erfahreneren Shell-Anwender sowie einem übersichtlichen Nachschlageteil zu allen Shell-Befehlen.
Pflichtlektüre also für alle, die optimal mit dem Amiga 500 arbeiten möchten!

**Aus dem Inhalt:**

- Erste Handgriffe auf der Workbench
- Umgang mit den mitgelieferten Programmen
- Kennenlernen der Amiga-Shell
- Der Texteditor MEMACS
- Automatisierung des Amiga mit Kommandofolgen
- Ausbaumöglichkeiten des Amiga 500
- Installation und optimale Nutzung von Festplatten
- Grundlagen der Amiga-Grafik
- Sounderzeugung mit dem Amiga 500
- Amiga-Basic im Überblick

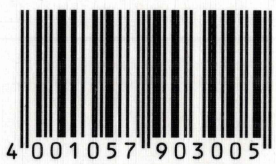
Das vorliegende Buch ist die vollständig überarbeitete und sorgfältig aktualisierte Neuausgabe eines der erfolgreichsten deutschen Bücher zum Commodore Amiga und enthält eine ausführliche Beschreibung der neuen Workbench 1.3.

**Hardware-Anforderung:**

Commodore Amiga 500, Monitor, Workbench 1.2 oder höher (Workbench 1.3 empfohlen)

ISBN 3-89090-300-2




DM 49,-
sFr 45,10
öS 382,-