MIKRO + KLEIN COMPUTER

87-5

DAS SCHWEIZER FACHMAGAZIN FÜR KLEINE UND MITTLERE COMPUTERSYSTEME Fr. 8.–

Von Prozessoren, Tasks und Coroutinen

GFU-General File Utility

ES STECKT NOCH MEHR IN IHREM MULTISYNC™!

SPEZIALAKTION - Fr. 200.–
(Regulärer Preis Fr. 1150.–)
Bis 15.11.87 Fr. 950.–)
Fragen Sie Ihren Dealer oder uns direkt an.

Sehr viele EGA-Karten kommen bis hierher

Einige kommen darüber hinaus

Jetzt holen Sie noch mehr aus Ihrem Multisync™: die SuperEGA™ von GENOA bringt Sie bis hierhin

Und so nutzen Sie ihn ganz aus: mit der SuperEGA HiRes™ von GENOA

15.75 KHz	18.75 KHz	21.8 KHz	30.0 KHz	31.5 KHz	35.0 KHz
CGA 320 x 200	MDA (Hercules) 720 x 348	EGA 640 x 350 1056 x 352 132 col. x 44	PGA resolution 640 x 480 Drivers for: Lotus 1-2-3, AutoCAD, GEM, Windows, and Desktop Publishing	CGA DoubleScan: 320 x 400 640 x 400	CAD/CAM 800 x 600 Drivers for: AutoCAD, Windows, GEM, and Desktop Publishing

Schöpfen Sie Ihren MultiSync™ oder kompatiblen Monitor voll aus, bis zu 800x600 Punkten, mit einer SuperEGA-Karte von GENOA.

Die SuperEGA™ von GENOA bietet Ihnen CGA, MDA (inkl. Hercules) und EGA-Modus, alle Modi voll kompatibel, ohne Software-Emulation. Zusätzlich dem PGA entsprechende 640x480 und DoubleScan. Damit lösen Sie normale CGA-Software mit bis zu 640x400 Punkten auf. 132 Zeichen in bis zu 44 Zeilen können Sie selbstverständlich auch auf Ihrem Bildschirm darstellen. Und fürs Desktop Publishing stehen Ihnen 80 Zeichen mal 66 Zeilen zur Verfügung.

Wenn Sie noch höher auflösen wollen, dann bietet Ihnen die SuperEGA HiRes™ darüber hinaus 800x600 Punkte bei 16 Farben für CAD/CAM und Desktop Publishing.

Beide SuperEGA-Karten synchronisieren automatisch. Mit allen MultiSync-Frequenzen zwischen 15,75 und 35 kHz.

Für alle Grafik-Anwendungen können Sie auf Ihre SuperEGA zählen. Wir holen mehr aus Ihrem MultiSync™ oder kompatiblen Monitor.

Und selbstverständlich können Sie auch einen Monochrom-, Color- oder EGA-Monitor anschließen.

Für weitere Informationen wenden Sie sich bitte an
ELECTRONIC MARKETING AG
Bahnhofstrasse 60, 4132 Muttenz
Tel. 061 61 53 53, Telex 965 669 elma, Fax 061 42 45 49

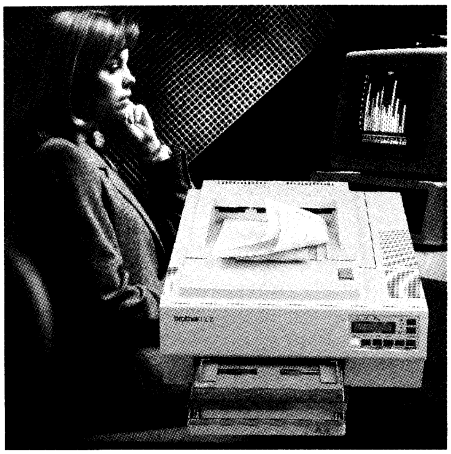
Your Swiss distributor for high technology.

ELECTRONIC MARKETING AG

YOUR SWISS DISTRIBUTOR FOR HIGH TECHNOLOGY

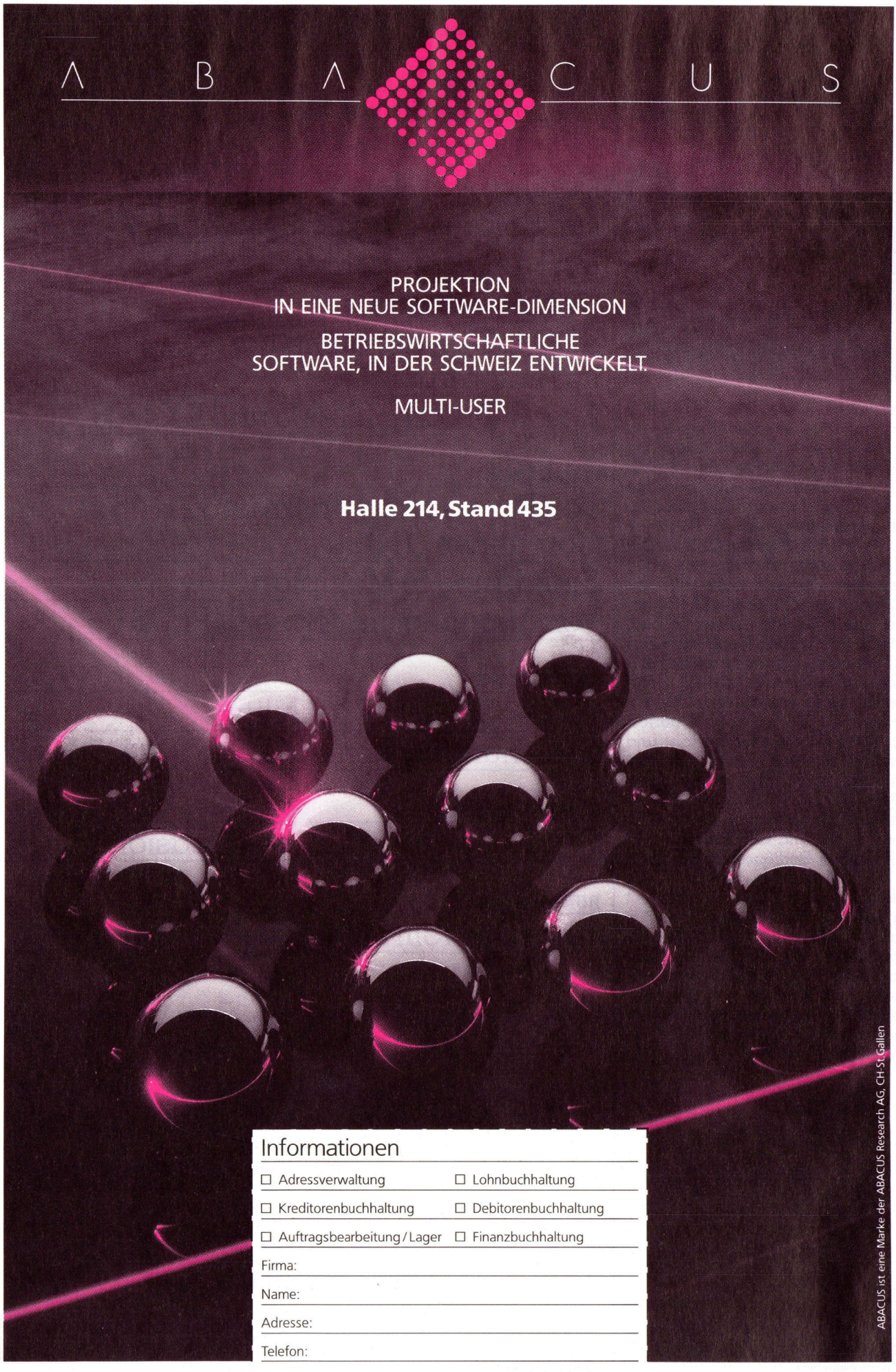
Electronic Marketing AG · Bahnhofstr. 60 · 4132 Muttenz-Basel · Tel. 061-61 53 53 · Tlx 965669 · Fax 061-42 45 49

DAS LESEN SIE IN DIESER NUMMER



Der Laserdrucker BROTHER HL-8 druckt mit einer Geschwindigkeit von acht A4-Seiten pro Minute. Insgesamt 30 Festfonts mit verschiedenen Schriften bieten zahlreiche typografische Möglichkeiten. Für Grafiken kann die Zeichenauflösung von 75 Punkten/Zoll bis zu 300 Punkten/Zoll individuell eingestellt werden. Der HL-8 verarbeitet Papierformate bis zur Standardgrösse A4. Spezielle Papierarten wie z.B. schwereres Papier, Couverts oder Folien lassen sich manuell zuführen. Alle wichtigen Funktionen wie Druckeremulationen, Font-Auswahl, Papierzuführung, Druckausrichtung (Hoch- oder Querformat), Formatierung, Kopienzahl und Selbst-Test können direkt auf dem Frontpanel angewählt werden. Das LCD-Display für 16 Zeichen gibt alle wichtigen Informationen an den Benützer weiter und dient gleichzeitig als hilfreiche Bedienerführung. Wie alle Brother-Drucker ist auch der Laserdrucker HL-8 serienmässig mit zwei Schnittstellen (einer Centronics parallel und einer RS-232C seriell) ausgestattet, mit sechs weiteren Druckern emulierbar und selbstverständlich kompatibel mit der handelsüblichen Software. Info: Brother Handels AG, Täfernstrasse 30, 5405 Baden, Tel. 056/84'02'21. □

Ausgabe Oktober 1987
Erscheint zweimonatlich
9. Jahrgang



ABACUS RESEARCH AG
Rorschacherstrasse 170, Postfach 694, 9006 St.Gallen, Telefon 071 25 93 25, Telex 71775 ABAC-CH, Telefax 071 253860
Betriebswirtschaftliche Software-Entwicklung, Beratung, Schulung

Von Prozessen, Tasks und Coroutinen (1. Teil)

Zwei Begriffe bestimmen zur Zeit die Hauptstossrichtung der Forschung auf dem Gebiet der Informatik: Künstliche Intelligenz und Parallelverarbeitung. Unsere zweiteilige Artikelserie wird sich mit dem zweiten Begriff, der Parallelverarbeitung, befassen. Nebst einer ausführlichen Einführung und Begriffsbestimmung wird der Schwerpunkt auf den verschiedenen Konzepten liegen, mit denen Konflikte im Zusammenhang mit Parallelität gemeistert werden können.

Gesteigerte Ansprüche sowohl an die Leistungsfähigkeit als auch an die Systemzuverlässigkeit von Rechenanlagen, eine zunehmende Vernetzung einzelner Rechner zu Verbünden sowie moderne, dezentrale Rechnerarchitekturen erfordern ein Softwarekonzept, das über die Fähigkeit verfügt, Arbeiten parallel und dezentral auszuführen. Erfüllt werden diese Anforderungen vom Prozesskonzept, einem Konzept, in welchem ein Konglomerat von Prozessen, das sogenannte Prozesssystem, das herkömmliche monolithische Programm ersetzt.

Andreas Pichler

Die parallele Verarbeitung bietet gegenüber der sequentiellen Vorteile in dreierlei Hinsicht. Zum einen gestattet sie es, Arbeiten oder Teilarbeiten einer geeigneten Arbeit gleichzeitig auf mehreren Rechnern zu verrichten. Man gewinnt damit an Leistungsfähigkeit, die proportional mit der Anzahl zur Verfügung stehender Rechner anwächst. Zum zweiten verhilft sie, die nicht zu vermeidenden Wartezeiten (etwa das Warten auf ein Peripheriegerät) besser zu verwenden, indem diese, im herkömmlichen Programm verlorene Zeit zur Verrichtung weniger wichtiger Arbeiten genutzt wird. Auch dies trägt entscheidend zur Steigerung der Leistungsfähigkeit bei. Schliesslich kann Parallelverarbeitung auch zur Sicherung der Systemzuverlässigkeit verwendet werden. Möglich sind etwa die parallele Ausführung gleicher Arbeiten mit anschliessendem Vergleich oder das Umschalten von Prozessen auf redundante Prozessoren, wenn der gerade benutzte Prozessor ausfallen sollte.

Die Aufspaltung in Prozesse erlaubt nicht nur eine Steigerung der Leistungsfähigkeit, sondern unterstützt auch dezentrale Verarbeitung. Erst damit wurde es möglich, ein System auch auf Software-Ebene an die Topologie der Rechnerstruktur anzupassen und die daraus resultierenden Vorteile, wie verkürzte Antwortzeiten und reduzierte Datenmenge, zu nutzen. So können beispielsweise dedizierte Lokalrechner in der Regel schneller auf Eingaben reagieren als Universalrechner. Zudem können bereits vor Ort alle Daten herausgefiltert werden, die nur von lokaler Bedeutung sind. Typische Beispiele für solch ausgeprägt dezentrale Systeme sind Robotersteuerungen. Ohne die entsprechenden Hard- und Software-Möglichkeiten zur Dezentralisierung wären sie kaum denkbar.

Schlüsselbegriffe

MODULA-2, Prozesse, Tasks, Coroutinen, Semaphoren, Monitore, Message Passing, Rendez-Vous, Interrupts, Prioritäten, Konflikte, Deadlock, Synchronisation, Kommunikation, gegenseitiger Ausschluss.

Das herkömmliche Softwarekonzept mit dem Programm als zentralen Begriff ist auf Einzelrechner und sequentielle Verarbeitung zugeschnitten. Ein Programm wird entsprechend der zu erwartenden Datenmenge, Datenhäufigkeit und Datenart programmiert. Als Resultat erhält man ein statisches Gebilde, das zu Beginn optimal in die Umgebung eingebettet ist. Dehnt sich die Umgebung später aber aus (etwa, weil ein neues Peripheriegerät angeschlossen wird), vermag das starre Programm diesen Leerraum nicht oder zumindest nur mit sehr hohem Aufwand auszufüllen.

Anders das Prozesssystem, das man sich aufgrund der Vielzahl seiner Bestandteile eher als verform-, verdünn- und verdichtbare weiche Masse vorzustellen hat und das daher in der Lage ist, sich dynamisch auf Veränderungen der Umgebung einzustellen, ja, sich bei gewissen Implementierungen sogar vermehren oder vermindern kann. Wird beispielsweise ein neuer Drucker angeschlossen, so braucht nur ein weiterer Druckerprozess gestartet werden. Da die Aufgaben meist die selben oder doch sehr ähnlich sind, genügt dazu der Start einer sogenannten Prozessinkarnation, also eines Duplikates des bereits laufenden Druckerprozesses.

Prozessinkarnationen bilden die ökonomische Grundlage eines Prozesssystemes. In vielen Anwendungen genügt es nämlich, einen Prozess einmal zu formulieren und diesen entsprechend den Anforderungen beliebig oft als Inkarnation zu starten. Doch auch andere Formen dynamischer Anpassung sind möglich. Anders als in einem Programm, wo die Reihenfolge der Arbeiten durch den Algorithmus fest vorgegeben ist, kann ein Prozesssystem die Reihenfolge in gewissen Grenzen selbst bestimmen. Produziert ein Datenkanal etwa mehr Daten, als das System erwartet, behilft es sich damit, während dieser Zeit weniger wichtige Aufgaben zurückzustellen. Selbstverständlich gilt es dabei, kausale Zusammenhänge zu berücksichtigen; nicht jede Teilarbeit kann zu einem beliebigen Zeitpunkt ausgeführt werden.

Natürlich lässt sich auch ein Programm, wenigstens auf einem Einzelrechner, mit genügend hohem Aufwand ähnlich flexibel gestalten. Es wird allerdings sehr schnell un-

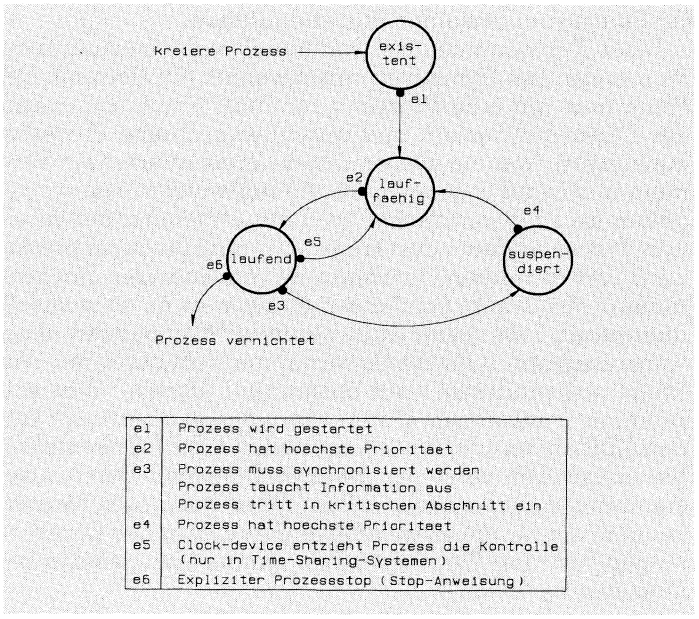


Abb. 1: Prozess-Zustandsdiagramm

übersichtlich (und in Konsequenz dazu sehr teuer), da es zu jedem Zeitpunkt in der Lage sein muss, auf alle möglichen Stimula der Umgebung zu reagieren. Ein Prozess hingegen braucht sich nur um diejenigen Stimula und Daten zu kümmern, die für seine Funktion relevant sind. Nur in den wenigsten Fällen - ein gut ausbalanciertes und weitgehend entkoppeltes Prozesssystem vorausgesetzt - ist er auf Entscheidungen oder Datensätze anderer Prozesse angewiesen. Vergleichen liesse sich ein solches Prozesssystem etwa mit einem gut eingespielten Team von Profis.

Offensichtlich gestattet uns die Formulierung der Problemlösung mittels eines Prozesskonglomerats eine Strukturierung ähnlich dem Prozedurkonzept. Operationen und Datenstrukturen können aus der Masse gelöst und an funktionsspezifische Einheiten, in diesem Fall Prozesse, gebunden werden. Man gewinnt damit eine Erhöhung der Lesbarkeit weit über die Möglichkeiten des Programmes hinaus.

Leider lässt sich diese Strukturierung nicht in der Form von einem Compiler überprüfen, wie wir uns das etwa von Prozeduren her gewöhnt sind. Es gibt zwar sprach-externe Methoden, beispielsweise in Form sogenannter Petri-Netze [4], mit denen sich die Konsistenz von Prozesssystemen mathematisch überprüfen lässt. Allerdings übersteigen solche Berechnungen schon bei Systemen vergleichsweise geringer Komplexität die Leistungsfähigkeit heutiger Rechner. Denn so unmöglich es ist, alle Interaktionsmöglichkeiten in einem Programm zu erfassen, so unmöglich ist es auch, sämtliche Interaktionsmöglichkeiten eines Prozesssystemes zu testen. Aber durch die Strukturierung und die damit gegenüber Programmen verbesserte Lesbarkeit fällt es dem Programmierer leichter, Fehlerquellen zu entdecken und zu reparieren. Angesichts der Tatsache, dass sich die Kosten nur für die Wartung bestehender Software auf etwa 80% der gesamten Softwarekosten belaufen, eine entscheidende Verbesserung.

Neben den aufgezählten Vorteilen bieten Prozesssysteme auch eine Reihe ungeahnter Schwierigkeiten, die hauptsächlich darauf zurückzuführen sind, dass der einzelne Prozess nicht weiss, in welchem Zustand sich die mit ihm kooperierenden Nachbarprozesse befinden. War der Zustand eines Programmes zu einem gegebenen Zeitpunkt noch vollständig durch die Anzahl abgearbeiteter Anweisungen bestimmt, so bestimmt in einem Prozesssystem noch eine weitere Komponente den Systemzustand: die Summe der einzelnen Prozesszustände.

Diese Komponente lässt traditionell einfache Aufgaben zu schwierigen Problemen anschwellen. Ein Beispiel: Ein Programm ruft eine Prozedur producer() auf, die irgend ein Objekt produziert, und anschliessend eine Prozedur consumer(), welche dieses Objekt konsumiert. Kein Problem in einem Programm, wo die Reihenfolge fest vorgegeben ist. Problematischer wird die Angelegenheit in einem Prozesssystem - dort kreisen die zwei Prozesse producer() und consumer() unabhängig von einander. Hat producer() ein Objekt produziert, so kann er es consumer() übergeben, falls dieser dazu bereit ist. Was passiert aber, wenn consumer() für die Uebernahme nicht bereit ist? Wie lange soll producer() auf consumer() warten? Was soll producer() mit seinem Objekt tun, wenn er überhaupt keinen Kontakt zu consumer() herstellen kann? Oder stellen Sie sich zwei Prozesse vor, die je ein Dokument auf dem Systemprinter ausdrucken wollen. Ohne geeignete Mechanismen, die für die folgerichtige Benutzung des Druckers sorgen, würden Sie wohl kaum jemals das gewünschte Dokument erhalten.

Die mit Parallelverarbeitung verbundenen Probleme wurden natürlich schon sehr früh erkannt. In der Folge entwickelten deshalb verschiedene Informatiker - darunter etwa E.W. Dijkstra [1] und C.A.R. Hoare [2] - Konzepte, mit denen sich solche Probleme grundsätzlich lösen lassen. Bevor wir uns aber diesen Koordinierungskonzepten

- dem Semaphoren-Konzept
- dem Monitor-Konzept
- dem Message Passing inklusive Mailbox-Konzept
- dem Rendez-Vous-Konzept

widmen werden, ist es angebracht, noch eine Reihe von Begriffen einführend zu erläutern.

Parallelität

In der Informatik unterscheidet man zwei Formen von Parallelverarbeitung: echt-parallele und quasi-parallele Verarbeitung. Die Unterscheidung ergibt sich aus der Art und Weise, wie die einzelnen Prozesse eines Prozesssystemes in Raum und Zeit verteilt werden:

- Verteilung in der Zeit *(Multi-Programming)*
Der Rechner besteht aus einem einzelnen Prozessor, der unter den lauffähigen Prozessen aufgeteilt wird. Der Ablauf von Prozessen auf einem solchen System erfolgt QUASI-PARALLEL.

- Verteilung im Raum *(Multi-Processing)*
Der Computer besteht aus mehreren identischen Prozessoren, wobei jeder Prozess seinen eigenen Prozessor besitzt. Der Ablauf von Prozessen auf einem solchen System erfolgt ECHT-PARALLEL.

Bemerkungen:

1) Auf Computersysteme mit mehreren unterschiedlichen Prozessoren (typischerweise CPU und Co-Prozessoren oder Controllern) wird hier nicht eingegangen.

2) In der englischen Literatur hat sich der Begriff «concurrent» (gleichzeitig) durchgesetzt. Der Begriff «parallel» ist kaum mehr zu finden.

3) Man beachte in diesem Zusammenhang, dass weder Multi-Programming noch Multi-Processing Echtzeiteigenschaften implizieren.

Das allgemeinste Rechnersystem umfasst P Prozessoren, auf denen p Prozesse ablaufen können. In aller Regel gilt dabei: $p > P$. Alle folgenden Erläuterungen werden sich auf diesen Fall beziehen, wenn nicht explizit ein anderer Fall genannt wird. Ist ein Prozesssystem zum Multi-Processing fähig, so beinhaltet dies in gewisser Weise auch die Fähigkeit zum Multi-Programming. Ein Prozess soll für jedes beliebige Rechnersystem in einheitlichen Begriffen formulierbar sein und muss deshalb so definiert werden, dass er den Anforderungen bezüglich Multi-Processing genügt. Ist das der Fall, so ist das Prozesssystem unabhängig von der späteren Hardware-Implementierung.

Der Prozess

Unter Prozess versteht man eine Einheit für parallele Verarbeitung. Die in einem Prozess spezifizierten Anweisungen werden streng sequentiell ausgeführt, doch der Prozess selbst läuft parallel mit anderen Prozessen ab. Das herkömmliche Konzept kann man als Spezialfall eines

Prozesssystemes mit nur einem einzigen Prozess betrachten.

Aehnlich einer Prozedur muss ein Prozess vor seiner Aktivierung definiert werden. Dies geschieht mittels einer sogenannten Prozessdefinition. Im Gegensatz zur Prozedurdefinition liefert die Prozessdefinition aber nicht nur den Code für einen Prozess, sondern es können beliebig viele Prozesse die selbe Prozessdefinition «fahren» und damit eine ganze Familie gleichartiger Prozesse bilden. Im Fachjargon spricht man dann bisweilen von sogenannten Prozessinkarnationen *(process instances)*. Die jeweils zu benützende Prozessdefinition wird dem Prozess erst beim Prozessstart, der entweder explizit durch eine Startanweisung (wie in Modula-2 und CHILL) oder implizit beim Hochfahren des gesamten Prozesssystemes (wie in ADA) erfolgt, bekannt gegeben.

In der Regel werden Prozesse in Form einer Endlosschleife programmiert, «loopen» also normalerweise während der gesamten Lebensdauer des Prozesssystemes (auf voll dynamisch konfigurierbare Prozesssysteme, wie sie etwa in CHILL möglich sind, wird hier nicht weiter eingegangen). Dabei können Umstände eintreten, die es erforderlich machen, dass ein Prozess auf ein bestimmtes Ereignis warten muss. In einem solchen Wartezustand ist es aber nicht nötig, dass er Betriebsmittel (Ressourcen) des Rechners, wie etwa Prozessorzeit, Speicher oder I/O-Medien, in Beschlag nimmt, die währenddessen von anderen Prozessen benutzt werden könnten. Er wird deshalb in einen Zustand versetzt, in welchem er auf das Ereignis warten kann, ohne dabei Ressourcen zu beanspruchen - man sagt, er wird suspendiert (im hier verwendeten Kontext ist suspendiert gleichbedeutend mit blockiert!). Dies geschieht normalerweise dadurch, indem man ihn in eine Warteschlange einreiht, die mit diesem Ereignis verknüpft ist. Nach Eintritt des Ereignisses wird der in der entsprechenden Warteschlange höchst priorisierte Prozess wieder in den lauffähigen Zustand versetzt, was letztlich nichts anderes bedeutet, als dass er gleich in eine weitere Warteschlange (meist als Ready-Queue bezeichnet) eingereiht wird. In dieser Queue warten lauffähige Prozesse darauf, dass ihnen Prozessorzeit zugeteilt wird. Erst wenn der Prozess die Kontrolle über den Prozessor zurückerlangt, befindet er sich wieder im laufenden Zustand. Zu diesen drei Zuständen - laufend, lauffähig und suspendiert - kommt noch ein weiterer hinzu, den man als existierend bezeichnet. Existierend ist ein Prozess dann, wenn er zwar kreiert, dem Prozesssystem aber noch nicht angegliedert wurde.

Gelegentlich bezeichnet man den Zustand existierend (zusammen mit den Pseudo-Zuständen definiert und terminiert) als unaktualisiert, die anderen drei Zustände als aktualisiert. Das Umschalten von einem Zustand in den anderen wird Prozesszustandsumschaltung genannt. Abb.1 zeigt ein Zustandsdiagramm, das ein mögliches (!) Verhalten von Prozessen beschreibt. Die Betonung liegt dabei auf möglich. Für spezifische Implementierungen kann sich durchaus ein anderes Bild ergeben. Im allgemeinen trifft das hier präsentierte Zustandsdiagramm allerdings zu, wobei im konkreten Fall der Suspendierungszustand meist verfeinert dargestellt wird.

Die dem Prozess zugrunde liegende Prozessdefinition enthält keinerlei Angaben für die Prozesszustandsumschaltung. Jeder Prozess wird vielmehr so formuliert, als ob er unterbrechungsfrei für sich alleine liefe. Dies ist nötig, um ihn implementierungsunabhängig zu halten. Soll er ein Betriebsmittel benutzen, auf einen anderen Prozess warten oder mit diesem kommunizieren, so verwendet man dafür die Anweisungen des entsprechenden Koordinierungskonzeptes. Diese sind derart implementiert, dass sie ein spezielles Kontrollprogramm, den sogenannten Scheduler, aufrufen. Er sorgt dafür, dass die zu Verfügung stehenden Ressourcen das Rechnersystemes optimal bezüglich vorher zu definierender Zielformulierungen genutzt werden können. Dies schliesst in erster Linie die Suspendierung von Prozessen und deren Einreihung in entsprechende Warteschlangen ein, wenn sie Systemkomponenten anfordern, die momentan nicht frei sind. Möglich ist aber auch die Suspendierung aufgrund irgendwelcher Zeitkriterien. Wir werden uns etwas später noch eingehend mit all diesen Umständen befassen.

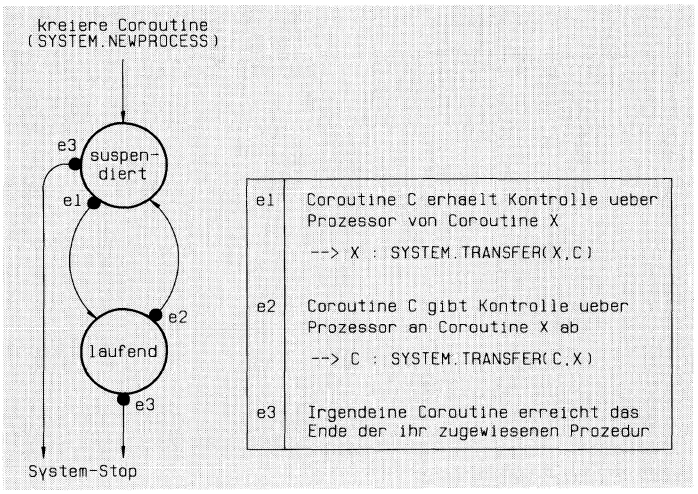


Abb. 2: Aktualisierte Zustände einer Coroutine

Im allgemeinen sind Scheduler, zumal dann, wenn sie sich noch über mehrere Einzelrechner verteilen, ziemlich komplex. Man verfrachtet sie deshalb auch meistens in ein Betriebssystem (UNIX, iRMX usw.). Zusammen mit einem solchen Betriebssystem und einer vom Konzept her se-

```
DEFINITION MODULE Processes;

 EXPORT QUALIFIED  SIGNAL,SEND,WAIT,StartProcess,Awaited,Init;

 TYPE   SIGNAL;

 PROCEDURE StartProcess (P: PROC; n: CARDINAL);
 (*
    Startet einen Prozess mit Prozessdefinition P und einem
    Arbeitsspeicher der Grösse n.
    P erhält die Kontrolle über den Prozessor !
 *)

 PROCEDURE SEND (VAR s: SIGNAL);
 (*
    Uebergibt die Kontrolle über den Prozessor an den Prozess
    mit der höchsten Priorität in der mit dem Signal s assozi-
    ierten Warteschlange.
    Wartet kein Prozess auf das Signal s, bleibt SEND() ohne
    Wirkung !
 *)

 PROCEDURE WAIT (VAR s: SIGNAL);
 (*
    Suspendiert den laufenden Prozess und reiht ihn in die mit
    dem Signal s assoziierte Warteschlange ein. Der Prozess
    bleibt so lange in der Warteschlange, bis ein anderer Pro-
    zess SEND(s) ausführt und er die höchste Priorität in der
    Warteschlange hat.
    Wenn alle Prozesse auf ein Signal warten, stoppt WAIT()
    das gesamte System !
 *)

 PROCEDURE Awaited (s: SIGNAL): BOOLEAN;
 (*
    Testet, ob mindestens ein Prozess auf das Signal s wartet !
 *)

 PROCEDURE Init (VAR s: SIGNAL);
 (*
    Zwingende Initialisierung des SIGNAL-Objektes s !
 *)

END Processes.
```

Abb. 3: Definitions-Modul des Scheduler Processes

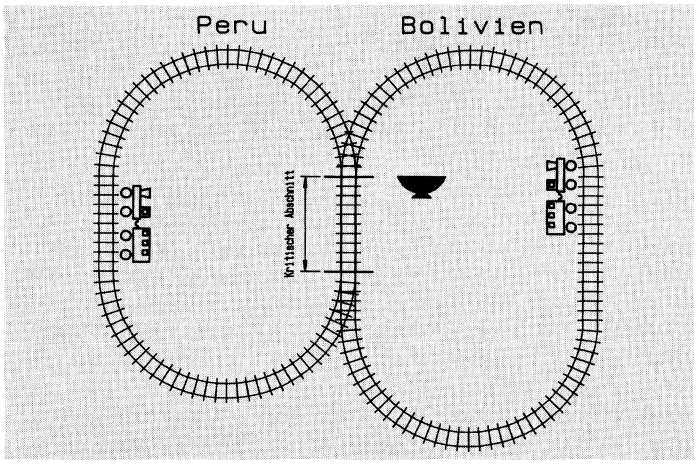


Abb. 4: Die Andenbahn

quentiellen Sprache wie etwa Pascal ist man in der Lage, ein Prozesssystem zu bilden, indem man an den entsprechenden Stellen des Programmes geeignete Aufrufe von Schedulerfunktionen aus dem Betriebssystem einfügt. Daneben gibt es aber auch Programmiersprachen, die den Scheduler bereits in ihrem Sprachumfang integriert haben. Typische Vertreter solcher hochentwickelten Sprachen sind beispielsweise CHILL (CCITT High Level Language, vom CCITT definierte und empfohlene Sprache für den Telecom-Bereich) und ADA. Nicht dazu gehört Modula-2. Dort verfolgt man eine andere Philosophie, die Sie bei der Diskussion zu den Coroutinen noch näher kennenlernen werden.

In den bisherigen Ausführungen wurde nur der Begriff Prozess verwendet. In der Literatur tauchen aber auch noch die Begriffe Task und Coroutine auf. Im folgenden werden diese beiden Begriffe jeweils dem Prozessbegriff gegenübergestellt:

a) Prozesse und Tasks

Der Begriff Task wird vor allem in der englischsprachigen Welt benutzt. Im allgemeinen bezeichnet man damit die Gesamtheit aller Aktionen, die für die Erledigung der gestellten Aufgabe notwendig sind. Meint man mit Task hingegen nur eine Parallelitätseinheit, so ist zwischen Task und Prozess kein Unterschied festzustellen; es sind dann vielmehr zwei Worte für den selben Begriff.

b) Prozesse und Coroutinen

In vielen Anwendungen sind Sprachen wie CHILL oder ADA nicht die geeigneten Mittel. Zum einen benötigen sie viel Speicherplatz, weil jeweils noch der gesamte Scheduler zum Programm hinzugebunden werden muss, zum anderen ist der auf Universalität getrimmte Scheduler oft viel zu gross und manchmal auch zu langsam für die gewünschte Applikation. Hinzu kommt der enorm hohe Preis, der für solche Sprachen und die dazu gehörenden Entwicklungssysteme zu bezahlen ist.

Eine andere Philosophie verfolgt man in Modula-2. Als äusserst kompakte und auf Effizienz ausgerichtete Implementierungssprache (Ersatz für PL/M, C oder Assembler) verfügt Modula-2 über keinen integrierten Scheduler, unterstützt aber über den Pseudo-Modul SYSTEM das sogenannte Coroutinenkonzept.

In gewisser Weise ähneln Coroutinen Prozeduren. Während aber eine Prozedur immer mit ihrer ersten Anweisung beginnt, dann vollständig abgearbeitet wird und nach getaner Arbeit zum aufrufenden Programmteil zurückkehrt, verfügen Coroutinen über die Möglichkeit, die Kontrolle über den Prozessor zwischenzeitlich und ohne Auftragsverhältnis (d.h. die Uebergabe bedingt nicht notwendigerweise eine Rückkehr) an eine andere Coroutine abzugeben. Dabei wird ihr momentaner Zustand eingefroren. Erhält sie die Kontrolle über den Prozessor zurück, so führt sie am Unterbrechungspunkt weiter. Coroutinen sind damit eine ideale Grundlage für ein Multi-Programming-System. Allerdings gilt es dabei folgende Unterschiede zu einem auf Prozessen beruhenden Prozesssystem zu beachten:

- Coroutinen sind keinem hierarchisch übergeordnetem Programmteil unterstellt. Daraus folgt, das die Uebergabe der Prozessorkontrolle explizit unter Nennung sowohl der übergebenden als auch der übernehmenden Coroutine zu erfolgen hat. Dies im Gegensatz zu Prozessen, deren Prozessdefinition keine Angaben zur Zustandsumschaltung enthält.
- Die explizite Abgabe der Prozessorkontrolle und das Fehlen eines Schedulers implizieren eine duale Zustandsform. Eine Coroutine kennt lediglich die aktualisierten Zustände laufend und suspendiert (vgl. Abb.2), während ein Prozess aufgrund des vom Scheduler verwalteten Warteschlangensystemes drei aktualisierte Zustände (laufend, lauffähig und suspendiert) kennt.
- Das Coroutinenkonzept basiert auf Einprozessorsystemen (Jump-Anweisung). Zudem impliziert auch die explizite Prozessorübergabe ein Einprozessorsystem, da in einem Multi-Processing-System der Prozess die Kontrolle über den Prozessor nicht notwendigerweise abgeben muss, wenn er auf system-globale Resourcen warten soll. Auch damit unterscheidet sich eine Coroutine grundsätzlich vom Prozess, der per Definition sowohl in einer echt- als auch in einer quasi-parallelen Umgebung laufen kann.

Offensichtlich gelingt es mit dem Coroutinen-Konzept, den aufwendigen Scheduler aus dem eigentlichen Sprachumfang zu streichen, dem Anwender aber dennoch eine einfache Form des Multi-Programming zu erlauben. Modula-2 ist aber noch weitaus flexibler, wie die folgende Aufstellung zeigen soll:

- Genügt dem Anwender das Coroutinen-Konzept, so kann er direkt mit den zur Verfügung gestellten Einrichtungen arbeiten.

Vorteil: Einfache Form des Multi-Programming.
Nachteil: Da die Zustandsumschaltung auf die Applikationsebene, also zu den einzelnen Coroutinen, verlagert wird, muss sich der Anwenderprogrammierer jeweils selbst um das Coroutinen-Management kümmern.

- Soll der Anwenderprogrammierer vom Coroutinen-Management entlastet werden, so kann der in den meisten Modula-Implementierungen vorhandene Scheduler Processes verwendet werden.

Vorteil: Entlastung vom Coroutinen-Management und Arbeiten mit prozessähnlichen Parallelitätseinheiten.
Nachteil: Beschränkung auf Einzelprozessorsysteme und quasi-parallele Verarbeitung.

- Benötigt der Anwender den allgemeineren Prozessbegriff, so kann ein entsprechender Scheduler in Modula-2 geschrieben und dem Anwender in Form eines Modules zur Verfügung gestellt werden.

Vorteil: Der Scheduler kann den Bedürfnissen entsprechend geschrieben werden.

Nachteil: Die Entwicklung, gegebenenfalls Weiterentwicklung, des Schedulers erfordert einen zusätzlichen Aufwand, der beim Einsatz von Sprachen wie CHILL und ADA enfällt.

Das Coroutinen-Konzept zusammen mit dem Modul-Konzept erlaubt es dem Anwender, die jeweils günstigste Form der Parallelverarbeitung zu verwenden. Er ist damit in der Lage, Modula-2 weit selektiver seinen Bedürfnissen anzupassen, als das mit den elefantös wirkenden Prozesssprachen CHILL und ADA möglich ist. Es kommt daher nicht von ungefähr, dass Experten Modula-2 weit höhere Chancen einräumen als etwa ADA, die ja ebenfalls Anspruch darauf erhebt, allgemein verwendbar zu sein (bei CHILL liegt der Fall etwas anders, da die Sprache speziell auf die Bedürfnisse der Telekommunikationsbranche zugeschnitten wurde).

Betrachten wir uns deshalb das Coroutinen-Konzept etwas genauer in der Implementierungsform von Modula-2:

a) Definition von Coroutinen

Als Definition für eine Coroutine dient eine Prozedurdefinition. Dabei sind folgende Regeln zu beachten:

- Als Coroutinendefinition sind nur parameterlose Prozeduren gestattet. Eingeschränkte Sichtbarkeit auf Daten erreicht man dadurch, indem man die Coroutinendefinition in einen lokalen Modul verkapselt.
- Das gesamte System stoppt, wenn eine Coroutine das Ende der ihr zugeordneten Prozedur erreicht. Insbesondere ist es daher nicht gestattet, eine der Coroutine zugrunde liegende Prozedur lokal zu einer anderen Prozedur zu deklarieren. Zudem sind Prozeduren im allgemeinen zyklisch geschlossen (LOOP-Anweisung) zu programmieren.
- Die Umwandlung einer Prozedur in eine Coroutine erfolgt unter Verwendung der Prozedur SYSTEM.NEWPROCESS(), welche die Coroutine vom unaktualisierten in den aktualisierten Zustand versetzt (definiert/existierend → suspendiert).

b) Ablauf

Der Ablauf von einem Coroutinen-System erfolgt quasiparallel auf einem Einprozessorsystem. Zu jedem Zeitpunkt hat genau eine Coroutine die Kontrolle über den Prozessor. Die Uebergabe der Kontrolle geschieht folgendermassen:
- Die Kontrolle über den Prozessor wird von einer Coroutine mittels der Anweisung SYSTEM.TRANSFER() an eine andere Coroutine abgegeben. Bei der Uebergabe ist sowohl die übergebende als auch die übernehmende Coroutine zu nennen. Selbstverständlich muss dabei die übernehmende Coroutine zuvor mit SYSTEM.NEWPROCESS() aktualisiert worden sein.
- Bei Abgabe der Kontrolle wird der momentane Zustand der Coroutine eingefroren. Erhält die Coroutine die Kontrolle über den Prozessor wieder zurück, so fährt sie an der Stelle weiter, die der Transferanweisung unmittelbar folgt.
- Erreicht eine Coroutine das Ende der ihr zugeordneten Prozedur, so stoppt das gesamte System.
- Der Run-Time-Support (RTS) von Modula-2 startet implizit eine erste Coroutine (definiert/existierend → laufend). Als Definition verwendet diese Coroutine den Codebereich des Programm-Modules. Die Zuweisung einer Prozessvariablen für diese Coroutine erfolgt ebenfalls implizit bei der ersten Transfer-Anweisung und nicht mittels SYSTEM.NEWPROCESS().

```
DEFINITION MODULE BinSema;

 EXPORT QUALIFIED SEMAPHORE,InitSemaphore,P,V;

 TYPE  SEMAPHORE;

 PROCEDURE  InitSemaphore(VAR s : SEMAPHORE);
   (* Initialisiere Semaphor s *)

 PROCEDURE  P(VAR s : SEMAPHORE);
   (* Semaphor-Operation "Proberen" *)

 PROCEDURE  V(VAR s : SEMAPHORE);
   (* Semaphor-Operation "Verhogen" *)

END BinSema.
```

Abb. 5a: Definitions-Modul BinSema

```
IMPLEMENTATION MODULE BinSema [7];

 FROM SYSTEM    IMPORT TSIZE;
 FROM Storage   IMPORT ALLOCATE;
 FROM Processes IMPORT SIGNAL,WAIT,SEND,Init;

 TYPE  SEMAPHORE = POINTER TO SemaphoreDescriptor;
       SemaphoreDescriptor = RECORD
                               CriticalSection : (free,used);
                               BecomeFree      : SIGNAL
                             END;

 PROCEDURE  InitSemaphore(VAR s : SEMAPHORE);
   (* Initialisiere Semaphor s *)
  BEGIN
   ALLOCATE(s,TSIZE(SemaphoreDescriptor));
   WITH s^ DO
    CriticalSection := free;
    Init(BecomeFree)
   END
  END InitSemaphore;

 PROCEDURE  P(VAR s : SEMAPHORE);
   (* Semaphor-Operation "Proberen" *)
  BEGIN
   WITH s^ DO
    IF CriticalSection = used THEN WAIT(BecomeFree) END;
    CriticalSection := used
   END
  END P;

 PROCEDURE  V(VAR s : SEMAPHORE);
   (* Semaphor-Operation "Verhogen" *)
  BEGIN
   WITH s^ DO
    CriticalSection := free;
    SEND(BecomeFree)
   END
  END V;

END BinSema.
```

Abb. 5b: Implementations-Modul Bin-Sema

Bemerkung:

Die meisten zur Zeit erhältlichen Implementierungen verwenden als Datentyp für Prozessvariablen noch SYSTEM.PROCESS. In der dritten Überarbeiteten Auflage von [6] wird als Datentyp jedoch SYSTEM.ADDRESS spezifiziert!

Wie bereits angedeutet, bieten die meisten Modula-Implementierungen den Modul Processes an. Processes stellt einen einfachen Scheduler dar, der uns von den maschinennahen Coroutinen wieder in die Abstraktion des Prozessbegriffes zurückführt. Zwar liegt der Implementierung von Processes in [6] das Coroutinen-Konzept zugrunde, doch dies ist, und darauf sei ausdrücklich hingewiesen, nur eine mögliche Implementierungsform. Ohne weiteres könnte nämlich die Implementierung auch auf einem bereits im unterlagerten Betriebssystem (etwa VMS auf einer VAX) integrierten Scheduler basieren. Unter Verwendung dieses Modules sind wir daher in der Lage, unsere Prozessformulierung von der zugrunde liegenden maschinellogischen Implementation zu abstrahieren.

Processes stellt einen Mechanismus zur Verfügung, der mit sogenannten Signalen arbeitet. Mit Signalen teilt ein Prozess seinen Nachbarprozessen mit, dass ein bestimmtes Ereignis eingetreten ist, beispielsweise, dass in einem

Buffer ein Datenobjekt bereitgestellt worden ist und nun abgeholt werden kann. Für das Aussenden solcher Signalen exportiert Processes zwei Prozeduren, die wir im folgenden etwas eingehender betrachten wollen:

- SEND(sig)
Bei Aufruf dieser Prozedur wird die Kontrolle vom gerade laufenden Prozess auf den Prozess übertragen, der in der mit dem Signal sig assoziierten Warteschlange auf dieses Ereignis wartet und dort die höchste Priorität hat. Wartet kein Prozess auf dieses Signal, bleibt SEND ohne Wirkung.

- WAIT(sig)
Bei Aufruf dieser Prozedur wird die Kontrolle vom gerade laufenden Prozess an den Prozess übergeben, der in der Ready-Queue die höchste Priorität hat. Der WAIT aufrufende Prozess wird suspendiert und in die mit dem Signal sig assoziierte Warteschlange eingereiht.

Offensichtlich besorgen diese beiden Prozeduren die Prozesszustandsumschaltung in diesem System. Sie benutzen dazu zwei Warteschlangen. Eine Warteschlange dient dabei der Blockierung von Prozessen, die mit WAIT signalisieren, dass sie auf ein bestimmtes Ereignis warten wollen. Die andere Warteschlange, die Ready-Queue, dient zur Aufnahme von Prozessen, die bereit sind, ihre Arbeit wieder aufzunehmen und sich deshalb um Rechnerzeit bewerben. Damit in späteren Erläuterungen keine Probleme im Zusammenhang mit diesem Modul auftauchen, zeigt Abb. 3 dessen Definitionsteil. Auf die Wiedergabe des Implementationsteiles möchte ich allerdings bewusst verzichten, da er für die weiteren Betrachtungen ohne Belang ist.

```
MODULE BoundedBuffer [7];

 EXPORT append,remove;
 IMPORT SIGNAL,SEND,WAIT,Init,portion;

 CONST  N = AnyValue;

 VAR    buffer   : ARRAY [0..N-1] OF portion;
        n        : [0..N];
        in,out   : [0..N-1];
        nonfull  : SIGNAL;
        nonempty : SIGNAL;

 PROCEDURE append (x : portion);

  BEGIN
   INC(n); IF n > N THEN WAIT(nonfull) END;
   buffer[in] := x; in := (in + 1) MOD N;
   IF n <= 0 THEN SEND(nonempty) END
  END append;


 PROCEDURE remove (VAR x : portion);

  BEGIN
   DEC(n); IF n < N THEN WAIT(nonempty) END;
   x := buffer[out]; out := (out + 1) MOD N;
   IF n >= N THEN SEND(nonfull) END
  END remove;


BEGIN (* init module *)
 n := 0; in := 0; out := 0;
 Init(nonfull); Init(nonempty);
END BoundedBuffer;
```

Abb. 6: Lokaler Monitor-Modul BoundedBuffer

Prozessinteraktionen

In der Regel kann ein Prozess den grössten Teil seiner Aufgaben verrichten, ohne sich dabei um die Umgebung kümmern zu müssen. Nur gelegentlich treten Umstände auf, die ihn dazu zwingen, sich mit Nachbarprozessen abzustimmen, sei es, um die Integrität eines Datensatzes oder eines Ablaufes sicherzustellen oder auch nur, um dem System die Gelegenheit zu geben, mögliche Wartezeiten durch Aktivierung eines anderen Prozesses zu nutzen. Trifft er auf einen solchen Umstand, signalisiert er dies und ruft dabei implizit den Scheduler auf (vgl. SEND und WAIT). Dieser kann dann entscheiden, ob der betreffende Prozess weiterfahren darf oder ob die Kontrolle einem anderen Prozess übergeben werden soll. Grundsätzlich lassen sich drei solche Umstände unterscheiden:

- Synchronisation
Zwischen zwei oder auch mehreren Prozessen ist eine zeitliche Reihenfolge einzuhalten. Ein Prozess «consumer» muss beispielsweise darauf warten, dass ein Prozess «producer» das entsprechende Konsumationsobjekt zu Verfügung stellt. Der Begriff Konsumationsobjekt darf dabei nicht zu eng gesehen werden. Es kann sich durchaus um ein Datenobjekt handeln, genauso gut aber auch einfach um eine Bedingung zur Steuerung des Ablaufes oder eines Buffers ohne eigentlichen Dateninhalt.

- Kommunikation
Zwischen zwei oder auch mehreren Prozessen müssen Informationen ausgetauscht werden. Ein solcher Austausch wird erreicht, indem ein Prozess die Information direkt an den oder die Empfänger schickt, oder, indem die Informa-

Literatur

[1] E.W. Dijkstra; Co-operating Sequential Processes, Programming Languages, Academic Press, 1968

[2] C.A.R. Hoare; Monitors: An Operating System Structuring Concept, Communication of the ACM, Vol. 17, Number 10, October 1974

[3] P. Wegner, A.S. Smolka; Processes, Tasks, and Monitors: A Comparative Study of Concurrent Programming Primitives, IEEE Transaction on Software Engineering., Vol. SE-9, No. 4, July 1983

[4] W. Reisig, Petrinetze; Eine Einführung Springer Verlag, 1986 (2. Auflage)

[5] CHILL - A Self-Introduction Manual Vol. I,II,III Philips, 1983

[6] N. Wirth; Programming in Modula-2, Third corrected edition, Springer Verlag, 1985

[7] Modula-2/86 - User's Manual Logitech SA, 1984

[8] Disk Operating System Version 3.00 - Technical Reference International Business Machines Corp., 1984

[9] Personal Computer - Technical Reference International Business Machines Corp., 1983

[10] Introduction to CHILL CCITT, Study Croup XI, 1980

[11] CHILL Language Definition CCITT, Rec. Z.200 (1984), 1984

tion in einem Buffer hinterlegt wird, von wo sie von dem oder den Empfängerprozessen ausgelesen werden kann. Im allgemeinen ist eine Prozesskoordination zum Zwecke der Kommunikation wesentlich lockerer gekoppelt als zu Synchronisationszwecken. So muss beispielsweise der Prozess «consumer» nur dann auf Konsumationsobjekte warten, wenn der Buffer völlig leer ist. Ist er hingegen teilweise gefüllt, laufen beide Prozesse unabhängig voneinander. Offensichtlich gleichen sich Synchronisation und Kommunikation in ihrer Wirkung, wenn der Buffer nur ein einziges Datenobjekt aufnehmen kann. In der Literatur werden die beiden Koordinationsumstände deshalb nicht immer scharf auseinandergehalten. Es sei aber ausdrücklich darauf hingewiesen, dass Synchronisation und Kommunikation verschiedene Ziele verfolgen: Synchronisation, um eine Reihenfolge zu erzwingen und Kommunikation, um Informationen auszutauschen!

- Gegenseitiger Ausschluss
Der gleichzeitige Zugriff auf Daten oder der gleichzeitige Gebrauch von Ressourcen von zwei oder auch mehreren Prozessen muss verhindert werden. So muss im allgemeinen sichergestellt werden, dass ein (höher priorisierter) Prozess nicht zwischendurch ein Datenobjekt verändert, während es gerade von einem anderen Prozess gelesen wird. Auch zwischen gegenseitigem Ausschluss und Synchronisation lässt sich nicht immer eindeutig unterscheiden, wenn man nur die Wirkung betrachtet.

Bei dieser Aufzählung gehe ich von der Annahme aus, dass sich das Prozesssystem selbständig verwaltet. Selbständig verwalten heisst dabei, dass ein Prozess solange die Kontrolle über den Prozessor behält, bis er auf einen in der Aufzählung erwähnten Umstand trifft. Erst mit der Signalisierung dieses Umstandes wird der Scheduler aktiv. Daneben gibt es natürlich noch andere Systeme, bei denen der Scheduler aktiv aufgrund bestimmter Zeitkriterien in das Prozessgeschehen eingreift. Den Trivialfall bildet dabei das Time-Sharing-System. Dort bildet ein fester Systemtakt das einzige Kriterium für die Prozesszustandsumschaltung. Prozesse können noch so dringende Arbeiten ausführen - wenn ihr Zeitschlitz abgelaufen ist, werden sie vom Scheduler erbarmungslos suspendiert. Komplexere Systeme berücksichtigen alle Umstände. So erfolgt beispielsweise die Zustandsumschaltung in einem System mit «deadline scheduling» im wesentlichen aufgrund der aufgezählten Umstände. Der Scheduler mischt sich erst dann aktiv in das Geschehen ein, wenn der jeweilige Prozess auch nach Ablauf einer gewissen Zeitspanne auf keinen Umstand getroffen ist, der den Scheduler aktiviert. Darüberhinaus können Prozesssysteme in ein Prioritätsschema verpackt werden, das Scheduler-Aktivitäten nur dann erlaubt, wenn höher priorisierte Prozesse den Prozessor verlangen. In der Regel gilt, dass man Time-Sharing- oder Deadline-Scheduling-Verfahren eher in der EDV antrifft, während rein selbstverwaltete, eventuell durch gewisse Echtzeiteigenschaften ergänzte Prozesssysteme die Domäne der PDV sind.

Koordinationsprobleme

Es gibt eine Unzahl von Problemen, die bei der Koordination von Prozessen auftreten. Eine Vielzahl davon gehört zum sogenannten Exception-Handling. Es sind dies alle Fälle, wo eine gewollte Synchronisation oder Kommunikation nicht stattfinden kann, weil die zeitliche Prozesskonstellation dies verunmöglicht. Allein die Behandlung all dieser Fälle würde Bände füllen und ginge weit über

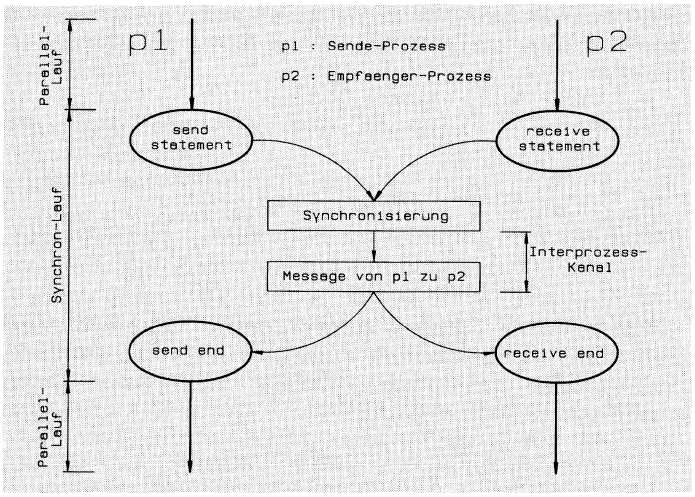


Abb. 7: Message Passing

den Rahmen dieses Artikels hinaus. Ich möchte die Diskussion deshalb eingrenzen und auf ein bestimmtes Gebiet konzentrieren, das die grundsätzliche Problematik sehr schön auszudrücken vermag. Anhand der schon fast legendären Andenbahn werden wir uns in die Schwierigkeiten des gegenseitigen Ausschlusses einarbeiten. Die dabei diskutierten Probleme sollen dazu dienen, das Vorstellungsvermögen bezüglich Parallel-Verarbeitung zu schärfen, indem sie die wirklich grundlegenden Gefahren DEADLOCK, CRASH und RESTRIKTION aufzeigen.

Zur Ausgangslage

Irgendwo in den Anden verbindet ein gemeinsames Teilstück die beiden nationalen Streckennetze von Peru und Bolivien (siehe Abb. 4). Selbstverständlich kann nur immer jeweils ein Zug durch dieses Teilstück, den kritischen Abschnitt (KA), fahren. Die zwei Zugführer benutzen eine Schale, um zu entscheiden, wer durch den kritischen Abschnitt fahren darf und wer warten muss. Dazu prüft jeder Zugführer zuvor, ob sich bereits ein Stein in der Schale befindet. Ist das der Fall, so wartet er; befindet sich hingegen kein Stein in der Schale, so sucht er einen, legt diesen hinein und fährt anschliessend durch den kritischen Abschnitt. Nach der Durchfahrt sorgt er mittels eines ausgeklügelten Schalenmechanismusses, dass der Stein in der Schale am Beginn des kritischen Abschnittes wieder entfernt wird und so den Weg für den nächsten Zugführer frei macht. Um die Analogie zum Computer noch weiter zu treiben, müssen wir davon ausgehen, dass die beiden Zugführer blind und taubstumm sind und lediglich ihren Tastsinn für die Schalenoperation zur Verfügung haben. Sie sehen selbst, es herrschen wahrlich seltsame Zustände in den Anden!

Etwas formaler ausgedrückt, folgen die beiden Zugführer(-Prozesse) dem folgenden Algorithmus:

```
LOOP (* Zugführerleben *)
 WHILE "Schale nicht leer" DO
  "Siesta"
 END
 "Suche Stein"
 "Lege Stein in Schale"
 "Fahre durch kritischen Abschnitt (KA)"
 "Entferne Stein aus Schale"
 "Fahre restliche Strecke"
END (* Zugführerleben *)
```

Halten sich beide Zugführer streng an dieses Verfahren, so werden sie in den weitaus meisten Fällen unfallfrei den kritischen Abschnitt (KA) passieren können. Was passiert

aber, wenn beide nahezu gleichzeitig zum gemeinsamen Teilstück kommen? Betrachten wir dazu den folgenden Ablauf:

```
|--------- Peru ---------------|----------- Bolivien ----------|

Findet leere Schale vor
"Suche Stein"          <---------->  Findet leere Schale vor
"Lege Stein in Schale"               "Suche Stein"
"Fahre durch KA"                     "Lege Stein in Schale"
Fährt durch KA                       "Fahre durch KA"
                          Zeit
```

Sich keines Vergehens schuldig, fährt der blinde und taubstumme bolivianische Zugführer dem blinden und taubstummen peruanischen Zugführer in dessen Zugskombination hinein → CRASH!

> Der gegenseitige Ausschluss ist nicht genügend gesichert; der Gebrauch des kritischen Abschnittes ist zu wenig exklusiv!

Es ist kaum anzunehmen, dass Sie in einem solchen Zug mitfahren wollen. Um das Geschäft mit dem Tourismus nicht zu gefährden, sollten wir eine andere Lösung suchen. Eine Möglichkeit wäre beispielsweise, dass sich die Zugführer unterschiedlicher Algorithmen bedienen:

```
LOOP (* Bolivianisches Zugführerleben *)
 WHILE "Schale nicht leer" DO
  "Siesta"
 END
 "Fahre durch kritischen Abschnitt"
 "Lege Stein in Schale"
 "Fahre restliche Strecke"
END (* Bolivianisches Zugführerleben *)

LOOP (* Peruanisches Zugführerleben *)
 WHILE "Schale leer" DO
  "Siesta"
 END
 "Fahre durch kritischen Abschnitt"
 "Entferne Stein aus Schale"
 "Fahre restliche Strecke"
END (* Peruanisches Zugführerleben *)
```

Durch die zueinander inversen Schalenbedingungen vermeiden wir nun zwar Auffahrunfälle, erzwingen aber auch eine streng alternierende Reihenfolge der Durchfahrt → RESTRIKTION!

> Der gegenseitige Ausschluss ist genügend gesichert, doch der Gebrauch des kritischen Abschnittes ist zu exklusiv!

Stellen Sie sich vor, die peruanische Regierung beschliesst eine Verbesserung der Zugverbindung in ihren Andenregionen und möchte deshalb ihre Züge doppelt so oft verkehren lassen - ein aussichtsloses Unterfangen, wenn die bolivianische Regierung nicht ebenfalls mitzieht. Oder gar, dass in einem der beiden Länder eine Revolution ausbricht und dort den Zugverkehr zum Erliegen brächte - binnen kürzester Zeit käme er auch im anderen Land zum Erliegen.

Die Lösung hat noch einen weiteren Nachteil. Es ist nicht gerade sehr effizient, wenn man für zwei ansich identische Prozesse (Zugführerleben) unterschiedliche Prozessdefinitionen benötigt. Dies unterläuft die eigentliche Oekonomie von Prozesssystemen, die davon ausgeht, dass ähnliche Prozesse mit der gleichen - eventuell durch Parameter leicht modifizierten - Prozessdefinition gefahren werden können. Verwerfen wir deshalb auch diese Lösung. Die beiden Bahnen fahren schon so genug Defizit ein und wir sollten sie nicht mit noch mehr Restriktionen und weiteren Ausgaben belasten. Versuchen wir es stattdessen lieber mit einem neuen Ansatz. Was passiert beispielsweise, wenn wir statt einer Schale deren zwei verwenden. Um wieder mit identischen Algorithmen für beide Zugführer arbeiten zu können, geben wir den beiden Schalen relative Namen:

```
LOOP (* Zugführerleben *)
 "Lege Stein in MEINE Schale"
 WHILE "DEINE Schale nicht leer" DO
  "Siesta"
 END
 "Fahre durch kritischen Abschnitt"
 "Entferne Stein aus MEINER Schale"
 "Fahre restliche Strecke"
END (* Zugführerleben *)
```

Beabsichtigt ein Zugführer durch den kritischen Abschnitt zu fahren, legt er einen Stein in seine bzw. MEINE Schale hinein. Damit blockiert er zugleich die Durchfahrtsberechtigung für den zweiten Zugführer. Dieser kann zwar seinen Stein auch schon in die Schale legen, hat dann aber zu warten, bis die andere bzw. DEINE Schale wieder geleert wird. Auf den ersten Blick scheint diese Lösung der Stein der Weisen zu sein; eine Kollision der Züge ist ausgeschlossen, auch wird keine feste Reihenfolge mehr vorgeschrieben. Doch wehe für die armen Passagiere, wenn die Züge wieder beinahe gleichzeitig durch den kritischen Abschnitt fahren wollen:

```
|--------- Peru ---------------|----------- Bolivien ----------|

"Lege Stein in MEINE Schale"   | "Lege Stein in MEINE Schale"
Siesta, bis DEINE Schale leer  | Siesta, bis DEINE Schale leer
                          Zeit
```

Da warten die beiden blinden, taubstummen Zugführer nun, derweil die Passagiere in der dünnen Luft der Anden darben. Durch die beiden Schalenoperationen blockieren sie sich gegenseitig bis in alle Ewigkeit → DEADLOCK!

> Der gegenseitige Ausschluss ist zu fest gesichert; das System kann blockiert werden!

Mit diesen drei Beispielen haben wir alle grundsätzlichen Probleme aufgezeigt. Fassen wir sie noch einmal etwas allgemeiner zusammen:

- *CRASH*
Die Integrität eines Datensatzes oder der folgerichtige Gebrauch eines Mediums konnte nicht gewährleistet werden.

- *DEADLOCK*
Durch die Benützung eines Datensatzes oder Mediums blockierten sich die Prozesse gegenseitig.

- *RESTRICTION*
Durch die Benützung des Datensatzes oder des Mediums wurde den Prozessen eine unnatürliche Restriktion auferlegt, die die Systemleistung entscheidend beeinträchtigt.

Zusätzlich können wir die Forderung erheben, dass identische Funktionen durch gleichartige Prozesse, also Prozesse mit gleicher Prozessdefinition, ausgeführt werden sollen. Diese Forderung tangiert zwar nicht direkt das Systemverhalten, trägt aber entscheidend zu den Entwicklungskosten bei. Angesichts der heutigen Softwarekosten ist ihr deshalb durchaus die gleiche Beachtung wie Aspekten der Systemsicherheit beizumessen. Schauen wir uns noch einen letzten Ansatz an, der die Lösung der Andenbahnproblematik darstellt. Er stammt von E.W. Dijk-

stra aus dem Jahre 1968. Bei dieser Lösung wird noch eine weitere, beiden Zugführern gemeinsame Schale verwendet, die wir deshalb UNSERE nennen wollen. Präsentiert wird allerdings nicht die Originalform der Lösung, sondern eine strukturierte und durch Parameter auf identische Prozessdefinition hin verallgemeinerte Version davon:

```
TYPE inhalt = (besetzt,leer);

VAR  DEINE,MEINE,UNSERE : inhalt;

PROCESS Zugführerleben (StatusUNSERE : inhalt);

BEGIN
 LOOP
  WHILE DEINE = besetzt DO
   MEINE := besetzt;
   WHILE (DEINE  = besetzt) AND (UNSERE <> StatusUNSERE) DO
    Siesta
   END;
   IF DEINE = besetzt
    THEN
     MEINE := leer;
     WHILE UNSERE = StatusUNSERE DO Siesta END
   END
  END;
  FahreDurchKritischenAbschnitt;
  UNSERE := StatusUNSERE;
  MEINE  := leer;
  FahreRestlicheStrecke
 END
END Zugführerleben;


Bolivien := START_PROCESS Zugführerleben(leer);
Peru     := START_PROCESS Zugführerleben(besetzt);
```

Bemerkung:

Die in diesem Beispiel verwendete Syntax gehört keiner spezifischen Sprache an!

Der Algorithmus funktioniert wie ein Flip-Flop. Im Konfliktfall, d.h., wenn beide Züge gleichzeitig zum kritischen Abschnitt kommen, erhalten sie abwechslungsweise das Durchfahrtsrecht, ansonsten kann der Zug passieren, der eher den kritischen Abschnitt erreicht. Da weder eine Kollision noch ein Deadlock entstehen kann, erfüllt die Lösung sämtliche obig geforderten Kriterien und kann daher zurecht als Lösung des Problems angesehen werden. Dennoch vermag sie nicht recht zu befriedigen. Verglichen mit dem ersten Ansatz wirkt sie umständlich und aufgeblasen - zu aufwendig, wenn man bedenkt, dass damit lediglich der sehr seltene Fall der gleichzeitigen Geleisebenutzung verhindert werden soll. Betrachten wir deshalb nochmals den ersten Lösungsansatz. Er wies die einfachste und übersichtlichste Form auf, hatte aber den Nachteil, dass die beiden Züge miteinander kollidieren konnten. Frage: Welcher Umstand verursachte die Kollisionsgefahr?

Offensichtlich ergab sich das ganze Dilemma nur deshalb, weil der zweite Zugführer die Schale auch dann noch prüfen und für leer befinden konnte, während der erste schon auf der Suche nach einem Stein ist, den er in die Schale zu legen gedenkt. Könnten wir dafür sorgen, dass die Aktionen:

- Prüfe Schale, und wenn leer
- Suche Stein
- Lege Stein in Schale

in Nullzeit durchgeführt werden, wäre das Problem gelöst bzw. die Kollisionsgefahr gebannt. Nun, so schnell werden wir niemals sein! Wir können aber dafür sorgen, dass diese Aktionen unterbrechungsfrei durchgeführt werden. Wir führen dazu den Begriff der «privilegierten Aktionen» ein. Privilegierte Aktionen laufen in einem Abschnitt höchster Priorität (privilegierter Abschnitt) eines gegebenen Rechnersystemes ab und können deshalb nicht von anderen Prozessen unterbrochen werden. Schematisch lässt sich das etwa derart verdeutlichen:

```
LOOP (* Zugführerleben *)

 LOCK (* Anfang des privilegierten Abschnittes *)
   |
   |  WHILE "Schale nicht leer" DO
   |   "Siesta"
   |  END
   |  "Suche Stein"
   |  "Lege Stein in Schale"
   |
 UNLOCK (* Ende des privilegierten Abschnittes *)

 "Fahre durch kritischen Abschnitt (KA)"
 "Entferne Stein aus Schale"
 "Fahre restliche Strecke"
END (* Zugführerleben *)
```

Wie sich privilegierte Abschnitte hardwaremässig implementieren lassen, ist für die Betrachtungen von untergeordneter Bedeutung. Wichtig ist zu erkennen, dass alle Konzepte mit privilegierten Aktionen auf Mechanismen der unterlagerten Hardware zurückgreifen. Der interessierte Leser und Benutzer eines IBM PC findet Angaben über mögliche Implementierungsformen in [8,9] oder der entsprechenden INTEL-Literatur.

In Modula-2 kann jedem Programm-, Implementierungs- oder lokalem Modul eine feste Prioritätsebene zugeteilt werden [6]. Das Logitech-System unterstützt acht solcher Prioritätsebenen, wobei 0 die tiefste und 7 die höchste Priorität bezeichnet [7]:

```
MODULE HighestPriorityModule [7];

  (* Modul-Spezifikation *)

END HighestPriorityModule.
```

Durch diese Prioritätsangabe ist es keinem anderen Modul mehr möglich, HighestPriorityModule zu unterbrechen. Selbst gleichrangige Module müssen warten, bis der entsprechende Code aus diesem Modul abgearbeitet wurde.

In einer Implementierung mit Coroutinen ist es natürlich per Definition unmöglich, dass eine Coroutine der anderen zufällig dazwischen fährt - eine Prioritätsangabe wäre deshalb überflüssig. Allerdings erlaubt Modula-2 auch die Programmierung von sogenannten Interrupt-Handlern. Im Gegensatz zu Coroutinen erhält ein Interrupt-Handler die Kontrolle nicht explizit durch eine andere Coroutine, sondern aufgrund eines äusseren Ereignisses - beispielsweise eines Tastendrucks auf der Tastatur - zugewiesen. Eine Coroutine könnte deshalb durch einen Interrupt-Handler an einer beliebigen, unvoraussehbaren Stelle unterbrochen werden, wenn beide, Coroutine und Interrupt-Handler, die gleiche Priorität besässen. Es ist deshalb sinnvoll,

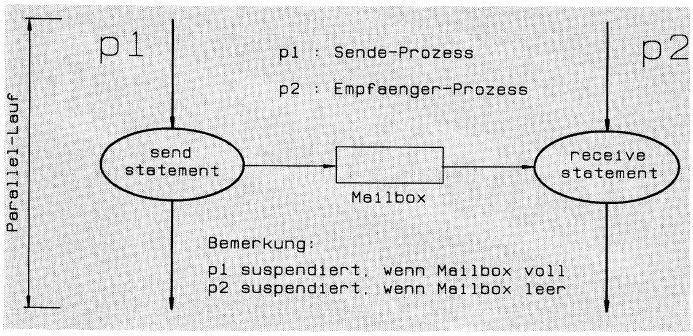


Abb. 8: Mailbox-Konzept

privilegierte Abschnitte immer in einen Modul mit Prioritätsangabe zu verkapseln - der Modul ist dann unabhängig von der späteren Umgebung.

Das folgende Beispiel soll Ihnen einen Eindruck vermitteln, wie solche Interrupt-Handler in Modula-2 implementiert werden. Der lokale Modul RTCHandler stellt einen Interrupt-Handler für die Systemuhr (Real Time Clock, RTC) des IBM PCs dar. Er könnte beispielsweise als Event-Generator in einem Time-Sharing-System eingesetzt werden.

```
MODULE RTCHandler [priority];

 IMPORT NEWPROCESS,PROCESS,TRANSFER,IOTRANSFER,
        SIZE,ADR,SetDeviceStatus,WorkSpace;
 EXPORT scheduler,count;

 CONST  RTCIntrVector   = 01C;
        RTCDeviceNumber = 0;

 VAR    scheduler,clock : PROCESS;
        count           : CARDINAL;

 PROCEDURE ClockISR;

  BEGIN
    LOOP
      IOTRANSFER(clock,scheduler,RTCIntrVector);
      count := (count + 1) MOD 19
    END
  END ClockISR;

 BEGIN
   count := 0;
   NEWPROCESS(ClockISR,ADR(WorkSpace),SIZE(WorkSpace),clock);
   SetDeviceStatus(RTCDeviceNumber,FALSE);
   TRANSFER(scheduler,clock)
 END RTCHandler;
```

Die Systemuhr generiert ca. alle 55 ms einen Interrupt, der eine Haltung zur Interrupt-Service-Routine clock (mit Definition ClockISR) zur Folge hat. Als einzige Aktivität führt clock ein Inkrement des Zählers count durch und beendet danach seine Ausführung mit dem Aufruf von IOTRANSFER(). Diese Prozedur erfüllt zwei Funktionen: zum einen meldet sie die Interrupt-Service-Routine für eine neue Interrupt-Behandlung an (Interrupt-Armierung) und zum anderen leitet sie die Kontrolle über den Prozessor an den Scheduler weiter. Dieser kann dann die eigentliche Umschaltung zwischen den Verarbeitungs-Coroutinen auslösen. Nützliche Dienste leistet dabei auch der vom Interrupt-Handler exportierte Zähler count. Dank seiner Hilfe ist es möglich, verschiedene Intervalle abzuleiten. Der Scheduler wird zwar alle 55 ms aufgerufen, könnte aber Umschaltungen beispielsweise nur dann vornehmen, wenn count den Wert 0 aufwiese (entspräche einem Intervall von etwa einer Sekunde). In allen anderen Fällen bliebe die Kontrolle bei der Coroutine, die den Prozessor belegte, als die Systemuhr die Verarbeitung unterbrach.

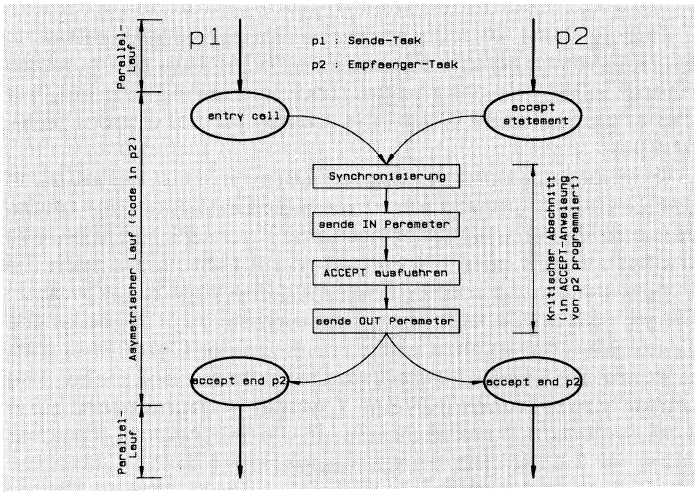


Abb. 9: Rendez-vous-Konzept

```
task body BoundedBuffer is

  N      : constant integer := AnyValue;
  Buffer : array (0..N-1) of Portion;
  In     : integer := 0;
  Out    : integer := 0;
  n      : integer := 0;

  begin
   loop
    select
     when n < N =>
      accept Append (x : in Portion) do
       Buffer(In) := x;
       n := n + 1;
       In := (In + 1) mod N;
      end;
    or
     when n > 0 =>
      accept Remove (x : out Portion) do
       x := Buffer(Out);
       n := n - 1;
       Out := (Out + 1) mod N;
      end;
    or
     terminate;
    end select;
   end loop;
  end BoundedBuffer;
```

Abb. 10: Task-Fragment BoundedBuffer

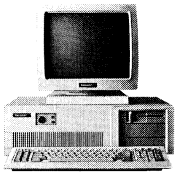
Bei der Modulinitialisierung wird davon ausgegangen, dass der Codeteil des Programm-Modules den Scheduler repräsentiert, der bereits vom RTS gestartet wurde und nun mit dem Aufruf von TRANSFER() in der Initialisierung des Interrupt-Handlers eine Prozessvariable zugewiesen bekommt. Mit SetDeviceStatus() wird die Erzeugung von Unterbrechungen der Systemuhr vorderhand unterbunden. Der Scheduler wird sie wieder freigeben, sobald die Initialisierung des gesamten Systemes beendet ist und mit der eigentlichen Verarbeitung begonnen werden kann.

Sollten Sie nun angeregt worden sein, auf der Basis des gezeigten Interrupt-Handlers einen eigenen Time-Sharing-Scheduler zu bauen, so gilt es folgendes zu beachten: MS/PC-DOS ist nicht re-entrant. Befindet sich DOS in einem kritischen Abschnitt, so darf nicht auf einen anderen Prozess umgeschaltet werden, der wiederum eine kritische Aktion im Sinne von DOS starten könnte. In [7] findet der interessierte Leser nähere Hinweise dazu.

Damit endet der erste Teil dieser Artikelserie, der als Begriffsbestimmung und Einführung in die Welt der Parallelverarbeitung gedacht ist. Im zweiten Teil befassen wir uns mit den Koordinierungskonzepten und Sie werden erfahren, was es mit Semaphoren, Monitoren, Message Passing, Mailboxes und Rendez-Vous auf sich hat. □

TANDON «SELECTION MENU»:

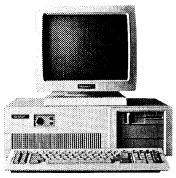
Aus dem breiten Angebot an Tandon AT's kann sich jeder seinen Favoriten



PCA: 80286-Prozessor, 1,2-MB-Floppylaufwerk, 1-MB-Hauptspeicher, serielle und parallele Schnittstelle, Herkules-kompatible Monochrom-Grafikkarte, 14"-Monochrom-Monitor, MS-Windows, MS-DOS 3.2, GW-Basic, VSM-Tastatur,
Fr. 5290.–.

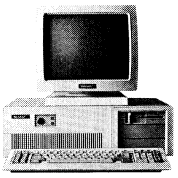
PCA 20 MIT 20-MB-FESTPLATTE.

aussuchen. Allen gemein ist das hohe Qualitäts- und das tiefe Preisniveau.



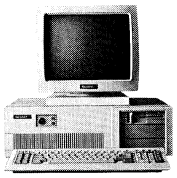
PCA 20: 20-MB-Festplatte, 80286-Prozessor, 1,2-MB-Floppylaufwerk, 1-MB-Hauptspeicher, serielle und parallele Schnittstelle, Herkules-kompatible Monochrom-Grafikkarte, 14"-Monochrom-Monitor, MS-Windows, MS-DOS 3.2, GW-Basic, VSM-Tastatur,
Fr. 5690.–.

PCA 40 MIT 40-MB-FESTPLATTE.



PCA 40: 40-MB-Festplatte, 80286-Prozessor, 1,2-MB-Floppylaufwerk, 1-MB-Hauptspeicher, serielle und parallele Schnittstelle, Herkules-kompatible Monochrom-Grafikkarte, 14"-Monochrom-Monitor, MS-Windows, MS-DOS 3.2, GW-Basic, VSM-Tastatur,
Fr. 6490.–.

PCA 70 MIT 70-MB-FESTPLATTE.



PCA 70: 70-MB-Festplatte, 80286-Prozessor, 1,2-MB-Floppylaufwerk, 1-MB-Hauptspeicher, serielle und parallele Schnittstelle, Herkules-kompatible Monochrom-Grafikkarte, 14"-Monochrom-Monitor, MS-Windows, MS-DOS 3.2, GW-Basic, VSM-Tastatur,
Fr. 8490.–.

Die AT's, die Tandon Ihnen auftischen kann, verfügen über bis zu 70 MB Speicherkapazität, bei Preisen bis max. 8490 Franken. Damit befindet sich das kompatible Tandon Programm eindeutig auf Erfolgskurs.
Tandon kommt aus Amerika und ist als weltgrösster Laufwerkhersteller und Entwickler von Massenspeichern bereits erfolgsgewohnt. Die Tandon Computer brachten es in Deutschland innert 2 Jahren zur Nr. 3 auf dem Markt.
In der Schweiz werden sie ein ähnliches Tempo anschlagen.

Tandon
Computer AG

MUK/8

☐ Bitte rufen Sie mich an.
☐ Bitte senden Sie mir Ihre Dokumentation über Tandon Computer.

Name:

Firma:

Strasse:

PLZ/Ort: Telefon:

Einsenden an: Tandon Computer AG, Industrie Werrikon 4, 8606 Werrikon/Uster, Telefon 01/941 41 30

M, H & LB

Texterfassung auf Ihrem PC – Satzproduktion auf der UD-Lichtsatzanlage

Einmalige Texterfassung spart Satzkosten, verhindert Übertragungsfehler. Alle Modifikationen wie Preis-, Text- oder Aufbau-Änderungen, zum Beispiel in Periodika, können problemlos wieder auf Ihrem PC vorgenommen werden. Sie erhalten die Texte wahlweise als Papierspalten oder als Film. Beides können Sie auf Wunsch selber umbrechen, montieren und maquettieren. Auch für diese Zeitschrift wird der Satz und nachher der Druck in dieser kostengünstigen Art hergestellt.

Setzen Sie jetzt Ihre Texte selber

Damit können Sie die Leistungspalette Ihres Computers um eine kostensenkende Möglichkeit erweitern.

Verlangen Sie unseren ausführlichen Prospekt oder lassen Sie sich von unseren Spezialisten in die kostensenkende Satzproduktion einführen.

Tel. 041 / 44 24 44

Unionsdruckerei Luzern
6005 Luzern, Kellerstrasse 6

UD

Zur starken Sprache

MODULA-2

starke Tools!

Passend zur Logitech-Entwicklungsumgebung liefern wir zwei Pakete, je mit vielen praxisnahen Basis-Moduln

EDITOREN zum simplen Editieren (nicht Programmieren) der Bedieneroberfläche (color und monochrome)

- **FORM** für feste Bildteile
- **MENU** für Auswahlfelder
- **MASK** für Eingabefelder

Fr. 480.–

FILES zum effizienten Umgang mit Dateien

- **ISAM** Basis-Moduln (multikey, B-tree mit cache)
- **QUEUE** allgemeiner Warteschlangen-Modul
- **PRINT** drucker-unabhängige LIST- und PRINT-Moduln

Fr. 470.–

Bühler Systemtechnik AG, Postfach 836
9001 St.Gallen, Telefon 071 23 63 73

Verlangen Sie Unterlagen!

UNIX.V2 auf PC/AT
Vollständige Implementation von Unix auf PC/AT, Mehrplatz-System von 2-17 Benutzern, je nach Applikation

Merge 286 - MS-DOS unter unix V.2
MS-Programme laufen unter UNIX im gleichen Filesystem, Datenaustausch und MS-DOS Kommandos in der Unix-Shell möglich

UNIX System V.2 Runtime System für 2 User, enthält über 180 Utilities, vi, UUCP usw.	**Fr. 850.–**
UNIX System V.2 Development System enthält C-Compiler, Fortran-Compiler, Entwicklungs-Tools wie SCCS MAKE SDB und Textverarbeitungs-Paket nroff/troff	**Fr. 1250.–**
UNIX System V.2 Upgrade 3–16 User	**Fr. 750.–**
UNIX System V.2 System komplett enthält Runtime-System für 2 User und Development System	**Fr. 1950.–**
DOSMERGE-Utility MS-DOS-Programme laufen unter UNIX	**Fr. 720.–**

Weitere System-Software für UNIX-Systeme wie Datenbank-SW, Compiler, Communication, Statistik- und Graphik ist auf Anfrage erhältlich.

Anfragen von Wiederverkäufern erwünscht

uniperform, Seebahnstrasse 145, 8036 Zürich
Tel. 01 / 463 05 00, Fax 01 / 462 76 48, Telex 814 003 ch

Sensationell! 12 MHz (Turbo) AT-kompatibler Computer

Standardversion
640 KB RAM, Herkules-Karte, 20 MB HD (Seagate), 1,2 MB/360 KB FD, CH- AT-Tastatur, 14"-Bernstein-Monitor, DOS und Dokumentation für **SFr. 3550.–**

Topversion
Gleich wie Standard, jedoch mit 1 MB RAM, Par.-Ser.- Adapter, Maus, Druckerkabel, Color (EGA, Herkules) und Farbmonitor (hochauflösend) für **SFr. 4850.–**

LOW-Soft AG

Postfach, 8185 Winkel, Telefon **01 / 820.00.66**

Besuchen Sie unseren Showroom: Di–Fr, 14–18.30 Uhr, Sa, 10–16 Uhr, Neugutstrasse 88, 8600 Dübendorf (beim P1 Dancing/Coop-BauCenter).

Fragen Sie nach dem interessanten Kurs-Konzept!

Für Gruppen oder Firmen wird die Schulung individuell den Bedürfnissen angepasst.

1227 **Carouge-Genève**
Tél. 022-43 13 60

ADCOMP AG
8953 **Dietikon**, Lerzenstr. 27
Tel. 01-741 41 11, Telex 58657

Einführung in MULTIPLAN

Multiplan ist eines der ältesten Tabellenkalkulationsprogramme. Es wurde von der amerikanischen Software-Firma Microsoft nach den Ideen eines kanadischen Studenten entwickelt und erfreut sich nach wie vor immer noch grosser Beliebtheit.

Inzwischen sind Weiterentwicklungen von Tabellenkalkulationsprogrammen auf den Markt gekommen. Bekannt sind vor allem Lotus 1-2-3, Symphony, Framework II, Open Access und Enable. Es handelt sich dabei um sogenannte

Marcel Sutter

integrierte Programmprodukte, da sie mindestens folgende Teilprogramme enthalten:

1. Ein Tabellenkalkulationsprogramm
2. Eine zugehörige Businessgrafik (Balken- und Kreisdiagramm)
3. Ein Textverarbeitungssystem
4. Ein Programm für den Aufbau und die Verwaltung von Dateien

Da Multiplan viel leichter zu handhaben ist als Lotus, Symphony, Framework, Open Access und Enable, im Preis weit unter diesen integrierten Programmpaketen liegt und überhaupt keine Hardware- und Programmierkenntnisse voraussetzt, wird es gern als Einstiegsprogramm für professionelle Software benützt.

1. Starten von Multiplan

Multiplan wird üblicherweise auf einer Diskette geliefert. Von dieser sollte unbedingt zuerst eine Sicherungskopie (ein Backup) hergestellt werden, damit man mit der Kopie und nicht mit dem Original arbeiten kann.

Um Multiplan zu starten, müssen Sie bei Ihrem PC (IBM oder kompatibel) in der DOS-Ebene sein. Auf Ihrem Bildschirm muss die Anzeige A> sichtbar sein. Wie man Multiplan von der Harddisk startet, wird hier nicht behandelt.

Legen Sie die Multiplan-Diskette ins linke Laufwerk (Laufwerk A), die Diskette mit den gespeicherten Multiplan-Dateien ins rechte Laufwerk (Laufwerk B), tippen Sie MP80 ein und drücken Sie die Returntaste. Nach kurzer Zeit erscheint auf dem Bildschirm ein Ausschnitt aus der noch leeren Kalkulationstabelle (Abbildung 1).

Falls Sie vorgängig in BASIC gearbeitet haben, müssen Sie SYSTEM eintippen, um ins DOS abzusteigen. Haben Sie in LOGO programmiert, so tippen Sie .DOS ein, um ins Betriebssystem zu gelangen.

Achtung: Die Multiplan-Systemdiskette muss ständig im Laufwerk A bleiben. Die Diskette, auf der Sie Ihre Tabellen speichern wollen, muss immer im Laufwerk B sein.

Es ist möglich, die Multiplan-Diskette selbststartend zu machen. Wie Sie das vorzunehmen haben, ist im Handbuch ausführlich erklärt.

2. Aufbau der Tabelle

Betrachten Sie die Abbildung 1. Die Tabelle besteht aus 255 Zeilen und 63 Spalten, umfasst also 255 x 63 = 16'065 Felder. Im Moment sehen Sie nur den linken oberen Ausschnitt mit 20 Zeilen und 7 Spalten.

Im Normalfall ist jedes Feld 10 Spalten breit. In ein Feld kann entweder nur ein Text (ohne Anführungszeichen), oder nur eine Zahl (Dezimalbrüche mit Komma schreiben), oder nur eine mathematische Formel geschrieben werden.

In der abgebildeten Tabelle sehen Sie im Feld Z1S1 (Feld in Zeile 1 und Spalte 1) ein helles waagrechtes Rechteck. Das ist der Tabellen-Cursor.

Unterhalb der Tabelle sehen Sie eine Textleiste zu vier Zeilen. Die ersten zwei Zeilen sind das sogenannte Befehlsmenü. Ein zweiter kleinerer Cursor zeigt im Moment auf den Befehl Text. Sie sehen das daran, dass der Befehl Text in inverser Farbe hervorgehoben ist. In der dritten Zeile steht, was Sie im Moment tun können. Tippen Sie den

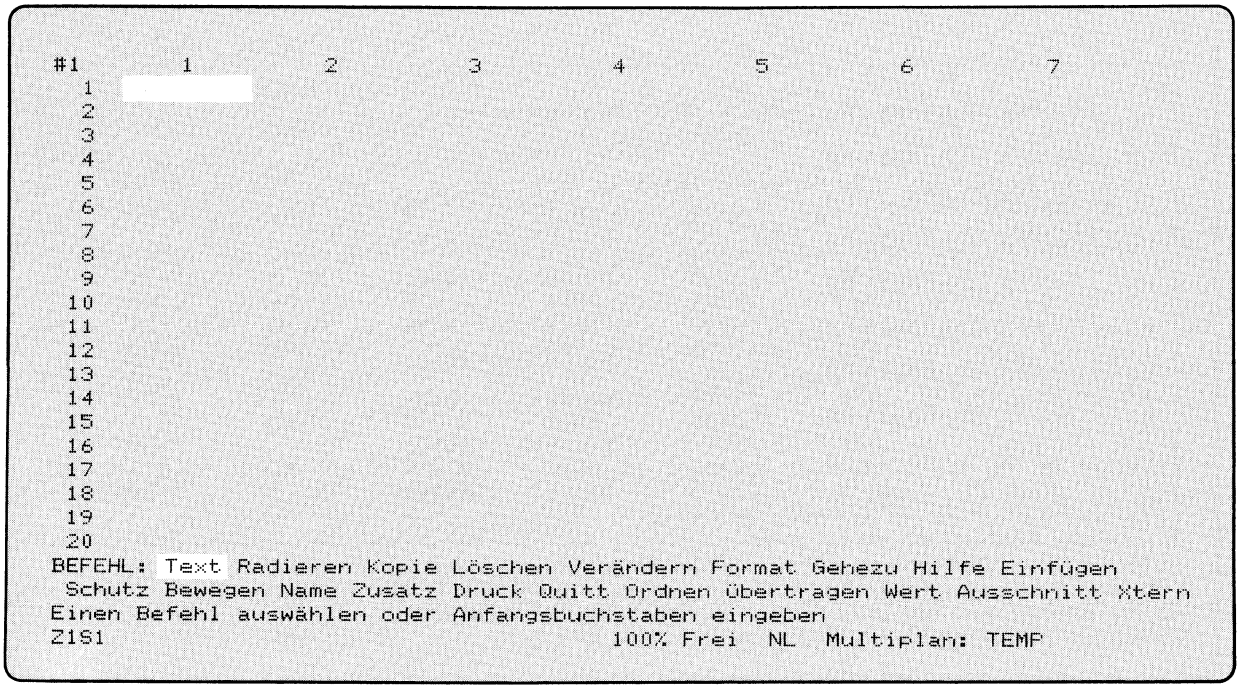


Abb. 1

Anfangsbuchstaben eines Befehls ein, so wird dieser sofort ausgeführt. Sie können aber auch mit der Leertaste (Space-Taste) nach rechts von Befehlswort zu Befehlswort springen und dann durch Drücken der Returntaste den ausgewählten Befehl in Aktion treten lassen. In der untersten Zeile steht links aussen die Adresse, wo der Tabellen-Cursor im Moment gerade steht und daneben, was der Inhalt dieses Feldes ist. In der Abbildung ist das Z1S1 und daneben steht nichts, da wir das Feld noch nicht beschrieben haben.

In der Mitte bedeutet 100%, dass noch 100% der Tabelle zur Verfügung stehen. Rechts unten steht Multiplan: TEMP. Temp (Temporary) ist der vorläufige Name der Tabelle. Wenn Sie die erstellte Tabelle später auf Diskette abspeichern wollen, dann müssen Sie ihr einen Namen geben. Multiplan hat schon einen ausgesucht, eben TEMP. Denken Sie daran, dass der Name der DOS-Konvention genügen muss, d.h. er darf aus höchstens acht Zeichen bestehen und keinen Zusatz haben.

Das ist wichtig:

1. Um den Cursor in der Tabelle zu bewegen, müssen Sie die Cursor- Kontrolltasten → ← ↑ ↓ benützen.

2. Um den Cursor im Befehlsmenü zu bewegen, müssen Sie die Leertaste benützen.

3. Sobald Sie die Taste Esc (Escape) drücken, kehren Sie sofort wieder ins Befehlsmenü zurück.

4. Multiplan ist nomalerweise immer in der Befehlsebene.

Die Abbildung 2 ist eine typische Multiplan-Tabelle. Sie zeigt die Vermehrung eines Kapitals von Fr. 10'000.-- innert 30 Jahren bei verschiedenen Zinssätzen.

Aendern Sie das Anfangskapital, so rechnet Multiplan augenblicklich die Tabelle erneut durch. Das ist typisch für Tabellenkalkulations-Programme, nämlich die Möglichkeit des Durchspielens von ...was wäre, wenn...?

Wie man eine solche Tabelle generiert, ausdruckt, speichert und wieder von der Diskette holt, wird anschliessend genau erklärt.

3. Uebersicht über die Befehle in Multiplan

Text Sie können einen Text in ein Feld schreiben.

Radieren Sie können den Inhalt eines oder mehrerer Felder löschen.

Kopie Sie können den Inhalt eines Feldes nach rechts oder nach unten oder in andere Felder kopieren.

Löschen Sie können eine ganze Zeile oder eine ganze Spalte in der Tabelle löschen. Es erfolgt eine Neunumerierung.

Verändern Sie können den Inhalt eines Feldes verändern, ohne ihn löschen und neu schreiben zu müssen.

Format Sie können die Breite der Felder einstellen, festlegen auf wieviele Dezimalen gerundet werden soll u.a.m.

Gehe zu Sie können direkt zu einem weit entlegenen Feld in der Tabelle gehen, ohne mit den Cursor-Kontrolltasten dieses mühsam anzusteuern.

Hilfe Wenn Sie den Cursor im Befehlsmenü auf einen Befehl lenken und ? eintippen, dann wird der Befehl auf dem Bildschirm genau erklärt.

Einfügen Sie können neue Zeilen oder neue Spalten irgendwo einfügen. Es wird automatisch umnumeriert.

Schutz Sie können bestimmte Felder gegen Ueberschreiben schützen.

Bewegen Sie können Zeilen oder Spalten an eine andere Stelle der Tabelle verschieben.

```
#1        1         2         3         4         5         6         7
  1 Vermehrung eines Kapitals k in n Jahren bei p%
  2 ------------------------------------------------------------------------
  3 Kapital:       10000
  4 ------------------------------------------------------------------------
  5 Prozente:          2         3         4         5         6         7
  6 Jahre:
  7 ------------------------------------------------------------------------
  8       1        10200     10300     10400     10500     10600     10700
  9       2        10404     10609     10816     11025     11236     11449
 10       3        10612     10927     11249     11576     11910     12250
 11       4        10824     11255     11699     12155     12625     13108
 12       5        11041     11593     12167     12763     13382     14026
 13      10        12190     13439     14802     16289     17908     19672
 14      15        13459     15580     18009     20789     23966     27590
 15      20        14859     18061     21911     26533     32071     38697
 16      25        16406     20938     26658     33864     42919     54274
 17      30        18114     24273     32434     43219     57435     76123
 18 ------------------------------------------------------------------------
BEFEHL: Text Radieren Kopie Löschen Verändern Format Gehezu Hilfe Einfügen
 Schutz Bewegen Name Zusatz Druck Quitt Ordnen Übertragen Wert Ausschnitt Xtern
Einen Befehl auswählen oder Anfangsbuchstaben eingeben
Z8S2      Z3S2*(1+Z5 S/100)^Z S1          99% Frei  NL  Multiplan: A:KAPITAL.MP
```

Abb. 2

Name Wenn Sie einer bestimmten Gruppe von Feldern einen Namen geben, dann können Sie diese Felder in Formeln über den Namen und nicht über die Feldadressen ansteuern. Das ist viel bequemer.

Zusatz Hier können Sie z.B. entscheiden, ob bei einer Aenderung eines numerischen Inhalts eines Feldes die ganze Tabelle neu durchzurechnen ist oder ob Multiplan noch warten soll u.a.m.

Druck Sie können die Tabelle auf dem Drucker herauslisten lassen. Verschiedene Darstellungsarten sind möglich.

Quitt Sie wollen Ihre Arbeit mit Multiplan beenden und ins DOS zurückkehren.

Ordnen Sie können die Zeilen längs einer bestimmten Spalte nach einem gemeinsamen Merkmal, z.B. alphabetisch ordnen.

Uebertragen Sie können eine Tabelle laden oder speichern.

Wert Sie können Zahlen oder Formeln eingeben.

Ausschnitt Wahl der Bildschirm- und Schreibfarbe.

Xtern Sie können zwei verschieden Tabellen verknüpfen.

Wir müssen noch auf ein wichtiges Detail hinweisen:

Wie Sie bereits wissen, können Sie in der Tabelle den Cursor nur mit den Cursor-Kontrolltasten bewegen, während Sie im Befehlsmenü immer mit der Leertaste den Cursor zum nächsten Befehl verschieben müssen. Haben Sie einen Befehl ausgewählt, z.B. den Befehl Format, dann erscheint im Befehlsfenster ein neues Befehlsmenü, ein sogenanntes Untermenü.

Sind in diesem Untermenü wieder Befehle aufgezeigt, müssen Sie erneut entweder den Anfangsbuchstaben eintippen oder mit der Leertaste auf den betreffenden Befehl vorrücken und die Returntaste drücken. Es ist aber oft im Untermenü eine Zahleneingabe vorzunehmen. Sie erkennen das daran, dass nach dem Eingabebefehl ein Doppelpunkt steht. Von Doppelpunkt zum nächsten Doppelpunkt können Sie nur mit der Tabulatortaste ⇤| vorwärtsrücken.

Beispiel:

Wählen Sie im Befehlsmenü den Befehl Format, indem Sie f eintippen. Es erscheint ein Untermenü (Abbildung 3). Wählen Sie im Untermenü den Befehl Felder, indem Sie erneut f drücken. Es erscheint ein neues Untermenü (Abbildung 4).

Angenommen Sie wollen, dass alle Zahlen innerhalb der Felder auf zwei Kommastellen gerundet erscheinen.

Bei Felder müssen Sie den Bereich eintippen, z.B. Z1:20S1:7. Darauf drücken Sie die Tabulatortaste. Bei Ausrichtung wollen Sie keine Eingabe machen, deshalb drükken Sie erneut die Tabulatortaste. Bei Formatcode müssen Sie den Befehl Fest (Festkomma) wählen. Rücken Sie daher mit der Leertaste bis zu Fest vor. Drücken Sie erneut die Tabulatortaste. Beim Befehl Dez.Stellen geben Sie die Zahl 2 ein, darauf drücken Sie die Returntaste.

Multiplan weiss jetzt, dass alle Zahlen in den Feldern von Zeile 1 bis 20 und Spalte 1 bis 7 sogenannte Festkommazahlen sind, wobei auf die zweite Stelle nach dem Komma gerundet werden muss. Die Ausrichtung der Zahlen erfolgt sofort nach dem Drücken der Returntaste.

Merke: Im Befehlsmenü bewegt man den Cursor
- mit der Leertaste von Befehl zu Befehl
- mit der Tabulatortaste von Doppelpunkt zu Doppelpunkt (Eingaben sind vorzunehmen).

Multiplan schreibt in weisser Farbe auf schwarzem Hintergrund. Das ist die Standardeinstellung. Wenn Ihnen diese Farben nicht gefallen, dann wählen Sie im Befehlsmenü den Befehl Ausschnitt, indem Sie a eintippen. Im Untermenü wählen Sie den Befehl Färben aus, indem Sie f eintippen. Sie können jetzt mittels Zahlen die Bildschirm-

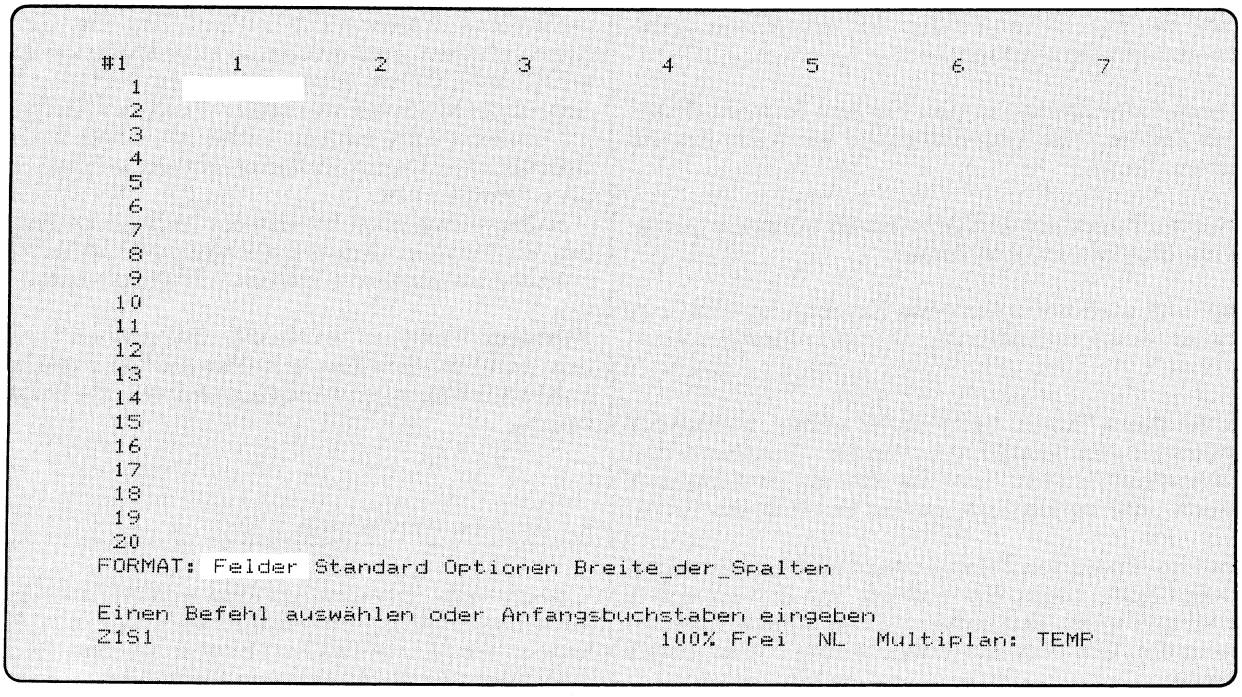


Abb. 3

farbe (sogenannte Hintergrundfarbe), die Zeichenfarbe (sogenannte Vordergrundfarbe) und die Farbe für den linierten Rahmen und die Tabellen einstellen. Mit der Tabulatortaste springen Sie von Eingabe zu Eingabe und geben die gewünschte Zahl ein.

Nummer der Farben beim IBM PC:

0	schwarz	4	rot
1	blau	5	violett
2	grün	6	braun
3	kobaltblau	7	weiss

4. Erstellen einer neuen Tabelle

Typische Multiplan-Tabellen sind:

- Kalkulationstabellen (vom Ankaufs- zum Verkaufspreis)
- Umsatzanalyse von verschiedenen Produkten
- Investitionsanalyse
- Zinseszinstabellen (bei Banken)
- Monatsbudget und Ausgabenkontrolle im Haushalt
- Jahresbudget
- Adresskartei
- Steuererklärungen
- Notentabellen für Schüler und Klassen
- Ranglisten bei sportlichen Anlässen
  usw.

Bevor Sie sich an den PC setzen und eine solche Tabelle aufbauen wollen, sollten Sie auf einem Blatt Papier einen groben Plan der Tabelle entwerfen. Folgende Fragen sind vorerst zu klären:

1. Wie sollen die Zeilen und die Spalten beschriftet werden?

2. Wie breit müssen daher die Felder der einzelnen Spalten sein?

3. Wie soll der Text in den Feldern formatiert sein (links- oder rechtsbündig oder zentriert)?

4. Wie sollen die Zahlen in den Feldern formatiert sein (ganze Zahlen oder Dezimalbrüche mit fester Anzahl Kommastellen)?

5. Welche mathematischen Formeln sind nötig, um gewisse Feldergruppen zu berechnen?

Achtung: Text wird normalerweise linksbündig und Zahlen normalerweise rechtsbündig in die Felder geschrieben.

Wenn Sie diese Fragen beantwortet haben, wählen Sie im Befehlsmenü den Befehl Format und im Untermenü den Befehl Breite_der_Spalten aus. Sie können jetzt Spalten vergrössern oder verkleinern, genau nach Ihrem Tabellenplan.

Wählen sie erneut den Befehl Format und im Untermenü den Befehl Felder. Legen Sie jetzt fest, wie Zahlen in den Feldern formatiert werden müssen, z.B. auf die 2. Kommastelle gerundet. Wie man das macht, wurde vorher genau beschrieben.

Legen Sie auch fest, wie der Text in den Feldern dargestellt werden soll. Insbesondere klären Sie, wann Sie für Ueberschriften die Feldergrenzen aufheben wollen.

Wenn Sie dies alles getan haben, liegt das Raster der Tabelle fest. Sie können somit mit der Eingabe von Text, Zahlen und Formeln beginnen.

4.1. Eingabe von Text

Sie wählen im Befehlsmenü den Befehl Text, indem Sie t eintippen. Das Befehlsfenster verschwindet und es erscheint das Wort TEXT. Schreiben Sie jetzt Ihren Text ein und verwenden Sie bitte keine Anführungszeichen. Der eingetippte Text steht zunächst unten und kann eventuell mit der Backspace-Taste noch korrigiert werden. Sobald Sie aber die Returntaste drücken, wird der Text in das Feld, wo der Tabellen-Cursor gerade steht, übertragen und es erscheint erneut das Befehlsmenü. Falls Ihr Text nicht ganz sichtbar ist, haben Sie die Felderbreite in jener Spalte falsch gewählt. Sie können das aber jederzeit korrigieren.

Sie können viel schneller Text oder Zahlen eingeben, wenn Sie nach einer Eingabe nicht die Returntaste drükken, sondern mit der Cursor-Kontrolltaste auf jenes Feld fahren, wo die nächste Eingabe erfolgen soll. Multiplan schreibt augenblicklich die Eingabe ins Tabellenfeld und bewegt danach den Tabellen-Cursor weiter.

Wenn Sie einen längeren Text wie z.B. eine Tabellenüberschrift schreiben wollen, dann verfahren Sie folgendermassen: Tippen Sie f für Format, dann f für Felder. Geben Sie jetzt den Bereich an, z.B. Z1S1:20, rücken Sie mit der Tabulatortaste bis zu Formatcode und mit der Leertaste bis unbeg (unbegrenzt) vor. Drücken Sie darauf die Returntaste. Jetzt sind in der Zeile 1 die Grenzen der Felder von Spalte 1 bis 20 aufgehoben.

4.2. Eingabe von Zahlen

Sobald Sie Ihre Eingabe mit einer Ziffer beginnen, kippt Multiplan automatisch vom Modus TEXT in den Modus WERT um und betrachtet Ihre Eingabe als Zahl. Danach verbleibt es im Modus TEXT/WERT. Beginnen Sie Ihre Eingabe mit einem Buchstaben, so kippt Multiplan von TEXT/WERT in TEXT um, tippen Sie aber zuerst eine Ziffer ein, dann kippt Multiplan von TEXT/WERT in WERT um.

Dezimalbrüche müssen leider mit einem Komma statt dem üblichen Dezimalpunkt eingetippt werden. Schreiben Sie 4,5 statt 4.5 usw.

Selbstverständlich können Sie im Befehlsmenü den Befehl Wert wählen, indem Sie w eintippen. Multiplan ist jetzt nur zur Aufnahme von Zahlen und Formeln bereit.

Achtung: Wenn Sie bei einer Eingabe einen Fehler begehen, dann drücken Sie am besten die Esc-Taste. Multiplan löscht das Eingetippte und kehrt sofort in die Befehlsebene zurück.

4.3. Eingabe von Formeln

Sie müssen unbedingt im Modus WERT sein. Viele Formeln beginnen mit einem Buchstaben. Damit Multiplan nicht in den Modus TEXT umkippt, schreiben Sie als erstes Zeichen ein + und dahinter die Formel.

Formeln mit direkter Adressierung:

Im Feld Z10S3 steht die Formel +Z8S3 - Z983. Was bedeutet das? Multiplan subtrahiert von der Zahl im Feld Z8S3 die Zahl im Feld Z9S3 und schreibt das Ergebnis formatiert in das Feld Z10S3.

Die direkte Adressierung hat verschiedene Nachteile:

1. Die Formel ist nicht in andere Felder kopierbar, da sie sonst überall das gleiche Ergebnis hinschreibt.
2. Die Formel wird sehr umständlich, wenn Sie viele Zahlen längs einer Zeile oder einer Spalte zusammenfassen müssen.

Wir zeigen daher einen eleganteren Weg, allgemeine gültige Formeln aufzustellen. Dazu benützen wir die sogenannte indirekte Adressierung bei mathematischen Formeln.

Formeln mit indirekter Adressierung:

Im Feld Z5S9 steht die Formel +Z S8*1,5. Beachten Sie bitte die Leerstelle zwischen Z und S, sie ist unbedingt nötig. Multiplan multipliziert die Zahl im Feld Z5S8 mit 1,5 und schreibt das Ergebnis formatiert in das Feld Z5S9, weil die Formel in diesem Feld steht. Da nach dem Z keine Zahl steht, wird das Feld in der 8. Spalte und der Zeile, in der die Formel steht, für die Rechnung benützt. Wenn Sie jetzt die *gleiche Formel* mit dem Kopie-Befehl in Feld Z6S9 kopieren, dann multipliziert Multiplan automatisch die Zahl im Feld Z6S8 mit 1,5 und schreibt das Ergebnis in das Feld Z6S9.

> **Merke:** Die Formel +Z S8*1,5 wirkt auf alle Felder der 8. Spalte, unabhängig von der Zeilennummer.

Hierin liegt das Geheimnis aller Tabellenkalkulationsprogramme. Fast immer müssen auf alle Felder einer Zeile (oder einer Spalte) die gleichen Rechenoperationen ausgeübt werden, z.B. ein gleicher prozentualer Zuschlag. Man programmiert dann eine Formel mit relativer Adressierung einer Zeile (oder Spalte) und kopiert sie mit dem Kopie-Befehl in jene Felder, wo man Resultate haben möchte.

Beispiel:

Um die Zinseszinstabelle in Abbildung 2 zu rechnen, ist eine einzige Formel nötig. Sie lautet mathematisch

$K(n) = K^*(1+p/100)^n$

Wenn Sie die Tabelle genau betrachten, werden Sie sehen, dass im Feld Z8S2 die Formel

+Z3S2*(1+Z5 S/100)^Z S1

einzugeben ist. Diese Formel kopieren Sie dann in alle Felder der Zeilen 8 bis 17 und der Spalten 2 bis 7.

Multiplan holt die Zahl aus dem Feld Z3S2 (K), holt in der 5. Zeile und der betreffenden Spalte die Zahl (p), berechnet die Klammer und potenziert sie mit der Zahl aus der betreffenden Zeile und der 1. Spalte (n). Das Ergebnis wird formatiert in das betreffende Feld, wo die Formel steht, eingeschrieben.

Wenn Sie das Anfangskapital im Feld Z3S2 abändern und die Returntaste drücken, dann rechnet Multiplan sofort die Tabelle neu durch. Aendern Sie in der 5. Zeile gewisse Zinssätze ab, so wird wiederum die Tabelle neu erstellt. Das ist typisch für Tabellenkalkulationsprogramme. Sie können beliebig oft Fragen stellen,

..... Was wäre, wenn ich K ändere? .....
..... Was wäre, wenn ich p ändere? .....
..... Was wäre, wenn ich n ändere? .....

Beachten Sie bitte die folgenden Schreibweisen:

- absolute Adressen: Z4S27, Z217S55, Z1S12
- relative Adressen: Z S27, Z217 S
- Bereich von Feldern: Z1:20S5, Z10S4:12, Z2:12S5:20
  Z S1:7, Z3:9 S
  usw.

Es gibt noch eine dritte Art, Formeln einzugeben. Im Feld Z10S4 soll die Formel stehen, die alle Zahlen in der Spalte 4 von Zeile 3 bis 7 addieren soll.

Führen Sie jetzt folgende Schritte aus:

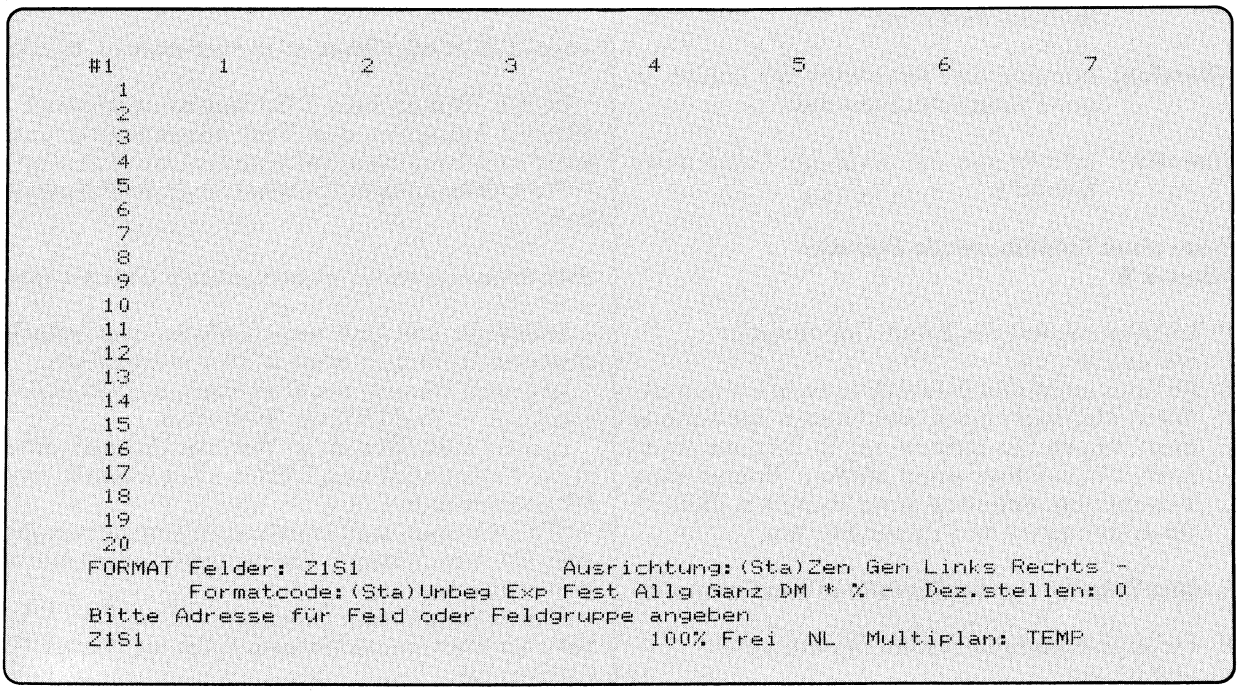


Abb. 4

1. Tabellen-Cursor in Feld Z10S4
2. Den Befehl Wert durch Drücken von w wählen
3. Tippen Sie ein summe(
4. Fahren Sie mit dem Tabellen-Cursor auf Feld Z3S4
5. Tippen Sie einen Doppelpunkt : ein
6. Fahren Sie mit dem Tabellen-Cursor auf Feld Z7S4
7. Tippen Sie die Klammer) ein
8. Drücken Sie die Returntaste

Die Formel ist jetzt mit relativer Adressierung eingeschrieben, wobei als Bezugspunkt das Feld Z10S4 dient. Da aber just diese absolute Adresse nirgends steht, können Sie die Formel in weitere Felder kopieren.

5. Operationszeichen

+	Addition
-	Subtraktion
*	Multiplikation
/	Division
^	Potenzieren
%	Prozentwert (= /100)
&	Aneinanderreihung von Textteilen

6. Einige wichtige Funktionen

abs(zahl), arctan(zahl), cos(zahl), exp(zahl), ln(zahl), sin(zahl), tan(zahl), wurzel(zahl)

Das sind die aus der Mathematik bekannten Funktionen. Daneben gibt es noch folgende bequeme Funktionen:

anzahl(Bereich) zählt alle Felder eines Bereiches, deren Inhalt ein Zahlenwert ist. Felder mit Text und Leerfelder werden nicht gezählt.

länge(Feld) zählt die Anzahl Zeichen des Textes im ausgewählten Feld.

max(Bereich) ermittelt die grösste Zahl in den Feldern des Bereiches.

min(Bereich) ermittelt die kleinste Zahl in den Feldern des Bereiches.

mittelw(Bereich) ermittelt den Mittelwert der Zahlen in den Feldern des Bereiches.

summe(Bereich) addiert alle Zahlen in den Feldern des Bereichs.

7. Speichern einer Tabelle auf die Diskette im Laufwerk B

1. Schritt: Im Befehlsmenü den Befehl Uebertragen wählen (ü).
2. Schritt: Im Untermenü den Befehl Speichern wählen (s).
3. Schritt: Wenn Multiplan nach dem Namen der Tabelle fragt, tippen Sie b:name ein (8 Zeichen maximal). Falls schon unter diesem Namen eine Tabelle gespeichert ist, fragt Multiplan «Soll ich überschreiben (j/n)? Tippen Sie j ein.

8. Laden einer Tabelle von der Diskette in den Computer

1. Schritt: Im Befehlsmenü den Befehl Uebertragen wählen (ü).
2. Schritt: Im Untermenü den Befehl Laden wählen (l).
3. Schritt: Geben Sie den Namen der Tabelle ein, also b:name. Sofort erscheint auf dem Bildschirm eine Tabelle.

9. Abruf des Inhaltsverzeichnisses der Diskette im Laufwerk B

1. Schritt: Im Befehlsmenü den Befehl Uebertragen wählen (ü).
2. Schritt: Im Untermenü den Befehl Laden wählen (l).
3. Schritt: Tippen Sie b: ein und drücken Sie danach die Cursor-Kontrolltaste →. Multiplan listet die Namen der gespeicherten Dateien auf den Bildschirm.

10. Ausdrucken einer Tabelle auf dem Printer

1. Schritt: Drucker einschalten und Papier richten.
2. Schritt: Im Befehlsmenü den Befehl Druck wählen (d).
3. Schritt: Im Untermenü den Befehl Drucken wählen (d). Multiplan listet die ganze Tabelle aus. Die Zeilen- und Spaltennummern sowie Formeln werden nicht ausgedruckt. Nur Text und Zahlen erscheinen.

Merke: Wenn Sie nur einen Teil der Tabelle ausdrucken wollen, dann müssen Sie im Untermenü den Befehl Optionen (o) wählen und darin den Felderbereich eintippen. Im Untermenü Optionen können Sie auch festlegen, ob die Zeilen- und Spaltennummern und die mathematischen Formeln ausgedruckt werden sollen.

11. Die IF-Anweisungs-Formel

Sie hat die allgemeine Form:

wenn(logische Bedingung; Dannwert; Sonstwert)

Beispiel:

wenn(Z3S2<=0; «Berechnung unmöglich«; ln(Z3S2))

Ist die Zahl im Feld Z3S2 kleiner oder gleich 0, dann schreibt Multiplan den Text «Berechnung unmöglich» sonst wird der natürliche Logarithmus der Zahl angezeigt.

Wir wollen noch ein Beispiel für diese IF-Anweisung geben:

Simultanes Lösen von quadratischen Gleichungen

Multiplan soll uns verschiedene quadratische Gleichungen der Form a*x*x + b*x + c = 0 auflösen.

Im Feld Z3S2 steht der Wert von a, im Feld Z4S2 der Wert von b und im Feld Z5S2 der Wert von c.

In das Feld Z7S2 soll der Wert der Diskriminanten D und in die Felder Z8S2 und Z9S2 die Lösungen x1 und x2 geschrieben werden.

Wir schreiben die Formeln spaltenunabhängig, sodass mehrere quadratische Gleichungen nebeneinander gelöst werden können.

Gehen Sie mit dem Tabellencursor in Feld Z7S2, tippen Sie w für Wert ein und schreiben Sie die Formel für die Diskriminante:

Z4 S*Z4 S - 4*Z3 S*Z5 S

Kopieren Sie diese Formel nach rechts.

Gehen Sie mit dem Tabellencursor in Feld Z8S2, tippen Sie w für Wert ein und schreiben Sie folgende IF-Anweisung:

wenn(Z7 S<0;«komplex»;(-Z4 S-wurzel(Z7 S))/2/Z3 S)

Kopieren Sie diese Formel nach rechts.

Gehen Sie mit dem Tabellencursor in Feld Z9S2, tippen Sie w für Wert ein und schreiben Sie folgende IF-Anweisung:

wenn(Z7 S<0;«komplex»;(-Z4 S+wurzel(Z7 S))/2/Z3 S)

Kopieren Sie diese Formel nach rechts.

Jetzt können Sie in Zeile 3 ab Spalte 2 verschiedene a-Werte, in Zeile 4 ab Spalte 2 ebenso viele b-Werte und in Zeile 5 ab Spalte 2 nochmals so viele c-Werte eingeben und Multiplan berechnet Ihnen die korrekten Lösungen. Schauen Sie Abbildung 5 an.

In ähnlicher Art können Sie mit Multiplan eine Vielzahl von mathematischen Problemen lösen, ohne dass Sie eine Programmiersprache lernen müssen.

Logische Verknüpfungen in der Bedingung der IF-Anweisung

wenn(oder(Z2S4<20;Z2S4>65); «nicht aktiv»; «aktiv»)

Ist der Inhalt von Feld Z2S4 kleiner als 20 oder grösser als 65, dann schreibt Multiplan «nicht aktiv» sonst das Wort «aktiv».

wenn(und(Z6S8>=1;Z6S8<=6); «gültige Note»;
«ungültige Note»)

Ist der Inhalt von Feld Z6S8 grösser gleich 1 und kleiner gleich 6, dann schreibt Multiplan «gültige Note» sonst «ungültige Note».

Wir möchten noch darauf hinweisen, dass man in Multiplan eine Berechnung innerhalb eines Feldes iterieren kann. So ist es sogar möglich, die Nullstellen von Gleichungen höheren Grades mit einem Iterationsverfahren wie z.B. dem Halbierungsverfahren zu berechnen. Wir treten aber darauf nicht ein. Dafür sind BASIC- oder Pascal-Programme besser geeignet.

Mathematische Funktionen

abs(Feld)	Absoluter Betrag der Zahl im Feld
anzahl(Liste)	Bestimmt die Anzahl der Felder mit Zahlen
arctan(Feld)	Arcustangens der Zahl im Feld
cos(Feld)	Cosinuswert der Zahl im Feld
exp(Feld)	Exponentialwert der Zahl im Feld
ganzzahl(Feld)	Ganzzahliger Wert der Zahl im Feld
länge(Feld)	Länge des Textes in dem Feld
ln(Feld)	Natürlicher Logarithmus der Zahl im Feld
log10(Feld)	Zehnerlogarithmus der Zahl im Feld
max(Liste)	Grösste Zahl in den Feldern der Liste
min(Liste)	Kleinste Zahl in den Feldern der Liste
mittelw(Liste)	Mittelwert der Zahlen der Listenfelder
pi()	Die Zahl pi
rest(Feld1;Feld2)	Divisionsrest, wenn die Zahl aus Feld 1 durch die Zahl aus Feld 2 dividiert wird
runden(Feld;n)	Die Zahl im Feld wird auf n Dezimalen gerundet
sin(Feld)	Sinuswert der Zahl im Feld
stabw(Liste)	Standardabweichung der Zahlen der Listenfelder
suchen(N;Liste)	In den Feldern der Liste wird nach der Zahl n gesucht
summe(Liste)	Summe der Zahlen der Listenfelder
tan(Feld)	Tangenswert der Zahl im Feld
teil(Textfeld;k;n)	Aus dem Text im Feld wird ab der k. Stelle genau n Zeichen nach rechts herausgeholt
wurzel(Feld)	Quadratwurzel aus der Zahl im Feld

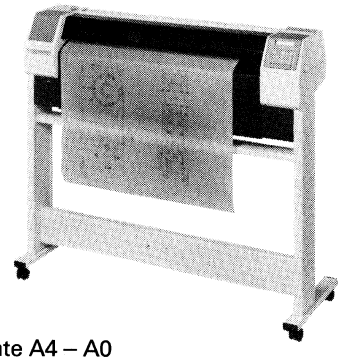
```
#1    |     1     |     2     |     3     |     4     |     5     |     6     |     7     |
  1 Auflösung von quadratischen Gleichungen a*x*x + b*x + c = 0
  2 ------------------------------------------------------------------------------------
  3 a :           1,000       1,000       1,000       1,000       2,000       1,000
  4 b :          -1,000       8,000       5,000       3,000      -3,000       0,000
  5 c :          -6,000      16,000       0,000      12,000      -6,000     -49,000
  6 ------------------------------------------------------------------------------------
  7 Diskr. :     25,000       0,000      25,000     -39,000      57,000     196,000
  8 x1 :         -2,000      -4,000      -5,000     komplex      -1,137      -7,000
  9 x2 :          3,000      -4,000       0,000     komplex       2,637       7,000
 10 ************************************************************************************
 11
 12
 13
 14
 15
 16
 17
 18
BEFEHL: Text Radieren Kopie Löschen Verändern Format Gehezu Hilfe Einfügen
 Schutz Bewegen Name Zusatz Druck Quitt Ordnen Übertragen Wert Ausschnitt Xtern
Einen Befehl auswählen oder Anfangsbuchstaben eingeben
Z1S6      "x + c = 0"                    99% Frei   NL   Multiplan: A:QUADGLEI
```

Abb. 5

GRAPHTEC

A0-Trommel-Plotter GP 9011

Hohe Leistung zu einem vernünftigen Preis



- Papierformate A4 – A0
- Plotgeschwindigkeit 400 mm/sec.
- Mechanische Auflösung 0,005 mm
- Wiederholgenauigkeit < 0,1 mm
- 4-Stifte-Magazin
- GP 9001-kompatibel
- GRAPHTEC-Befehlssatz GP-GL- oder HP-GL-Emulation
- Centronics-RS 232C- oder GP-IB-Interface
- 256 KB RAM-Speicher

SEYFFER+CO. AG
8048 Zürich, Hohlstr. 550, Tel. 01/62 82 00

INCO SERVICE AG ist

Ihr Partner für PC und Zubehör

Beispiele aus unserem Angebot:

PC AT-Comp. Graphik
6/8/10 MHz umschaltbar, Harddisk 20 MB, Floppy 1,2 MHz, Echtzeituhr, 1 parallel und 1 seriell Port, Grafikkarte VISION 7 (EDA/480), 14"-Farbmonitor (NEC MULTISYNC), Swiss Enhanced Keyboard, inkl. graphics Package Dr. HALO II **nur** Fr. 7262.–

20 MB Harddiskkarte, 85ms	Fr. 880.–
20 MB HDU-Drive, 85ms inkl. Controller	Fr. 849.–
Barcodeleser (Datalogic)	Fr. 1165.–
HERKULES	Fr. 149.–
Schönschriftdrucker (Typenrad)	Fr. 480.–
14"-Farbmonitor (NEC Multisync)	Fr. 1759.–

Alle Preise inkl. WUST

Wir liefern Ihnen alle Zusatzkarten für IBM und Kompatible PCs, HDU, Streamer, Floppytape, unterbrechungsfreie Stromversorgungen usw.

Wir beraten Sie gerne und unterstützen Sie auch nach dem Kauf. Wiederverkäufer verlangen bitte Katalog und Preisliste.

INCO SERVICE AG
Albulastrasse 57, 8048 Zürich
Tel. 01/432 23 43

ATARI®

...die professionellste Lösung

ATARI (Schweiz) AG · Baden
Telefon 056 21 14 22

Endlich ist sie da, die komplette Arbeitsstation mit **PC** und **Laser-Drucker** zu einem sensationellen Preis!

- Geräuscharm
- Geringer Platzbedarf
- Schreibmaschinenqualität
- Einfachste Bedienung durch Maus-Steuerung
- Leistungsstark
  - 16/32 Bit Prozessor
  - 2/4 MB Hauptspeicher
  - Eurobus
  - 8 Seiten p/Min. Text und Grafik

Komplettes System inklusive Monitor und Laser-Drucker ab **Fr. 5490.–**

Besuchen Sie uns an der Büfa 87
Halle 331/Stand 571

Gegen Abgabe dieses Inserates erhalten Sie an unserer Info-Bar ein nettes Geschenk (solange Vorrat)

M+K

Vom Umgang mit dBase III PLUS (4)

In M+K 87-4 haben wir ausführlich die listenmässige Datenausgabe besprochen, hierbei sind sicherlich Eingabefehler zum Vorschein gekommen, die es nun zu eleminieren gilt. In der heutigen Folge befassen wir uns daher mit der Mutation der Adressen. Zuvor soll jedoch der Druck von Adressetiketten, eine der Listenausgabe verwandte Datendarstellung, behandelt werden.

Druck von Etiketten

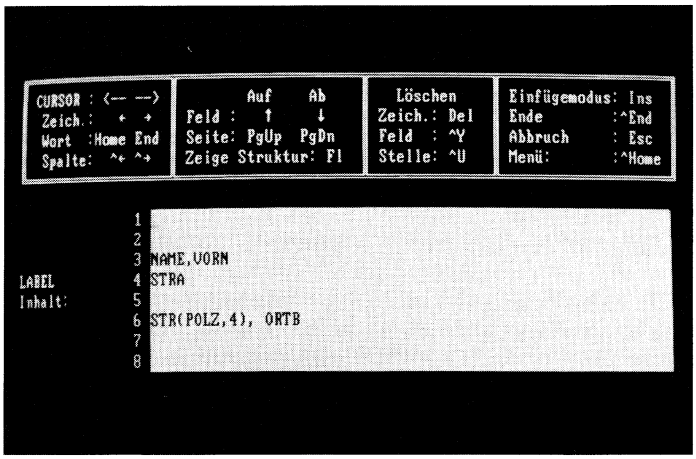
In nahezu allen Textverarbeitungssystemen mit integriertem Mailmerge werden bei Serienbriefen die Anschriften direkt in den Schemabrief eingesetzt. Im Gegensatz dazu muss gerade beim Versand von Drucksachen, wie sie im Vereinswesen bei Rundschreiben oder Protokollen üblich sind, eine direkte Adressierung des Briefcouverts vorgenommen werden. Für diese Adressierungsart werden Endlosetiketten in den verschiedensten Ausführungen im Handel angeboten. Diese Selbstklebeetiketten sind sowohl ein- als auch mehrspaltig und in den verschiedensten Breiten und Höhen lieferbar. Für Adressen haben

Heinz Kastien

sich Etiketten mit einer Breite von 35 Zeichen und einer Höhe von 8 Zeichen als ideal erwiesen. Zum Ausdruck der Etiketten bietet dBase III Plus eine Reihe von Befehlen an, die eine Gestaltung des Etikettenformats und deren Ausdruck ermöglichen. Der wichtigste ist der Makrobefehl «CREATE LABEL». Mit der Befehlsfolge

USE MITGLIED
CREATE LABEL VEREIN

wird der Labelgenerator zum Entwurf der Etiketten aufgerufen, der gleichzeitig auch eine Etikettenformatdatei VEREIN.LBL auf der Disk anlegt. Der Generator gliedert sich in zwei Bildschirmseiten, von denen die erste die Breite der Etikette, Höhe der Etikette, Linker Rand, Zeilen zwischen den Etiketten (vertikal), Platz zwischen den Etiketten (horizontal) sowie Anzahl der Etiketten nebeneinander umfasst und deren Defaultwerte vorgibt. Auf der zweiten Seite können auf den acht definierten Zeilen die gewünschten Felder der Datei übernommen werden.



Wie schon bei der Besprechung des Befehls «CREATE REPORT» im letzten Heft, so kann auch hier mit der Funktionstaste F1 die Struktur der Datei zur Unterstützung beim Etikettenentwurf aufgerufen werden. Numerische Werte müssen über die Funktion STR(Feld) in einen String umgewandelt werden. Felder, die durch Komma getrennt sind, werden in der gleichen Zeile nebeneinander ausgedruckt. Ist die Labeldatei einmal abgespeichert, so können die Etiketten mit der Befehlsfolge

USE MITGLIED
LABEL FORM VEREIN

ausgedruckt werden. Eine Sonderform dieses Befehls lautet

LABEL FORM VEREIN SAMPLE

```
 1. * Etikettenprogramm by H. Kastien 16.06.1987 *
 2. SET ECHO OFF
 3. SET TALK OFF
 4. CLEAR
 5. STORE SPACE(20) TO TEXT
 6. @ 10,20 SAY "Standardtext : " GET TEXT
 7. READ
 8. @ 12,20 SAY "Bitte warten, Ausgabe läuft !"
 9. USE MITGLIED
10. SORT ON NAME, VORN, ORTB/A TO NAMSORT
11. USE NAMSORT
12. SET CONSOLE OFF
13. SET PRINT ON
14. ? CHR(27)+"C"+CHR(9)
15. DO WHILE .T.
16. IF EOF()
17. SET PRINT OFF
18. EXIT
19. ENDIF
20. ? CHR(27)+"!"+CHR(145)
21. ? TRIM(TEXT)
22. ? CHR(27)+"!"+CHR(1)
23. ? TRIM(NAME)+chr(32)+VORN
24. ? STRA
25. ? CHR(27)+"!"+chr(16)
26. ? STR(POLZ,4)+CHR(32)+CHR(27)+"!"+CHR(144)+TRIM(ORTB)
27. ? CHR(12)
28. SKIP+1
29. LOOP
30. ENDDO
31. SET PRINT OFF
32. USE
33. DELETE FILE NAMSORT.DBF
34. SET CONSOLE ON
35. RETURN
```

Es wird eine Probeetikette ausgedruckt, bei der die definierten Zeilen als Punkte dargestellt sind. Sie dient zum Justieren der Etiketten auf dem Printer.

Eine wichtige Option des LABEL-Befehls lautet

LABEL FORM (Datei) Bereich FOR/WHILE

In der Ergänzung kann ein Bereich definiert werden, der ausgedruckt werden soll.

LABEL FORM VEREIN FOR STR(POLZ,1)=«8» TO PRINT

Diese Befehlsfolge bewirkt den Ausdruck aller Adressen auf einem Lineprinter, deren Postleitzahl mit 8 beginnt. Ohne die Option TO PRINT erfolgt die Ausgabe auf dem Monitor. Die Option TO FILE (Name) schreibt die ausgewählten Etiketten unter dem angebenen Namen auf Disk.

Schliesslich kann die Labeldatei jederzeit mit dem Befehl
MODIFY LABEL VEREIN

veränderten Bedürfnissen angepasst werden. Prinzipiell gilt für die Labelbefehle das gleiche, das bereits bei den Listen gesagt wurde. Eine individuelle Gestaltung ist nur im Rahmen der Vorgaben des Labelgenerators möglich. Zur freizügigerer Gestaltung der Etiketten, z.B. mit Fixtexten, verschiedenen Schriftarten oder Unterstreichen der Ortsbezeichnung muss zur direkten Programmierung eines Etikettenprogramms geschritten werden, das wir nachfolgend vorstellen möchten.

1. Zeile
Remark
2. Zeile
Abschalten der Programmkommandos
3. Zeile
Abschalten der Systemmeldungen
Diese drei Zeilen können beim Aufruf aus dem Menü entfallen.
4. Zeile
Löschen des Bildschirms
5. Zeile
Vorbelegung der Variable «TEXT»
6.- 7. Zeile
Eingabe eines Fixtextes (z.B. Einschreiben, Drucksache usw.)
8. Zeile
Systemmeldung
9.-11. Zeile
Initialisierung der Datei und Sortieren
12.-13. Zeile
Unterdrücken der Anzeigen auf dem Bildschirm
14. Zeile
Einstellen der Zeilenlänge auf neun Zeilen
15. Zeile
DO WHILE ... Schleife
16.-18. Zeile
Wird das EOF der Datei erreicht, verlässt das Programm durch die IF .... ENDIF-Verzweigung die Schleife
19.-25. Zeile
Ausdruck der Etikette, die Umschaltung des Printers erfolgt über den kombinierten Druckmode ESC «!» (n)
n = 1 Normalschrift
n = 145 Eliteschrift-Fettdruck-Unterstrichen
n = 144 Normalschrift-Fettdurck-Unterstrichen
n = 16 Normalschrift-Fettdruck
26. Zeile
Formfeed
27. Zeile
Aufruf des nächsten Records
28.-29. Zeile
Schleife und ENDDO
30. Zeile
Abschalten des Printers
31. Zeile
Schliessen der Datenfile
32. Zeile
Löschen des Sortfiles
33. Zeile
Einschalten der Bildschirmkontrolle
34. Zeile
Rückkehr zum Menü oder Interpreter

Das Programm eröffnet durch den Standardtext und die unterschiedlichen Schriftarten zahlreiche Gestaltungsvarianten. Werden andere Etikettenformate verwendet, so muss der Wert für die Etikettenlänge in Zeile 14 geändert werden. Das Programm unterscheidet sich nur unwesentlich vom Listenprogramm in M+K 87-4. Auch hier werden die Adressen sortiert. Im Gegensatz zum Listenprogramm werden diesmal aber mit SET PRINT ON/OFF nicht nur die SAY-Befehle an den Printer umgeleitet, sondern die gesamte Kontrolle dem Printer übergeben, daher entfallen bei diesem Programm die PCOL() und PROW()-Befehle. Neu in diesem diesem Programm ist lediglich der TRIM-Befehl, er entfernt nachfolgende Leerstellen eines Strings.

Editieren von Datensätzen

Unter Editieren oder Mutieren von Datensätzen versteht man die Aenderung der Feldinhalte oder das Löschen ganzer Records. Prinzipiell gliedert sich jede Mutation in die drei Schritte Record suchen, Record mutieren und Record abspeichern.

Record suchen

Auf jeden Record eines Datenfeldes kann auf drei Arten zugegriffen werden, nämlich Zugriff über die Datensatznummer, direkter Zugriff auf ein Feld und Zugriff über den Index.
Listet man alle Records einer Datei mit

USE MITGLIED
DISPLAY ALL

auf dem Bildschirm oder mit

TYPE MITGLIED TO PRINT

auf einem Lineprinter auf, so erkennt man, dass die Datensätze durch dBase III Plus fortlaufend numeriert werden. Diese Datensatznummern werden auch beim Einfügen oder Löschen von Datensätzen ständig nachgeführt. Es ist also möglich, mittels der Datensatznummern auf die einzelnen Records zuzugreifen, sofern eine Verbindung zwischen dem gesuchten Namen und der Datensatznummer besteht.

USE MITGLIED
GO 5
DISPLAY

oder

USE MITGLIED
DISPLAY RECORD 5

zeigt den 5. Datensatz auf dem Bildschirm an. Die Suche bestimmter Datensätze über die Satznummer ist sicherlich sehr umständlich, denn es müsste zu diesem Zweck immer eine ausgedruckte Liste der Datensätze vorhanden sein, aus der die Recordnummer ersichtlich ist. Diese Art der Aufrufs definierter Datensätze hat daher auch nur rein theoretisches Interesse. Vielmehr soll an diesem einfachen Beispiel der GO-Befehl näher beleuchtet werden.

GO RECORDNUMMER
Springt auf den genannten Datensatz, hierbei kann die Recordnummer auch eine Variable sein
GO TOP
Springt auf den ersten Datensatz der Datei
GO BOTTOM
Springt auf den letzten Datensatz der Datei

Editieren

Diesem Befehl werden wir in den Programmen noch mehrfach begegnen. Wird nun der Befehl DISPLAY durch den Befehl EDIT ersetzt, so wird der Datensatz angezeigt und kann gleichzeitig editiert werden, da sich die Felder beliebig überschreiben lassen und auf der Disk wieder zurückgeschrieben werden. Die Befehlsfolge

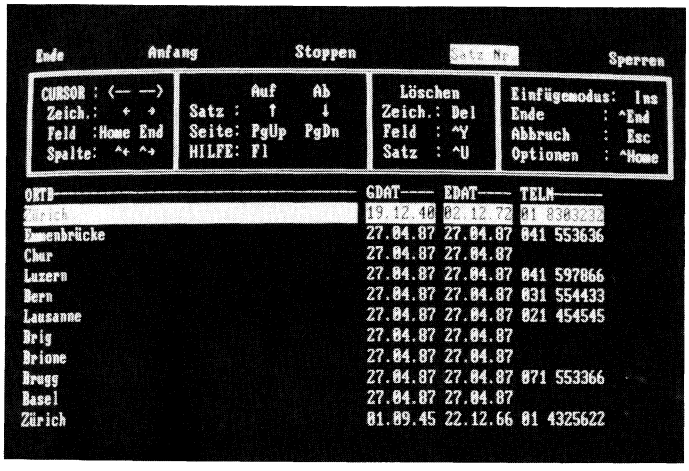
GO 5
EDIT

kann durch

EDIT 5

ersetzt werden. Es wird die Adresse mit der Datensatznummer 5 aufgerufen, auf dem Bildschirm dargestellt und nach der Aenderung unter der gleichen Satznummer wieder abgespeichert. Der Editmode wird durch die «ESC»-Taste unterbrochen und zum dBase-Interpreter zurück gesprungen. Die Darstellung der Daten auf dem Monitor wurde bereits in M+K 87-2 gezeigt.

Ein dem EDIT-Befehl nahestehender Befehl lautet «BROWSE». Er erlaubt nicht nur das Editieren sondern auch das Anfügen neuer Datensätze an eine Datei. Der BROWSE-Befehl stellt im Gegensatz zum EDIT-Befehl jeweils in Abhängigkeit vom Hilfsmenü, zwischen 11 bis 17 Records auf dem Bildschirm dar. Durch weitere Untermenüs werden zusätzliche Editierbefehle, wie Suchen, Sperren usw. in die Kopfzeilen eingeblendet. Ueberschreitet die Recordlänge die Bildschirmbreite von 80 Zeichen, so können auch die nicht sichtbaren Felder mit CTRL (Cursor rechts/links) auf dem Bildschirm angezeigt werden. Leider sind die Befehle BROWSE und EDIT mit dem Clipper nicht compilierbar. Daher muss auf diese Befehle verzichtet werden, wenn eine Compilierung vorgesehen ist.



Da die Mutation mittels EDIT und BROWSE sehr umständlich ist, beschreiben verschiedene Methoden, mit denen die Suche innerhalb der Datei nach Schlüsselworten möglich ist. Bei einer Adressdatei ist das Schlüsselwort folgerichtig immer der Name. Andere Schlüsselworte spielen nur bei der Listenausgabe eine Rolle. Wird nach diesem Schlüsselworten gesucht, so ist es entscheidend, ob die Suche in der Hauptdatei oder in der Indexdatei erfolgt. Festzuhalten ist, dass die Suche in der Indexdatei die schnellste und eleganteste Methode ist und hierfür spezielle Befehle zur Verfügung stehen. Vorab soll aber mit einer Reihe bekannter Befehle die Hauptdatei direkt nach dem Schlüsselbegriff durchsucht werden. Bei dieser Suchmethode ist eine Indexierung der Datei nicht erforderlich. Das folgende Listing zeigt eine einfache Programmstruktur, welche die Stammdatei nach einem Schlüsselwort absucht und die Adresse anzeigt.

```
* Editierroutine Demoprogramm 2 *
CLEAR
USE MITGLIED
SNAME=SPACE(40)
@ 5,10 SAY "Name          : " GET SNAME
READ
DO WHILE .T.
IF SNAME = NAME
EDIT
ENDIF
IF EOF()
EXIT
ENDIF
SKIP+1
LOOP
ENDDO
```

1. Zeile
Remark
2. Zeile
Ausschalten der Kommandoanzeige
3. Zeile
Ausschalten der Systemanzeigen
4. Zeile
Löschen des Bildschirms
5. Zeile
Eröffnen der Datei Mitglied
6. Zeile
Die Variable SNAME wird gleich einem String von 40 Leerschlägen gesetzt.
7. Zeile
Der Text «Name : » wird auf dem Bildschirm ausgegeben und gleichzeitig der zu suchende Name mittels GET abgefragt
8. Zeile
READ Anweisung zum GET-Befehl
9. Zeile
DO WHILE-Schleife zur Erhöhung der Recordnummer
10. Zeile
IF-Befehl, es wird verglichen, ob das Feld NAME des eingelesenen Records mit dem zu suchenden Namen identisch ist.
11. Zeile
Ist die Bedingung erfüllt, wird der Datensatz angezeigt und kann mit EDIT mutiert werden.

12. Zeile
Beendigung des Vergleichs mit ENDIF
13. Zeile
Ist das Ende der Datei erreicht, bricht die Erhöhung des Schleifenzählers ab. Dieser Vergleich erfolgt durch die Abfrage ob EOF() erfüllt ist. IF EOF() .T.
14. Zeile
Bei Erfüllung der Bedingung kehrt das Programm zum Interpreter zurück
15. Zeile
Beendigung der Verzweigung mit ENDIF
16. Zeile
Ist die Bedingung NAME = gesuchtem Schlüssel nicht erfüllt, wird mit SKIP+1 der Datensatzzeiger um 1 erhöht.
17. Zeile
Schleife
18. Zeile
Ende des Programms mit ENDDO

Da bei diesem Demoprogramm die Systemanzeige eingeschaltet worden ist, kann man auf dem Bildschirm sehr gut erkennen, wie nach der Eingabe des Schlüsselwortes der Datensatzzeiger ständig erhöht wird bis die Identität des Feldes «NAME» und des Schlüsselwort gegeben ist.

```
* Editierroutine Demoprogramm 3 *
SET ECHO OFF
SET TALK OFF
CLEAR
USE MITGLIED
DO WHILE .T.
CLEAR
SNAME=SPACE(40)
GO TOP
@ 5,10 SAY "Name          : " GET SNAME
READ
DO WHILE .T.
IF SNAME = NAME
@ 5,10 SAY "Name          : "
@ 5,26 SAY NAME
@ 7,10 SAY "Vorname       : "
@ 7,26 SAY VORN
@ 9,10 SAY "Strasse & Nr. : "
@ 9,26 SAY STRA
@ 11,10 SAY "Postleitzahl  : "
@ 11,26 SAY POLZ
@ 13,10 SAY "Ort           : "
@ 13,26 SAY ORTB
@ 15,10 SAY "Telefonnr.    : "
@ 15,26 SAY TELN
ENAME = NAME
EVORN = VORN
ESTRA = STRA
EPOLZ = POLZ
EORTB = ORTB
ETELN = TELN
@ 5,26 GET ENAME
IF ENAME = SPACE(40)
CLEAR
RETURN
ENDIF
@ 7,26 GET EVORN
@ 9,26 GET ESTRA
@ 11,26 GET EPOLZ
@ 13,26 GET EORTB
@ 15,26 GET ETELN
READ
REPLACE NAME WITH ENAME
REPLACE VORN WITH EVORN
REPLACE STRA WITH ESTRA
REPLACE POLZ WITH EPOLZ
REPLACE ORTB WITH EORTB
REPLACE TELN WITH ETELN
EXIT
ENDIF
SKIP +1
LOOP
ENDDO
LOOP
ENDDO
```

Die Adresse steht nun zur Mutation mittels des EDIT-Befehls bereit. Selbstverständlich ist es noch ein weiter Weg, bis aus diesem primitiven Suchalgorithmus eine komfortable Editierroutine geworden ist, denn mit diesem Demoprogramm können Mehrfachnennungen nicht erfasst werden, ebenso werden nicht vorhandene Adressen nicht als solche ausgewiesen, sondern es erfolgt nur ein Abbruch des Programms. Im nächsten Listing ist die Suche nach dem richtigen Schlüsselbegriff die gleiche und bedarf daher keiner weiteren Erklärung, jedoch unterscheidet sich dieses Demoprogramm vom vorhergehenden dadurch, dass hier nicht der EDIT-Befehl verwendet worden ist, sondern die einzelnen Felder individuell dargestellt werden und mit der bekannten GET-READ-Routine editiert werden können. Schliesslich erfolgt die Abspeicherung durch den schon mehrfach erwähnten REPLACE-Befehl.

Die Suche nach einem Schlüsselwort innerhalb der Stammdatei ist sehr langsam, da immer der ganze Datensatz eingelesen werden muss, der bei grossen Dateien sehr lang sein kann. Wesentlich effizienter kann in der Indexdatei nach einem Schlüsselbegriff gesucht werden. Da dieser Dateityp im Vergleich zur Hauptdatei immer sehr klein ist, wird die Suche dementsprechend schnell. Es darf allerdings nicht verschwiegen werden, dass diese Suchmethode bei Mehrfachnennungen eines Schlüsselbegriffs völlig versagt. Entweder muss dann nach der oben beschriebenen Methode die Hauptdatei durchsucht werden oder es muss nach mehreren Schlüsselbegriffen wie z.B. NAME, VORNAME und GEBURTSDATUM ermittelt werden. Vorerst wollen wir jedoch auf diese Einschränkungen keine Rücksicht nehmen und im folgenden Demoprogramm die prinzipielle Arbeitsweise besprechen.

```
* Editierroutine Demoprogramm 4 *
CLEAR
USE MITGLIED INDEX NAMIND
SNAME = SPACE(40)
@ 10,20 SAY "Name : " GET SNAME
READ
SEEK SNAME
EDIT
```

Dieses Programm ist nahezu identisch mit dem oben besprochenen Suchprogramm in der Hauptdatei, jedoch wird hier neben der Hauptdatei Mitglied auch die Indexdatei NAMIND.NDX initialisiert, die nach dem Schlüsselbegriff Namen aufgebaut ist. Der dBase-Befehle SEEK durchsucht die Indexdatei nach dem definierten Schlüsselbegriff. Sobald er gefunden ist, ruft das Programm auf Grund der Satznummer aus der Hauptdatei den gesamten Record auf und stellt ihn auf dem Monitor zur Mutation bereit. Mittels dieses neuen Befehls ist also ein sehr schneller Zugriff über die Indexdatei möglich. Eine Erhöhung des Zählers oder ein Zurücksetzen vor einem neuen Suchdurchlauf ist hier nicht nötig, da dies bereits in den Suchalgorithmus des SEEK-Befehls integriert ist. Wird das Ende der Indexdatei erreicht, ohne den Suchbegriff gefunden zu haben, so erfolgt die Anzeige «Nicht gefunden» auf dem Monitor und EOF() wird auf .T. gesetzt. Aehnlich wie der SEEK-Befehl arbeitet FIND. Da jedoch bei Mehrfachnennungen immer nur das erste Schlüsselwort erkannt wird, muss in diesem Fall durch Erhöhung des Schleifenzählers von der Startposition der ersten Nennung ausgehend, in der Hauptdatei weiter gesucht werden.

Zusammenfassung der Editiervarianten

Die Mutation der Datensätze erfolgt entweder global mit den Befehlen EDIT oder BROWSE oder in einer individuell

aufgebauten Datenmaske durch Darstellung des Datenfeldes mit SAY und Uebernahme der alten oder mutierten Felder mit GET sowie der Rückspeicherung auf der Disk mit REPLACE und USE. Diese Befehlsfolgen wurden aber bereits in der letzen Folge beim Eingabeprogramm besprochen und bedürfen hier keiner weiteren Erklärung.

Löschen von Datensätzen

Zum Mutieren von Dateien gehört auch das Löschen von Records. Eine Besonderheit des dBase III Plus beim Löschen, ist die Unterteilung dieses Makrobefehls in die Teilbefehle «Markierung von Datensätzen zum Löschen» und «Löschen der markierten Datensätze». Einzelne Felder in einem Record können nicht gelöscht, sondern nur überschrieben werden. Vier Befehle vereinigen alle Funktionen, die zum Löschen von Datensätzen erforderlich sind.

DELETE 5
Markiert den Datensatz 5 zum Löschen
DELETE WHILE RECNO() < 5
Markiert alle Datensätze zum Löschen, deren Record-Nummer kleiner als 5 ist.
DELETE ALL
Markiert alle Datensätze zum Löschen

Bei der Markierung werden die Datensätze nicht aus der Datei entfernt, sondern nur mit einem Stern «*», der Löschmarkierung, versehen. Wird die Datei nach einem Feldbegriff abgesucht, so werden die Datensätze nicht mehr angezeigt. Der Befehl DISPLAY ALL zeigt jedoch auch weiterhin die markierten Datensätze an.

RECALL
Macht die Löschmarkierung wieder rückgängig.

Zur definitiven Entfernung der Datensätze aus der Datei muss nach der Markierung der Befehl

PACK

verwendet werden. PACK entfernt die markierten Datensätze definitiv aus der Datei und führt eine Neunumerierung der restlichen Datensätze durch. Die Befehlsfolge

DELETE ALL
PACK

die alle Datensätze löscht, kann durch den Befehl

ZAP
oder
ERASE (Name)

ersetzt werden. ZAP löscht definitv alle Datensätze einer Datei, ERASE löscht die Datei selbst.

Dieses Programm markiert nur die zum Löschen vorgesehenen Datensätze, das eigentliche Löschen muss mit dem PACK-Befehl erfolgen. Selbstverständlich lässt sich diese Erweiterung auch in das Programm einbauen. Um unbeabsichtigtes Löschen während der Uebungen zu vermeiden, haben wir dies hier unterlassen.

```
* Löschroutine Demoprogramm 5 *
CLEAR
USE MITGLIED
SNAME=SPACE(40)
@ 5,10 SAY "Name          : " GET SNAME
READ
DO WHILE .T.
IF SNAME = NAME
DELETE
ENDIF
IF EOF()
EXIT
ENDIF
SKIP+1
LOOP
ENDDO
```

Die hier beschriebenen Demoprogramme können selbstverständlich nur die prinzipielle Arbeitsweise der Befehle erklären, ohne auf eine praxisgerechte, komfortable Programmierung solcher Programme einzugehen. Wir werden daher in der nächsten Folge mit einem Mutationsprogramm aufwarten, das alle Finessen enthalten wird. □

Jede Disk Fr. 8.–

Public Domaine Software

für IBM-PC und kompatible Systeme

Wir versichern Ihnen, dass Sie Ihren PC mit unserer SHAREWARE noch viel, viel effizienter einsetzen können, denn ein PC ohne Computer Freeware SHAREWARE ist nur ein halber PC. Unser Wort darauf!

Unser SHAREWARE-Angebot umfasst:

Weit über 1100 Disketten aus über 80 Themengebieten für Ihren PC/XT/AT:

Gegen Fr. 10.– in Brief oder Überweisung auf PC 69-5953-0 Sercoex SA, 6948 Porza-Lugano, senden wir Ihnen drei Disketten mit Katalog und Demo-Programmen zum Kennenlernen. Viele Programme in Basic zum Anschauen oder Kopieren. Computermodell angeben.

Weitere Informationen über Telefon 091/52 80 33 oder Postkarte an:

E. Marbach, via Cantonale 42, 6948 Porza

Deutsche Programme

Obwohl zurzeit dem englischsprachigen Shareware-Angebot zahlenmässig noch unterlegen, wächst unser Angebot an deutscher Software beständig. Dieses Paket enthält hauptsächlich allgemeine Geschäftsprogramme wie Buchhaltung usw. Auch eine deutsche Spieldiskette ist dabei!

PC-Write 2.7

Professionelles, superschnelles Textverarbeitungssystem mit allen Merkmalen von 1000 Fr.-Programmen wie: Blocksatz, voller IBM-Zeichensatz, Seitenumbruch, Dutzende Drucker fertig angepasst, DOS-Zugriff, Löschen, Verschieben, Maus-Unterstützung.

Direktimport von Computern und Elektronik

Unser heutiges Angebot:

XT 8088 Turbo 10 Mhz inkl. 640 KB, 2 × 360 Floppy, Schnittstellen, 14" Monitor monochrom **Fr. 1750.–**

dito HD 20 **Fr. 2650.–**

AT 80286 10 Mhz 1 MB HD 20 (Seagate), 1,2 MB Floppy, Schnittstellen, 14" Monitor monochrom, Herkules-Karte **Fr. 3600.–**

dito HD 40 **Fr. 4100.–**

Sonderangebot

Genius Mouse GM 6 Mouse + Micro-Soft Compatibel, inkl. Software (solange Vorrat) **Fr. 150.–**

NEU

Verwandeln Sie Ihren PC in einen Telefax Fax-Karte inkl. Software **Fr. 1900.–**

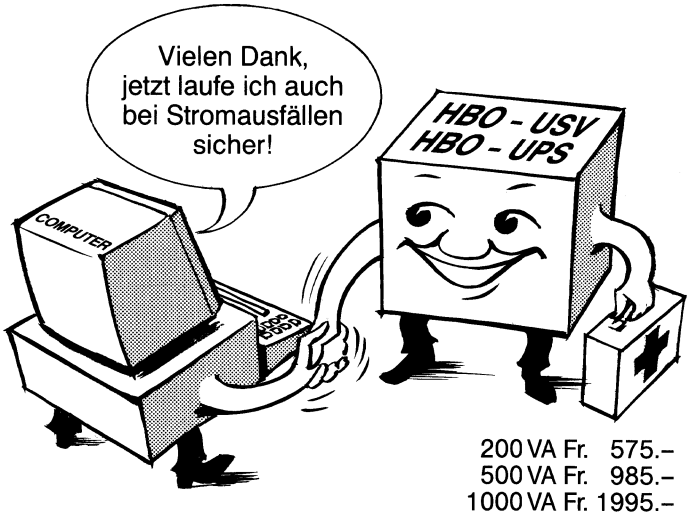
Diverse günstige Angebote von Software, Zubehör und Drucker

Preise ab Zofingen inkl. Wust

Verkauf-, Service- und Schulung für Hard- und Software

SWIF TRADE AG

Software- und Informatik-Fachhandel
CH-4800 Zofingen, Pommerngut
Tel.- und Fax-Nr.: 062/51 82 00



200 VA Fr. 575.–
500 VA Fr. 985.–
1000 VA Fr. 1995.–

Unterbrechungsfreie Stromversorgungsanlagen (USV) 200 VA bis 12 kVA

HBO Hans Baumann Optoelektronik
CH-8605 Gutenswil
Telefon 01/945 28 29

Original USA-Teile

Speed-Karte 286 für XT	**Fr. 690.–**
Intel Speed-Karte 386	**Fr. 2990.–**
2 MB RAM-Karte Intel (LIM)	**Fr. 490.–**
20 MB Harddisk 65 ms inkl. Controller	**Fr. 948.–**
PT-386 AT/40 Speedkarte Intelboard 386 40 MB Harddisk 40 ms	**Fr. 8990.–**
EGA-Monitor 14"	**Fr. 998.–**
Set: HEGA-Karte + Monitor 14"	**Fr. 1450.–**
Set: Genova Super EGA + Muliscan-Monitor 14"	**Fr. 1990.–**
HEGA-Karte	**Fr. 450.–**
Cenova Super EGA-Karte	**Fr. 798.–**

Händler-Anfragen erwünscht

ACHTUNG Grossabnehmer: Industrie, Softwarehäuser, Schulen, TAIWAN-PREIS!

Original Qualitäts-Hardware und Zubehör dank Direktimport zu US-Preisen!

RUFEN SIE UNS AN – WIR LIEFERN.

Generalimporteur:

Computer
M + M Data AG
Lerchenweg 3, 6210 Sursee
Tel. 045 / 21 62 29
Tel. 045 / 21 44 44

Wenn die Kopie besser, günstiger ist als das Original!

PT-16 AT-02 Turbo 6/8/10 MHz Takt

100% kompatibel zum Industrie-Standard

- CPU 80286 16/32 Bit
- 1×20-MB-Festplatte
- Centronics parallel, Clock, Multikarte
- MSDOS 3.1
- RAM 512 KB
- 1×1,2-MB-Laufwerk
- Tastatur VSM, sep. Cursor

Fr. 3590.– mtl. Fr. 195.– Leasing

PT-16 AT-03 Turbo

dito PT-16 AT-02, jedoch 40-MB-Festplatte

Fr. 4990.–

PT-16 PC Turbo

- CPU 8088 16 Bit
- 1×Laufwerk 360 KB
- RS 232 C + 2×Centronics parallel, Clock, Multikarte
- Tastatur VSM, 4,77 oder 8 MHz Takt
- 640-KB-RAM
- Graphic 720×348 P.

Fr. 1490.–

jedoch Harddisk 25 MB, Festplatte inkl. Controller 65 ms

Fr. 2490.–

Portable LCD 286/20 MB

Compaq-kompatibel

Fr. 5995.–

Monochrom-Monitor 14" bei AT/XT/20 gratis

Für alle öffentlichen und privaten Schulen!!!

S M S - II
(Stundenplan-Management-System, Release 1.0)!

Das Programm-Paket für die computerunterstützte Stundenplan-Erstellung auf Ihrem **IBM- oder 100%ig kompatiblen PC**. Bedienung über **«Pulldown»-Menüs** und Funktionstasten; **Plan**: 6 Tage mit **bis zu 20 Stunden; 2 Fächer/Stunde**; Parallelkurse; **Eingabe über Klassenpläne**, Lehrer- und Zimmerpläne automatisch verwaltet! Max. **je 200 Lehrer-, Klassen- und Zimmerpläne** möglich. Umfassende «Online»-Hilfe. Unterstützt MS-DOS-Struktur.

Jakob Heider, Jr./Hard- & Software-Beratung/Hertensteinstrasse 25/CH-8052 Zürich

FINANZ-
BUCHHALTUNG
PFIB – 4 standardmässig MIT
Budget- und Vorjahresvergleich, freiem
4-stelligem Kontoplan, Standardtexten,
Probebilanz, Mandantenfähigkeit, Passwort-
schutz, Anleitung mit Beispiel, Fenstertechnik,
Ausgaben auf Drucker oder Disk, Sortier-
möglichkeiten. OHNE Kopierschutz.
Alles inklusiv Fr.950.–, vollständige
Probier-Version Fr.50.–
ARNOLD, 061/22 17 52
Rütlistrasse 39
4051 Basel

Und wenn es zwölf wären: es wäre immer noch keine Dutzendware.

WENGER ist das Schweizer Unternehmen, das nur eines baut: Printer, Printer und noch einmal Printer.

Da erstaunt es nicht, dass WENGER Printer keine Dutzendware sind. Denn ein Spezialist wie wir kann auf Kundenbedürfnisse besser und präziser eingehen.

Deshalb deckt allein schon unser Standardsortiment an Matrix- und Laserprintern ein breites Anwendungsfeld ab. Und deshalb gibt es von den Standardmodellen viele Varianten und dazu viele Sonderausstattungen.

Was die WENGER Printer, ausser ihrem Namen, verbindet, ist, dass sie überdurchschnittlich leise sind. Und überdurchschnittlich schnell. Und überdurchschnittlich belastbar. Und von überdurchschnittlicher Qualität in den Materialien und der Verarbeitung. Und von überdurchschnittlicher Qualität im Ausdruck.

Kein Wunder, dass sie heute schon in Hunderten von Firmen in ganz Europa mit Tausenden von Computern eng verbunden sind. Hätten Sie von einem Spezialisten etwas anderes erwartet?

☐ Schicken Sie uns bitte eine Dokumentation über Ihr Druckersortiment.

☐ Wir würden gerne mit einem Ihrer Aussendienst-Mitarbeiter reden.

Name: _____

Firma: _____

Strasse: _____ MK

PLZ/Ort: _____

Telefon: _____

Einsenden an: WENGER PRINTERS AG,
Im Kägen 23/25, 4153 Reinach 1, Tel. 061/76 87 87.

WENGER

Wer nur eines macht, kann vieles besser machen.

WALKOM PORTABLE AT

Erhältlich im Fachhandel oder direkt durch uns:

Fr. 5'765.–
(AT mit 20 MB HD)

Volle AT Desktop Leistung in einem kompakten tragbaren LCD Computer. Die neuen tragbaren WALKOM LCD Computer stellen alles in den Schatten, was man bisher von LCD-Computern gewohnt war. Im Gegensatz zu anderen Laptop Computern besitzt der WALKOM ein sehr kontrastreiches hintergrundbeleuchtetes LCD-Display, das mit einem normalen externen Monitor absolut konkurrenzfähig ist. Und beide, der XT kompatible LCD-88 und der AT kompatible LCD-286 haben die Kapazität und Ausbaufähigkeit eines grossen Tischgerätes. Die 6 Erweiterungssteckplätze, 2 x 5¼" Laufwerke (oder 1 Festplatte und 1 Laufwerk), das automatisch umschaltbare Netzteil 110V/220V machen den WALKOM zu einem vollwertigen tragbaren Computersystem.

Hinzu kommt eine 86-Key Tastatur mit 12 Funktionstasten, bis zu 45° neigbares LCD-Display, kompaktes attraktives Gehäuse mit Luxus-Lederkoffer auf Wunsch; kurzum die beste Lösung für tragbare Unabhängigkeit und volle AT-Leistung.

ELECTRONIC MARKETING AG
Your Swiss distributor for high technology
Bahnhofstrasse 60 4132 Muttenz-Basel
Tel. 061-615353 Tlx 965669 Fax 061-424549

EDV LEHMANN AG

Superpreise Computerzubehör

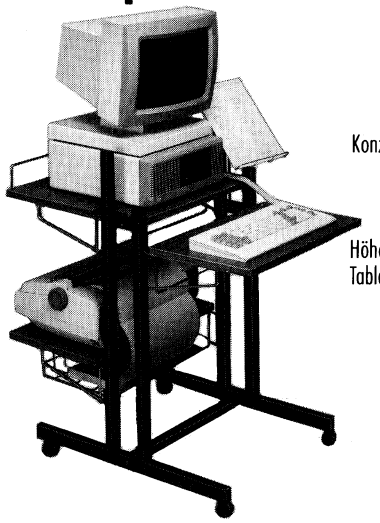
Disketten:

10 No Name 3 1/2" MF2DD	Fr. 29.--
10 Maxell 3 1/2" MF2-DD	Fr. 55.--
10 No Name 5 1/4" 2D	Fr. 9.--
10 SKC 5 1/4" MD 2D	Fr. 18.--
10 Maxell 5 1/4" MD2-D	Fr. 29.90
10 Maxell 5 1/4" MD2-HD	Fr. 59.--
Diskettenbox 3 1/2", abschliessbar	Fr. 17.--
Diskettenbox 5 1/4", abschliessbar	Fr. 12.80
Monitorständer BMC, frei beweglich	Fr. 38.--

TV Lehmann AG , Oltnerstr. 18 ,
5012 Schönenwerd, Tel. 064/41 58 21

EDV LEHMANN AG

REXEL
Personal Computer Arbeitstisch



Idealer, äusserst preisgünstiger Arbeitstisch für alle Einheiten eines Personal Computers oder Textsystems. Mobiler Arbeitsplatz, der je nach Bedarf von einem Büro in ein anderes verschoben werden kann. Viele Möglichkeiten zum Verstellen der einzelnen Tablare. Gestell aus dunkelbraunem, vierkantigem Stahlrohr, Tablare aus Holz mit Kunststoffabdeckung in Holzimitation. Platzbedarf: Breite 700 mm / Tiefe 900 mm.

In guten Bürofachgeschäften und Computershops erhältlich.
Bezugsquellennachweis und Prospektunterlagen durch:
Rexel Signa AG, Flughofstrasse 50, 8152 Glattbrugg, Tel. 01/810 66 71

BON für Gratis-Dokumentation: CMK
Firma:
Name/Abteilung:
Strasse:
PLZ/Ort:

GFU - General File Utility

Die General File Utility ist als eine Art Betriebssystemerweiterung gedacht, die es auf einfache Weise erlaubt, die Dateiattribute (read-only, hidden, system, archive) frei zu manipulieren. Dabei können auch Unterverzeichniseinträge manipuliert (z.B. versteckt) werden. Zusätzlich existiert eine Möglichkeit, für beliebige Dateien deren Attribute zu kontrollieren und es ist möglich, alle Dateien, die ein bestimmtes Attribut haben, aufzulisten.

Marc Scheuner

Programmablauf

Die Dateiattribute können zwar auch mit Hilfe von DOS Debug, Norton Utilities, PC-Tools oder ähnlichen Utilities manipuliert werden, dies erfordert jedoch etliche Sucherei und Kenntnis der DOS-Directory Struktur und man kann damit jeweils nur eine Datei aufs Mal bearbeiten. GFU ermöglicht es, mit einem einfachen DOS-Befehl auf einen Schlag beliebig viele Dateien zu manipulieren. Zudem braucht sich der Benutzer nicht darum zu kümmern, wie DOS seine Directories aufbaut.

Normalerweise hat eine Datei keine Attribute, oder nur das archive-Attribut. Dieses gibt an, dass diese Datei seit dem letzten Backup der Platte (Festplatte oder Diskette) verändert wurde. Dieses Attribut wird von den DOS-Befehlen BACKUP und RESTORE verwendet, um herauszufinden, welche Dateien zu sichern sind.

Das readonly-Attribut schützt die Datei vor versehentlichem Löschen mit den DOS-Befehlen DELETE oder ERASE. Damit können zum Beispiel Programmdateien (COM- und EXE-Dateien) geschützt werden. Zu beachten ist aber, dass sich dieser Schutz nicht für Datendateien eignet, die von Programmen beschrieben werden sollen.

Das hidden-Attribut schützt die Datei ebenfalls vor versehentlichem Löschen und versteckt sie zudem vor dem DOS-Befehl DIR. Damit können Dateien versteckt werden, die andere nicht zu Gesicht bekommen sollen oder die nicht immer im DIRrectory erscheinen sollen/müssen. Dieses Attribut ist geeignet z.B. für Message-Dateien (.MSG, wie in Turbo Pascal und anderen Programmen), für Help-Dateien (.HLP) und z.B. auch Konfigurationsdateien. Allerdings reagieren verschiedene Programme verschieden auf versteckte Dateien. Turbo Pascal scheint keine Probleme zu kennen, es findet die versteckte TURBO.MSG Datei ohne Probleme - andere haben mehr Mühe.

Das system-Attribut wird für DOS-Systemdateien verwendet. Dieses dürfte kaum von Bedeutung sein. Es versteckt die Datei ebenfalls. Files, die versteckt oder blockiert sind, können ohne Probleme gestartet werden (wenigstens unter DOS 3.1), ebenso Files in versteckten Directories.

Der Aufruf von GFU erfolgt nach folgendem Schema:

A>GFU [filespec] [/action]

Dabei steht «filespec» für die Angabe der gewünschten Dateien. Hierbei können auch Laufwerks- und Unterverzeichnisangaben verwendet werden sowie die sogenannten Wildcards ? und * benutzt werden.

«action» steht für die gewünschte Aktion, die mit den Dateien ausgeführt werden soll. Folgende Aktionen werden unterstützt:

/N	Normalisieren, d.h. read only-, hidden- und system-Attribut werden zurückgesetzt
/R+,/R–	Setzen (+) respektive Löschen (-) des readonly-Attributs
/H+,/H–	dito für das hidden-Attribut
/S+,/S–	dito für das system-Attribut
/A+,/A–	dito für das archive-Attribut
/C	Kontrolle der Attribute
/LN	Listet alle «normalen» Dateien auf
/LR	Listet alle readonly-Dateien auf
/LH	Listet alle hidden-Dateien auf
/LS	Listet alle system-Dateien auf
/LA	Listet alle archive-Dateien auf

Folgende Punkte sind zu beachten:

1) Der Aufruf von GFU kann auch ohne Parameter geschehen, dann werden die nötigen Daten direkt am Bildschirm abgefragt. Man kann auch z.B. nur die «filespec» angeben, dann wird die Aktion abgefragt.
2) Gibt man keine «filespec«, aber eine gültige «action» an, dann wird automatisch die Dateispezifikation *.* angenommen, d.h. es sind alle Dateien des aktuellen Laufwerks und Verzeichnisses betroffen.
3) Gibt man eine ungültige Aktion an, so erscheint eine entsprechende Meldung, ohne dass unerwünschte Effekte auftreten sollten.
4) Werden zu der angegebenen Dateispezifikation keine passenden Dateien gefunden, so wird dies ebenfalls gemeldet.
5) Sollen alle Dateien eines Laufwerks/Verzeichnisses angesprochen werden, so kann man *.* weglassen und nur z.B. B: oder A:\SUB\ angeben.
6) Will man nicht die Dateien eines Verzeichnisses ansprechen (mit SUB\ ), sondern das Verzeichnis selber, um es z.B. zu verstecken, so muss nur A:\SUB angegeben werden.

Dazu einige Beispiele:

A>GFU IBM*.COM /C
kontrolliert die Attribute der Betriebssystemdateien IBMBIO.COM und IBMDOS.COM.

A>GFU IBM*.COM /N
normalisiert diese Dateien, so dass sie mit dem normalen COPY-Befehl von DOS kopiert werden können.

A>GFU /LH
Listet alle (*.*) Dateien des aktuellen Laufwerks/Verzeichnisses, die das hidden-Attribut haben (und eventuell auch noch andere Attribute).

A>GFU B:\PASCAL\SOURCE\ /A+
Verpasst allen Dateien im Verzeichnis \PASCAL\SOURCE des Laufwerks B: das archive-Attribut.

A>GFU *.COM /R+
Setzt allen COM-Dateien das readonly-Attribut.

A>GFU *.* /H-
Löscht das hidden-Attribut aller Dateien des momentanen Verzeichnisses.

Durch aufeinander folgendes Ausführen mehrerer Attribut-Setzaktionen können einer Datei oder Dateigruppe auch mehrere Attribute zugeordnet werden (z.B. readonly, hidden, system wie bei IBMBIO.COM/IBMDOS.COM). Ob dies Sinn macht, muss der Benutzer selber entscheiden.

Programmaufbau

Der grobe Programmaufbau und -ablauf ist eigentlich sehr einfach: es wird eine Dateispezifikation und eine Aktion eingelesen, entweder via DOS-Kommandozeile als Parameter oder durch Aufruf der Einleseroutinen, die die notwendigen Abfragen am Bildschirm vornehmen, dann wird eine Liste der passenden Dateien angelegt und die gewünschte Aktion auf diese Dateien angewandt. Dabei werden möglichst alle denkbaren und zu erwartenden Fehler abgefangen.

Datenstrukturen

Die Datenstrukturen sind ebenfalls recht einfach, der DOS_Str ist die längste von DOS akzeptierte Zeichenkette, die eine Datei inklusive Laufwerks- und Verzeichnisangaben spezifiziert. RegRec ist im wesentlichen der Registersatz der Intel 8086/8088/80186/80286 Prozessoren und wird für die Aufrufe der DOS-Funktionen via Interrupt 21H benötigt. Frame und die sechs Konstanten UpLeft ..Vertic definieren einen Rahmen; ursprünglich war Frame ein Array von Rahmen, so dass verschiedenartige Rahmen benutzt werden konnten (Einfachlinien, Doppellinien, Mischlinien, sonstige ASCII-Zeichen), da dies hier aber nicht notwendig ist, wurde Frame gestutzt. Die Konstanten Normal..Invers dienen der Bildschirmattributsteuerung mittels der Prozedur SetScr.

Die meisten Variablen sollten ebenfalls mehr oder weniger selbsterklärend sein, CurrFA und NewFA definieren das aktuelle und das neue File-Attribut; i ist ein Hilfsvariable und action definiert die gewünschte Aktion. FileSpec ist die Dateispezifikation, die eingelesen wird, SubDirs wird daraus entnommen und enthält die Laufwerks- und Verzeichnisangaben, die allenfalls in FileSpec enthalten sind. CurrFN und CompleteFN definieren das aktuelle File, wobei CurrFN nur den Dateinamen ohne Laufwerks- und Verzeichnisangaben enthält (wird von den DOS Directory Routinen so geliefert), CompleteFN diese jedoch enthält (wird von den Attribut-Routinen benötigt). found und okay sind zwei boolesche Hilfsvariablen zur Fehlererkennung. Am interessantesten dürfte DTA sein: DTA steht für Disk-Transfer-Area, einen 43-Byte Puffer, der von DOS für seine Directory-Routinen benutzt wird. Der Aufbau der DTA ist wie folgt:

```
PROGRAM General_File_Utility;

(* Utility zum Kontrollieren und Setzen aller DOS-Fileattribute. Gedacht
   als Betriebssystemerweiterung zum Schutze vor ungewolltem Löschen
   wichtiger Programmdateien, zum Anlegen versteckter Unterverzeichnisse
   und anderem mehr

   Autor   : Marc Scheuner, 04.03.1987
   Sprache : Turbo Pascal 3.0
   OpSys   : MS-DOS 2.0 oder höher
*)

CONST MaxL   = 64;                          (* max. Dateiname inkl. Pfad *)
      Frame : ARRAY [1..6] OF Char =        (* Rahmen aus Doppellinien *)
              ('I',';','H','<','M',':');

      UpLeft = 1; UpRight = 2;    (* Position der einzelnen Rahmenzeichen *)
      LoLeft = 3; LoRight = 4;
      Horiz  = 5; Vertic  = 6;

      Normal = 1; Hell = 2; Invers = 3; (* Bildschirmattribute monochrom *)

TYPE DOS_Str = STRING[MaxL];       (* längstmöglicher String für DOS-File *)
     RegRec  = RECORD              (* Registersatz 8088/80286 *)
                 ax,bx,cx,dx,bp,di,si,ds,es,flags : Integer;
               END;

VAR  CurrFA, NewFA, i, action  : BYTE;
     FileSpec, CurrFN, SubDirs,
     CompleteFN                : DOS_Str;
     found, okay               : Boolean;
     DTA                       : RECORD
                                   res  : ARRAY [1..21] OF BYTE;
                                   attr : BYTE;
                                   time,
                                   date : Integer;
                                   Size : ARRAY [1..2] OF Integer;
                                   Name : ARRAY [1..13] OF Char;
                                 END;

{-------------------------------------------------------------------------}
PROCEDURE SetScr (Attrib : BYTE);

(* Prozedur zum Setzen der Bildschirmattribute einer Monochrom-Karte
   (Hercules). Farbgrafikkartenbesitzer können diese Werte ändern (unter
   Verwendung der vordefinierten Turbo-Konstanten) und verschiedenfarbige
   Darstellungsmodi erreichen *)

BEGIN
  CASE attrib OF
    1 : BEGIN     (* Normale Schrift auf schwarzem Grund *)
          TextColor (lightgray);   TextBackground (black)       END;
    2 : BEGIN     (* Helle Schrift auf schwarzem Grund   *)
          TextColor (white);   TextBackground (black)           END;
    3 : BEGIN     (* Schwarze Schrift auf normalem Grund *)
          TextColor (black);   TextBackground (lightgray)       END;
  END;
END;
{-------------------------------------------------------------------------}
PROCEDURE DrawFrame (a1, b1, a2, b2 : BYTE);

(* Diese Prozedur zeichnet einen Rahmen mit den Eckpunkten (a1,b1) oben
   links und (a2,b2) unten rechts.*)

VAR i : BYTE;

BEGIN
  SetScr (Normal);
  GotoXY (a1, b1);  Write (Frame [UpLeft]);
  FOR i := Succ (a1) TO Pred (a2) DO Write (Frame [Horiz]);
  Write (Frame [UpRight]);
  FOR i := Succ (b1) TO Pred (b2) DO BEGIN
    GotoXY (a1, i); Write (Frame [Vertic]);
    GotoXY (a2, i); Write (Frame [Vertic])
  END;
  GotoXY (a1, b2);  Write (Frame [LoLeft]);
  FOR i := Succ (a1) TO Pred (a2) DO Write (Frame [Horiz]);
```

```
  Write (Frame [LoRight])
END;
{------------------------------------------------------------------------}
PROCEDURE EstablishDTA;

(* Richtet eine DTA (Disk Transfer Area) für die nachfolgenden
   Diskettenzugriffe von DOS ein. Näheres siehe Beschreibung. *)

VAR Regs : RegRec;

BEGIN
  FillChar (DTA, SizeOf (DTA), 0);
  Regs.AX := $1A00;
  Regs.DS := Seg (DTA);    Regs.DX := Ofs (DTA);
  MsDos (Regs);
END;
{------------------------------------------------------------------------}
PROCEDURE GetFirstEntry (    Pattern       : DOS_Str;
                         VAR FileNameFound : DOS_Str;
                         VAR FileAttribute : BYTE;
                         VAR Found         : Boolean);

(* Liest ersten Eintrag im Directory, sofern einer vorhanden ist, der
   auf das Suchmuster PATTERN passt (* und ? verwendbar). Wenn einer
   gefunden, ist Found = TRUE und FileNameFound enthält den gefundenen
   Dateinamen (ohne Laufwerks- und Subdirectory-Angaben), sonst ist
   Found = FALSE und FileNameFound leer. *)

VAR  error,i : Integer;        Regs : RegRec;

BEGIN
  FileNameFound := '';   i := Length (Pattern);
  IF (Pattern = '') OR (Pattern[i] IN  ['\',':'])
  THEN Pattern := Pattern + '*.*';
  Pattern[Succ (Length (Pattern))] := #0;
  Regs.AX := $4E00;
  Regs.DS := Seg (Pattern[1]);
  Regs.DX := Ofs (Pattern[1]);
  Regs.CX := $37;
  MsDos (Regs);
  error := Regs.AX AND $FF;
  IF error = 0 THEN BEGIN
    i := 0;
    REPEAT
      i := Succ (i);   FileNameFound[i] := DTA.Name[i]
    UNTIL NOT (FileNameFound[i] IN[' '..'~']);
    FileNameFound[0] := Chr (i-1);
    FileAttribute := DTA.Attr;
    Found := True
  END
  ELSE Found := False
END;
{------------------------------------------------------------------------}
PROCEDURE GetNextEntry (    Pattern       : DOS_Str;
                        VAR FileNameFound : DOS_Str;
                        VAR FileAttribute : BYTE;
                        VAR Found         : Boolean);

(* Liest weiteren Eintrag im Directory, sofern einer vorhanden ist,
   der auf das Suchmuster PATTERN passt (* und ? verwendbar). Wenn
   einer gefunden, ist Found = TRUE und FileNameFound enthält den
   gefundenen Dateinamen (ohne Laufwerks- und Subdirectory-Angaben),
   sonst ist Found = FALSE und FileNameFound leer. *)

VAR  error,i : Integer; Regs : RegRec;

BEGIN
  FileNameFound := '';    i  := Length (Pattern);
  IF (Pattern = '') OR (Pattern[i] IN ['\',':'])
  THEN Pattern := Pattern + '*.*';
  Pattern[Succ (Length(Pattern))] := #0;
  Regs.AX := $4F00;
  Regs.DS := Seg (Pattern[1]);
  Regs.DX := Ofs (Pattern[1]);
  Regs.CX := $37;
  MsDos (Regs);
  error := Regs.AX AND $FF;
```

Byte 1-21	reserviert, nicht zu benutzen
Byte 22	Dateiattribut
Byte 23-24	Zeitpunkt der letzten Dateiänderung
Byte 25-26	Datum der letzten Dateiänderung
Byte 27-28	Grösse der Datei (niederwertiger Teil)
Byte 29-30	Gröse der Datei (höherwertiger Teil)
Byte 31-43	Dateiname, beendet durch Null-Byte (CHR (0))

Für GFU werden die Byte 22 und 31-43 benutzt. Die anderen bleiben in GFU unverwendet, können aber für andere Zwecke durchaus gebraucht werden. Die DTA ist in GFU als Record aufgebaut, der übersichtlicher ist als ein blosser ARRAY [1..43] OF BYTE. Für nähere Erläuterungen, auch der folgenden DOS-Funktionen, empfehle ich das «MS-DOS Handbuch» von Richard A. King, erschienen im Sybex-Verlag.

Das Dateiattributbyte ist folgendermassen aufgebaut:

Bit 6/7	: nicht benutzt
Bit 5	: Archive-Bit
Bit 4	: Directory-Bit (kann nicht gesetzt werden)
Bit 3	: Volume-Bit (Disketten-volumen
Bit 2	: System-Bit
Bit 1	: Hidden-Bit
Bit 0	: Readonly-Bit

Prozeduren und Funktionen

SetScr (Set Screen): Diese Prozedur kann sehr universell verwendet und nach Belieben ausgebaut werden. Der Zweck ist das Setzen von Bildschirmattributen. Die vorliegende Version unterstützt nur drei Darstellungsarten: normal, hell und invers. Auf Monochrom-Bildschirmen wären noch unterstrichen (weiss auf schwarz), invers plus hell (schwarz auf weiss), unsichtbar (schwarz auf schwarz) möglich, zusätzlich kann der Vordergrund blinken (z.B. white plus blink). Dies alles kann mit den vordefinierten Turbo Pascal-Konstanten erreichbar. Auf Farbbildschirmen sind alle Farbkombinationen möglich, die der Monitor zulässt.

DrawFrame zeichnet einen Doppelrahmen, wobei a1,b1 den oberen linken und a2,b2 den unteren rechten Eckpunkt definieren. In dieser Version können nur Doppellinienrahmen mit normaler Intensität gezeichnet wer-

den, DrawFrame lässt sich aber erweitern, so dass weitere Rahmentypen zur Auswahl stehen und man diese auch invers, hell, blinkend usw. darstellen könnte.

EstablishDTA errichtet eine neue DTA, die für die Operationen verwendet wird. Die vorliegende Version von GFU zerstört damit die alte DOS DTA. Sollten Teile von GFU in ein eigenes Programm eingebunden werden, so muss man unter Umständen zuerst die alte DTA saven und nach Beendigung der Operation, welche die GFU über die neu erstelle DTA abwickelt, die alte DTA wieder aktivieren.

GetFirstEntry liefert mittels DOS-Funktion 4E den ersten Eintrag eines Directory. Dabei wird in Pattern ein Muster definiert, das die Wildcards ? und * enthalten darf, das angibt, welchen Bedingungen der Eintrag zu genügen hat (z.B. kann Pattern = *.COM sein). Ist Pattern leer oder ist das letzte Zeichen ein \, so wird *.* an Pattern angehängt, d.h. nach allen Files gesucht. Anschliessend wird Pattern noch ein Nullbyte angehängt. Hierauf wird die DOS-Funktion 4E aufgerufen. Trat ein Fehler auf, so wäre Regs.AX bei der Rückkehr von $00 verschieden. In diesem Fall wird Found auf False gesetzt. Trat kein Fehler auf, so befinden sich die Informationen zum gefundenen File in der DTA, aus der hierauf der Filename und das Dateiattribut entnommen wird. Diese Angaben werden in FileNameFound und FileAttribute zurückgeliefert.

GetNextEntry liefert mit den gleichen Parametern wie GetFirstEntry und demselben Vorgehen mit Hilfe der DOS-Funktion 4F den nächsten Directory-Eintrag, der dem Muster in Pattern entspricht.

ExtractSubdirs ist eine Hilfsprozedur, die einem String, der DOS-Files spezifiziert, die Angaben über Laufwerk und Verzeichnis entnimmt.

Translate verwandelt den übergebenen String in eine Aktionsnummer. Dabei findet eine Kontrolle statt, ob der String überhaupt eine gültige Aktion definiert oder nicht. Wenn ja, wird eine Zahl zwischen 0 und 14 zurückgegeben, allenfalls auch 255 (für Abbruch), sonst wird 99 gemeldet.

Manipulate manipuliert das in CurrAttr übergebene Attribut gemäss der in action übergebenen Aktion und liefert dieses zurück. Manipulate

```
  IF error = 0 THEN BEGIN
    i := 0;
    REPEAT
      i := Succ (i);      FileNameFound[i] := DTA.Name[i]
    UNTIL NOT (FileNameFound[i] IN[' '..'^']);
    FileNameFound[0] := Chr (i-1);
    FileAttribute := DTA.Attr;
    Found := True
  END
  ELSE Found := False
END;
{-------------------------------------------------------------------}
FUNCTION ExtractSubdirs (origstr : DOS_Str) : DOS_Str;

(* Extrahiert einem DOS-String die Laufwerks- und Unterverzeichnis-
   Angaben. *)

VAR  k,m : BYTE;

BEGIN
  m := 0;
  k := Pos('\',origstr);
  WHILE k <> 0 DO
  BEGIN
    m := m + k;
    k := Pos('\',Copy (origstr,m+1,MaxL))
  END;
  IF m = 0 THEN m := Pos (':',origstr);
  ExtractSubDirs := Copy (Origstr,1,m)
END;
{-------------------------------------------------------------------}
FUNCTION Translate (InStr : DOS_Str) : BYTE;

(* Uebersetzer für den zweiten Parameter von GFU in eine Aktionsnummer,
   gleichzeitig Test auf Gültigkeit des Parameters. *)

VAR  act : BYTE;

BEGIN
  IF ((InStr[1] = '/') AND (Length (InStr) > 1)) THEN BEGIN
    Delete (InStr, 1, 1);
    CASE Upcase (InStr[1]) OF
      'N' : Act := 0;
      'R' : CASE Upcase (InStr[2]) OF
              '+' : Act := 1;
              '-' : Act := 2;
              ELSE Act := 99
            END;
      'H' : CASE Upcase (InStr[2]) OF
              '+' : Act := 3;
              '-' : Act := 4;
              ELSE Act := 99
            END;
      'S' : CASE Upcase (InStr[2]) OF
              '+' : Act := 5;
              '-' : Act := 6;
              ELSE Act := 99
            END;
      'A' : CASE Upcase (InStr[2]) OF
              '+' : Act := 7;
              '-' : Act := 8;
              ELSE Act := 99
            END;
      'C' : Act := 9;
      'L' : CASE Upcase (InStr[2]) OF
              'N' : Act := 10;
              'R' : Act := 11;
              'H' : Act := 12;
              'S' : Act := 13;
              'A' : Act := 14;
              ELSE Act := 99
            END;
      'X' : act := 255;
    END
  END
  ELSE Act := 99;
  Translate := Act
```

```
END;
{-------------------------------------------------------------------}
FUNCTION ManipAttr (CurrAttr, Action : BYTE) : BYTE;

(* Manipuliert das übergebene Attribut CurrAttr gemäss Action *)

VAR NewAttr : BYTE;

BEGIN
  CurrAttr := CurrAttr AND $2F;
  CASE action OF
    0 : NewAttr := CurrAttr AND $30;     (* Normalisieren *)
    1 : NewAttr := CurrAttr  OR $01;     (* Readonly setzen *)
    2 : NewAttr := CurrAttr AND $3E;     (* Readonly löschen *)
    3 : NewAttr := CurrAttr  OR $02;     (* Hidden setzen *)
    4 : NewAttr := CurrAttr AND $3D;     (* Hidden löschen *)
    5 : NewAttr := CurrAttr  OR $04;     (* System setzen *)
    6 : NewAttr := CurrAttr AND $3B;     (* System löschen *)
    7 : NewAttr := CurrAttr  OR $20;     (* Archive setzen *)
    8 : NewAttr := CurrAttr AND $0F      (* Archive löschen *)
  END;
  ManipAttr := NewAttr
END;
{-------------------------------------------------------------------}
PROCEDURE AskFileSpec (VAR FileSpec : DOS_Str);

(* Abfrage der Datei-Spezifikation (mit * und ?) *)

BEGIN
  DrawFrame (1,2,80,6);
  GotoXY (5,4);  Write ('File Spez. : ');  Read (FileSpec);
  IF FileSpec = '' THEN BEGIN
    Write ('*.*');  FileSpec := '*.*'
  END
END;
{-------------------------------------------------------------------}
PROCEDURE AskAction (VAR Act : BYTE);

(* Abfrage der gewünschten Aktion *)

VAR AuxStr : DOS_Str;   okay : Boolean;

BEGIN
  DrawFrame (1,7,80,23);
  GotoXY (4,10);  Write (
  '/N  = Normalisieren     /A+ = Archive setzen /LN = List normal');
  GotoXY (4,11);  Write (
  '/R+ = Readonly setzen  /A- = Archive löschen /LR = List Readonly');
  GotoXY (4,12);  Write (
  '/R- = Readonly löschen                       /LH = List hidden');
  GotoXY (4,13);  Write (
  '/H+ = Hidden setzen                          /LS = List system');
  GotoXY (4,14);  Write (
  '/H- = Hidden löschen                         /LA = List archive');
  GotoXY (4,15);  Write (
  '/S+ = System setzen');
  GotoXY (4,16);  Write (
  '/S- = System löschen    /C  = Dateiattribute kontrollieren');
  GotoXY (4,17);  Write (
  '                        /X  = GFU verlassen');
  GotoXY (4,20); Write ('Gewünschte Aktion : ');
  REPEAT
    GotoXY (24,20);   Read (AuxStr);   Act := Translate (AuxStr);
    IF Act = 99 THEN BEGIN
      GotoXY (24,20);  Write (' ':50);  Write (^G)
    END
  UNTIL (act < 99) OR (act = 255);
  ClrScr
END;
{-------------------------------------------------------------------}
FUNCTION PrettyFormat (fn : DOS_Str) : DOS_Str;

(* Formatieren eines Strings, so dass Filename aus acht Zeichen vor dem
   Punkt, einem Punkt und drei Zeichen nach dem Punkt besteht, wobei
   Leerzeichen für fehlende Zeichen gebraucht werden  *)

CONST spc = '                ';
```

macht ausgiebig Gebrauch von der Turbo Pascal-Fähigkeit, die logischen Operatoren AND sowie OR auch auf numerische Datentypen wie BYTE anzuwenden, um einzelne Bits eines solchen Einzel- oder Doppelbytes zu manipulieren.

AskFileSpec öffnet ein Pseudo-Fenster und fragt nach der gewünschten Filespezifikation. Return wird als *.* interpretiert.

AskAction öffnet ebenfalls ein Pseudo-Fenster und zeigt alle verfügbaren Aktionen an und erwartet die Eingabe einer gültigen Aktion. Die Eingabe wird sofort übersetzt (mit Translate), und wenn eine gültige Aktion angegeben wurde, wird AskAction beendet, sonst ertönt ein Warnton und die Eingabe wird erneut verlangt.

PrettyFormat wandelt den übergebenen String in ein einheitliches Format um, und zwar acht Zeichen vor dem Punkt, wobei eventuelle Leerzeichen fehlende Zeichen ersetzen, einem Punkt und danach drei Zeichen der Erweiterung (ebenfalls mit Leerzeichen, falls nötig).

DisplayCurr zeigt das aktuelle File (in CurrFN übergeben) an und verwandelt das in CurrFA übergebene Attribut in einen String um, der ebenfalls angezeigt wird. Count dient dazu, dass jeweils 20 Files auf einmal angezeigt und danach eine Pause eingelegt wird.

SetNewFA verwendet ebenfalls die DOS-Funktion 43H, um für das angegebene File das in NewAttr übergebene Attribut zu setzen.

DisplayExplanations ist eine Art Hilfe, die eine kurze Erklärung von GFU auf dem Bildschirm anzeigt. Sie erscheint, wenn A>GFU ? eingegeben wird. In der vorliegenden Version ist diese Prozedur leer, jeder Benutzer kann selber einen individuellen Text eingeben.

Hauptprogramm

Zunächst eine sehr erfreuliche Nachricht für alle Turbo Pascal-Programmierer: man kann sehr einfach während eines Programmes die Funktion von Ctrl-C kontrollieren, indem die boolesche Variable **CBreak** entsprechend gesetzt wird. FALSE bedeutet, dass Ctrl-C keine Wirkung haben soll, TRUE ermöglicht wieder, mit Ctrl-C abzubrechen. Damit kann

man, im Gegensatz zur Compiler-Direktive {$C } den Abbruch mittels Ctrl-C selektiv erlauben und unterbinden.

Die erste CASE-Anweisung dient zum Einlesen der notwendigen Angaben. Je nach Anzahl und Art der Parameter wird die noch benötigte Information am Bildschirm verlangt.

Wenn anschliessend action den Wert 255 hat, wird abgebrochen. Sonst wird kontrolliert, ob action = 99 ist (ungültige Aktion). Wenn nein, so wird entsprechend action weitergefahren.

Ist action zwischen 0 und 9 (setzen oder löschen eines Attributs), so werden nach und nach alle Files mit den Directory-Prozeduren bestimmt, die FileSpec genügen. Dabei müssen die Pseudo-Files . und .. , die in Unterverzeichnissen auftreten, ausgefiltert werden, denn wenn man versucht, einem solchen File ein Attribut «anzuhängen», stürzt das Programm ab. Aus den Angaben der Directory-Prozeduren und den extrahierten Laufwerks- und Verzeichnisangaben entsteht CompleteFN. Soll nur kontrolliert werden (action = 9), so werden Filename und Attribute angezeigt, sonst wird das gelesene Attribut entsprechend der vorgegebenen Aktion manipuliert und dann gesetzt.

Programm auf Diskette

Allen Lesern, denen das Abtippen des Programmlistings zu mühsam ist, stellt der Autor gegen einen Unkostenbeitrag von sFr. 20.-- das Programm in voller Source inklusiv Dokumentation zur Verfügung. **Achtung!** Das Programm ist nur für IBM-Kompatible erhältlich, auf Anfrage auch auf 3.5-Zoll-Disketten. Benützen Sie bitte für Ihre Bestellung eine Leserdienst-Kontaktkarte (am Schluss des Heftes).

Liegt action zwischen 10 und 14 (Listing-Aktion), so werden nach und nach alle Files, die FileSpec genügen, daraufhin kontrolliert, ob sie das entsprechende Attribut gesetzt haben oder nicht. Dabei werden normale Files hell, Directories normal dargestellt. Die Ausgabe erfolgt im Format des DOS-Befehls DIR /W, also fünf Files pro Zeile. Ein Zähler läuft mit, um am Schluss die Anzahl der gefundenen Files angeben zu können.

```
VAR   p : BYTE;     TmpStr : DOS_Str;

BEGIN
  p := Pos ('.',fn);
  IF (p = 0) OR (p = 1)
  THEN TmpStr := Copy (fn + spc,1,16)
  ELSE TmpStr := Copy (Copy (fn,1,Pos ('.',fn)-1) + spc,1,8) + '.' +
                 Copy (fn + spc, Pos ('.',fn) + 1,3) + Copy (spc,1,4);
  PrettyFormat := TmpStr
END;
{-------------------------------------------------------------------}
PROCEDURE DisplayCurr (    CurrFN : DOS_Str;
                           CurrFA : BYTE;
                       VAR count  : BYTE);

(* Anzeige des Filenamens und des/der zugehörigen Attributs/bute.
   Attribute werden in einem String von 6 Zeichen Länge dargestellt,
   der die folgenden Zeichen enthalten kann:
   N = Normales File
   R = Readonly gesetzt
   H = Hidden gesetzt
   S = System gesetzt
   D = Directory
   A = Archive gesetzt
   Ausgabe wird in heller Schrift für Files und schwächerer Schrift
   für Directories dargestellt.*)

VAR  AttrString : STRING[6];

BEGIN
  NormVideo;
  AttrString := '      ';
  IF (CurrFA AND $0F = $00) THEN AttrString[1] := 'N';
  IF (CurrFA AND $01 = $01) THEN AttrString[2] := 'R';
  IF (CurrFA AND $02 = $02) THEN AttrString[3] := 'H';
  IF (CurrFA AND $04 = $04) THEN AttrString[4] := 'S';
  IF (CurrFA AND $10 = $10) THEN AttrString[5] := 'D';
  IF (CurrFA AND $20 = $20) THEN AttrString[6] := 'A';
  IF (CurrFA AND $10 = $10) THEN LowVideo;
  Writeln (PrettyFormat (CurrFN), '':10, AttrString);
  NormVideo;
  count := Succ (count);
  IF (count MOD 20) = 0 THEN BEGIN
    Writeln; Writeln ('Weiter mit RETURN ... ');  Readln;
    Writeln; Writeln
  END
END;
{-------------------------------------------------------------------}
PROCEDURE SetNewFA (FileName : DOS_Str;  NewAttr : BYTE);

(* Setzen eines neuen Attributs, ebenfalls mit DOS-Funktion 43H. *)

VAR  Regs : Regrec;

BEGIN
  FileName[Succ (Length (FileName))] := #0;
  NewAttr := NewAttr AND $2F;
  Regs.DS := Seg (FileName[1]);
  Regs.DX := Ofs (FileName[1]);
  Regs.AX := $4301;
  Regs.CX := NewAttr;
  MsDos (Regs)
END;
{-------------------------------------------------------------------}
PROCEDURE DisplayExplanations;

(* Routine für Erklärungstexte *)

BEGIN
END;
{-------------------------------------------------------------------}
BEGIN

  (* Ctrl-C wird wirkungslos. So einfach geht das! Diese vordefinierte
     Turbo-Variable wird leider nirgends in den Handbücher erwähnt,
     obwohl sie (wenigstens in Turbo 3.0) problemlos funktionert. *)
  CBreak := False;
```

```
  i := ParamCount;
  CASE i OF
    0 : BEGIN
          ClrScr;
          AskFileSpec (FileSpec);
          AskAction (Action)
        END;
    1 : IF ParamStr (1) = '?' THEN DisplayExplanations
        ELSE BEGIN
          action := Translate (ParamStr (1));
          IF action = 99 THEN BEGIN
            FileSpec := ParamStr (1);
            ClrScr;
            AskAction (Action)
          END
          ELSE FileSpec := '*.*'
        END;
    2 : BEGIN
          FileSpec := ParamStr (1);
          Action := Translate (ParamStr (2))
        END;
  END;

  IF Action = 255 THEN Halt;
  IF Action <> 99 THEN BEGIN
    EstablishDTA;
    SubDirs := ExtractSubDirs (FileSpec);
    CASE Action OF
      0..9 : BEGIN
               i := 0;
               GetFirstEntry (FileSpec, CurrFN, CurrFA, found);
               IF found THEN
                 WHILE found DO BEGIN
                   IF (CurrFN <> '.') AND (CurrFN <> '..') THEN BEGIN
                     CompleteFN := SubDirs + CurrFN;
                     IF Action = 9 THEN DisplayCurr (CurrFN, CurrFA, i)
                     ELSE BEGIN
                       NewFA := ManipAttr (CurrFA, Action);
                       SetNewFA (CompleteFN, NewFA)
                     END;

                   END;
                   GetNextEntry (FileSpec, CurrFN, CurrFA, found)
                 END
               ELSE Writeln ('Keine solche Files gefunden')
             END;
      10..14 : BEGIN
                 i := 0;
                 GetFirstEntry (FileSpec, CurrFN, CurrFA, found);
                 WHILE found DO BEGIN
                   CASE Action OF
                     10 : okay := (CurrFA AND $0F = $00);
                     11 : okay := (CurrFA AND $01 = $01);
                     12 : okay := (CurrFA AND $02 = $02);
                     13 : okay := (CurrFA AND $04 = $04);
                     14 : okay := (CurrFA AND $20 = $20);
                   END;
                   IF okay THEN BEGIN
                     i := Succ (i);
                     IF (CurrFA AND $10 = $10) THEN LowVideo
                                               ELSE NormVideo;
                     Write (PrettyFormat (CurrFN))
                   END;
                   GetNextEntry (FileSpec, CurrFN, CurrFA, found)
                 END;
                 Writeln; Writeln;
                 CASE i OF
                   0 : Writeln ('Keine Files gefunden');
                   1 : Writeln ('1 File gefunden');
                   ELSE Writeln (i:1, ' Files gefunden')
                 END
               END;
    END
  END
  ELSE Writeln (^G,'Ungültige Parameterangabe !!');
  NormVideo
END.
```

Schlussbemerkungen

Die Prozeduren sind zum Teil sehr problembezogen (Translate, DisplayCurr), viele aber können für andere Zwecke genutzt werden. Dies sind insbesondere die Directory-Prozeduren. Mit diesen Prozeduren hat man ein leistungsfähiges Hilfsmittel in der Hand, das es erlaubt, von Turbo Pascal aus direkt auf die DOS-Directories zuzugreifen. Dies kann in vielen Fällen wünschenswert sein, wenn man dem Benutzer auf Knopfdruck ein Inhaltsverzeichnis eines Laufwerks geben will. Man könnte GFU noch weiter ausbauen, z.B. für Dateien und Dateigruppen Datums- und Zeitangaben zu manipulieren oder sortierte Inhaltsverzeichnisse nach verschiedenen Kriterien erstellen (durch Erstellen einer sortierten linearen Liste).

Mit etwas Phantasie und Geduld liesse sich sogar ein Programm à la Norton Utilities oder Xtree erstellen. Und man könnte GFU eventuell auch speicherresident machen, wenn man genügend Speicher zur Verfügung hat. Aber auch in der vorliegenden Form kann GFU durchaus hilfreich sein, denn mit der Möglichkeit, die wichtigsten Files zu blockieren, kann die Gefahr einer irrtümlichen Löschung erheblich vermindert werden. □

COMPUTER SPLITTER

Hercules Grafikkarte für PS/2

(428/eh) Von Hercules Tech. wird im Herbst dieses Jahres eine Grafikkarte auf den Markt gebracht werden, die für die Geräte der PS/2-Serie eingesetzt werden kann. □

IBM's Superstart?

(426/eh) An einer Tagung Ende Juni in New York wurden vom IBM-Direktor für den Bereich Entry Systems, W. Lowe, auch einige Zahlen zum neu eingeführten Personal System /2 bekanntgegeben. Demnach hat IBM in den ersten zwei Monaten nach Ankündigung bereits 250'000 Geräte der Serie /2 ausgeliefert und verfügt über einen Bestellüberhang von weiteren 500'000 Systemen. Gemäss derselben Informationsquelle werden zur Zeit täglich 3'800 Systeme produziert, nämlich 2'000 Modell 30, 1'000 Modell 50 und 800 Modell 60. □

Dataland GmbH
Oberdorfstrasse 143
CH-9100 Herisau Tel. 071 52 21 20

Top-Angebot 1

Vicki der Personalcomputer

- 640 KByte Hauptspeicher
- Turbo 8088 CPU, 4,77/8 MHz
- voll IBM-kompatibel
- 12" hochauflösender Grafikmonitor
- Herkules-kompatible Grafikkarte
- 2 x 360 KByte-Diskettenlaufwerke
- DIN-Tastatur
- MS-DOS 3.2 und MS-Basic
- serielle und parallele Schnittstelle

Fr. 1'990.–

VICTOR

Top-Angebot 2

Vicki der Personalcomputer

- 640 KByte Hauptspeicher
- Turbo 8088 CPU, 4,77/8MHz
- voll IBM-kompatibel
- 12" hochauflösender Grafikmonitor
- Herkules-kopmatible Grafikkarte
- 1 x 360 KByte-Diskettenlaufwerk
- 1 x 20 MByte-Festplatte
- DIN-Tastatur
- MS-DOS 3.2 und MS-Basic
- serielle und parallele Schnittstelle

Fr. 2'790.–

Top-Angebot 3

VICKI-Markendrucker
NEC P6

- 24-Nadel Druckwerk
- 216 Z./sec. EDV
- 60 Z./sec. NLQ
- Grafik 360 x 360 dpi

Profi-Drucker mit Pin Feed Tractor
Fr. 1'590.–

Komplett-Paketpreis VICKI
Doppelfloppy-Version (wie oben) **Fr. 3'390.–**
Festplatten-Version (wie oben) **Fr. 4'090.–**

EPSON LX 800

- 9-Nadel Druckwerk
- 180 Z./sec. EDV
- 28 Z./sec. NLQ
- Grafik-Modus

LOW COST-Drucker mit Einzelblattführung und Tractor
Fr. 750.–

Komplett-Paketpreis VICKI
Doppelfloppy-Version (wie oben) **Fr. 2'450.–**
Festplatten-Version (wie oben) **Fr. 3'190.–**

Optionen

- 14" an Stelle von 12"-Monitor **Fr. 150.–**
- EGA-Karte und EGA-Bildschirm 14" **Fr. 990.–**
- VICTOR-Mouse **Fr. 199.–**
- MICROSOFT-Mouse mit Windows, dt. **Fr. 500.–**
- zusätzl. 3,5 Zoll Laufwerk zur Festplatten-Version **Fr. 500.–**

● 9100 Herisau-Heinrichsbad 071 52 21 20 ● 9470 Buchs 085 6 18 56 ● 9430 St. Margrethen 071 71 15 53
● 8590 Romanshorn 071 63 40 03 ● 8200 Schaffhausen 053 4 33 19 ● 8623 Wetzikon-Kempten 01 930 79 49
● 8750 Glarus 058 61 11 89 ● 8640 Rapperswil 055 27 27 00 ● 6430 Schwyz 043 21 46 56

dataland
INFORMATIK-ZENTRUM
071 52 21 20

SEIKOSHA
NEC
MICROSOFT
EPSON
star
3M
Commodore
VICTOR

Klar und deutlich MITSUBISHI

KEY COMPUTER AG

☎055-48 22 83

FREE-SCAN
EUM 1471-A

MITSUBISHI
Monitore
erhalten Sie
bei Ihrem Fachhändler

FREE-SCAN
EUM 1471-A

- 14"-Monitor inkl. Dreh-/Schwenkfuss
- Anschliessbar an alle Personal-Computer und Videos
- **Auflösungen:** CGA, MDA, EGA, PGA, TGA und CAD/CAM (800 x 560)
- **Eingänge:** Comp. Video, RGB TLL, RGB analog, Monochrom TTL und «FAST BLANKING» zum Mischen von Digital- und Videosignalen
- **Für höchste Ansprüche und individuellste Auflösungen:** FREE-SCAN mit GENOA SUPER EGA HiRes-Grafikkarte; noch günstiger im **Kombipaket.**

Generalvertretung: Schnellmann Interhandels AG, 8808 Pfäffikon/SZ

Ihr Fachhändler:
Key-Computer AG
Industriestrasse 2
8808 Pfäffikon
Telefon 055-48 22 83

SNAP **Faszinierend, was dieses menügesteuerte Gesamt-Systemverwaltungsprogramm alles kann!**

Mit seinen über 200 Systemfunktionen verwaltet SNAP Software-Programme, Festplatten und Hardware-Konfigurationen.

Unentbehrlich für den **Anfänger,** weil er das mühsame Lernen und Manipulieren mit MS-DOS-Befehlen vergessen kann.

Geschätzt und beliebt bei den **Experten,** weil SNAP enorm schnell und leistungsstark ist.

In der Zeit, die Sie brauchen, um diesen Textabschnitt zu lesen, hätte SNAP:

105 Dateien übertragen
246 Dateien in
41 Verzeichnissen löschen
107 Dateien kopieren
und eine 360KB-Diskette
formatieren können . . .

in nur 25 Tastenanschlägen.

SNAP **der unübertroffene PC-MANAGER**

die Software, die in keinem PC fehlen darf!

deutsch, französisch, italienisch, spanisch, englisch, niederländisch

Für nur Fr. 190.– können Sie schon morgen SNAP für Sie arbeiten lassen!

SNAP benötigt einen IBM PC/XT/AT oder Komatiblen MS/PC-DOS 2.11 oder höher

Marctech S.A., CH-2000 Neuchâtel, Tel. 038 42 48 43

Coupon
Senden Sie mir bitte ausführliche Informationen über SNAP ☐

Sprachversion _____

Ich bestelle _____ SNAP

Name _____

Firma _____

Strasse _____

Plz, Ort _____

Bitte einsenden an:
Marctech S.A., Verkauf
Chemin des Noisetiers
1261 Givrins/VD

CLIPPER

Technische Daten des Clipper 2:

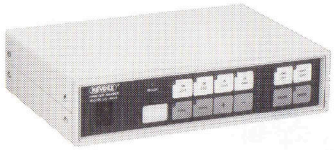
CPU
80286-10 Mikroprozessor
80287 Co-Prozessor opt.
BIOS-ROM
27128/27256 EPROM CHIP
RAM
640 KB standard, erweiterbar auf 1 MB auf Motherboard
Takt-Frequenz
6/10 MHz, schaltbar jederzeit durch Hard-/Software
Wartezyklus
0/1 Wait State schaltbar
Slots
6 PC-AT Interface
2 PC-XT Interface
Disk-LW
1.2 MB/360 KB 5.25 zoll
720 KB 3.5 zoll
360 KB 5.25 zoll
Harddisk
20–800 MB Seagate/Maxtor
I/O-Karte
Centronics, 2 Seriell (2.opt.)
Netzteil
200 W / low noise
Abmessung
43.6×16.2×42.2 cm
MHz-Test
0 WS = 13,5 MHz!

Clipper 2 inkl. Seagate ST225 20 MB HD Fr. 4600.–
inkl. Seagate ST4096 80 MB HD Fr. 5750.–

NS Soft

NIEDERHAUSER & CIE. AG
Hard- & Software
Gaswerkstrasse 33
4900 **Langenthal**
Telefon 063/22 27 88

ELEKTRONISCHE DRUCKER-UMSCHALTER



- mehrere PC's und 1 oder mehrere Drucker/Plotter
- mit oder ohne Speicher
- automatische Drucker-Belegung oder manuelle Zuordnung
- Kopierfunktion
- auch manuelle Umschalter bis 12-Weg

Wiederverkäufer gesucht!

TELTEC · 3250 Lyss
Knospenweg 4 · Tel. 032 / 84 42 40 · Telex 934446

Alles für Ihren PC

Witty Mouse + Testsoftware	XT/AT	Fr. 159.–
Gameadapter für zwei Joysticks	XT/AT	Fr. 59.–
RS-232 Karte (serielle Karte)	XT	Fr. 85.–
Parallel-Printerport	XT	Fr. 59.–
Uhrenkarte batteriegepuffert	XT	Fr. 85.–
PC Joy-Stick	XT/AT	Fr. 49.–
3.5 MB RAM-Karte ohne RAM	/AT	Fr. 399.–
2 MB RAM-Karte ohne RAM	XT	Fr. 359.–
640 KB RAM-Karte ohne RAM	XT	Fr. 129.–
Druckerkabel par/cent 3 m	XT/AT	Fr. 35.–
14 Zoll EGA Monitor und EGA Karte	XT/AT	Fr. 1'490.–

Es hat vieles mehr: Streamer, Harddisk, Floppies, RAM Monitoren, diverse I/O-Karten, Prozessoren, Disketten. Lieferung ab Lager gegen Nachnahme. Eigener Reparatur-Service. 12 Monate Garantie.

Tel. 057/444 740 ... Es hat, solange es hat...

Aus einem Gegengeschäft verkaufen wir sehr günstig eine grössere Menge:

LCD Bildschirme für Overheadprojektion

Der DAVIS Transview ist ein norwegisches Qualitätsprodukt höchsten Ranges!
Er zeichnet sich u. a. durch folgende Besonderheiten aus:

- Einzigartige eingebaute Kühlung für den Bildschirm!
- Brillante Auflösung von 640x200 Punkten
- hoher Kontrast, absolut flimmerfrei
- Erhältlich mit Toshiba oder IBM - Interface
- Kinderleichter Einbau
- Einfach zu transportieren, weniger als 3 kg schwer!
- sehr flach, solide Verarbeitung, höchste Zuverlässigkeit

DAVIS Transview kann man einsetzen, z.B. für:

Schulungen aller Art, Präsentationen, oder als zweiten Bildschirm ....

Der Transview ist kein Billigprodukt! Der Verkaufspreis ist Fr. 4750.-

Wir verkaufen den Transview zum **Spezialpreis von Fr. 2550.-**
(anschlussfertig, inkl. WUST + inkl. Toshiba oder IBM - Interface und 6 Monate Garantie.)

Greifen Sie zu und nutzen Sie diese einmalige Gelegenheit!

->->->->->->->->->->->->->->->- ausschneiden und einsenden an: -<-<-<-<-<-<-<-<-<-<-<-

ATARI (Schweiz) AG, Bahnhofstr. 7, 5400 Baden 1. Tel: 056/211'422, Fax: 056/211'081

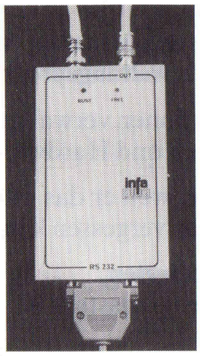
BESTELLUNG:

Bitte senden Sie uns: Anzahl : ____ DAVIS Transview mit ____ Toshiba IF ____ IBM IF

Rechnung und Lieferung geht an:

Ort, Datum und Unterschrift:

Tel: _____ / ___________



Keine Kanonen gegen Spatzen:

INFAPLUG
Local Area Network

- *fuer Office Automation und Industrie*
- *einfachste Installation und Anwendung*
- *fuer alle Mini–, Micro– und PC–Systeme*
- *flexibel im Ausbau*
- *preisguenstig*

MAP INFORMATIK 5732 ZETZWIL
Hauptstrasse 115 064/73 22 15

Bereits ab SFr. 180.– pro Monat können Sie Ihren

EPSON-, NCR- oder HP-COMPUTER

(auch Drucker) bei uns mieten.

Miete – Leasing – Kauf – Beratung – Realisierung durch:

DIMOTRON AG
Informationssysteme
Buggenacher 26
CH-6043 Adligenswil
Tel. 041 / 31 69 71

Gleichungssysteme in Logo

Benützt man die Gleichungen zweier Geraden in der Ebene, so kann man als Logo-Prozeduren leicht elementare Matrizenoperationen ableiten. Wendet man diese Erfahrung bei linearen Differentialgleichungen an, ergeben sich daraus die Definitionen ihrer singulären Punkte. Die Eigenschaften eines Paraboloids bestimmen dann die Stabilität dieser Punkte. An Beispielen (je einem aus Oekologie, Elektronik und Chemie) gelangt man schliesslich auch zu nichtlinearen Differentialgleichungssystemen. Mit anderen Worten, mittels rechnerischer und grafischer Logo-Hilfe erreicht man einen intuitiven Einblick in die mathematische Theorie der dynamischen Systeme. Für diese Exkursion benötigt man, ausser der Gleichung einer Geraden, keine anderen mathematischen Kenntnisse. Die Prüfung der Einsatzmöglichkeiten von Logo auf diesem speziellen Gebiet hat sich ebenfalls als aufschlussreich erwiesen.

Dr. Branco Milicevic

Die Theorie der dynamischen Systeme kann schon auf Leonhard Euler (1774) zurückgeführt werden; ihre Bedeutung für den Fortschritt der Natur- und Ingenieurwissenschaften erkannte man aber erst in unserem Jahrhundert [1].

Die Fundamente dieser Theorie sind von H. Poincaré (1885) und A.M. Liapunov (1892) erstellt worden (Gurel in [1]). Die Anwendungen dieser Theorie in den frühen zwanziger Jahren von B. van der Pol erweckten aber erst das Interesse anderer Mathematiker. So begannen in der UdSSR A.A. Andronov [2] und in den USA S. Lefschetz [3] etwas später mit ihren Mitarbeitern dieses neue Gebiet systematisch zu untersuchen. Aus der Reihe der zahlreichen Mathematiker, die dabei mitwirkten, seien nur die folgenden Namen herausgegriffen: R. Bellman [4], M.W. Hirsch und S. Smale [5]. Die Werke dieser Autoren dienten hier nämlich als Ausgangsbasis.

Wir haben hier die schwierige und umfangreiche Mathematik im Hintergrund belassen und nur die rein rechnerische Seite hervorgehoben. Damit ist die entsprechende Veranschaulichung mit Logo-Hilfe erst möglich geworden.

Matrizen

Betrachten wir zwei Geraden in der Ebene und schreiben wir ihre Gleichungen folgendermassen:

$$c_1 = a_1x_1 + a_2x_2$$
$$c_2 = a_3x_1 + a_4x_2$$

Dabei sind x_1, x_2 Variablen und alle anderen Bezeichnungen stellen Konstanten dar. Man kann das betrachtete Gleichungssystem auch in eine andere Form bringen,

$$\begin{bmatrix} c_1 \\ c_2 \end{bmatrix} = \begin{bmatrix} a_1 & a_2 \\ a_3 & a_4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

und dies nach Vereinbarung auch abkürzen mit

$$\underline{c} = \underline{A}\ \underline{x}$$

Man sagt dabei: Der Vektor $\underline{x}$ multipliziert mit der Matrix $\underline{A}$ ergibt den Vektor $\underline{c}$. Die Koeffizienten $a_1,...$ nennt man Elemente der Matrix $\underline{A}$ und $x_1,...$ bzw. $c_1,...$ stellen die Komponenten der Vektoren $\underline{x}$ bzw. $\underline{c}$ dar.

Da man ja deutlich sehen kann, wie die Vorschrift für die Wandlung eines linearen algebraischen Gleichungssystems in die Matrixform lautet, kann man berechtigterweise fragen: «Warum einfach, wenn es auch komplizierter geht?». Dem ist aber nicht so, da Matrizen gewisse Eigenschaften besitzen, die es erlauben, die Berechnungen wesentlich zu vereinfachen. Bevor wir uns diesem Thema widmen, müssen wir noch abklären, wie wir in Logo mit Matrizen umgehen können.

Bekanntlich verfügt Logo über Listen, aber sind dies auch Matrizen? Nein, von vornherein sind Listen keine Matrizen. Da aber Logo eine Programmiersprache ist, können wir z.B. auf Listen Matrizenoperationen definieren, um dann mit geeigneten Prozeduren gewünschte Berechnungen durchzuführen. Betrachten wir zu diesem Zweck die Matrixform unseres Gleichungssystems. Wir vergeben willkürliche Zahlenwerte und schreiben probeweise $\underline{A}\ \underline{x}$ als

```
MAKE «P [[[1 2][3 4]][5 6]]
```

Diese Art von Schreibweisen für «Vektor multipliziert mit Matrix» erscheint auf den ersten Blick wegen der vielen Klammern ziemlich verwirrend, ist es aber in Logo nicht.

Mit FIRST :P erhalten wir [[1 2][3 4]], also die Matrix, und mit LAST :P folgt [5 6], also der Vektor. LAST FIRST :P ergibt [3 4], demnach die zweite Zeile der Matrix, usw. Man muss bei dieser «Pflichtübung für Anfänger in Lisp» aber beachten, dass man nicht «das Kind mit dem Bad ausschüttet». BUTLAST BUTFIRST :P ergibt [], das heisst eine leere Liste in Logo, jedoch kennt man in der Algebra keine «leere Matrix«. Durch Anwendung weiterer Logo-Primitiva (z.B. ITEM, PIECE) erhält man in Logo eine sehr geeignete Umgebung zur Definition der linearen Algebra.

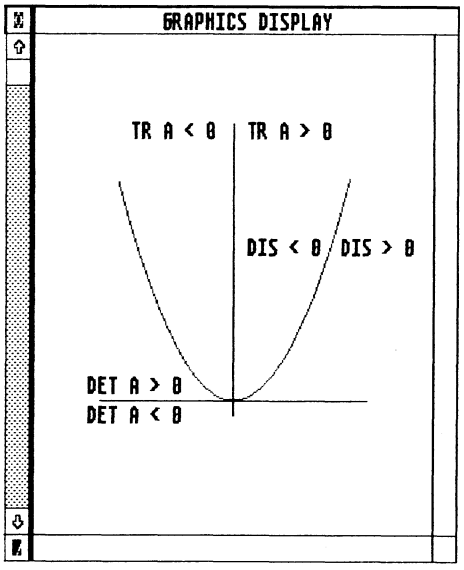


Abb. 1: Singuläre Punkte dynamischer Systeme, charakterisiert durch die Relationen (Prädikate) : DET $\underline{A} = 0$, TR $\underline{A} = 0$, DIS = 0.

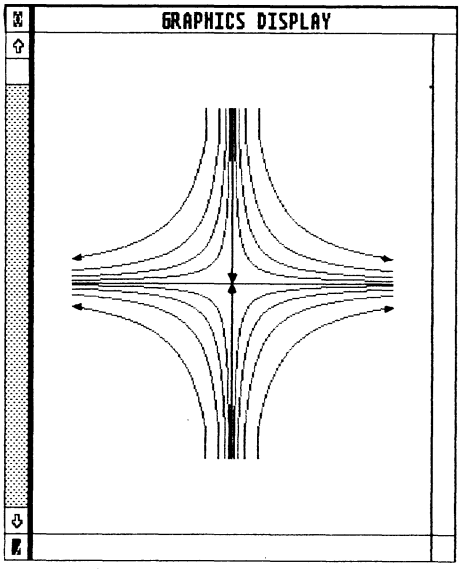


Abb. 2: Sattelpunkt. Beispiel: x' = x, y' = -y. $\underline{A}$ = [[1 0][0 -1]]. Hyperbeln : y = p/x.

```
?PO "VM
TO VM :a1 :a2 :a3 :a4 :x1 :x2
PR "
PR "
(LOCAL "c1 "c2 "DET "V "M)
MAKE "DET :a1 * :a4 - :a2 * :a3
IF = :DET 0 [PR [Matrix is singular !] STOP]
MAKE "c1 :a1 * :x1 + :a2 * :x2
MAKE "c2 :a3 * :x1 + :a4 * :x2
MAKE "V SE :x1 :x2
MAKE "M (LIST (SE :a1 :a2) (SE :a3 :a4))
TYPE "VECTOR : SHOW :V
TYPE "MATRIX : SHOW :M
TYPE "c1 : PR :c1
TYPE "c2 : PR :c2
END

?
?
?
?VM 1 2 3 4 5 6


VECTOR :[5 6]
MATRIX :[[1 2] [3 4]]
c1 :17
c2 :39
?
```

Listing 1

Matrizenrechnung

Jeder quadratischen Matrix kann man eine Zahl zuordnen, die die Determinante der Matrix genannt wird. Für unseren Fall (2x2-Matrix) ist die Determinante wie folgt definiert:

$\text{DET } \underline{A} = a_1a_4 - a_2a_3$

Es ist nun offensichtlich, dass diese Zahl positiv oder negativ, aber auch gleich Null sein kann.

Eine Matrix, deren Determinante gleich Null ist, heisst singulär. Eine singuläre Matrix wird durch eine lineare Abhängigkeit verursacht, das heisst in unserem Fall, die beiden Geraden sind parallel, bzw. im Trivialfall besteht nur eine Gerade. Wir schliessen in unseren weiteren Betrachtungen singuläre Matrizen aus und berücksichtigen dies in allen Prozeduren.

In der Prozedur VM (Listing 1) wird nun die Multiplikation eines Vektors mit einer Matrix gegeben. Die Rechenvorschrift ist uns ja bekannt. Zur Eingabe/Ausgabe eine Bemerkung: Dies ist in Logo nur eine Frage der Textgestaltung. Jedermann/frau kann sich dabei nach seinem eigenen Geschmack richten.

Nun ist vielleicht mancher Leser enttäuscht, da er/sie die Lösung des Gleichungssystems - also die Ermittlung des Schnittpunktes der beiden Geraden - erwartet hat. Um dies zu erreichen müssen wir

$\underline{x} = \underline{A}^{-1}\underline{c}$

berechnen, wobei mit $\underline{A}^{-1}$ die inverse Matrix bezeichnet wird. In unserem Fall ist die inverse Matrix mit

$\underline{A}^{-1} = (1/\text{DET } \underline{A}) * [[:a_4 \text{ -}:a_2][\text{-}:a_3 :a_1]]$

definiert. Die Prozedur IM (Listing 2) errechnet sowohl die inverse Matrix, als auch den Schnittpunkt. Es wird auch klar, dass es für $\text{DET } \underline{A} = 0$ weder eine inverse Matrix noch einen Schnittpunkt geben kann.

In der Prozedur MM (Listing 3) ist die Multiplikation Matrix mal Matrix gegeben, wobei jetzt als Resultat eine Matrix folgt. Wie man sieht, ist die Rechenvorschrift jetzt etwas länger. Interessierte Leser können solche Vorschriften leicht finden und das hier Beschriebene verallgemeinern.

Mit der Prozedur MM können wir uns noch einiges Wissen experimentell bestätigen lassen. Die Multiplikation von Matrizen ist nicht eine komutative Operation, das heisst im allgemeinen Fall gilt

```
?PO "IM
TO IM :a1 :a2 :a3 :a4 :c1 :c2
PR "
PR "
(LOCAL "x1 "x2 "DET "I)
MAKE "DET :a1 * :a4 - :a2 * :a3
IF = :DET 0 [PR [Matrix is singular !]]
MAKE "I (LIST (SE :a4 / :DET -:a2 / :DET) (SE -:a3 / :DET :a1 / :DET)) SHOW :I
PR [INVERSE MATRIX]
MAKE "x1 :c1 * (FIRST ITEM 1 :I) + :c2 * (LAST ITEM 1 :I) PR :x1
MAKE "x2 :c1 * (FIRST ITEM 2 :I) + :c2 * (LAST ITEM 2 :I) PR :x2
PR [Components of x]
END

?
?
?
?IM 22 3 0 100 4.5 .2


[[0.045455 -0.001364] [0 0.01]]
INVERSE MATRIX
0.204273
0.002
Components of x
?
```

Listing 2

$\underline{A}\,\underline{B} <> \underline{B}\,\underline{A}$

Im Fall einer Matrix und ihrer Inversen gilt jedoch

$\underline{A}\,\underline{A}^{-1} = \underline{A}^{-1}\underline{A} = \underline{I}$

wobei $\underline{I}$ die Einheitsmatrix

$\underline{I} = [[1\ 0][0\ 1]]$

darstellt.

Im Hinblick auf unser späteres Vorhaben wollen wir gleich noch einige andere benötigte Eigenschaften von Matrizen definieren.

Die Spur (Trace) von $\underline{A}$ ist durch

$TR\ \underline{A} = a_1 + a_4$

gegeben. Mit

$DIS = (TR\ \underline{A})^2 - 4\ DET\ \underline{A}$

ist die Diskriminante DIS definiert. Nun können wir auch die charakteristische Gleichung unserer Matrix angeben; sie lautet

$e^2 - TR\ \underline{A}\,e + DET\ \underline{A} = 0$

Dies ist eine quadratische Gleichung mit folgender Lösung:

$e_1/e_2 = (TR\ \underline{A} \pm SQRT\ DIS)/2$

wobei e_1 und e_2 die Eigenwerte (englisch: eigenvalues!) der Matrix $\underline{A}$ darstellen. Die Prozedur EV (Listing 4) berechnet diese Werte, die reelle, imaginäre oder komplexe Zahlen sein können.

Gewöhnliche Differentialgleichungen

Die einfachste Differentialgleichung kann wie folgt geschrieben werden:

$x' = a\,x$

Hier ist x = x(t) eine Funktion der Zeit (t), und wir sagen, dass ihre zeitliche Aenderung (x' = dx/dt) sich selbst proportional ist.

Zur Klärung der Begriffe hier das einfachste Beispiel: Wenn wir einen grossen Wasserbehälter mit einem kleinen Loch im Boden betrachten, so wird die Abflussgeschwindigkeit (x') des Wassers (z.B. in kg/h) dem Wasservorrat (x) des Behälters (z.B. in kg) proportional sein.

Die Proportionalitätskonstante (a) dient dabei zwei Zwecken. Erstens wird die Abflussgeschwindigkeit auch vom Lochdurchmesser (z.B. in cm) abhängig sein. Wir berücksichtigen dies mit dem Wert von a. Zweitens haben wir es hier mit ganz verschiedenen Masseinheiten - Gewicht pro Stunde, Gewicht, Länge - zu tun. Diese verschiedenen, sogenannten physikalischen Dimensionen gleicht a mit seiner eigenen Dimension aus, bzw. erlaubt uns, die Differentialgleichung als rein mathematischen Ausdruck zu behandeln.

Die betrachtete Differentialgleichung kann man nach bekannten Regeln lösen (integrieren) und erhält damit eine Schar von Integralkurven. Wollen wir bei fixem a nur eine solcher Kurve erstellen, können wir dies mit der Anfangsbedingung (Wert für t = 0) festlegen, das heisst das Problem folgendermassen formulieren:

$x' = a\,x,\ x(0) = c$

Zwei wichtige Bemerkungen: Man kann einerseits beliebig viele Beschreibungen finden, für die die betrachtete Differentialgleichung als mathematisches Modell dienen kann. Andererseits gibt es aber auch andersartige Differentialgleichungen, für die wir keine Lösungsregeln kennen.

Betrachten wir jetzt ein System von Differentialgleichungen

$x_1' = a_1x_1 + a_2x_2 - c_1$
$x_2' = a_3x_1 + a_4x_2 - c_2$

oder

$\underline{x}' = \underline{A}\ \underline{x} - \underline{c}$

Man bezeichnet dies als ein dynamisches System und wir erkennen sofort eine gewisse Aehnlichkeit mit unserem vorherigen algebraischen System.

Die erste Frage, die uns bei einem solchen System interessiert, ist: Wie können wir es in seiner Ruhelage beschreiben? In der Ruhelage sind definitionsmässig alle Flüsse gleich Null, d.h. einfach $\underline{x} = \underline{0}$ ($\underline{0}$ heisst Nullvektor).

Mit der Prozedur IM können wir nun auch einen existierenden Ruhelagepunkt berechnen, nur wird er jetzt nicht mehr den Schnittpunkt zweier Geraden darstellen. Wir nennen ihn auch anders und sagen: Es ist der singuläre Punkt des dynamischen Systems.

Eine Zwischenbemerkung: Falls wir die beiden Differentialgleichungen des betrachteten dynamischen Systems dividieren, erhalten wir eine neue Differentialgleichung, die keine Zeit mehr enthält (links vom Gleichheitszeichen steht jetzt dx_1/dx_2). Wir sprechen in solchen Fällen von autonomen (zeitinvarianten) dynamischen Systemen. Die Integralkurven dieser Gleichung werden sich im singulären Punkt (und zwar nur in diesem Punkt) schneiden. Es ist üblich, diese Integralkurven Trajektorien (Bahnen) zu nennen und die Ebene, in der sie verlaufen, als die Phasenebene des dynamischen Systems zu bezeichnen.

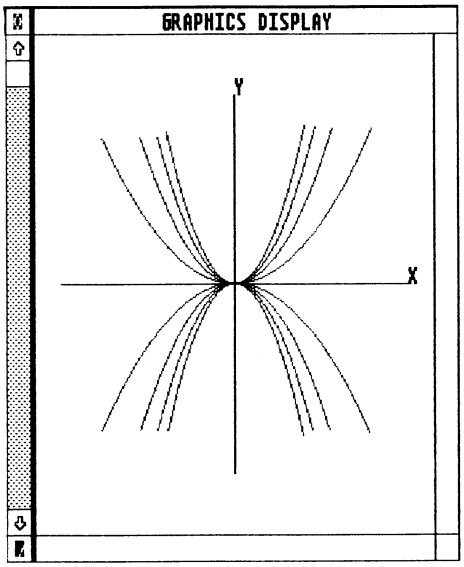


Abb. 3: Knotenpunkt. Beispiel: x' = x, y' = 2y. A = [[1 0][0 2]]. Parabeln : y = px².

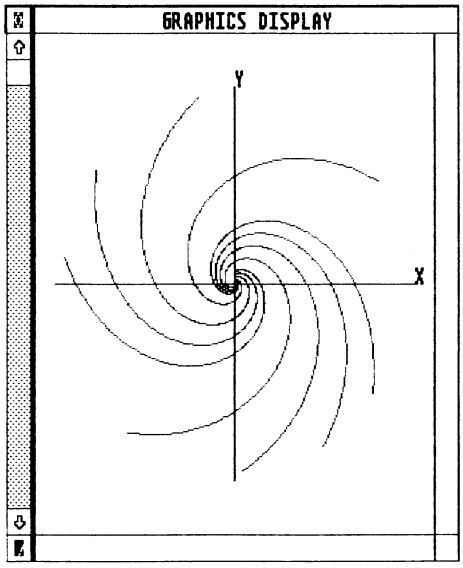


Abb. 4: Brennpunkt. Beispiel: x' = x - y, y' = x + y. A = [[1 -1][1 1]]. Logarithmische Spiralen (Listing 5).

Singuläre Punkte

Mittels der Prozedur EV können wir jetzt verschiedene singuläre Punkte definieren (Abb. 1). Grafisch veranschaulichte Beispiele sind in Abb 2, 3 und 4 gegeben. Die Trajektorien wur-

```
?PO "MM
TO MM :a1 :a2 :a3 :a4 :b1 :b2 :b3 :b4
PR "
PR "
(LOCAL "DETA "DETB "A "B "M "M1 "M2)
MAKE "DETA :a1 * :a4 - :a2 * :a3
MAKE "DETB :b1 * :b4 - :b2 * :b3
IF OR :DETA = 0 :DETB = 0 [PR [Matrix is singular !] STOP]
MAKE "M1 (SE (:a1 * :b1 + :a2 * :b3) (:a1 * :b2 + :a2 * :b4))
MAKE "M2 (SE (:a3 * :b1 + :a4 * :b3) (:a3 * :b2 + :a4 * :b4))
MAKE "A (LIST (SE :a1 :a2) (SE :a3 :a4)) SHOW :A PR [MATRIX A]
MAKE "B (LIST (SE :b1 :b2) (SE :b3 :b4)) SHOW :B PR [MATRIX B]
MAKE "M LIST :M1 :M2 SHOW :M PR [MATRIX C]
END

?
?
?
?MM 12 41 33 50 0 1 2 3


[[12 41] [33 50]]
MATRIX A
[[0 1] [2 3]]
MATRIX B
[[82 135] [100 183]]
MATRIX C
?
```

Listing 3

den mit der jeweils angepassten Prozedur CURVE gezeichnet (Beispiel Listing 5 für Abb 4.). Dabei wurde auch die Notation vereinfacht ($x_1 = x$, $x_2 = y$).

Nun zu den singulären Punkten und ihren Namen:

- Für DET $\underline{A} < 0$ exisitiert nur der Sattel (Abb. 2). Bei allen anderen Singularitäten gilt DET $\underline{A} > 0$.
- Für DIS > 0 existieren Knoten (Abb. 3).
- Für DIS < 0 existieren Brennpunkte (Abb. 4).

Für diese Klassifikation kann man auch Eigenwerte verwenden, was der allgemeinere Weg ist. Uns genügt das schon Erwähnte, allerdings müssen wir es noch mit Betrachtungen über die Stabilität der singulären Punkten ergänzen.

Stabilität

Die Theorie der Stabilität wurde 1892 von Liapunov [2, 3] formuliert. Diese Theorie ist erstaunlich einfach, jedoch benötigen wir hier noch eine kurze Regel, um sie darzustellen. Es geht um das Differenzieren von Funktionen. Betrachten wir die Funktion $y = x^2$, dann lautet ihre zeitliche Ableitung $y' = 2 x x'$. Das ist alles.

Nun zum Liapunov'schen Theorem an einem Beispiel. Wir suchen für ein lineares dynamisches Systeme eine stets positive Funktion L(x,y). Diese Bedingung ist z.B. mit

$$L(x,y) = \tfrac{1}{2} x^2 + \tfrac{1}{2} y^2 > 0$$

erfüllt, da Quadrate immer positiv sind. Die zweite Bedingung, nämlich, dass diese Funktion im singulären Punkt gleich Null ist, gilt ebenfalls, also

$$L(0,0) = 0$$

Wenn jetzt auch die zeitliche Ableitung dieser Funktion kleiner als Null (also negativ) ist,

$$L(x,y)' = x x' + y y' < 0$$

dann ist der singuläre Punkt asymptotisch stabil, d.h. alle Trajektorien streben diesem Punkt wie einer Senke

```
?PO "EV
TO EV :a1 :a2 :a3 :a4
PR "
PR "
(LOCAL "DET "TR "DIS "e1 "e2)
MAKE "DET :a1 * :a4 - :a2 * :a3
MAKE "TR :a1 + :a4
MAKE "DIS :TR * :TR - 4 * :DET
IF > :DIS 0 [(PR "e1 = :TR / 2 + 0.5 * SQRT :DIS)
             (PR "e2 = :TR / 2 - 0.5 * SQRT :DIS)]
IF = :DIS 0 [(PR "e1,e2 = :TR / 2)]
IF < :DIS 0 [(PR "real = :TR / 2)
             (PR [imag = (+ / -)]) (PR 0.5 * SQRT -:DIS)
TYPE "DET = PR :DET
TYPE "TR = PR :TR
TYPE "DIS = PR :DIS
END

?
?
?
?EV 1 -1 1 1


real = 1
imag = (+ / -)
1
DET =2
TR =2
DIS =-4
?
```

Listing 4

zu; eine solche Funktion heisst Liapunov'sche Funktion. Asympthotisch stabile Punkte nennt man auch Attraktoren.

Nehmen wir jetzt den Brennpunkt (Abb. 4). Die ersten beiden Bedingungen bleiben gültig, jedoch die dritte,

$$L(x,y)' = x(x-y) + y(x+y)$$
$$= (x^2 + y^2) > 0$$

nicht, und der Brennpunkt wirkt wie eine Quelle, das heisst alle Trajektorien entfernen sich von ihm. Einen solchen singulären Punkt nennt man Reppelor.

Ausser zwei Ausnahmen sind singuläre Punkte entweder Attraktoren oder Reppeloren. Der Sattelpunkt (Abb. 2) ist instabil, und für TR $\underline{A} = 0$ exisitiert der Mittelpunkt (Punkt umgeben mit Kreisen), der nur stabil ist (es gilt dabei $L(x,y)' = 0$).

Betrachtet man L(x,y) als eine Fläche im Raum (Abb. 5), so erkennt man, dass es ein Paraboloid ist. Die Trajektorien können Kreise sein oder streben als Parablen (bzw. als Spiralen) dem singulären Punkt zu. Dies ist die geometrische Deutung des Stabilitätstheorems von Liapunov.

Liapunov'sche Funktionen können aber verschiedenster Art sein (nur die Bedingungen müssen erfüllt sein). Wir möchten jedoch eine Prozedur LIAPUNOV (Listing 6) schreiben und benötigten dazu eindeutige Regeln. Für unseren Fall gibt es dazu einen eleganten Weg; wir werden ihn nun skizzieren.

Wir beginnen mit unserem dynamischen System und nehmen nur eine seiner Gleichungen (z.B. die zweite), differenzieren sie nochmals, entfernen durch Substitution (aus der ersten Gleichung) x', x, und erhalten

$$y'' - (TR\ \underline{A})y' + (DET\ \underline{A})y = 0$$

als äquivalente Gleichung für das dynamische System. Durch Einführung einer neuen Variable z = y' wechseln wir die Phasenebene, wobei jetzt folgendes gilt:

$$y' = z$$
$$z' = (TR\ \underline{A})z - (DET\ \underline{A})y$$

Für dieses äquivalente dynamische System lässt sich eine allgemeine Liapunov'sche Funktion leicht finden

$$L(x,z) = \tfrac{1}{2}\ (DET\ \underline{A})y^2 + \tfrac{1}{2}\ z^2$$
$$L(0,0) = 0$$
$$L(x,z)' = (TR\ \underline{A})\ z^2$$

Mit anderen Worten hängt die Stabilität der singulären Punkte einzig vom Vorzeichen der Spur der Matrix $\underline{A}$ ab. Für TR $\underline{A} < 0$ haben wir Attraktoren und für TR $\underline{A} > 0$ Reppeloren.

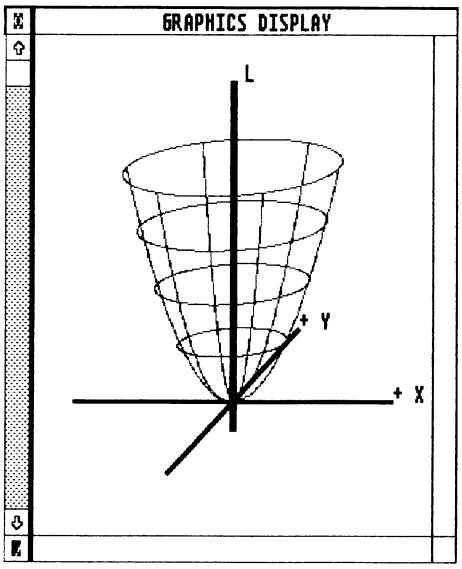


Abb. 5: Geometrische Deutung der Liapunov'schen Funktion als ein Paraboloid. Trajektorien: Kreise (Mittelpunkt), Parabeln (Knoten), (nicht eingezeichnet) Spiralen (Brennpunkt).

Damit wird Abb. 1 unsere kleine Wissensbasis und die Prozedur LIAPUNOV ein dafür in Logo geschriebener IF-Interpreter (Listing 6).

Nichtlineare Systeme

Bei nichtlinearen dynamischen Systemen haben die Bifurkation (Lösungsverzweiung) und die Existenz von selbsterregenden Oszillationen (Grenzzyklen) bei Mathematikern besondere Aufmerksamkeit ausgelöst. Beides wurde schon von Poincaré entdeckt, jedoch müssten die Anwendungen noch genau untersucht werden. Mit drei solchen Anwendungsbeispielen werden wir uns in den nächsten Abschnitten etwas eingehender beschäftigen.

Durch die Linearisierung von nichtlinearen Gleichungen (partielle Ableitungen an der Stelle des singulären Punktes) erhält man sogenannte Jacobi'sche Matrizen, deren Eigenschaften ein analoges Vorgehen, wie bei linearen Systemen, erlauben. Einzig beim Mittelpunkt ist dies nicht erlaubt.

Durch ein Theorem von E. Hopf (1942) ist es jetzt auch möglich zu prüfen, ob ein Grenzzyklus exisitiert [6]. Dieses Theorem können wir hier nicht besprechen, jedoch eine von den Bedingungen für seine Gültigkeit erwähnen: Der singuläre Punkt, den ein asympthotisch stabiler Grenzzyklus umgibt, ist ein Brennpunkt und zwar ein Reppelor. Mehr benötigen wir nicht.

Da wir bei nichtlinearen Differentialgleichungen nur selten eine ana-

```
?POALL
TO CURVE :A :I :P
(LOCAL "X "Y "R)
MAKE "X :P * (SIN :A) * EXP 0.01 * :A
MAKE "Y :P * (COS :A) * EXP 0.01 * :A
MAKE "R SQRT (:X * :X + :Y * :Y)
IF :R > 130 [STOP]
;IF OR :X > 100 :X < -100 [STOP]
;IF OR :Y > 130 :Y < -130 [STOP]
SETPOS SE :X :Y
CURVE :A + :I :I :P
END

TO CROSS
PU SETPOS [0 -135] PD SETPOS [0 135] TT "Y
PU SETPOS [-135 0] PD SETPOS [135 0] TT "X
END

TO S
PU HOME PD
END

TO SING3
CROSS
S CURVE 0 1 0.4
S CURVE 0 1 0.6
S CURVE 0 1 0.8
S CURVE 0 1 1
S CURVE 0 1 2
S CURVE 0 1 4
S CURVE 0 1 6
S CURVE 0 1 8
END
```

Listing 5

lytische Lösung kennen, ist eine grafische Darstellung der Trajektorien des Systems nach einer gelungenen numerischen Integration der Gleichungen am einfachsten. Hier bietet Logo aussergewöhliche Möglichkeiten an. Einerseits ist die Definition der Rechenvorschriften in Logo viel einfacher als z.B. in FORTRAN [7], der «klassischen» Sprache der Numerik. Andererseits ist aber auch die in Logo mögliche Umgehung der unnötigen Zwischenspeicherung der numerischen Daten von erheblichem Vorteil.

Ohne uns um die vielen Details der Numerik zu kümmern, übernehmen wir einfach aus [7] zwei geeignete Integrationsverfahren: Die etwas gröbere aber schnellere Methode nach Euler und die Standardmethode nach Runge-Kutta (Listings 7, 8).

Oekosystem

In den zwanziger Jahren haben ein Biologe, A. J. Lotka (1920), und ein Mathematiker, V. Volterra (1928), das nichtlineare «Räuber-Beute-Modell» zur Diskussion gestellt. Dieses für jedermann leicht verständliche Modell erfreut sich auch heute grosser Beliebtheit (obwohl schone eine unübersehbare Anzahl Publikationen darüber erschienen ist). Einzig über die Benennung des Modells ist man sich noch nicht einig: Beide Seiten wollen den Namen ihres Mannes vorne haben.

Worum handelt es sich? Betrachten wir z.B. in einer Bucht die Fische als «Räuber» (y) und ihr einziges Nahrungsmittel Plankton (x) als «Beute». Um zu einem mathematischen Modell zu kommen, müssen wir aber auch einige vereinfachende Annahmen vereinbaren. Ohne Fische gibt es (aus dem Meer) immer mehr Plankton. Ohne Plankton sterben die Fische aus. Fische vermehren sich proportional zum erbeuteten Plankton. Als zeitliche Aenderung ergibt dies (a, b positive Konstanten):

$x' = ax - xy$
$y' = -bx + xy$

Dieses dynamische System hat offensichtlich zwei singuläre Punkte: 1(0,0) und 2(b,a). Wir linearisieren das System und erhalten als Jacobi'schen Matrizen

$\underline{J1} = [[a\ 0][0\ -b]]$, $\det \underline{J1} = -ab$
$\underline{J2} = [[0\ -b][a\ 9]]$, $\det \underline{J2} = ab$, $TR\ \underline{J2} = 0$

das heisst im ersten Fall, $\det \underline{J1} < 0$, also ein Sattel und im zweiten Fall, $\det \underline{J2} > 0$ und $TR\ \underline{J2} = 0$, also ein Mittelpunkt. Nun, wie schon erwähnt, ist ausgerechnet in diesem Fall kein Verlass auf die Linearisierung.

Versuchen wir es mit der numerischen Integration, z.B. nach der Methode von Euler (Listing 7). Das Resultat ist aber enttäuschend (Abb. 6,A.). Anstatt einer geschlosssenen Kurve um den Mittelpunkt erhalten wir eine nicht geschlossene Kurve, die ihren Durchmesser ständig vergrössert.

Nun, wie schon erwähnt, die Methode von Euler ist ein grobes Tangenten-Verfahren. Die rechte Seite der Differentialgleichung ergibt ja

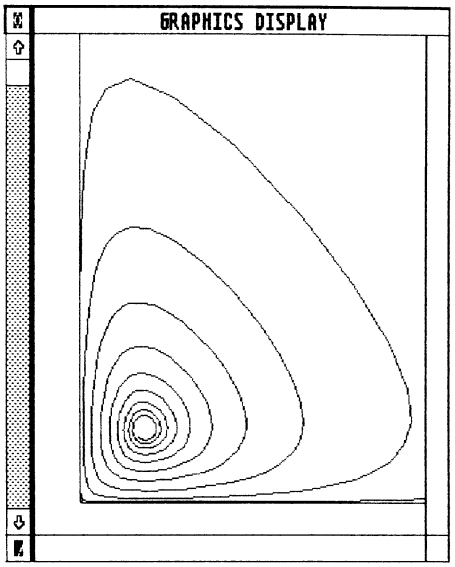


Abb. 6,A

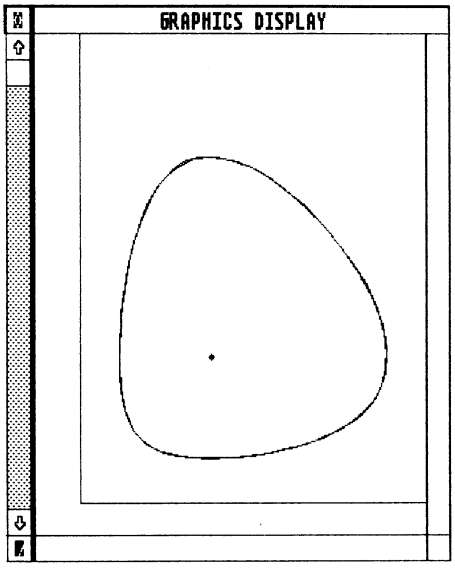


Abb. 6,B

Abb. 6: Numerische Integration des Lotka-Volterra-Modells. A. Methode nach Euler (Listing 7). B. Methode nach Runge-Kutta (Listing 8).

```
?PO "LIAPUNOV
TO LIAPUNOV
(LOCAL "a1 "a2 "a3 "a4 "DET "TR "DIS)
PR "
PR [Singular points and theirs stability]
PR [via Liapunov's second method.]
PR "
PR [Please enter elements of]
PR [the matrix A:]
PR "
PR [a1 a2 a3 a4]
MAKE "M RL
IF :M = [] [STOP]
MAKE "DET (FIRST :M) * (LAST :M) - (FIRST BF :M) * (LAST BL :M)
MAKE "TR (FIRST :M) + (LAST :M)
MAKE "DIS :TR * :TR - 4 * :DET
IF :DET = 0 [PR [Matix is singular] STOP]
IF :DET < 0 [PR [The singular point is a SADDLE] STOP]
IF :TR = 0 [PR [The singular point is a CENTER] STOP]
IF AND :TR < 0 :DIS > 0 [PR [Attracting NODE] STOP]
IF AND :TR < 0 :DIS = 0 [PR [Attracting improper NODE] STOP]
IF AND :TR < 0 :DIS < 0 [PR [Attracting SPIRAL] STOP]
IF AND :TR > 0 :DIS > 0 [PR [Instable NODE] STOP]
IF AND :TR > 0 :DIS = 0 [PR [Instable improper NODE] STOP]
IF AND :TR > 0 :DIS < 0 [PR [Instable SPIRAL] STOP]
END
```

Listing 6

schon die Gleichung der Tangente. Man setzt dort die Anfangswerte ein, multipliziert dies mit der Schrittlänge und erhält damit den ersten Näherungspunkt. Dies wiederholt man so oft, bis man alle gewünschten Punkte erhalten hat. Die Abweichung vom als richtig vermuteten Resultat ist also sehr gut verständlich.

Benützt man jedoch das optimierte Verfahren nach Runge-Kutta (Listing 8), erhält man genau das, was man vermutet hat (Abb. 6,B.). Dies ist natürlich kein Beweis, aber für dieses System wurde auch mit strengen mathematischen Methoden bewiesen, dass es sich tatsächlich um einen Mittelpunkt handelt.

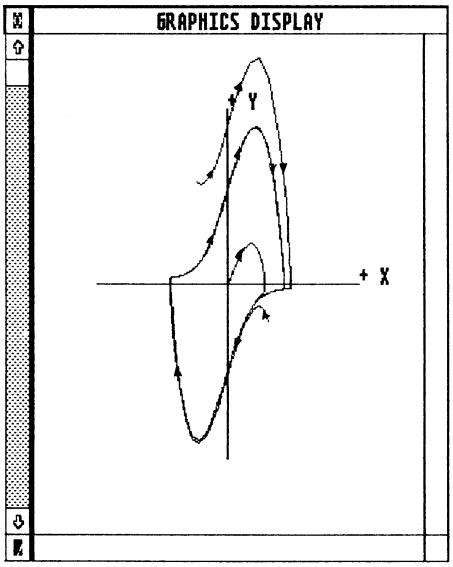


Abb. 7,A

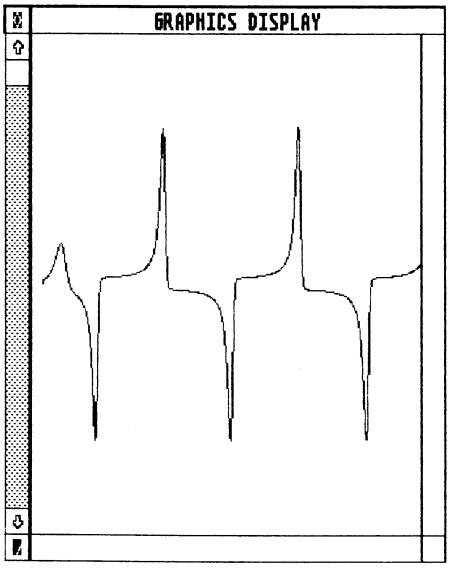


Abb. 7,B

Abb. 7: Numerische Integration (Euler) des Oszillators nach van der Pol. A. Der Grenzzyklus. B. Zeitlicher Verlauf der Oszillation.

Nun mögen sich manche Leser fragen, ob dieses simulierte ökologische Gleichgewicht mit den recht vagen Annahmen bei seiner Formulierung nicht zu naiv ist, um die Realität zu beschreiben. Im quantitativen Sinne trifft dies sicherlich zu.

Im qualitativen Sinne vermittelt das Modell doch eine plausible Beschreibung eines ökologischen Gleichgewichtes. Damit stellt sich auch die Frage: Wie verträgt sich dies mit dem klassischen Begriff des Gleichgewichtes (Ruhelage ist ein asymptotisch stabiler Punkt)?

Triode

Wer sich daran erinnern kann, weiss: Die Triode ist eine Elektronenröhre. Wenn man nun die Gitterelektrode der Triode mit einem Schwingungskreis (also Widerstand, Kondensator, Induktionsspule) koppelt, kann man damit unter Umständen elektromagnetische Wellen produzieren. So begann unsere Radiotechnik.

B. van der Pol veröffentlichte 1922 eine Arbeit, in der er mittels bekannter Gesetze von Ohm, Kirchhoff und Faraday die Wirkungsweise der Triode als dynamisches System zu deuten versuchte. Wie bereits erwähnt, löste dies eine Lawine von Arbeiten aus. Ein Beispiel: Band II des Werks von Andronov [2] ist ausschliesslich der mathematischen Theorie der damaligen Radiotechnik gewidmet.

Die Gleichung die van der Pol abgeleitet hat, sieht folgendermassen aus:

$$x'' + a(x^2 - 1)x' + x = 0$$

also eine Differentialgleichung zweiter Ordnung, wobei a ein Parameter ist. Mit einer neuen Variable kann daraus, wie gezeigt, das äquivalente System von zwei Gleichungen erster Ordnung abgeleitet werden:

$$x' = y$$
$$y' = -a(x^2 - 1)y - x$$

Dieses System hat nur einen singulären Punkt 1(0,0). Nach der Linearisierung folgt aus der Jacobi'schen Matrix, dass es sich um einen reppelierenden Brennpunkt handelt.

Wenn wir jetzt das Theorem von Hopf anwenden würden, könnten wir exakt feststellen, ob ringsherum ein Grenzzyklus existiert. Wir begnügen uns mit der numerischen Integration der Gleichungen nach der Methode von Euler (Abb. 7,A.). Wir sehen, dass bei verschiedenen Anfangswerten

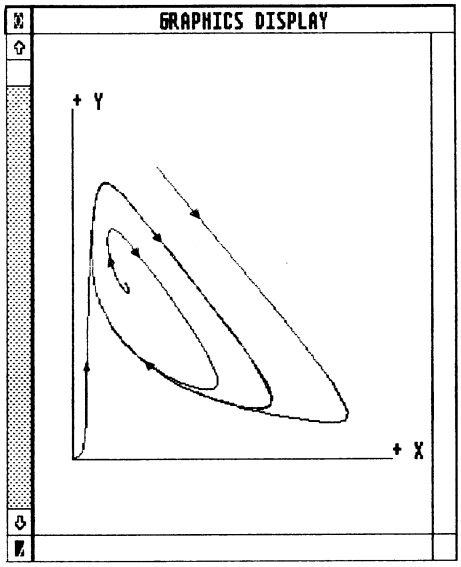


Abb. 8,A

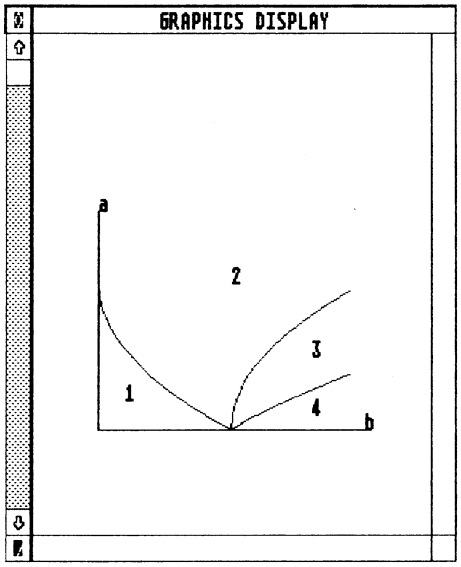


Abb. 8,B

Abb. 8: Brusselator. A. Numerische Integration (Runge). B. Bifurkationsdiagramm: 1 = attrative Knoten, 2 = attraktive Brennpunte, 3 = reppelierende Brennpunkte (mit Grenzzyklus), 4 = reppelierende Knoten.

(nächträglich eingezeichnete Pfeile beachten) die verschiedenen Trajektorien zum Grenzzyklus streben und dort auch bleiben, das heisst die Triode im Schwingungskreis ist ein selbsterregender Oszillator. Man nennt dies auch seltsamer Attraktor.

Wie schon erwähnt, hat Poincaré Grenzzyklen bei analytisch lösbaren Gleichungssystemen entdeckt, jedoch war dies nur für reine Mathematiker von Interesse. Wie gross aber das technische Intresse war sieht man sogar bei Andronov [2]. Er verwendete im erwähnten Schwingungskreis mit der Triode auch einen Oszillo-

graphen, um Grenzzyklen auf dem Schirm zu erzeugen. Die einfache Schaltung mit zwei einstellbaren Potentiometern stammt aus dem Jahr 1936 und stellt wahrscheinlich den ersten Analog-Computer dar.

Eine Bemerkung zur Methode von Euler. Dieses grobe, aber schnelle Rechenverfahren eignet sich besonders für Grenzzyklen. Der inherente Fehler dieses Verfahrens wird meistens durch die Attraktivität des Grenzzyklus korrigiert.

Van der Pol veröffentlichte 1928 auch eine Arbeit, in der er seinen Oszillator als Modell für die Funktion unseres Herzens vorschlug. Sehen wir uns dies mal an.

In beiden numerischen Prozeduren ist als Eingabevariable ein :N vorgesehen. Dies ist eine einfache Logo-Laufvariable. Man kann damit z.B. eine Abbruchbedingung formulieren. Wir können aber auch vereinbaren, damit die Zeit zu simulieren. Wenn man nun :X oder :Y gegen :N*:H (Zeit) zeichnet (Abb. 7,B.), erhält man, wie erwartet, wellenartige Kurven, Besonders interessant ist dabei, dass Herzspezialisten diese Kurven als Elektrokardiogrammen ähnliche bezeichnen.

Nun, von einer Triode bis zu mehr als 300'000 Transistoren auf einem Chip ist der Zeitabstand gar nicht so gross gewesen.

Brusselator

Bevor wir zur Erklärung des Titels kommen, müssen wir noch einiges nachholen. B. P. Belousov entdeckte 1958 eine neuartige katalytische Reaktion zwischen Bromat und Malonsäure. A.M. Zhabotinskii untersuchte 1964 die Kinetik dieser Reaktion (Tyson in [1]).

Nun, diese Belousov-Zhabotinskii-Reaktion erregte die Geister vieler Chemiker. Diese Reaktion oszilliert nämlich. Wenn man in einem Becherglas alle sechs Ingredienzien in Wasser löst und diese Mischung rührt, wechselt sie ihre Farbe rot-blau-rot-usw. Mit zwei geeigneten Elektroden in dieser reagierenden Mischung lässt sich der Zauber auch als wellenartige Aenderung des elektrochemischen Potentials auf einem Oszillographen bestätigen. Man führte diese Reaktion an verschiedensten Universitätsinstituten vor; man konnte sie sogar Mitte der siebziger Jahre an einer Mustermesse in Basel bewundern.

Unabhängig von alledem begann I. Prigogine 1947 in Brüssel die «Thermodynamik der irreversiblen Prozesse» zu entwickeln. Die chemische Ki-

Literatur

[1] *O. Gurel, O.E. Rössler.* Ed. Bifurcation Theory and Applications in Scientific Disciplines; Annals N.Y.Acad.Sci. 316(1979) 1-708

[2] *A. Andronow, A.A. Witt, S.E. Chaikin,* Theorie der Schwingungen; Akademie-Verlag, Berlin, 1965/69

[3] *S. Lefschetz,* Differential Equations: Geometric Theory; Intersci.Pub., New York, 1963

[4] *R. Bellman,* Methods of Nonlinear Analysis; Academic Press, New York, 1973

[5] *M.W. Hirsch, S. Smale,* Differential Equations, Dynamical Systems and Linear Algebra; Academic Press, New York, 1974

[6] *J.E. Marsden, M. McCracken,* Ed. The Hopf Bifurcation and Its Applications; Springer, New York, 1976

[7] *C.W. Gear,* Numerical Initial Value Problems in Ordinary Differential Equations; Prentice-Hall, Englewood Cliffs, 1971

[8] *P. Glansdorf, I. Progogine,* Thermodynamic Theory of Structure, Stability and Fluctuations; Wiley-Intersci., London, 1971


```
?PO "EULER1
TO EULER1 :N :X0 :Y0 :A :B :H
(LOCAL "K1 "L1 "X1 "Y1)
MAKE "K1 (:A * :X0 - :X0 * :Y0) * :H
MAKE "L1 -(:B * :Y0 - :X0 * :Y0) * :H
MAKE "X1 :X0 + :K1
MAKE "Y1 :Y0 + :L1
SETPOS SE :X1 :Y1
EULER1 :N + 1 :X1 :Y1 :A :B :H
END
```

Listing 7

```
?PO "RUNGE1
TO RUNGE1 :N :X0 :Y0 :A :B :C :H
(LOCAL "K1 "L1 "K2 "L2 "K3 "L3 "K4 "L4 "K "L "X1 "Y1)
MAKE "K1 (:A * :X0 - :C * :X0 * :Y0) * :H
MAKE "L1 -(:B * :Y0 - :C * :X0 * :Y0) * :H
MAKE "K2 (:A * (:X0 + 0.5 * :K1) - :C * (:X0 + 0.5 * :K1) * (:Y0 + 0.5 * :L1))
* :H
MAKE "L2 (-(:B * :Y0 + 0.5 * :L1) + :C * (:X0 + 0.5 * :K1) * (:Y0 + 0.5 * :L1))
 * :H
MAKE "K3 (:A * (:X0 + 0.5 * :K2) - :C * (:X0 + 0.5 * :K2) * (:Y0 + 0.5 * :L2))
* :H
MAKE "L3 (-(:B * :Y0 + 0.5 * :L2) + :C * (:X0 + 0.5 * :K2) * (:Y0 + 0.5 * :L2))
 * :H
MAKE "K4 (:A * (:X0 + :K3) - :C * (:X0 + :K3) * (:Y0 + :L3)) * :H
MAKE "L4 (-(:B * :Y0 + :L3) + :C * (:X0 + :K3) * (:Y0 + :L3)) * :H
MAKE "K (:K1 + 2 * :K2 + 2 * :K3 + :K4) / 6
MAKE "L (:L1 + 2 * :L2 + 2 * :L3 + :L4) / 6
MAKE "X1 :X0 + :K
MAKE "Y1 :Y0 + :L
SETPOS SE :X1 :Y1
RUNGE1 :N + 1 :X1 :Y1 :A :B :C :H
END
```

Listing 8

netik gehört natürlich auch dazu. So kam es auch, dass zwei seiner Mitarbeiter, R. Lefever und G. Nicolis, 1971 [8] bei einem rein hypothetischen Reaktionsmechanismus einen Grenzzyklus fanden. Dieses an sich mathematische Modell taufte Tyson 1973 auf den Namen Brusselator. Für alle seine Beiträge zur Thermodynamik erhielt I. Prigogine 1977 den Nobelpreis für Chemie.

Bevor wir uns diesem Modell widmen, sei noch folgendes erwähnt: R. M. Noyes von der Universität Oregon und Mitarbeiter untersuchten 1972 sehr genau die Belousov-Zhabotinskii-Reaktion. Daraus entstand später ein realistisches Modell dieser Reaktion; es bekam den Namen Oregonator. Heute kennt man schon eine ganze Anzahl oszillierender Reaktionen; die interessantesten stammen aus der Biochemie.

Nun zum Brusselator. Die Differentialgleichungen lauten:

$x' = a - (b + 1)x + x^2y$
$y' = bx - x^2y$

Dieses System hat nur einen singulären Punkt $1(a, b/a)$. Der im Original gefundene Grenzzyklus (für $a = 1$, $b = 3$) ist hier nach der Methode von Runge-Kutta ermittelt worden (Abb. 8,A.).

Nach der Linearisierung des Gleichungssystems folgt aus der Jacobi'schen Matrix

$$\text{DET } J = a^2$$
$$\text{TR } J = b - a^2 - 1$$
$$\text{DIS} = (b - a^2 - 1)^2 - 4a^2$$

Setzt man nun TR J = 0 und DIS = 0, so erhält man die Realtionen:

$a = \text{SQRT}(b - 1), a = 1 - \text{SQRT}b,$
$a = -1 + \text{SQRT}b$

Damit lässt sich für den Brusselator leicht ein Bifurkationsdiagramm (Abb. 8,B.) erstellen. Abhängig von den Parameterwerten kann sich dieses System verschieden verhalten (Bifurkation bzw. Lösungsverzweigung). Die asympthotisch stabilen Knoten oder Brennpunkte sind als Senken im Einklang mit der kassischen Thermodynamik der Gleichgewichte, die Quellen dagegen nicht. Selbsterregende Oszillationen, also Grenzzyklen, schliesslich stellen für die Thermodynamik etwas ganz Fremdes dar.

Man könnte meinen, dass bis heute das Problem gelöst wurde. Dem ist aber nicht so, denn die hier diskutierten Grenzzyklen sind nur ein Bruchteil

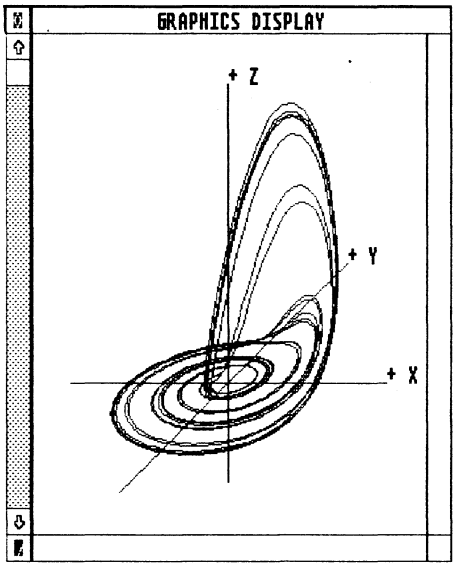


Abb. 9: Chaotischer Attraktor von Rössler [1]. Gleichungen: $x' = -y - z$, $y' = x + ay$, $z' = bx - cz + xz$. Werte: $a = 0.36$, $b = 0.4$, $c = 4.5$. Zeit pro Zyklus 1-2 Min. Die für Chaos charakteristische selbstähnliche Aufspaltung der Trajektorien nach ca. 15 Min.

dessen, was wir nicht kennen. Das können wir am Beispiel des chaotischen Attraktors von Rössler (Abb. 9) sehen. Deswegen hat bis heute die Bifurkationstheorie nichts an Aktualität verloren.

Wozu eignet sich Logo?

«Logo wurde zwar für Kinder entwickelt, ist jedoch keine Kinder-Programmiersprache». So etwas kann man sehr oft lesen. Genügt aber diese Definition?

Wir haben hier am Beispiel eines weit fortgeschrittenen technisch-wissenschaftlichen Gebietes den Einsatz von Logo nebenbei untersucht. Wir können feststellen: Von der Matrizenrechnung bis zur numerischen Integration von nichtlinearen Differentialgleichungssystemen konnte Logo ohne besonderen Aufwand angewendet werden.

Die direkte Kombinaiton von rechnerischer Erzeugung von Daten, deren Wandlung und deren grafischer Darstellung auf dem Bildschirm (z.B. Abb. 9) zeichnet Logo besonders aus, denn es exisitiert kaum eine andere Programmiersprache, in der man das gleiche Resultat mit weniger Aufwand realisieren kann. Demnach erscheint «Personal Programming Language» eine gerechtere Qualifikation für Logo zu sein, einer Programmiersprache nach jedermanns Wunsch, also auch für die Bedürfnisse der Kinder geschaffen. □

COMPUTER SPLITTER

Textverarbeitung: aber bitte mit Konzept!

(513/fp) Das papierlose Büro wird zwar oft herbeigesungen, und gleichzeitig wächst der Ausstoss an Papier im Büro munter weiter. Und dies wird vorübergehend so bleiben, meinen jedenfalls die Marktforscher. Ein guter Teil dieser Papierproduktion wird durch die betriebsinterne Textverarbeitung verursacht. Trotz ihrer betrieblichen Bedeutung auch in der papierenen Form wird die Textverarbeitung oft als Stiefkind behandelt. Und wenn man den Korrespondenzkräften Modernisierung zugesteht, dann passiert dies meist nach dem Konzept Wurstelei. Das Buch «Fachwissen Textverarbeitung», erschienen bei der Klaes GmbH, Agentur und Verlag in Essen, räumt mit diesem Missstand auf und bringt Konzept in die Textverarbeitung. Die zum Teil sehr ins Detail gehenden Vorschläge erläutern einen ganzen Projektverlauf für die Einführung oder Modernisierung in der Textverarbeitung. Als Beispiele für den erwähnten Detailreichtum sollen uns einige Stichworte aus dem Buch dienen: Disketten-Organisation, Kopf- und Rückenschmerzen, Token-Ring, Teletex. Am Ende des Buchs finden wir Tabellen zur Aufnahme von Ist/Soll-Analysen und eine Uebersicht über die wesentliche Hard- und Software bezüglich Textverarbeitung. Auch Leute aus der Host-Welt werden ihre Bedürfnisse abgedeckt sehen, finden doch auch modernste Dokumenten-Architekturen des Marktführers Eingang. In der Schweiz ist das Buch bei ADV-Info-Markt, Klosbachstrasse 123, 8032 Zürich, Tel. 01/251'34'40 erhältlich. □

20 MHz-80386-Prozessor

(435/eh) INTEL liefert nun seriemässig ihre Einplatinencomputer für Multibus-Architekturen mit dem neuen 20 MHz-80386-Prozessor aus. Diese Platinen sind standardmässig auch mit dem Coprozessor 80387 bestückt. Für Einsätze mit kleineren Anforderungen an die Rechenleistung sind auch weiterhin die 16 MHz-Versionen erhältlich, auch diese jedoch neu mit einer Aufnahme für den Coprozessor 80387. □

civem Computer

Profi-Hardware enorm günstig!

HARD DRIVES

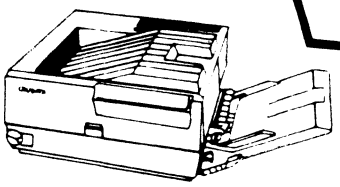
MiniScribe

MINISCRIBE	20 Mega	53 ms	695.-
MINISCRIBE	30 Mega	53 ms	795.-
MINISCRIBE	41 Mega	40 ms	1'280.-
MICROPOLIS	44 Mega	28 ms	1'630.-
MICROPOLIS	71 Mega	28 ms	2'125.-

HARDCARDS
30mB
MiniScribe
W/Western Digital
1'275

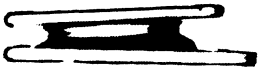
Citizen
80 Column Printer
Hi-Speed
599.00

LASERLINE
by OKIDATA



EASY TO USE. EASY TO BUY. EASY TO SELL.
BEST VALUE ON THE MARKET TODAY!
Fantastic Sale!!

EGA WONDER
845.-



ADJUSTABLE MONITOR STAND
49.-

Der Preishammer

10 MHz Profi-AT

NUMBER ONE
80286 COMPUTER
IN EUROPE!

2'998.-

ADD ON CARDS

- RS-232C CARD
- 2 MB RAM CARD
- MULTI-I/O CARD
- I/O PLUS II CARD
- HARD DISK CONTROLLER
- CLOCK
- CALENDAR CARD

Handy-Scanner
8086 SPEED CARD
2.5 MB MULTIFUNCTION CARD
ENHANCED GRAPHIC ADAPTER
HARD FLOPPY DISK CONTROLLER

Disketten-Kopierstation 945.-

Star - Star - Star - Star - Star
Star NL-10 **595.-**

MODEM
CCITT V21 V22
AUTOANSWER
FULL HAYES KOMPAT.
299.-

SOFTWARE
VERLANGEN SIE
UNSEREN
SOFTWARE-KATALOG

Seagate
20 MEGABYTE
HARD DISK
ONLY 845.-
inkl. Controller/Kabel

Günstige Möglichkeit der Finanzierung durch Ratenkredit.

CALL NOW **01-44'32'14**

Wir suchen ständig günstige Einkaufsquellen für die angebotenen und neue innovative Produkte.

COMPUTER MARKT

DIE AKTUELLE COMPUTERINFORMATION 5/87

Portabel und extrem leistungsfähig

Der Compaq Portable III ist der erste tragbare Rechner im Industriestandard, der mit einem 12 MHz getakteten 80286 Mikroprozessor ausgestattet ist; er ist voll kompatibel zu allen Hochleistungs-PCs führender Hersteller, arbeitet jedoch bis zu 50% schneller.

Der Compaq Portable III stellt im Hinblick auf sein umfassendes Leistungsangebot den derzeit höchsten Standard bei portablen Rechner für die professionelle Anwendung dar. Der Compaq Portable III braucht den Vergleich mit einem Hochleistungs-PC in keiner Hinsicht zu scheuen. Im Gegenteil, bei voller Funktionalität bietet dieser leicht zu t... rende Computer mit s... Gehäuse höhere ... schwindigkeit, ... cherkapazit... optione... Zoll-...

Abo-Bestellkarte vorne im Heft jetzt bestellen.

Infosys

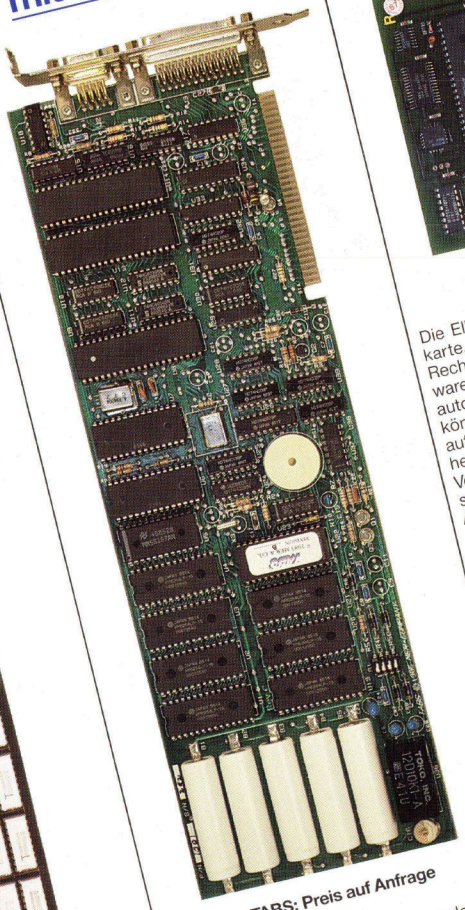
Die Zukunft Ihres PC's steht in diesen Karten!

Multi 4 AT – eine für alles, die Slotsparkarte!

Multi 4 AT inkl. 2 MB Memory: Fr. 2'675.–
Basisaufrüstung 512/640 KB: Fr. 250.–

Die Multi 4 AT von CE-Infosys ist eine Multifunktionskarte für den IBM XT 286 und alle AT's. Sie beinhaltet bis zu 4 MB RAM-Speicher, aufgebaut in 2 MB-Schritten mit 1 MB DRAM im Extended Memory, ausserdem alle Funktionen der Farbgrafikkarte EGA sowie eine parallele und eine serielle Schnittstelle. Über ein Modul kann ausserdem der Basisspeicher des Rechners auf 640 KB aufgerüstet werden.
Dank modernster SMD-Fertigungstechnik sind alle diese Funktionen auf einer Karte vereint; im Rechner wird nur 1 Slot belegt. In Rechnern bis zu 8 MHz Taktfrequenz läuft die Multi 4 AT ohne «Waitzyklen».
Zum Lieferumfang gehören ein deutsches Handbuch, die Info-PC-Software und kostenlos das Infosys Memory Managerprogramm, das einen Expanded Memory-Bereich unter Verwendung des Extended Memory der Multi 4 AT-Karte simuliert. Programme wie Lotus 1-2-3, Symphony und Framework benutzen den Expanded-Speicher.

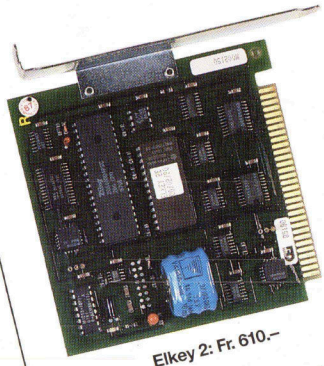
TABS hört mit. Telefonkosten-Analyse (Abrechnung) mit Ihrem PC!



TABS: Preis auf Anfrage

TABS ermöglicht es, Ihren Personalcomputer als private Gebühren- und Datenverarbeitungsanlage (GDV) an Ihre PTT-Haustelefonzentrale anzuschliessen.
TABS sammelt die von Ihrer Telefonvermittlungsanlage gelieferten Daten über abgehende oder eventuell auch ankommende Gespräche und wertet diese aus. Mit TABS können bis zu 2000 Zweiganschlüsse einer Haustelefonzentrale überwacht werden.
Die TABS-Auswertung beinhaltet z.B. Detailangaben über angewählte Telefonnummern, Gesprächsdauer und Gesprächskosten pro Zweigstelle. TABS ermöglicht die Zusammenfassung von Telefonkosten pro Abteilung. Statistiken über die Belegung der Amtsleitungen einer Zentrale während 24 Stunden ergeben Aufschluss über mögliche Kommunikationsengpässe. Im Bereich Hotelerie erstellt TABS eine ausführliche Telefonabrechnung für jeden Gast u.a.m.
TABS ist eine Zusatzkarte mit eigenem Prozessor und Stromversorgung (Akkus), für IBM XT AT's und 100%-kompatible Rechner. Das TABS Programm funktioniert somit im Hintergrund, selbst bei ausgeschaltetem Rechner. Der Anschluss an eine Haustelefonzentrale erfolgt mittels PTT-geprüften Modems (RS 232).

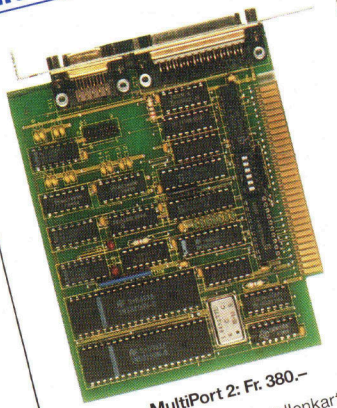
Elkey 2 – Zutritt für Unbefugte verboten!



Elkey 2: Fr. 610.–

Die Elkey 2 ist eine elektronische Sicherheitskarte, welche die unerlaubte Benutzung des Rechners, unabhängig von der Anwendersoftware, verhindert. Elkey 2 stellt sicher, dass nur autorisierte Personen den Rechner benutzen können. Insbesondere Rechner mit Festplatte, auf denen unternehmenswichtige Daten stehen, benötigen einen derartigen Schutz.
Vertrauliche Files können mit der Elkey 2 verschlüsselt werden.
Auf der Elkey 2 wird ein elektronisches Logbuch geführt.
Zum Lieferumfang gehören ein deutsches Handbuch, ein Logbuch für Super User und die Info-PC-Software zur Installation der Schutzmechanismen.

MultiPort 2 – Karte mit Schnitt(stellen)!



MultiPort 2: Fr. 380.–

MultiPort 2 ist eine Schnittstellenkarte mit 1 parallelen und 2 seriellen Schnittstellen. Mit der MultiPort 2 können Rechner, PC-XT's oder AT's, auf 4 serielle Schnittstellen ausgebaut werden. Mousetreiber für COM3: und COM4: werden kostenlos mitgeliefert. Die seriellen Schnittstellen der MultiPort 2 können als COM1: und COM2:, oder mittels einem mitgelieferten Device-Treiber als vollwertige COM3: und COM4: angesprochen und betrieben werden.

BÜFA
Halle 221
Stand 347

Computer Elektronik Infosys GmbH, Am Kümmerling 2, 6501 Bodenheim, Telefon 061 35/30 81
Computer Elektronik Infosys AG, Im Rötel 10a, CH-6301 Zug, Telefon 042/21 78 57
Lager und Vertrieb Schweiz: Oberdorfstrasse 11, CH-8953 Dietikon, Telefon 01/741 30 41

ATW, Regensdorf