inforum

INFOrmationsforum des Rechenzentrums der Universität Münster
Jahrgang 9, Nr. 2 April 1985

Inhalt

Impressum

Redaktion *inforum*

A. Achilles (Tel. 83-2488)
W. Bosse (Tel. 83-2461)
H. Pudlatz (Tel. 83-2472)
E. Sturm (Tel. 83-2609)

Satz: P. Bigdon
Druck: H. Mecke

Rechenzentrum der Universität
Einsteinstr. 60
4400 Münster

Auflage dieser Ausgabe: 1000
Redaktionsschluß der nächsten Ausgabe: 14.06.85

Editorial

W. Bosse

Die wachsende Bedeutung der Informationstechnologie für die Belange der Universität ist längst unbestritten. Dies betrifft nicht allein den Bereich der Forschung, in dem DV-Geräte und -Verfahren bereits in großem Umfang eingesetzt werden, sondern immer stärker auch den Bereich der Lehre. Um dem zukünftigen Bedarf gerade im letztgenannten Bereich zu entsprechen, haben Bund und Länder das Computer-Investitions-Programm (CIP) aufgelegt, um alle Hochschulen bedarfsgerecht mit Rechenkapazität speziell für Lehre und Ausbildung auszustatten. Da hierbei Mikrocomputer, die untereinander vernetzt sind, als modernes Arbeitsmittel eingesetzt werden sollen, müssen die Beschaffungswünsche der einzelnen Fachbereiche universitätsintern dahingehend koordiniert werden, daß die Probleme der Wartung und Betreuung sowie die Software-Problematik mit vertretbarem Aufwand lösbar sind. Diese *inforum*-Ausgabe enthält einen ersten Artikel zum CIP, in dem der derzeitige Stand angesprochen wird. Das Thema wird auch für die nächsten Ausgaben aktuell sein.

Hinsichtlich der Probleme bei der Auswahl von Mikrocomputern ist der Aspekt der Kompatibilität verschiedener Geräte zueinander einigermaßen wichtig, wie dies am Markt durch das Beispiel des IBM PC zu beobachten ist (Hannover-Messe!). Der Begriff der Kompatibilität wird anhand dieses Beispiels in einem Artikel verdeutlicht.

Da es an der Universität Münster einen großen Bedarf an anwendungsorientierter Forschung und Entwicklung auf dem Gebiet der Informatik gibt, ist der Beschluß, ein neues Institut für Angewandte Informatik zu gründen, sehr zu begrüßen. Wegen der Beteiligung des Rechenzentrums der Universität an diesem Institut berichten wir über die Pressekonferenz, auf der die Industrie- und Handelskammer zu Münster (IHK) und die Universität den Beschluß der Institutsgründung erläutert haben.

Wie in der letzten *inforum*-Ausgabe angesprochen, steht jetzt auch UNIX als Dialogsystem auf einem Großrechner (zusätzlich) zur Verfügung. Das Tutorial gibt dazu eine erste Einführung. Im Lehrangebot des Rechenzentrums hat UNIX schon seit einigen Semestern seinen Platz.

Angesichts all dieser für die Datenverarbeitung an der Universität bedeutsamen Neuigkeiten sollte aber die traditionelle Komponente nicht vergessen werden: der Stapelbetrieb (MVS-Batch), der nach wie vor die Produktionslast trägt. Außer zu den geplanten Wartungszeiten ist der Betrieb in den letzten Monaten nicht unterbrochen worden; vom 26. Februar bis zum 25. März lief die Maschine "rund um die Uhr" ohne einen einzigen Neustart. Notwendige Systemarbeiten und die Umstellung auf die Sommerzeit haben anschließend zwar zu zwei Unterbrechungen geführt, wodurch aber der Gesamteindruck eines stabilen Betriebs nicht getrübt wird. Da es offensichtlich typisch für die Datenverarbeitung ist, daß man als Benutzer Fehlersituationen beklagt, einen störungsfreien Betrieb aber stillschweigend hinnimmt, mag es gut sein, die 'selbstverständlichen' Dinge auch einmal zu erwähnen (oder hätten Sie's gewußt?).

Neben einigen Ankündigungen zu FORTRAN, zu denen auch die "Elementary Math Library" (EML) gehört, sollte der Artikel über "ACRITH" Aufmerksamkeit finden, in dem über Fortschritte auf dem Gebiet der Computer-Numerik berichtet wird. Wer Zahlen als korrekt betrachtet, nur weil sie auf einem Computer berechnet worden sind, ohne sich über die Grundlagen der Berechnung und die verwendeten Verfahren Rechenschaft abzulegen, der muß seine Denkweise schnellstens ändern. Wie so oft, liegt das Problem nicht allein beim Computer, sondern vor allem in der Fehleinschätzung seiner Möglichkeiten und der ungeprüften Übernahme von Ergebnissen.

RUM-Aktuell

Computer-Investitions-Programm (CIP)

W. Held

Bund und Länder haben auf Anregung des ehemaligen Vorsitzenden der Kommission für Rechenanlagen der Deutschen Forschungsgemeinschaft, Herrn Prof. Dr. Haupt, für die Jahre 1985 bis 1988 Sondermittel in Höhe von insgesamt 240 Millionen DM zur Beschaffung vernetzter Mikrorechnerarbeitsplätze für Lehre und Studium an Universitäten und Fachhochschulen bereitgestellt.

Das Programm orientiert sich an Vorhaben verschiedener amerikanischer Hochschulen, die von jedem Studenten erwarten, daß er zu Beginn seines Studiums über einen Mikrorechner verfügt. Damit diese Rechner an das Hochschulnetz angeschlossen werden können, müssen sie den von der Universität vorgegebenen Konventionen entsprechen. Die von Bund und Ländern veranschlagten Sondermittel reichen zwar nicht, jedem Studenten in der Bundesrepublik einen "eigenen" Mikrocomputer zu geben, durch sie wird aber jedem Studenten zeitweise der Zugang zur Datenverarbeitung ermöglicht werden können.

Mit diesen Geräten sollen natürlich nicht nur die Programmierkurse ausgeweitet werden. Vor allem sollen sie in möglichst vielen Lehrveranstaltungen genützt werden. Darüber hinaus sollen Studenten und Doktoranden im Rahmen ihrer Studienabschluß-Arbeiten Zugang zu den Geräten haben.

Nutzungsmöglichkeiten dürften sich - eventuell nach entsprechender Einbeziehung in die Lehrveranstaltungen - nahezu in jedem Fach bieten. Neben den schon traditionellen mathematischen und statistischen Auswertungen sind z. B. denkbar: Planspiele, Datenbankanwendungen und Literatur-Recherchen, Einsatz in Praktika zur Meßwerterfassung und Steuerung, anspruchsvolle Textverarbeitung, Kommunikation mit Hilfe der Datenverarbeitung, grafische Darstellungen. Einsetzbar sind die Geräte auch bei der Erfassung und Gestaltung von Diplom- und Examensarbeiten mit Hilfe des Rechners.

Da viele Studenten im späteren Berufsleben mit der Datenverarbeitung zu tun haben werden, ist dieses Investitionsprogramm zur Förderung der Lehre sehr zu begrüßen. Die Studenten werden von den Möglichkeiten sicher reichlich Gebrauch machen.

Die Personal-Computer sollen - verteilt über die Universität - in Pools, die Studenten zugänglich sein müssen, aufgestellt werden. Die Universität Münster hat unter Beteiligung der Fachbereiche einen ersten Teilantrag für 1985 erarbeitet und an die zuständigen Stellen weitergeleitet. Die Geräteauswahl steht noch bevor. Dem Universitätsrechenzentrum liegen dazu mehr als 20 zum Teil sehr günstige Angebote vor. Mit der Prüfung der Geräte wurde begonnen. Es ist wahrscheinlich, daß die in diesem Jahr zu beschaffenden Gerätepools im Wintersemester 85/86 einsetzbar sein werden.

EDV-Kolloquium in unserer Universität

W. Held

Um einen Informationsaustausch zwischen allen EDV-Anwendern in unserer Universität anzuregen, hat der Rektor, Herr Prof. Dr. Schlüter, zu einem zweitägigen EDV-Kolloquium in unserer Universität am 11. und 12.6.1985 aufgerufen.

Die Vorträge sollen die DV-Aktivitäten aller Fachbereiche widerspiegeln. Gedacht ist z.B. an Überblicke über DV-Anwendungen im Fachbereich oder einzelne Themen aus den Bereichen Großrechner-, Minirechner- und Mikrorechnereinsatz, Prozeßdatenverarbeitung, graphische Datenverarbeitung, numerische und statistische Anwendungen, linguistische Datenverarbeitung und Textverarbeitung, Kommunikation, DV-Organisation, DV in der Lehre sowie gesellschaftliche Auswirkungen des DV-Einsatzes. Denkbar sind natürlich auch Vorträge über ungelöste Aufgaben. Ebenso willkommen sind Hinweise auf Probleme aller Art beim Einsatz von DV-Geräten und -Programmen.

Zur Zeit wird das Vortragsangebot zusammengestellt. Sobald die nach Schwerpunkten zu gliedernde Veranstaltungsfolge festliegt, werden die Themen und der Ort des EDV-Kolloquiums universitätsweit bekanntgemacht.

Da angestrebt wird, daß die einzelnen Vorträge relativ kurz und für Hörer aller Fachbereiche verständlich sind, kann ein interessantes, das weite Spektrum der DV-Anwendungen in der Universität repräsentierendes Programm erwartet werden.

Personalia

Am 1.2.1985 hat Herr Dipl.-Math. W. Stockhofe seine Tätigkeit als wissenschaftlicher Mitarbeiter am Rechenzentrum aufgenommen. Er war vorher am Fachbereich Mathematik beschäftigt und wird sich im Rahmen eines DFG-Projekts hauptsächlich mit Fragen der Vernetzung von Rechnern und dem UNIX-Betriebssystem auseinandersetzen.

Ebenfalls zum 1.2.1985 sind Frau S. Koppenstein und Herr D. Stüken als studentische Mitarbeiter am Rechenzentrum eingestellt worden.

Wir freuen uns mit Herrn B. Süselbeck über seine Promotion zum Dr. rer. nat. am 13.2.1985.

Zum 31.3.1985 sind die Herren K.-D. Fusenig und A. Jaron als studentische Mitarbeiter ausgeschieden.

VS FORTRAN Release 4.0

St. Ost

Am 29. April dieses Jahres wird das Release 3.1 des VS-FORTRAN-Compilers im MVS und CMS durch das verbesserte und erweiterte Release 4.0 ersetzt. Das Release 4.0 unterscheidet sich in folgenden Punkten von seinem Vorgänger:

1. Unterstützung von (VSAM-) Dateien mit der Organisationsform KEYSEQUENCED:

   Die Datensätze einer Datei der Organisationsform KEYSEQUENCED sind nach einem Satzschlüssel logisch geordnet. Um einen Bezug auf diese Satzschlüssel herstellen zu können, wurden die FORTRAN-Anweisungen OPEN, READ, WRITE und INQUIRE erweitert und zwei neue FORTRAN-Anweisungen DELETE und REWRITE hinzugefügt, die das Löschen oder Zurückschreiben eines Datensatzes nach vorhergehendem Lesen ermöglichen.

2. Effizientere Bearbeitung von CHARACTER-Ausdrücken:

   Bislang wurde jede Manipulation von CHARACTER-Daten in einem FORTRAN-Programm beim Übersetzungvorgang vom Compiler in Unterprogramm-Aufrufe umgewandelt. Das Release 4.0 des Compilers fügt stattdessen den notwendigen Programmcode direkt in das übrige Maschinenprogramm ein, was zu einer schnelleren Ausführung des übersetzten Programmes führt.

3. Die unformatierte Ein/Ausgabe wurde in zwei Punkten ergänzt:

   - Das Satzformat (RECFM) ist nicht mehr auf VBS oder U beschränkt.
   - Die unformatierte READ-Anweisung wurde um einen Parameter NUM=**anzahl** erweitert, wobei **anzahl** eine Variable vom Typ INTEGER*4 ist, die nach Beendigung der READ-Operation die Anzahl der übertragenen Bytes enthält.

4. CHARACTER- und REAL*8-Variablen können nunmehr in der Typ-Vereinbarung initialisiert werden.

5. Die VS-FORTRAN-Dokumentation wurde neu überarbeitet, womit eine bessere Handhabbarkeit und Lesbarkeit erreicht wurde, leider auf Kosten eines mehr als doppelt so hohen Preises.

Kompatibilitätsprobleme zwischen dem alten und neuen VS-FORTRAN-Release sind nicht zu erwarten. Sollten sich beim Übergang dennoch Probleme ergeben, so kann das Release 3.1 für eine gewisse Übergangszeit im MVS mit Hilfe des symbolischen Parameters VERSION=31 und im CMS mittels der Option VERSION(OLD) benutzt werden. Im MVS steht das (uralte) Release 2.0 mit Einführung des Release 4.0 nicht mehr zur Verfügung.

Um etwaige Unverträglichkeiten zwischen altem und neuem Compiler-Release wirksam untersuchen zu können, bitten wir um eine entsprechende Meldung an die Abteilung Systemsoftware des Rechenzentrums (Tel. 2607).

Elementary Math Library

B. Süselbeck

Die Elementary Math Library (EML) ist eine Sammlung von Unterprogrammen zur Berechnung elementarer mathematischer Funktionen wie

- Wurzel (SQRT, DSQRT)
- Exponentialfunktion (EXP, DEXP)
- Logarithmen (ALOG, ALOG10, DLOG, DLOG10)
- komplexer Absolutbetrag (CABS, CDABS)
- trigonometrische Funktionen (SIN, COS, TAN, COTAN, DSIN, DCOS, DTAN, DCOTAN)
- Inverse trig. Funktionen (ARSIN, ARCOS, ATAN, ATAN2 DARSIN, DARCOS, DATAN, DATAN2,)
- Exponentiation (FRXPR, FDXPD)

Die Routinen stehen in einfacher und doppeltgenauer Version zur Verfügung und können von VS-FORTRAN-Programmen aus aufgerufen werden. Die Namen der Funktionen (s.o.), Anzahl der Argumente und Vereinbarungen sind identisch mit denen der entsprechenden Unterprogramme aus der VS-FORTRAN Mathematical Library, weisen allerdings eine höhere Genauigkeit auf. Einige Funktionen liefern für alle Argumente die dem exakten Wert am nächsten liegende Maschinenzahl. Bei allen Routinen ist aber höchstens das letzte Bit der Mantisse falsch und dies bei maximal 5% der vom Hersteller getesteten Argumente. Im Gegensatz dazu sind bei den entsprechenden Funktionen der VS FORTRAN Mathematical Library an nicht wenigen Stellen mehrere Ziffern des Ergebnisses falsch. Die erhöhte Genauigkeit der Routinen wird durch neue Algorithmen erzielt, die im wesentlichen folgendermaßen vorgehen:

1. Durch Ausnutzen mathematischer Eigenschaften (Additionstheoreme, Beziehungen der Funktionen untereinander) werden die Argumente auf einen kleinen Bereich zurückgeführt.

2. Dieser Bereich ist an gewissen Punkten, die so gewählt sind, daß das zu berechnende Ergebnis möglichst nahe an einer maschinendarstellbaren Zahl liegt, mit hoher Genauigkeit in einer oder mehreren Tabellen vertafelt.

3. Für Argumente in der Nähe der vertafelten Werte wird der Funktionswert mittels einer Minimax-Polynom Approximation niedrigen Grades berechnet.

Diese erhöhte Genauigkeit wird nicht durch größeren Zeitbedarf erkauft, vielmehr ergibt sich auf Anlagen des Typs 4381 Model 2 durch hardwaremäßige Unterstützung der Wurzel-, Exponential- und Logarithmusfunktion sogar eine Rechenzeitersparnis.

Es ist geplant, die EML zusammen mit dem Release 4.0 des VS-FORTRAN Compilers so zu installieren, daß bei Aufruf der oben angegebenen Programme automatisch auf die EML zugegriffen wird. Weitere Informationen (Algorithmen, Angaben zur Genauigkeit, Fehlermeldungen) findet man im ELEMENTARY MATH LIBRARY Program Reference and Operations Manual (IBM SH20-2230-1) sowie in der Literatur zu VS-FORTRAN (Release 4.0).

Neuauflage der CMS-Broschüren

W. Bosse

Die in der letzten *inforum*-Ausgabe bereits angekündigten Neuauflagen in der Reihe *Software-Informationen* sind inzwischen erschienen. Dabei handelt es sich um folgende Titel:

SI 10 CMS für Anfänger
A. Achilles, 2. Auflage, Februar 1985. 1,50 DM

SI 11 Benutzerhandbuch VM/CMS
A. Achilles, W. Held, B. Neukäter, H. Pudlatz, E. Sturm
2. Auflage, Februar 1985. 5,50 DM

Diese Broschüren können zu den festgelegten Ausgabezeiten

Mo 13.45 - 15.00 Uhr
Di/Do 10.30 - 11.45 Uhr

im Sekretariat des Rechenzentrums bei Frau Luth (Zimmer 101) erworben werden.

Zu dem *Benutzerhandbuch VM/CMS* (SI 11) sind noch folgende Hinweise zu geben:

- Es sind einige neue Kommandos aufgenommen worden.
- Im Anhang liegt eine Kurzbeschreibung der Kommandosprache REXX vor.
- Obwohl auch einzelne Beschreibungen modifiziert worden sind, verliert die 1. Auflage nicht ihre Gütigkeit. Eine Ergänzungslieferung zur 1. Auflage ist jedoch nicht vorgesehen.

Neue Datensichtgeräte

A. Achilles

Nach Installation einer neuen Terminalsteuereinheit konnten Mitte März sechs neue Datensichtgeräte (Typ IBM 3180 Mod.1) in Betrieb genommen werden. Die hinzugekommenen Geräte ermöglichten eine funktionale Gruppierung der Datensichtgeräte im Terminalraum des Rechenzentrums:

- 12 Bildschirmgeräte für die Eingabe größerer Datensätze oder Programme stehen auf der der Einsteinstraße zugewandten Seite des Terminalraums. Diese Geräte verfügen über die unten beschriebene Einrichtung "ENTRY ASSIST", die neue Funktionen für das Edieren bereitstellt. Von diesen Terminals aus kann keine Hardcopy-Ausgabe auf dem Terminal-Drucker erstellt werden.
- Auf der gegenüberliegenden Seite stehen 10 Bildschirme, von denen sieben mit der APL-Einrichtung ausgestattet sind. Diese stehen vornehmlich für Programmentwicklung und -test zur Verfügung. Sie sind mit dem Hardcopy-Drucker verbunden.
- Die Geräte der linken Seite des hinteren Raumes sind der Arbeit mit interaktiver graphischer Software vorbehalten.

Die neuen Datensichtgeräte werden - im Gegensatz zu den bekannten Sichtgeräten IBM 3278 - über die Tastaturfunktion "SET UP" eingerichtet. Nach einem Druck auf diese Taste erscheint in der Statuszeile eine blinkende Raute. Nun können die Tasten mit den blauen Symbolen zur Einrichtung des Bildschirms benutzt werden. Dieser Zustand wird durch einen erneuten Druck auf die "SET UP"-Taste beendet.

Mit Hilfe des "ENTRY ASSIST" wird die Dateneingabe durch Formatierungshilfen, erweiterte Möglichkeiten der Cursor-Bewegung und zusätzliche Edierfunktionen unterstützt. "ENTRY ASSIST" wird aktiviert bzw. beendet, indem zunächst die "EXSEL"- und dann die Funktionstaste 13 gedrückt wird. Einzelheiten können dem ebenfalls ausgelegten Auszug aus dem User's Guide entnommen werden.

Eine Auswahl der Möglichkeiten, die "ENTRY ASSIST" bietet, ist:

- Einrichtung von Randbegrenzungen,
- Setzen von Tabulatoren,
- Setzen eines Zeilenendsignals,
- Cursor-Bewegungen zum nächsten oder vorhergehenden Wort,
- Cursor-Bewegungen zum nächsten oder vorhergehenden Tabulatorstop,
- Löschen von Wörtern,
- Wiederholtes Löschen von Zeichen,
- Automatischer Wortumbruch,
- Löschen fehlerhafter Texte im Insert-Modus mit der Rücksetztaste.

Neues Institut für Angewandte Informatik

W. Bosse

Auf einer gemeinsamen Pressekonferenz wurde am 13. Februar 1985 die Öffentlichkeit über den Beschluß der Industrie- und Handelskammer zu Münster (IHK) und der Universität unterrichtet, ein Institut für Angewandte Informatik an der Westfälischen Wilhelms-Universität Münster zu gründen, das von der Wirtschaft getragen und finanziert wird.

Wie der Präsident der IHK, H.G. Borgmann, betonte, ist ein Engagement im Bereich der Angewandten Informatik deshalb naheliegend, da der Informationstechnik für die Unternehmen im Kammerbereich große Bedeutung zukommt. Demgemäß sollen durch das interdisziplinäre Institut, an dem neben der Angewandten Mathematik und den Wirtschaftswissenschaften auch das Universitätsrechenzentrum beteiligt ist, Fragestellungen und Probleme aus der wirtschaftlichen Praxis aufgegriffen und untersucht werden. Schwerpunkte des Instituts sind dabei die anwendungsorientierte Forschung und Entwicklung sowie die Unterstützung der Anwendung wissenschaftlicher Erkenntnisse in der Praxis auf dem Gebiet der Angewandten Informatik.

Der Rektor der Universität, Prof. Dr. W. Schlüter, begrüßte das Engagement der IHK. Da das neue Institut "nicht als einseitige, bedingungslose Serviceleistung der Universität" zu verstehen sei, werde "durch die Aufnahme von praxisnahen Fragestellungen und Problemen grundsätzlicher Art ... auch die

Wissenschaft von dieser Einrichtung profitieren." Dadurch werde auch die notwendige Forschung, die gegenwartsbezogen ist, möglich. Eine direkte Auftragsforschung für einzelne Firmen ist nicht beabsichtigt.

Nun gibt es schon einzelne Institute, die aufgrund externer Finanzierung den Status *'an* der Universität' haben. Das Besondere an diesem Institut für Angewandte Informatik ist seine interdisziplinäre Struktur, die auch die fachübergreifende Bedeutung der Informatik widerspiegelt. Direktoren des Instituts sind Prof. Dr. N. Schmitz, Direktor des Instituts für Mathematische Statistik, Prof. Dr. H. Wagner, Lehrstuhl für Betriebswirtschaftslehre, insbesondere Organisationstheorie und elektronische Datenverarbeitung und Dr. W. Held, Leiter des Universitätsrechenzentrums.

Das Institut wird voraussichtlich seine Arbeit im Mai aufnehmen können. Die notwendigen Satzungen und Vereinbarungen liegen vor, die zuständigen Gremien der Universität - Rektorat, Senat und die beteiligten Fachbereiche Mathematik und Wirtschaftswissenschaften - haben bereits positiv entschieden.

Schriftarten des Agfa-Druckers

W. Kaspar

Zur Ausgabe von CMS-Dateien auf dem Agfa-Drucker P400 stehen zur Zeit folgende nicht-proportionale Schriften zur Verfügung:

Bulletin22.N.6.P

```
a b c d e f g h i j k l m n o p q r s t u v w x y z
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
1 2 3 4 5 6 7 8 9 0
@ # $ % & * ( ) + = ! , . ; : ? " '
```

Mit dieser Schrift können bei einem (standardmäßig) vorgegebenen linken Rand von 34 Millimetern 140 Zeichen in eine Zeile gedruckt werden. Setzt man den linken Rand auf 2 Millimeter (LEFTMARGIN 2MM) so passen 165 Zeichen in eine Zeile.

Sind beim Aufruf von FPRINT die Optionen "NOCC FCB 38DOTS FONT Bulletin22.N.6.P TOPMARGIN 0" wirksam, so können bis zu 123 Zeilen auf einer Seite gedruckt werden, indem die LINECOUNT-Option auf den gewünschten Wert gesetzt wird.

Antic15.N.8.P

```
a b c d e f g h i j k l m n o p q r s t u v w x y z
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
1 2 3 4 5 6 7 8 9 0
@ # $ % & * ( ) + = ! , . ; : ? " '
```

Mit dieser Schrift können bei einem (standardmäßig) vorgegebenen linken Rand von 34 Millimetern 103 Zeichen in eine Zeile gedruckt werden. Setzt man den linken Rand auf 2 Millimeter (LEFTMARGIN 2MM) so passen 122 Zeichen in eine Zeile.

Sind beim Aufruf von FPRINT die Optionen "NOCC FCB 49DOTS FONT Antic15.N.8.P TOPMARGIN 0" wirksam, so können bis zu 95 Zeilen auf einer Seite gedruckt werden, indem die LINECOUNT-Option auf den gewünschten Wert gesetzt wird.

Modern10.N.12.P

```
a b c d e f g h i j k l m n o p q r s t u v w x y z
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
1 2 3 4 5 6 7 8 9 0
@ # $ % & * ( ) + = ! , . ; : ? ” ’
```

Mit dieser Schrift können bei einem (standardmäßig) vorgegebenen linken Rand von 34 Millimetern 84 Zeichen in eine Zeile gedruckt werden. Setzt man den linken Rand auf 2 Millimeter (LEFTMARGIN 2MM) so passen 100 Zeichen in eine Zeile.

Sind beim Aufruf von FPRINT die Optionen ”NOCC FCB 66DOTS FONT Modern10.N.12.P TOPMARGIN 0” wirksam, so können bis zu 71 Zeilen auf einer Seite gedruckt werden, indem die LINECOUNT-Option auf den gewünschten Wert gesetzt wird.

Courier10.N.10.P

```
a b c d e f g h i j k l m n o p q r s t u v w x y z
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
1 2 3 4 5 6 7 8 9 0
@ # $ % & * ( ) + = ! , . ; : ? " '
```

Mit dieser Schrift können bei einem (standardmäßig) vorgegebenen linken Rand von 34 Millimetern 68 Zeichen in eine Zeile gedruckt werden. Setzt man den linken Rand auf 2 Millimeter (LEFTMARGIN 2MM) so passen 80 Zeichen in eine Zeile.

Sind beim Aufruf von FPRINT die Optionen "NOCC FCB 65DOTS FONT Courier10.N.10.P TOPMARGIN 0" wirksam, so können bis zu 72 Zeilen auf einer Seite gedruckt werden, indem die LINECOUNT-Option auf den gewünschten Wert gesetzt wird.

Übersicht

Schrift	Zeich. Breite in Dots	Zeichen/Zeile Rand 34mm	Zeichen/Zeile Rand 2mm	Pitch	max. Zeilen pro Seite FCB in Dots	max. Zeilen pro Seite Zeilen
Bulletin22.N.6.P	20	140	165	22	38	123
Antic15.N.8.P	27	103	122	15	49	95
Modern10.N.12.P	33	84	100	12	66	71
Courier10.N.10.P	41	68	80	10	65	72

Beispiele

Die Datei "PRG LISTING A" mit einer Record-Länge von 120 kann mit

```
FPRINT PRG LISTING A (DEV P400 LEFT 2 FONT Antic15.N.8.P
```

auf dem P400 gedruckt werden.

Will man die Datei "BUCH LISTE A" möglichst platzsparend ausgeben, so sollte folgendes Kommando abgesetzt werden:

```
FPRINT BUCH LISTE A (DEV P400 FONT Bulletin22.N.6.P FCB 38D0 LI 118)
```

Auf diese Weise können bis zu 140 Zeichen pro Zeile mit einem linken Rand von 34 Millimetern und mit 3 Leerzeilen am oberen und 2 am unteren Rand einer jeden Seite gedruckt werden. (Die 3 Leerzeilen am oberen Rand sind über die implizite Setzung der Option "TOPMARGIN" vorgegeben.)

Soll mit einem linken Rand von 40 Zeichen und einem oberen und unteren Rand von je 5 Zeilen gedruckt werden, so muß folgendes Kommando eingegeben werden:

```
FPRINT BUCH LISTE A (DEV P400 LEFT 40 FONT Bulletin22.N.6.P FCB 38D0
                     TOP 5 LI 113)
```

Datenbanken A. Achilles

Am Rechenzentrum werden Datenbankanwendungen im Rahmen eines DFG-Forschungsvorhabens untersucht. Um den Bedarf an Plattenplatz, Rechenkapazität, Unterstützung usw. abschätzen und geeignete Projekte gezielt unterstützen zu können, fordern wir Sie auf, geplante Datenbankanwendungen umgehend anzumelden. Eine rechtzeitige Übersicht ermöglicht es uns, die Resourcen angemessen zu verteilen. Bitte richten Sie Ihre Wünsche möglichst bald an mich (Tel. 2488 oder schriftlich).

RUM-Lehre

Lehrveranstaltungen im Sommersemester 1985

1. Einführende Lehrveranstaltungen

320042	Programmieren in FORTRAN mi 13-15, Hörsaal: M6; Beginn: 10.4.1985	Reichel, K.
320057	Programmieren in FORTRAN mo 13-15, Hörsaal: M2; Beginn: 15.4.1985	Süselbeck, B.
320061	Programmieren in Pascal mo 13-15, Hörsaal: M4; Beginn: 15.4.1985	Achilles, A.
320076	Programmieren in Pascal mi 13-15, Hörsaal: M5; Beginn: 10.4.1985	Neukäter, B.
320080	Programmieren in PL/I (nichtnumerische Anwendungen) di 13-15, Hörsaal: M3; Beginn: 16.4.1985 Übungen: fr 13-15	Benduhn-Mertz, A.
320095	Programmieren in PL/I (numerische Anwendungen) di 13-15, Hörsaal: M4; Beginn: 16.4.1985 Übungen: fr 13-15 (M5)	Mertz, K.-B.
320100	Statistische Datenanalyse mit dem Programmsystem SPSS-X di 15-17, Hörsaal: M3; Beginn: 16.4.1985	Nienhaus, R./ Steinhausen, D.

2. Weiterführende Lehrveranstaltungen

320114	Programmieren in PL/I für Fortgeschrittene mi 9-11, Hörsaal: M6; Beginn: 10.4.1985	Kaspar, W.
320129	Datenstrukturen und Programmierverfahren in Pascal di 13.30-15, Hörsaal: M2; Beginn: 16.4.1985	Bosse, W.
320133	Einführung in das Betriebssystem UNIX mo 13-15, Hörsaal: M6; Beginn: 15.4.1985	Kisker, H.-W.

320148	Programmieren in C do 13-15, Hörsaal: M4; Beginn: 25.4.1985	Richter, G.
320152	SAS-Programmierung mi 9-11, Hörsaal: M4; Beginn: 17.4.1985	Zörkendörfer, S.
320167	Graphische Datenverarbeitung mi 15-17, Hörsaal: M5; Beginn: 17.4.1985	Sturm, E.
320171	Strukturierte Programmierung mo 13-15, Hörsaal: M5; Beginn: 22.4.1985	Ost, St.
320186	Einführung in das relationale Datenbanksystem VDN di 11-13, Hörsaal: M6; Beginn: 16.4.1985	Achilles, A.
(064611)	Einführung in INTERLISP mi 11-13, Hörsaal: M4; Beginn: 17.4.1985 Übungen: fr 13-15 Vorbesprechung: 12.4.1985 um 13 Uhr im M4	Göttsche, H.
(071280)	Forschungskolloquium "Computermusik" mi 8.30-10, Hörsaal: Bibliothek Rechenzentrum Beginn: 17.4.1985	Achilles, A./ Brockhoff, M.E./ Slaby, W.A.
320190	Kolloquium über Themen der Informatik fr 15-17, Hörsaal: M5	Held, W. und die wiss. Mitarbeiter des Rechenzentrums
320205	Anleitung zum Einsatz der EDV bei wissenschaftlichen Arbeiten	die wiss. Mitarbeiter des Rechenzentrums

FILL-AREA in FORTRAN

O. Tuchel

Die FORTRAN77-Version von RUM/GKS enthielt bisher die drei graphischen Grundelemente GPL (POLYLINE), GPM (POLYMARKER) und GTX (TEXT). Als weiteres Element wird ab sofort GFA (FILL_AREA) zur Verfügung gestellt, welches der Darstellung von Flächen dient und beim Plotter durch Schraffur realisiert wird.

Der Aufruf von FILL_AREA geschieht wie bei POLYLINE unter Angabe der Punktanzahl sowie der X- und Y-Koordinaten eines Polygonzuges. Das so beschriebene Gebiet - der erste und letzte Punkt werden automatisch verbunden - kann nun auf mehrere, vom Benutzer definierbare Arten schraffiert werden. Hierbei ist es auch erlaubt, daß sich die Umrandungslinie beliebig überschneidet. Nach der Definition des GKS liegt ein Punkt innerhalb eines zu schraffierenden Bereiches, wenn ein von dort bis ins Unendliche gehender Strahl eine ungerade Anzahl von Linien überquert.

Die Darstellungsweise der Fläche kann mit SET_FILL_AREA_INTERIOR_STYLE eingestellt werden:

```
CALL GSFAIS (STYLE)
```

STYLE kann folgende Werte annehmen:

```
0 - 'HOLLOW'   (nur die Umrandung wird gezeichnet)
1 - 'SOLID'    (vollständig einfärben)
2 - 'PATTERN'  (mit Muster füllen)
3 - 'HATCH'    (Schraffur ohne Umrandung)
```

'SOLID' wird durch eine enge Schraffur angedeutet, 'PATTERN' ist nicht implementiert und wirkt wie 'HOLLOW'. Wenn man den Stil 'HATCH' wählt, hat man mit SET_FILL_AREA_STYLE_INDEX einen weiteren Parameter für die Art der Schraffur.

```
CALL GSFASI (INDEX)
```

Für INDEX-Werte zwischen 1 und 12 wird die Schraffur auf dem Plotter mit wachsendem Wert immer dunkler. Mit Werten größer als 100 kann man den Abstand der Linien, ihre Orientierung und den Linientyp selbst bestimmen. Die Zehnerstelle gibt den Winkel zur positiven x-Achse an:

```
0 - 0 Grad (waagerecht),
1 - 45 Grad,
2 - 90 Grad (senkrecht),
3 - 135 Grad,
```

und die Einerstelle hat folgende Bedeutung:

1 - durchgezogene Linien,
2 - abwechselnd durchgezogen und gestrichelt,
3 - abwechselnd durchgezogen und punktiert,
4 - nur gestrichelte Linien,
5 - abwechselnd gestrichelt und punktiert,
6 - nur punktiert,
7 - durchgezogene Linien lang kariert,
8 - durchgezogene Linien quadratisch kariert.

Alle Stellen davor geben den Abstand der Schraffurlinien in 1/10 mm an. Somit ist der Wert 3021 zu lesen als 3,0 mm Abstand, senkrechte sowie durchgezogene Linien.

Hat man auf diese Weise allgemein festgelegt, wie Flächen aussehen sollen, so kann man nun eine spezielle Fläche zeichnen lassen:

```
CALL GFA (ANZAHL, XKOORD, YKOORD)
```

Zur Demonstration diene das folgende Beispielprogramm, mit dem zwölf Reihenhäuser gebaut werden:

```
* Westfälische Nikolaus-Reihenhäuser als Beispiel für FILL_AREA
      INTEGER     ANZAHL, I, INDEX
      PARAMETER  (ANZAHL = 12)
      INTEGER     STYLE  (ANZAHL) /
     &             4014     4.0 mm, 45 Grad, nur gestrichelt
     &             4012,    4.0 mm, 45 Grad, durchgezogen und gestrichelt
     &             4011,    4.0 mm, 45 Grad, durchgezogen
     &             2014,    2.0 mm, 45 Grad, nur gestrichelt
     &             2012,    2.0 mm, 45 Grad, durchgezogen und gestrichelt
     &             2011,    2.0 mm, 45 Grad, durchgezogen
     &             1014,    1.0 mm, 45 Grad, nur gestrichelt
     &             1012,    1.0 mm, 45 Grad, durchgezogen und gestrichelt
     &             1011,    1.0 mm, 45 Grad, durchgezogen
     &             1017,    1.0 mm, 45 Grad, lang kariert
     &             1018,    1.0 mm, 45 Grad, quadratisch kariert
     &              511 /   0.5 mm, 45 Grad, durchgezogen
      REAL X(11) /10., 10., 13., 16., 16., 10., 16., 10., 16., 10., 16/,
     &     Y(11) /10., 20., 25., 20., 10., 15., 20., 20., 15., 10., 10./
```

```
      CALL GSPLOT (0)
      CALL GSFAIS (3)
      DO 2 INDEX = 1, ANZAHL
        CALL GSFASI (STYLE(INDEX))
        CALL GSFACI
     &          (MOD(INDEX-1,3)+1)
        CALL GFA (11, X, Y)
        CALL GPL (11, X, Y)
        DO 1 I = 1, 11
          X(I) = X(I) + 6
1       CONTINUE
2     CONTINUE
      CALL GEPLOT

      END
```

```
START_PLOT
SET_FILL_AREA_INTERIOR_STYLE ('HATCH')

SET_FILL_AREA_STYLE_INDEX
SET_FILL_AREA_COLOUR_INDEX

FILL_AREA
POLYLINE




END_PLOT
```

In diesem Beispiel sind die Elemente von STYLE so definiert, daß der Aufruf von GSFASI auch mit dem Argument INDEX direkt erfolgen könnte. Mit dem Aufruf von SET_FILL_AREA_COLOUR_INDEX

```
CALL GSFACI (MOD (INDEX - 1, 3) + 1)
```

werden die Farben schwarz (1), rot (2) und blau (3) zyklisch gesetzt. Diese sind natürlich wie immer bei GKS unabhängig von den Farben der anderen Grundelemente.

RUM-Tutorial

Einführung in das Betriebssystem UTS-Unix

H.-W. Kisker

Am Rechenzentrum der Universität Münster steht seit Mitte Februar 1985 das Betriebssystem UTS (Universal Timesharing System) der Firma Amdahl auf einem IBM-Großrechner zur Verfügung, das aus Mitteln der DFG finanziert wird. UTS (Release 2.3) ist eine Anpassung des weitverbreiteten Betriebssystems Unix (Version 7) an die VM-Umgebung.

Ich möchte Sie mit diesem Artikel in das Betriebssystem einführen und Ihnen die grundsätzlichen Konzepte von Unix vorstellen. Aufgabe eines Betriebssystems ist es, dem Benutzer Zugang zu den Betriebsmitteln eines Rechners zu verschaffen. Ein gutes Betriebssystem wird dabei den normalen Benutzer weitgehend vom Umgang mit rechnerinternen Einzelheiten entlasten. Es wird ihm stattdessen an seiner Oberfläche ein einfaches Szenario vorgaukeln, das es ihm gestattet, auch komplexe Zusammenhänge leicht zu überschauen. In diesem Sinne ist es zu verstehen, wenn Kaare Christian in seinem Buch "The Unix Operating System" (S. 239) schreibt:

> The UNIX System supports two powerful illusions: that the file system has "places" and that processes have "life".

Diese Illusionen gilt es im folgenden zu verstärken. Dabei hat das ausgebreitete Szenario einiges zu bieten. Da erzeugen Väter Töchter und warten dann auf ihr Ende. Und da ist die Unix-Welt bevölkert von Dämonen, die hier aber gute Geister sind, und von Zombies, die dem Systemverwalter das Leben schwer machen.

Unix wird oft als Kugel dargestellt. Im Inneren befindet sich der Kern, der die Benutzer von der Hardware isoliert. Darum herum gruppieren sich die Prozesse, in deren Rahmen die Aktivitäten unter Unix ablaufen. Die Prozesse, deren Aufgabe es ist, die Kommunikation mit den Benutzern durchzuführen, werden als äußerste Schale ('shell') angesehen. Jede Nutzung eines Betriebsmittels durch einen Prozeß ist nur durch die Vermittlung des Kerns möglich. Insbesondere gilt dies auch für einen Zugriff auf das File-System, das den Platz und die Ordnung für die zu verwaltenden Datenbestände darstellt. Dieses File-System wird mit Recht häufig als die Grundlage des Betriebssystems bezeichnet. Böse Zungen behaupten sogar, Unix sei insgesamt nicht viel mehr als ein Ein/Ausgabe-Verwaltungssystem.

Der Begriff des Files im Sinne von Unix ist dabei sehr weit gefaßt. Darunter fallen neben den üblichen Datenbeständen auf einer Magnetplatte auch Daten, auf die durch Geräte-Ein/Ausgabe oder Interprozeßkommunikation zugegriffen wird. Für den Benutzer bedeutet dies, daß ihm für alle Arten der Ein/Ausgabe gleiche Mechanismen zur Verfügung stehen.

Ich werde Ihnen jetzt die wesentlichen Leistungen vorstellen, die Unix Ihnen anbietet. Dabei werde ich die Mechanismen erläutern und Ihnen gleichzeitig die 'shell'-Notation angeben, durch die Sie diese Leistung auf Kommandoebene anfordern können.

Üblicherweise werden heute zwei Kommandooberflächen angeboten:

- die sogenannte 'Bourne shell' und
- die 'C shell'.

Beide stehen wahlweise unter UTS zur Verfügung. Ich werde in diesem Artikel die Notation der 'Bourne shell' verwenden.

Das File-System

Um die Erfüllung seiner Aufgaben zu gewährleisten, verwaltet das Betriebssystem zu jedem File eine Reihe von Angaben. Hierzu gehören z.B.

- der File-Name, durch den das File identifiziert wird,
- der Besitzer des Files, i.a. ist dies der das File anlegende Benutzer,
- die Projektgruppe, die das File bearbeitet,
- der 'mode' des Files, durch den die Zugriffsberechtigung geregelt wird und
- der Typ des Files.

Vom strukturellen Standpunkt aus gesehen, ist der File-Typ die wesentliche Angabe, durch die der Aufbau des File-Systems festgelegt wird. Es gibt drei File-Typen:

- ordinary files,
- directories und
- special files.

Die 'ordinary files' sind die eigentlichen "Arbeitstiere" des File-Systems. Sie repräsentieren Datenbestände, die innerhalb des Unix-Systems abgelegt werden. Dabei ist anzumerken, daß für das Betriebssystem ein Datenbestand einfach aus einer Folge von Zeichen besteht. Eine weitere innere Struktur wird von Unix nicht gefordert. Das bleibt den das File bearbeitenden Anwenderprogrammen vorbehalten. So unterstützt z.B. ein Editor selbstverständlich eine Einteilung in Zeilen, und der Systemlader liest und schreibt seine Daten natürlich in einem wohldefinierten Object-Code-Format.

Die 'directories' sind die "Schaltstellen" des File-Systems. Unix vermittelt die Illusion, daß eine 'directory' sozusagen ein Behälter ist, der keinen Datenbestand, sondern ganze Files - ja ganze File-Systeme - enthält. Dabei kann ein enthaltenes File von jedem der unterstützten File-Typen sein, also auch wieder eine 'directory'.

Der verbleibende File-Typ 'special file' dient dem Zugriff auf angeschlossene Geräte wie Drucker, Bildschirme, Magnetplatten usw. Hinter den 'special

files' versteckt sich stets ein Gerätetreiber, der die Bedienung und Parametrierung des ihm zugeordneten Geräts übernimmt und der die Abbildung der gerätespezifischen Datenrepräsentation auf die Unix-spezifische Datenschnittstelle vornimmt. Daten, die in einen solchen 'special file' geschrieben werden, werden also nicht im Unix-System unter dem Namen des 'special files' abgelegt, sondern durch das File hindurch auf das zugeordnete Gerät ausgegeben. Entsprechendes gilt für die Eingabe.

Angemerkt sei noch, daß ab Unix System III weitere File-Typen hinzukommen, z.B. die sogenannten 'fifos', die einen Kommunikationsweg zwischen beliebigen Prozessen zur Verfügung stellen.

Wieder vom strukturellen Gesichtspunkt aus gesehen, ist die 'directory' der wesentliche File-Typ, durch den der Aufbau des File-Systems festgelegt wird. Die Struktur ergibt sich einfach aus der Möglichkeit, daß eine 'directory' wieder 'directories' enthalten kann. Das ganze File-System beginnt in der Anfangs- oder Wurzel-'directory' 'root'. Diese enthält neben 'ordinary files' wieder 'directories' (bin, etc, usr usw.). Jede dieser 'directories' enthält neben 'ordinary files' wieder 'directories' usw. usw.

Die sich ergebende Struktur ist wohl offensichtlich. Es entsteht ein Baum, eine Hierarchie. Diese Hierarchie wird im Unix-System konsequent genutzt, um die Datenbestände sachgerecht zu ordnen. Unter anderem werden auf diese Weise auch die Files der einzelnen Benutzer voneinander isoliert. Jedem Anwender wird eine 'directory' zugewiesen, die seiner exklusiven Nutzung vorbehalten ist. Dies ist die sogenannte 'home directory'. Da jeder Benutzer auch wieder beliebig eigene 'subdirectories' anlegen kann, ist ihm das Werkzeug an die Hand gegeben, um seine Datenbestände entsprechend seinem Problem zu ordnen.

Ein Problem handelt man sich jedoch ein mit dem Konzept des hierarchischen File-Systems, nämlich das des eindeutigen Ansprechens eines bestimmten Files. Zwar verfügt jedes File über einen Namen, und dieser muß innerhalb der direkten Vorgänger-'directory' eindeutig sein, aber zwei Files, die sich in verschiedenen 'directories' befinden, dürfen durchaus gleiche Namen haben. Eine kanonische Lösung dieses Problems bietet sich an. Zu jedem beliebigen File gibt es stets einen Weg, der von der Wurzel-'directory' zu diesem File führt. Dieser Weg ist eindeutig, wenn man nur vorwärts gehen darf. Reiht man nun, bei 'root' beginnend, die Namen aller 'directories' dieses Weges aneinander, so erhält man mit dem Namen des Files zusammen eine eindeutige Identifikation des Files. In der 'shell'-Notation werden diese Namen durch das Zeichen "/" voneinander getrennt, und die Angabe von 'root' wird unterdrückt. Die entstehende Zeichenkette wird als 'path name' bezeichnet.

Beispiel:

/usr/urz/urz10/cms/memo

Die Nachteile dieser Methode zur Identifikation sind offensichtlich, sie liegen in der Länge der entstehenden Namen. Eine Abhilfe wird durch einen einfachen Trick erreicht. Es wird dem Benutzer die Möglichkeit zur Verfügung gestellt, eine andere 'directory' als 'root' zum Startpunkt für Pfadnamen zu definieren. Diese 'directory' wird als 'current directory' bezeichnet. Sie

kann dynamisch vom Anwender geändert werden und so zu jedem Zeitpunkt den Bedürfnissen angepaßt werden.

Beispiel:

Ist /usr/urz/urz10 als 'current directory' festgesetzt, so läßt sich der obige File ansprechen als:

cms/memo

Welcher der beiden möglichen Startpunkte gemeint ist, wird in der 'shell'-Notation festgelegt durch das erste Zeichen des Pfadnamens. Ist dieses das Zeichen "/", so hat der 'path name' seinen Ursprung in der 'root directory'. Bei jedem anderen Zeichen wird die 'current directory' als Startpunkt benutzt.

Integrität des File-Systems

In einem Mehrbenutzer-Betriebssystem ist die Integrität des File-Systems stets ein kritischer Punkt. Unix unterstützt hier flexible Schutzmechanismen auf der Basis des einzelnen Files. Das Werkzeug hierzu steht jedem Benutzer zur Verfügung. Jeder kann den Zugang zu seinen Datenbeständen selbst regeln. Er kann den externen Zugriff vollständig verwehren, er kann aber auch einzelne Files für die Nutzung durch andere Anwender freigeben. Der Schutz wird nach zwei beliebig kombinierbaren Kriterien festgelegt. Das erste Kriterium ist die für das File zulässige Operation. Man kann festlegen, ob ein File

- gelesen,
- geschrieben oder
- ausgeführt

werden kann. Letzteres ist natürlich nur für Files sinnvoll, die ausführbaren Code oder Kommandolisten enthalten. Die Zulässigkeit einer Operation kann für drei Benutzerkreise getrennt festgelegt werden. Dabei umfaßt

- der erste Kreis nur den Besitzer des Files selbst,
- der zweite Kreis alle Mitglieder einer über einen Namen spezifizierten Projektgruppe und
- der letzte Kreis schließlich alle übrigen Unix-Benutzer.

Man kann sagen, daß sich dieser Schutzmechanismus als wirkungsvoll und hinreichend flexibel erwiesen hat, um auf die verschiedensten Anforderungen zu reagieren. Beachten Sie bitte auch, daß der File-Schutz für alle Typen von Files gilt, insbesondere auch für Geräte.

Prozesse

Alle Aktivitäten unter Unix laufen im Rahmen von Prozessen ab. Bringt der Benutzer ein Programm zur Ausführung, so wird hierfür stets ein eigener, programmspezifischer Prozeß erzeugt: für die Übersetzung eines C-Programms wird ein C-Prozeß erzeugt, will man Texte edieren, so wird ein Editor-Prozeß zum Ablauf gebracht, und die Kommandoeingaben nimmt der 'shell'-Prozeß entgegen.

Beim Arbeiten unter Unix wird der Benutzer also ständig neue Prozesse erzeugen. Dem Anfänger wird dies nicht bewußt werden, dem etwas Fortgeschrittenen jedoch eröffnen sich durch das Verständnis des Zusammenspiels der Prozesse eine Reihe von komfortablen Möglichkeiten. Die Prozeßerzeugungsmechanismen sind relativ simpel, und ich möchte sie am Beispiel des Ablaufs bei einer Kommandoeingabe verdeutlichen.

Für das Folgende wollen wir einen Prozeß als aus zwei Teilen bestehend ansehen, aus dem Programm und aus der Umgebung.

process := (program,environment)

Das Programm steht dabei sowohl für den geladenen ausführbaren Code als auch für die zu bearbeitenden Datenbereiche - die Variablen. Die Umgebung besteht aus einer Reihe von Angaben, die unabhängig sind von dem gerade ausgeführten Programm (z.B. Benutzerkennung, Gruppenzugehörigkeit, geöffnete Files usw.).

Bei der Prozeßerzeugung spielen vier einfache Systemaufrufe eine Rolle, die kurz erläutert werden sollen:

- fork,
- execute,
- sleep und
- exit.

Der wichtigste davon ist wohl der 'fork'-Aufruf, denn er stellt die einzige Möglichkeit dar, um neue Prozesse zu erzeugen. Ein Prozeß, der einen solchen Aufruf ausführt, verdoppelt sich. Statt des einen Prozesses laufen anschließend zwei von einander unabhängige Prozesse, die sich konkurrierend um den Zentralprozesser bewerben wie andere Prozesse auch. Sie sind "völlig" identisch (gleicher Code, gleiche Belegung der Variablen, gleiche Umgebung) bis auf eine Kleinigkeit: der eine Prozeß ist als "Vaterprozeß" der andere als "Tochterprozeß" ausgewiesen. Dies ist nicht viel mehr als ein Bit, das in dem einem Prozeß gesetzt ist und in dem anderen nicht.

Der 'execute'-Aufruf dient dazu, das "Gesicht" eines Prozesses zu wandeln. Durch ihn wird der Programmteil des Prozesses ausgewechselt. Ein Argument des Aufrufs ist der Name eines Files, der ladbaren Code enthält. Dieser Code verdrängt den alten Programmteil, und der Prozeß beginnt mit der Ausführung des neuen Programms. Das alte Programm verschwindet vollständig, d.h. es gibt anschließend keinen Prozeß mehr, der das alte Programm ausführt.

Die weiteren Aufrufe sind trivial. Durch einen 'sleep'-Aufruf wird der Prozeß in einen schlafenden Zustand versetzt. Er bewirbt sich anschließend nicht mehr um den Zentralprozessor. Aus diesem Zustand kann er nur durch bestimmte Signale, die als Reaktion auf externe Ereignisse ausgelöst werden, aufgeweckt werden.

Mit einem 'exit'-Aufruf schließlich beendet sich ein Prozeß. Er "stirbt" und ist anschließend nicht mehr existent.

Abläufe bei einer Kommandoeingabe

Damit haben wir uns das Rüstzeug verschafft, um das bei einer Kommandoeingabe ablaufende Szenario zu beschreiben. Eines ist (hoffentlich) aus dem Vorangegangenen klar geworden: um ein Programm zum Ablauf zu bringen, muß ein Prozeß erzeugt werden; um einen Prozeß zu erzeugen, benötigt man wenigstens einen laufenden Prozeß. Für jeden am Unix-System arbeitenden Anwender gibt es nun einen Prozeß, der für ihn arbeitet - das ist der 'shell'-Prozeß. Seine Aufgabe ist es, Kommandos vom Terminal entgegenzunehmen und die angeforderten Programme zur Ausführung zu bringen. Die Grundeinheit der 'shell'-Sprache ist dabei das einfache Kommando, das aus einer Folge von einem oder mehreren durch Leerzeichen getrennten Strings besteht.

```
command := name argument ...
```

Der erste String wird als Kommando-Identifikation - konkret als Name eines ausführbaren Files - aufgefaßt, die weiteren als Argumente, die dem auszuführenden Programm zu übergeben sind.

Beispiele:[1]

```
ls
cd /usr/urz/urz10
pwd
```

---

[1] Zusammenstellung der hier erwähnten UNIX-Kommandos:

ls	Auflisten der in der 'current directory' enthaltenen Files
cd	Einstellen der 'current directory'
pwd	Ausgeben des Namens der 'current directory'
cc	Aufruf des C-Compilers
pr	Aufbereitung einer Druckausgabe
opr	Ausgabe eines Files auf einem realen Drucker
sort	Sortierung der Eingabedaten

Erkennt nun der 'shell'-Prozeß den Abschluß einer Kommandoeingabe, so führt er ein 'fork' aus. Für den Benutzer laufen dann zwei 'shell'-Prozesse, die sich allerdings sehr unterschiedlich verhalten.

Der Vaterprozeß führt einen 'sleep'-Aufruf aus und geht dadurch in den schlafenden Zustand über. Er stellt damit bis auf weiteres seine Aktivitäten ein.

Der Tochterprozeß dagegen führt einen 'execute'-Aufruf aus auf den im Kommando enthaltenen Programmnamen. Damit ist der Tochterprozeß kein 'shell'-Prozeß mehr. War das Kommando z.B. der Aufruf des C-Compilers, so ist der Tochterprozeß jetzt ein C-Prozeß geworden. Der Prozeß beginnt dann das neue Programm auszuführen. Irgendwann hat das Programm seine Aufgabe erfüllt, führt einen 'exit'-Aufruf durch und beendet damit die Existenz des Tochterprozesses. Durch dieses Ereignis wird ein Signal ausgelöst, das den Vaterprozeß aufweckt. Dieser nimmt seine Arbeit wieder auf und fordert das nächste Kommando vom Terminal an.

Asynchrone Prozeßausführung

Der beschriebene Ablauf benutzt zwar die Hilfsmittel eines Multi-Tasking-Betriebssystems, die gleiche Leistung ließe sich jedoch auch mit einem Single-Tasking-Betriebssystem erbringen. Eine kleine Modifikation im Ablauf ermöglicht es dem Benutzer jedoch, sich die Leistungen des Multi-Tasking nutzbar zu machen:

Die Änderung liegt im Verhalten des Vaterprozesses. Dieser ist nämlich in keiner Weise daran gebunden, nach dem 'fork'- einen 'sleep'-Aufruf abzugeben. Er kann stattdessen durchaus seine normale Arbeit fortsetzen. In unserem Beispiel kann der 'shell'-Prozeß nach dem Start eines Programms also weitere Kommandoeingaben bearbeiten. Auf Kommandoebene kann der Benutzer dies spezifizieren, indem er das Kommando mit dem Zeichen "&" abschließt.

Beispiel:

```
cc test.c &
```

Der Anwender kann auf diese Weise beliebig viele Prozesse für sich parallel zu den Kommandoeingaben arbeiten lassen. Die Prozesse werden dann im Hintergrund abgearbeitet.

Standard-Ein/Ausgabe

Ein weiterer Unix-typischer Punkt bedarf noch der Erläuterung: die sogenannte Standard-Ein/Ausgabe. Einem Prozeß werden üblicherweise bei seiner Erzeugung zwei logische Datenwege zur Verfügung gestellt - für lesenden

Zugriff die Standard-Eingabe und für schreibenden Zugriff die Standard-Ausgabe. Ein Programm, das diese logischen Kanäle nutzt, legt weder die Art des Files noch gar das File selbst fest. Dies ist vielmehr Aufgabe des den Prozeß erzeugenden Vaterprozesses. Bei Prozessen, die als Folge einer Kommandoeingabe entstehen, ist dies also eine der Leistungen der 'shell'. Programme, die nur je ein File für Eingabe bzw. Ausgabe bearbeiten, können durch dieses Konzept sehr flexibel aufgebaut werden. Bei der Programmierung legt man nur fest, daß man lesen bzw. schreiben möchte. Quelle bzw. Ziel des Zugriffs werden jedoch erst bei der Ausführung auf Kommandoebene festgelegt.

Normalerweise verbindet die 'shell' für einen von ihr erzeugten Prozeß die Standard-Eingabe mit der Terminaleingabe und die Standard-Ausgabe mit der Terminalausgabe. Möchte der Benutzer dies ändern, so kann er das Kommando um die Angabe der Files erweitern, auf die er die Standard-Ein/Ausgabe umdirigieren möchte. Dem Standard-Eingabe-File ist dabei das Zeichen "<" voranzustellen und dem Standard-Ausgabe File das Zeichen ">".

Beispiele:

```
ls > liste
sort < daten > /dev/tty02
```

Auf diese Art und Weise kann auf beliebige Typen von Files umdirigiert werden:

- auf einen Datenbestand des File-Systems,
- auf ein Gerät wie Termial oder Drucker und
- auf einen Kommunikationskanal zu einem anderen Prozeß.

Solch ein Kommunikationskanal wird als 'pipe' bezeichnet. 'shell' bietet eine Notation an, um zwei Prozesse gleichzeitig zu starten, die über eine solche 'pipe' Daten austauschen. Dabei wird die Standard-Ausgabe des einen Prozesses der Eingabeseite der 'pipe' zugeordnet und die Standard-Eingabe des anderen Prozesses der Ausgabeseite.

Insgesamt erscheint es so, als ob die Standard-Ausgabe mit der Standard-Eingabe verbunden sei. Was der eine Prozeß schreibt, wird vom anderen Prozeß gelesen. Auf der Kommandoebene sind beide Kommandos hintereinander zu schreiben und durch das Zeichen "|" zu trennen.

Beispiele:

```
ls | opr

ls | sort | pr | opr
```

Die Notation ist also nicht auf zwei Prozesse beschränkt, sondern es können beliebig viele Prozesse verkettet werden. Es entsteht so ein Datenstrom, der von mehreren Programmen sozusagen gefiltert wird. Aus diesem Grund wird ein Programm, das von der Standard-Eingabe liest und auf die Standard-Ausgabe schreibt, als 'filter' bezeichnet.

Viele Dienstleistungsprogramme benutzen ausschließlich Standard-Ein/Ausgabe. Daneben stehen für Anwenderprogramme natürlich die üblichen Aufrufe für Datenzugriffe (wie open, close, read, write usw.) zur Verfügung.

Ausblick

Wenn Sie in den letzten Monaten die einschlägige Literatur verfolgt haben, wird Ihnen eine gewisse Unsicherheit in der Beurteilung von Unix nicht entgangen sein. Ich zitiere den einleitenden Satz eines hierfür typischen Zeitschriftenartikels:

> "'Unix ist ein Standard' und 'Unix wird nie ein Standard werden', diese Aussagen spiegeln die konträren Anschauungen von Unix-Fachleuten und solchen, die sich dafür halten, wider."
>
> (Markt & Technik, Heft 5, 1. Febr. 1985, S. 71)

Bei dieser Diskussion - die in kontroverser Form auch in unserem Hause geführt wird - ist jedoch zu beachten, daß es ausschließlich um die zukünftige Marktverbreitung von Unix geht. Die Leistungsfähigkeit des Betriebssystems ist dagegen weitgehend unumstritten.

Die jetzige Bereitstellung von UTS auf einem Großrechner ist Teil eines Forschungsvorhabens des Rechenzentrums, das von der DFG gefördert wird. Gegenstand der Untersuchung ist die Eignung von Unix als Standard-Betriebssystem in einem Hochschulrechenzentrum mit seinem weit gestreuten Aufgabenspektrum. Die Beurteilungskriterien sind dabei in zwei Richtungen festzulegen. Zum einen ist zu überprüfen, ob Unix und die unter Unix ablauffähige Software eine geeignete Alternative zu dem Angebot der bei uns verfügbaren IBM-Betriebssysteme sein kann. Zum anderen halten wir es für in einem hohen Maße wünschenswert, bei der zunehmenden Verteilung von Rechnerkapazität durch die Verbreitung von Mikro-Rechnern eine einheitliche Benutzeroberfläche für alle Leistungsklassen von Rechnern zur Verfügung zu haben. Die Kompatibilität der verschiedenen Unix-Portierungen wird somit Gegenstand unserer Untersuchung sein.

Ein weiterer wesentlicher Gesichtspunkt ist die Akzeptanz durch die Benutzer des Rechenzentrums. Wir werden Sie zu gegebener Zeit um Äußerung Ihrer Meinung bitten. Für Anregungen und Kritik werden Sie in uns zu jedem Zeitpunkt interessierte Gesprächspartner finden.

Kompatibilität zum IBM-PC

Hans-Werner Kisker

Seit nunmehr drei Jahren tritt die Firma IBM als Hersteller von Personal Computern auf. Das von ihr angebotene Grundgerät - der IBM-PC - ist mit einer 16-Bit-Zentraleinheit und einem 8-Bit-Datenbus ausgestattet. Als Betriebssystem ist das Single-User/Single-Task-Betriebssystem PC-DOS vorgesehen. Die Leistungsfähigkeit des IBM-PCs ist also nur im mittleren Bereich des denkbaren Leistungsspektrums angesiedelt. Trotzdem hat dieser Rechner zu einer weitgehenden Nivellierung des Mikro-Rechner-Marktes geführt. Viele Firmen, die heute neue Personal Computer vorstellen, legen Wert darauf, daß ihre Produkte "kompatibel" zum IBM-PC sind. Diesem Trend Rechnung tragend, verlagern auch die großen Software-Firmen den Schwerpunkt ihrer Aktivitäten auf die Entwicklung von auf dem IBM-PC ablauffähigen Programmen. So läßt das heute für diesen Rechner zur Verfügung stehende Angebot kaum noch Wünsche offen. Von vielen Fachleuten wird diese Entwicklung mit einem gehörigen Schuß Skepsis betrachtet. Nach meiner Meinung ist eine gewisse Standardisierung zwar wünschenswert, sie erfolgt durch den IBM-PC jedoch auf zu niedrigem Niveau. Dies ist jedoch ein Thema, das an dieser Stelle nicht weiter diskutiert werden soll. Mein Artikel will vielmehr den etwas schillernden Begriff "Kompatibilität" konkretisieren. Dazu ist es erforderlich, den konzeptionellen Aufbau des IBM-PCs näher zu beschreiben. Ich möchte zu diesem Zweck drei aufeinander aufbauende Ebenen unterscheiden:

- die Hardware-Ebene,
- die ROM-BIOS-Ebene und
- die Betriebssystemebene (i.a. PC-DOS).

Alle diese Ebenen bieten dem Anwender eine Reihe von Leistungen an, die er nach seinem Bedarf nutzen kann. Es ist sinnvoll, auch die Kompatibilität entsprechend einzuteilen und für die einzelnen Ebenen getrennt zu beurteilen.

Volle Kompatibilität wird natürlich nur durch einen identischen Nachbau aller einzelnen Ebenen erreicht, was zu einer wettbewerbswidrigen Nachahmung führt. Diese läßt sich jedoch durchaus vermeiden, wenn man von einer pragmatischen Definition der Kompatibilität ausgeht, die für den Anwender i.a. vollkommen ausreichend ist.

Zwei Rechner sind kompatibel zu nennen, wenn beliebige Software, die für den einen Rechner entwickelt wurde, ohne Änderung auch auf dem anderen Rechner ablauffähig ist. Um nun diese Art der Kompatibilität zu gewährleisten, ist es völlig ausreichend, nur die Leistungen der verschiedenen Ebenen nachzubilden; die unzulässige Kopie kann vermieden werden. Dies ist ein Weg, den die meisten Anbieter kompatibler Geräte beschritten haben. Jede Leistung einer Ebene wird durch bestimmte Parametersetzungen angefordert. Damit bedeutet Überprüfung der Kompatibilität also Überprüfung des Verhaltens des Rechners bei entspechender Parametrierung der einzelnen Ebenen.

Die Hardware-Ebene

Die Leistungen der Hardware-Ebene werden durch die verschiedenen Bausteine erbracht, aus denen der Rechner aufgebaut ist. Die oben angesprochenen Parameter stellen sich in dieser Ebene also bausteinspezifisch dar:

- für die Zentraleinheit ist hierunter der Instruktionssatz zu verstehen,
- beim Speicher ist der Adreßbereich und die Einteilung in ROM- und RAM-Bereiche zu beachten und
- für die Ein/Ausgabe-Bausteine sind neben den Adressen vor allem die Festlegungen der Steuerworte für Kontroll- und Statusregister wichtig.

Häufig wird die Kompatibilität hier dadurch erreicht, daß zumindest für die hochintegrierten Komponenten die gleichen Bausteintypen gewählt werden wie für den IBM-PC. Insbesondere gilt dies für

- die Zentraleinheit 8088 (4.77 MHz),
- den Interrupt-Controller 8259,
- den DMA-Controller 8237,
- den Video-Controller 6845,
- den Floppy-Controller µPD 765 und
- den seriellen Baustein 8250.

Nach meinen Beobachtungen vermeiden es die meisten auf dem Markt angebotenen Programme, die Hardware-Ebene direkt zu nutzen. Dies ist ein Prinzip, das auch für eigene Entwicklungen dringend zu empfehlen ist. Allerdings gibt es einige Ausnahmen von dieser Regel. So ist es manchmal zu beobachten, daß ein Programm, das eine komfortable visuelle Oberfläche anbieten möchte, direkt in den Bildwiederholspeicher schreibt. Da jedoch das ROM-BIOS ausreichende Funktionen zur Bildschirmbearbeitung anbietet, halte ich diese Nutzung der Hardware-Ebene in den meisten Fällen für nicht erforderlich.

Anders ist dagegen das direkte Ansprechen des seriellen Bausteins zu beurteilen. Für diese Komponente bieten die oberen Ebenen nur eine ungepufferte und nicht interrupt-gesteuerte Schnittstelle an. Um auch bei höheren Übertragungsraten einen zuverlässigen Betrieb garantieren zu können, ist diese nicht ausreichend. Bei bestimmten Anwendungen ist es also unbedingt erforderlich, eine eigene 'interrupt service routine' zu etablieren. Dies ist jedoch nur über die direkte Programmierung des Interrupt-Controllers und des seriellen Bausteins möglich.

Die ROM-BIOS-Ebene

Zum Lieferumfang des IBM-PCs gehört eine im ROM abgelegte Software, die für die meisten Hardware-Komponenten Basis-Schnittstellen anbietet: das sogenannte ROM-BIOS. Die Leistungen des ROM-BIOS sind für den Anwender über Software-Interrupts zugänglich. Die Nummern dieser Interrupts und die jeweils

erforderlichen Registerwerte bilden die zu überprüfenden Parameter dieser Ebene. Im einzelnen stehen Funktionen zur Verfügung für

- das programmierte Urladen,
- den Selbsttest,
- die serielle Schnittstelle,
- Tastatureingaben,
- Disketten-Zugriffe,
- Druckerausgaben,
- Bildschirmsteuerung,
- die Abfrage der Systemkonfiguration,
- den ladbaren Zeichensatz der Grafik-Karte,
- die Software-Uhr und
- Hardcopy des Bildschirms.

Durch das ROM-BIOS werden eine Reihe von Leistungen erbracht, die auf der oberen Betriebssystemebene nicht zur Verfügung stehen: z.B. 'window scrolling', nicht standardgemäße Formatierung von Disketten usw. Aus diesem Grund wird diese Ebene häufig von Systemprogrammierern direkt genutzt, und deshalb ist die Quelle von Inkompatibilitäten meistens hier zu finden. Um die Überprüfung der Kompatibilität zu erleichtern, habe ich ein Programm mit Namen PCTEST geschrieben. Dieses setzt direkt auf der BIOS-Ebene auf und bietet für die Komponenten Bildschirm, Tastatur, Drucker und serielle Schnittstelle je einen Bedienungsbildschirm an. Der Anwender kann so die wichtigsten Leistungen dieser Schnittstellen über Funktionstasten abrufen und deren Wirkung visuell überwachen.

Die Betriebssystemebene

Für den IBM-PC stehen mehrere Betriebssysteme zur Verfügung: z.B. PC-DOS, UCSD-PASCAL oder CP/M-86. Standardbetriebssystem ist PC-DOS, unter dem auch der größte Teil der auf dem Markt erhältlichen Software ablauffähig ist. PC-DOS besteht aus zwei Teilen:

- dem Modul IBMBIO.COM, der die Gerätetreiber enthält, und
- dem Modul IBMDOS.COM, der den Betriebssystemkern darstellt.

PC-DOS ist eine Anpassung des Betriebssystems MS-DOS der Firma Microsoft an den IBM-PC. Der Modul IBMDOS dürfte weitgehend identisch sein mit entsprechenden Moduln anderer MS-DOS-Implementierungen. So wird man i.a. davon ausgehen können, daß auf dieser Ebene nur wenige Inkompatibilitäten versteckt sind. Diese Ansicht wird auch durch die Erfahrung gestützt, daß Programme, die sich nur des Betriebssystems bedienen (wie z.B. Compiler), ohne Probleme auf den verschiedensten MS-DOS-Rechnern ablauffähig sind.

Die PC-DOS-Leistungen stehen über System-Calls zur Verfügung. Die Parameter dieser System-Calls sind die zu überprüfenden Parameter dieser Ebene. Für eine genaue Beschreibung dieser Leistungen sei auf die entsprechenden Handbücher verwiesen.

Ein Beispiel

Die verschiedenen Stufen des Leistungsangebots möchte ich noch am Beispiel der Bildschirmausgabe verdeutlichen. Aufgabe sei es, ein bestimmtes Zeichen - z.B. einen Stern - an der momentanen Cursor-Position auf den Bildschirm zu bringen. Auf Betriebssystemebene steht hierzu der System-Call 2 (Bildschirmausgabe) zur Verfügung. Es ist zu empfehlen, auf dieser Ebene grundsätzlich höhere Programmiersprachen zu verwenden. In C wird der obige System-Call indirekt durch die Funktion 'putchar' abgegeben. Man kann also schreiben:

```
putchar ('*');     /* Stern ausgeben   */
```

Will der Anwender die ROM-BIOS-Ebene nutzen, so muß er das Register AH mit der Wert 14 und das Register AL mit dem auszugebenden Zeichen laden und dann einen Software-Interrupt 16 auslösen. In Assembler-Notation kann man schreiben:

```
MOV AH,14          ; Bildschirmausgabe
MOV AL,'*'         ; auszugebendes Zeichen
INT 16             ; BIOS-Funktion: video interrupt
```

Schließlich besteht die Möglichkeit, direkt in den Bildwiederholspeicher zu schreiben. Dieser befindet sich für den monochromen Adapter an der Adresse B000H und hat eine Größe von 4000 Bytes. Bytes mit gerader Adresse enthalten den Zeichencode, das jeweils folgende Byte mit ungerader Adresse das zugehörige Video-Attribut. Das folgende Assembler-Programmstück schreibt den Stern in Zeile 11 Spalte 41:

```
MOV AX,0B000H      ; Extrasegment mit der Adresse
MOV ES,AX          ; des Video-RAM's laden.
MOV DI,1600        ; Zeile 11 positionieren
MOV ES:80[DI],'*'  ; Stern in Spalte 41 ausgeben
```

Weitere Gesichtspunkte

Neben der reinen Überprüfung der Software-Verträglichkeit sind natürlich noch einige weitere Gesichtspunkte zu beachten. Besondere Bedeutung kommt z.B. der Austauschbarkeit von Disketten zu. Nur wenn diese gewährleistet ist, können die verschiedenen Rechner wirklich gleichberechtigt und im Wechsel als Arbeitsgeräte genutzt werden. Ähnliches gilt für die Auslegung der Tastatur und die Auflösung des Bildschirms, insbesondere des Grafik-Bildschirms. Weiterhin verfügt der IBM-PC über ein nach außen geführtes Bus-System, über das Erweiterungskarten integriert werden können. Normalerweise bieten kompatible Rechner ebenfalls diese Erweiterungsmöglichkeit an. In diesem Zusammenhang sei auch noch auf eine unerfreuliche Konsequenz der Kompatibilität hingewiesen. Wie bereits erwähnt, sind die Leistungsmerkmale des IBM-PCs im mittleren Bereich des Leistungsspektrums angesiedelt. Dies veranlaßt Firmen häufig, bestimmte Vorgaben zu ignorieren und ihren Rechner mit leistungsfähigerern Komponenten auszustatten: z.B. mit Disketten größerer Kapazität, einer CPU mit

höherer Taktrate oder mit einem Bildschirm feinerer Auflösung. Alle diese Maßnamen sind durchaus sinnvoll und wünschenswert, in bezug auf die Kompatibilität schaffen sie jedoch leider eine Reihe von Problemen.

Vorgehen bei der Kompatibilitätsüberprüfung

Ich habe in den letzten Wochen eine Reihe von Mikro-Rechnern, die uns von verschiedenen Firmen zur Verfügung gestellt wurden, auf Kompatibilität zum IBM-PC hin untersucht. Diese Untersuchung gliederte sich in drei Teile.

Begonnen wurde stets mit einem systematischen Test der wichtigsten Komponenten der Hardware- und ROM-BIOS-Ebene, soweit das Diagnose-Programm der Firma IBM und das von mir erstellte Programm PCTEST die Mittel hierzu anboten. Die Diskettenverträglichkeit wurde einfach dadurch sichergestellt, daß ich für alle Arbeiten ausschließlich meine normalen, für den IBM-PC bestimmten Disketten verwendete. Außerdem wurden Disketten, die auf dem zu überprüfenden Rechner formatiert und beschrieben wurden, auf einem Vergleichs-IBM-PC weiterverarbeitet. Danach wurde die Ablauffähigkeit einiger Standard-Programme überprüft. Dabei wurden vor allem solche Programme berücksichtigt, die auch Leistungen unterhalb der Betriebssystemebene in Anspruch nehmen. Mit jedem dieser Programme wurden eine Reihe von programmspezifischen Aktionen durchgeführt. Im einzelnen handelte es sich hierbei um

- die Benutzeroberfläche "File Command",
- den Digital Research C-Compiler mit Assembler und Linker,
- den "Professional Editor",
- das Textverarbeitungsprogramm "WordStar" mit "WSGERM",
- das Textverarbeitungsprogramm "Easy Writer",
- das Formatierungsprogramm "FORMAT",
- den Drucker-Spooler "PRINT",
- das Tabellenkalkulationsprogramm "LOTUS123",
- das grafische Programm "ZELLAUTO" mit "GRAPHICS" und
- das Kommunikationsprogramm CMS.

Bei den Programmen WSGERM, ZELLAUTO und CMS handelt es sich um von mir geschriebene Software. WSGERM ist ein Rahmenprogramm für WordStar, das eine Schwäche bei der Druckerausgabe von deutschen Umlauten umgeht. Das Programm ZELLAUTO simuliert die Entwicklung linearer Zellautomaten auf einem Grafik-Bildschirm. CMS erlaubt den Anschluß eines IBM-PCs an den Protokollkonverter des Rechenzentrums. Der Rechner verhält sich dann wie ein IBM-327x-Terminal. Das Programm setzt nur den asynchronen Adapter voraus.

Für die Zeit, in der mir ein Rechner zur Verfügung stand, wurde er von mir als Standard-Arbeitsgerät benutzt. Hierdurch war es am einfachsten festzustellen, wie er sich im täglichen Umgang bewährte.

Das beschriebene Vorgehen gewährleistet in manchen Bereichen sicher nur eine exemplarische Überprüfung. Insgesamt erhält man hierdurch jedoch mit relativ geringem Arbeitsaufwand einen Überblick über die Leistungsfähigkeit

des einzelnen Rechners und die Kompatibilität der verschiedenen Rechner zueinander.

ACRITH

B. Süselbeck

Jeder, der umfangreiche numerische Berechnungen mit Gleitpunktzahlen durchführt, sollte sich der dabei auftretenden Fehlerquellen bewußt sein. Mit ACRITH wird der Versuch unternommen, diese Probleme zu reduzieren. Zunächst sollen die ohne die Verwendung von ACRITH auftretenden Schwierigkeiten beschrieben werden.

Obwohl die Gleitpunktoperationen +, -, *, / fast immer maximale Genauigkeit liefern, d.h. es wird die dem exakten Ergebnis am nächsten liegende in der Maschine darstellbare Zahl ausgegeben, so trifft dies bei mehrfacher Anwendung arithmetischer Operatoren, insbesondere bei komplizierten Algorithmen im allgemeinen nicht mehr zu. Ein Beispiel möge dies verdeutlichen: Der arithmetische Ausdruck **10E50 + 911 - 10E50** wird keineswegs das exakte Ergebnis **911**, sondern den Wert **0** liefern (selbst bei vierfacher Genauigkeit).

Die folgenden Beispiele enthalten einige der üblichen numerischen Rechnungen, bei denen überraschende Ergebnisse zu tage treten.

Skalarprodukt:

Für die Vektoren A und B mit den Komponenten

A(1) = 2.718281828	B(1) = 1486.2497
A(2) = -3.141592654	B(2) = 878366.9879
A(3) = 1.414213562	B(3) = -22.37492
A(4) = 0.5772156649	B(4) = 4773714.647
A(5) = 0.3010299957	B(5) = 0.000185049

liefert ein FORTRAN-Programm mit dem Statement

```
SP = A(1)*B(1) + A(2)*B(2) + A(3)*B(3) + A(4)*B(4) + A(5)*B(5)  ,
```

aber auch die NAG-Routine X02AAF für SP in doppelter Genauigkeit den Wert +0.335349457333448215E-09. Der exakte Wert hingegen lautet -1.00657107E-11.

Arithmetische Ausdrücke:

Berechnet man den arithmetischen Ausdruck

$$(1682xy^4 + 3x^3 + 29xy^2 - 2x^5 + 832)/107751$$

in doppelter Genauigkeit für x = 192119201 und y = 35675640, so erhält man anstelle des exakten Wertes 1783 das Ergebnis -0.718056003706102612E+21.

Polynomberechnung:

Wenn man das Polynom

$$P(x) = 8118x^4 - 11482x^3 + x^2 + 5741x - 2030$$

mit dem Horner-Schema in doppelter Genauigkeit an der Stelle x = 0.707107 berechnet, so ergibt sich: -1.97815097635611892E-11
Exakter Wert: P(x) = -1.91527325270810000E-11

Lineare Gleichungen:

Die korrekten Formeln zur Lösung des Gleichungssystems

64919121x - 159018721y = 1
41869520.5x - 102558961y = 0

lauten:

y = 41869520.5/(64919121*102558961-41869520.5*159018721)

x = (102558961/41869520.5)*y

Als korrekte Lösungen ergeben sich für x und y die Werte

x = 205117922 und y = 83739041, hingegen

x = 84814646.4057059884 und y = 34625434.4014264978

bei Berechnung mit doppelter Genauigkeit.

Die Grundlage zur Überwindung dieser Schwierigkeiten besteht in der Angabe einer fünften elementaren arithmetischen Operation, nämlich des Skalarprodukts zweier Vektoren (in Zeichen: ∘). Für

$x = (x_1,\ldots,x_n)$ und

$y = (y_1,\ldots,y_n)$ gilt also

$$x \circ y = \sum_{i=1}^{n} x_i y_i \ .$$

Dieses Skalarprodukt ist so auszurechnen, daß das Ergebnis wieder (wie bei den anderen Operatoren) die dem wahren Ergebnis am nächsten kommende Maschinenzahl ist. Dies kann natürlich nicht durch Zurückführen auf die Anwendung der üblichen Operatoren + und *, wie es die obige Definition nahelegt, erreicht werden, da Rundungsfehler sich akkumulieren können. Die höhere Genauigkeit wird maschinenintern durch einen sehr langen Akkumulator (328 hexadezimale Stellen) erreicht, der es erlaubt, die Summation "stellengerecht" durchzuführen, und sogar einen Overflow bei Zwischenergebnissen berücksichtigt. Dieses

Skalarprodukt ist dann die Grundlage für viele Matrixoperationen, auf die sich wiederum zahlreiche numerische Algorithmen gründen. Auch die exakte Berechnung arithmetischer Ausdrücke geschieht durch Zurückführen auf lineare Gleichungssysteme und damit auf Matrixoperationen.

Diese fünfte Grundoperation (und zusätzlich die Möglichkeit, gerichtet zu runden) ist in dem IBM-Programmpaket ACRITH realisiert. ACRITH besteht aus einer Sammlung von Unterprogrammen, die von VS FORTRAN aus aufgerufen werden können, und einer On-line-Trainingskomponente. Die Programmbibliothek setzt sich aus drei Teilen zusammen:

1. Full Precision Primitives (Routinen zur direkten Rechnung mit dem erweiterten Akkumulator)

2. Basic Arithmetic Routines (Routinen zur Berechnung von Skalar- und Matrixprodukten sowie Konvertierungen zwischen hexadezimaler und dezimaler Darstellung)

3. Problem Solving Routines (Routinen zur Berechnung mit maximaler Genauigkeit). Es gibt Routinen für folgende Aufgaben:

   - Arithmetische Ausdrücke
   - Polynomberechnung
   - Nullstellen von Polynomen
   - Lineare Gleichungssysteme
   - Invertierung einer Matrix
   - Lineare Optimierung
   - Eigenwerte und Eigenvektoren
   - Standardfunktionen

Einige der Routinen beinhalten die Möglichkeit zur Intervallrechnung, insbesondere den Nachweis von Existenz und Eindeutigkeit einer Lösung innerhalb gewisser Schranken.

Eine Gegenüberstellung von herkömmlicher Rechnung und ACRITH bildet das folgende Beispiel zur Reihenberechnung:

Die Auswertung des Ausdrucks

$$1 - \frac{\frac{3}{2}}{(1!)^2}x + \frac{\frac{3}{2}\frac{5}{2}}{(2!)^2}x^2 - \frac{\frac{3}{2}\frac{5}{2}\frac{7}{2}}{(3!)^2}x^3 + - \ldots$$

mit den FORTRAN-Statements für x = 50:

```
      S=1D0
      T=1D0
      A=5D-1
      X=5D1
      DO 10 I=1,N
           A=A+1
           T=-T*A/(I*I)*X
           S=S+T
 10        CONTINUE
```

ergibt für N=10(10)170 folgende Tabelle, bei der die Abweichungen von den ACRITH-Werten (s.u.) unterstrichen sind:

```
  10                  83413677355.7331085
  20            14433718298358430.0
  30          1375082076438158 6700.0
  40         4463054732141434 63000.0
  50        1173650140013734 070000.0
  60         4165118696008040 45000.0
  70          279435114009802 71100.0
  80            45032263228782 7712.0
  90              208510573415 5502.0
 100                 3187564540120.130860
 110                    1797941610.026829720
 120                       553022.6276921327
 130                       143433.1513405994
 140                       143392.4866717651
 150                       143392.4847966468
 160                       143392.4847966043
 170                       143392.4847966043
```

Mit ACRITH hingegen erhält man

```
  10                  83413677355.73315
  20            14433718298358456.0
  30          1375082076438163 0500.0
  40         4463054732141452 33000.0
  50        1173650140013740 290000.0
  60         4165118696008067 32000.0
  70          279435114009803 48900.0
  80            45032263228768 8416.0
  90              208510573401213 1.0
 100                 3187564396727.682
 110                    1797798217.541219
 120                        409630.1420592959
 130                            40.66570775702985
 140                             0.001038922760929940
 150                            -0.0008361954386099667
 160                            -0.0008362378592663932
 170                            -0.0008362378597601083
```

Alle bisher behandelten Beispiele sind natürlich konstruiert, und es ist sehr schwer, qualifizierte Aussagen darüber zu machen, inwieweit solch kritische Situationen bei realen Anwendungen wirklich auftreten. Wenn sie aber wirklich vorkommen, ist ACRITH sicherlich ein nützliches Werkzeug, z.B. bei schlecht konditionierten Problemen oder unsicheren Eingabedaten, die dann in Intervallform eingegeben werden können. Deshalb ist ACRITH auch nicht als Ersatz, sondern als Ergänzung zu der bisher vorhandenen Software zu sehen, zumal der Zeitbedarf in der Regel höher sein wird. Allerdings lassen sich mit ACRITH bei kritischen Problemen aufwendige Wiederholungsläufe vermeiden. Zum Abschluß sei noch bemerkt, daß sich mit ACRITH natürlich nur die Fehler der herkömmlichen Rechnerarithmetik vermeiden lassen, aber nicht solche, die durch schlechte Modellierung oder ungünstige Diskretisierung bzw. Approximation entstehen.

Ziel dieses Artikels war es, über eine neuere Entwicklung auf dem Gebiet der numerischen Datenverarbeitung zu berichten. Leider verfügt das Rechenzentrum zur Zeit nicht über die zur Beschaffung von ACRITH nötigen Mittel. Um trotzdem einen Überblick bezüglich der Nachfrage nach dieser Software zu bekommen, mögen sich interessierte Benutzer bitte mit dem Rechenzentrum in Verbindung setzen. Eventuell besteht die Möglichkeit einer zeitlich befristeten Testinstallation.

Literatur: High-Accuracy Arithmetic Subroutine Library: General Information Manual (IBM GC33-6163-1)
High-Accuracy Arithmetic Subroutine Library: Program Description and User's Guide (IBM SC33-6164-1)
Kulisch: Grundlagen des Numerischen Rechnens
Kulisch/Ullrich: Wissenschaftliches Rechnen und Programmiersprachen

Die Statistikseite

Programmquerschnitt Januar bis März 1985

A. Ahrens

Im ersten Quartal 1985 ergab sich im Batch-Betrieb auf Jobstep-Basis folgende Verteilung:

Programm	Anzahl	in %	CPU-Zeit	in %
PL/1-Optimizing-Compile	17483	8.14	12:54:23	1.12
PL/1-Optimizing-Linkage	6682	3.11	2:36:05	0.23
PL/1-Optimizing-Execute	26024	12.12	175:48:37	15.26
PL/1-Checkout-Compile	1303	0.61	1:22:47	0.12
PL/1-Checkout-Linkage	0	0.00		0.00
PL/1-Checkout-Execute	748	0.35	2:34:15	0.22
FORTRAN-66-Compile	12000	5.59	5:48:18	0.50
FORTRAN-66-Linkage	4281	1.99	2:48:33	0.24
FORTRAN-66-Execute	25475	11.87	262:28:33	22.78
FORTRAN 77-Compile	27136	12.64	15:41:47	1.36
FORTRAN 77-Linkage	1118	0.52	0:39:58	0.06
FORTRAN 77-Execute	18355	8.55	315:57:23	27.42
Pascal-Compile	2845	1.33	2:37:10	0.23
Pascal-Linkage	387	0.18	0:07:17	0.01
Pascal-Execute	2839	1.32	132:46:31	11.52
SPSS	22952	10.69	37:37:58	3.27
SAS	6711	3.13	26:38:41	2.31
IEB / IEH - Programme	11418	5.32	2:22:13	0.21
SORT	6546	3.05	3:48:59	0.33
Service	10851	5.06	6:24:42	0.56
andere RZ-Dienstprogramme	2250	1.05	2:53:47	0.25
Anzahl der Jobsteps gesamt :	214658			
mit einer CPU Zeit von :			1152:25:48	