



von Franz-Josef Reichert

Steigen wir gleich ein mit der Beschreibung der Prozeß-Kommunikation (Bild 1). Wir hatten schon geklärt, daß sich Amiga-DOS bei der Nachrichtenübermittlung des Message-Systems von Exec bedient.

Ähnlich wie Prozesse auf der Exec-Task-Struktur basieren, bauen auch »DosPackets« auf der Exec-Message-Struktur auf: In der Struktur »Message« verweist der Eintrag »In_Name« aus der Unterstruktur »mn_Node« auf die Dos-Packet-Struktur; in der DosPacket-Struktur verweist der Eintrag »dp_Link« auf die Message-Struktur. Dadurch ist die Koppelung vollzogen. Die Message-Struktur wird Exec übergeben. Exec sendet sie über die PutMsg()-Funktion an den Prozeß-MsgPort des Empfängers. Mit dem Antworten ist es allerdings nicht mehr so einfach. Unter Exec würde man einfach ReplyMsg() aufrufen. Voraussetzung dazu wäre, daß der Eintrag mn_ReplyPort in der Message-Struktur auf den MsgPort des Absenders zeigt, was aber nicht der Fall ist. Beim DosPacket ist dieser Zeiger im Eintrag »dp_Port« abgelegt. Will ein Prozeß ein Packet nach Erhalt und Bearbeitung zurücksenden, muß er zunächst den Absender aus »dp_Port« lesen, dann dort den Zeiger auf seinen eigenen MsgPort eintragen und die Message-Struktur an Exec übergeben. Die Dos-Packet-Struktur enthält außerdem den Typ des Packets »dp_Type«, zwei Felder für Rückgabewerte »dp_Res1« und »dp_Res2«, sowie maximal sieben Argumente von »dp_Arg1« bis »dp_Arg7«. Packets sind für die unterschiedlichsten Operationen definiert. Je nach Typ kann das Format der Argumente und Resultate variieren, die wichtigsten Packets dienen zum Lesen und Schreiben von Daten. In diesem Falle wird der Bezug zum FileHandle über das erste Argument hergestellt, das den gleichen Wert wie »fh_Arg1« erhält.

Wir wollen uns nun die Arbeit eines DOS-Handler-Prozesses anhand eines Beispielprogramms klarmachen. Das Listing dazu finden Sie im Artikel »Boot-Handler« auf Seite 118 in dieser Ausgabe des AMIGA-Magazins. Unser Handler ist in »C« geschrieben, trägt den Namen »Boot-Handler« und wird über die »mountlist« eingebunden. Übernehmen Sie also die Eintragsdefinition aus dem Artikel in Ihre Mountlist. Rufen Sie jetzt

mount BOOT:

auf. Wenn Sie alles richtig gemacht haben, kennt Amiga-DOS ab jetzt das Gerät »BOOT:«, was Sie durch Aufruf von ASSIGN überprüfen können. Doch was kann das neue Gerät? Stellen Sie sich das BOOT-Device als Diskette vor, auf der alle Bootblöcke Ihrer angeschlossenen Laufwerke, ob Festplattenpartitionen, RAM-Disks oder Disketten-Laufwerke, als Dateien gespeichert sind.

Sie haben dort ab jetzt nahezu unbegrenzte Manipulationsmöglichkeiten auf CLI-Ebene, wo Sie früher nur mühsam mit Diskettenmonitoren zu Rande kamen. Probieren Sie die folgenden Beispiele:

type boot:df0 opt h	; Anzeigen
copy boot:df0 to df1:boot.dat	; Sichern
copy df1:boot.dat boot:df0	; Ersetzen
newzap boot:df0	; Editieren (Fish-Disk 58)
copy * to boot:df0	; Beschreiben (<Ctrl-> Ende)
type boot:df0 to prt: opt h	; Ausdrucken
copy boot:df0 boot:df1	; Duplizieren

Tabelle. Ein paar nützliche Befehle

Die Arbeitsweise des Programms können Sie anhand des reich kommentierten Listings leicht nachvollziehen. Der Code für unseren Handler-Prozeß muß bestimmten Eigenschaften genügen, die eine besondere Programmierdisziplin bei der Entwicklung in »C« oder Assembler erfordern — BCPL-Programme erfüllen diese Voraussetzung übrigens von »Haus« aus. Das komplette Programm muß nämlich »entrant«, d. h. wiedereintrittsfähig sein. Sie dürfen keine globalen Daten verwenden, die verändert werden könnten. Erlaubt sind nur lokale Variablen, die im Stackbereich liegen. Größere Datenstrukturen müssen wie benötigt auf dynamischem Weg beschafft und nach Gebrauch wieder freigegeben werden.

Die meisten von Ihnen werden diese Eigenschaft bereits kennen, schließlich genügen diesen auch einige CLI-

Befehle, die durch das gesetzte »pure-bit« erkennbar sind. Nur solche Programme können durch das CLI-Kommando RESIDENT geladen und danach mehrfach ausgeführt werden, ohne sie erneut von Diskette laden zu müssen. Residente Programme werden von Amiga-DOS in einer Liste verkettet, auf die der Eintrag »di_NetHand« in der DosInfo-Struktur verweist (siehe Bild 2). Dies ist übrigens ein BPTR und kein APTR, wie in den Include-Dateien fälschlicherweise angegeben. Die »ResidentList«-Strukturen sind über den BPTR »rl_Next« verkettet. »rl_UseCount« bezeichnet die Zahl der Prozesse, die gerade das Programm benutzen. Ist der Wert größer als Null, verhindert dies ein Entfernen des Codes aus dem Speicher durch Amiga-DOS. »rl_SegList« zeigt auf die uns schon bekannte SegmentList-Struktur, die das Ergebnis des Ladevorgangs ist. »rl_SegName« schließlich enthält den Na-

me der Prozesse. Sie sind im nächsten Abschnitt Gegenstand unserer Betrachtung sein.

DOS-Routinen zerfallen in zwei Kategorien. Die erste ent-

In der Ausgabe 8/89 des AMIGA-Magazins haben wir im Insider-Kurs mit der Besprechung des Amiga-DOS begonnen. Diesmal geht es um Kommunikation, Handler-Prozesse und das Amiga-DOS-Diskettenformat.

AMIGA

Teil 5

KURSÜBERSICHT

Amiga-Insider verdeutlicht die Zusammenhänge und Abläufe im Betriebssystem des Amiga. Vorkenntnisse im Programmieren sind notwendig. Wenn Sie den Kurs verfolgen, machen wir Sie zum Amiga-Insider.

TEIL 1: Übersicht, Betriebssystem-Kern (Exec)

TEIL 2: Intuition — Die Bedieneroberfläche

TEIL 3: Geräte (Devices), input.device, keyboard.device, gameport.device, console.device, timer.device, serial.device, parallel.device, printer.device, audio.device, clipboard.device

TEIL 4: Zugriff auf die Dateien — dos.library, trackdisk.library

TEIL 5: DOS-Prozeßkommunikation, Prozesse, Programmausführung, Aufbau der Ketten, Directories, Hashing, File-System, DOS-Handler

TEIL 6: Grafik und Text auf dem Bildschirm — graphics.library, layers.library

TEIL 7: Kochrezept — Wie man sich ein eigenes Icon oder eine eigene Library

hält alle Funktionen, die der Ein- und Ausgabe dienen. Die Routinen der zweiten Kategorie übernehmen die Steuerung der DOS-Prozesse, welche das »File-Handling« durchführen.

Zur Bearbeitung von DOS-Objekten stehen uns Routinen zur Verfügung, die die Packet-Kommunikation mit den entsprechenden Handler-Prozessen für uns erledigen. Drei von ihnen, Lock(), DupLock() und Unlock() dienen dazu, Locks zu belegen und freizugeben. Über die FileLock-Struktur lassen sich mit den Funktionen Examine(), ExNext() und Info() weitere Informationen in FileInfoBlock- und InfoData-Struktur übertragen. Zum Anlegen, Verfolgen und Wechseln von Directories werden die Funktionen CreateDir(), CurrentDir() und ParentDir() verwendet. Kommentare und Protection-Maske unserer DOS-Objekte werden mit den Funktionen SetComment() und SetProtection() beeinflusst, um-

reits bekannte FileHandle-Struktur für uns. Schreiben und Lesen können wir über die Funktionen Read() und Write(); Seek() übernimmt dabei die Positionierung innerhalb des Files. Eventuell auftretende Fehler werden über IoErr() ausgelesen. Speziell für CLI-Prozesse sind die beiden letzten Funktionen dieser Kategorie vorgesehen. IsInteractive() liefert uns die Antwort, ob der CLI-Prozeß ein Terminal ist und Benutzereingaben akzeptiert oder keine Kommunikation zuläßt. WaitForChar() teilt mit, ob ein Zeichen über ein interaktives Terminal eingegeben wurde.

Die zweite Kategorie befaßt sich mit dem Prozeß-Handling. Bevor ein Prozeß gestartet werden kann, muß sein Programmcode geladen werden. LoadSeg() erledigt dies für uns, UnLoadSeg() gibt den Speicherbereich wieder frei. Danach können Sie den Prozeß mit CreateProc() starten. Diese drei Funktionen lassen sich übr-

können Sie die Prozeßkennung, dies ist die Adresse des Message-Ports eines Handler-Prozesses, für DOS-Devices in Erfahrung bringen.

Wenn ein Prozeß für eine gewisse Zeitspanne stillgelegt werden soll, ohne deswegen anderen Tasks Rechenzeit zu rauben, erfolgt das am einfachsten durch Aufruf von Delay(). Sie sollten in einem Multitaskingsystem stets sicherstellen, daß keine Rechenzeit durch unnützes Abzählen von sogenannten »Verzögerungsschleifen« verschwendet wird! Die Systemzeit kann ebenfalls komfortabel mit DateStamp() in Form des bereits bekannten Datumsstempels ausgelesen werden.

Prozesse

Eine letzte Funktion namens Execute() ermöglicht uns die Erzeugung von CLI-Prozessen, dies können interaktive oder Hintergrundprozesse sein.

Wir wollen uns nun mit der Frage beschäftigen, wie ein Programm auf dem Amiga zur Ausführung gebracht wird. Jedes Programm liegt zunächst als sogenanntes »executable file« auf Diskette vor. Das ist ausführbarer Programmcode,

derlichen Stackbereich zur Verfügung. Ist dieser Vorgang erfolgreich abgeschlossen, wird sofort mit der Ausführung begonnen. Zu bemerken ist hierbei, daß kein neuer Prozeß erzeugt wird. Das geladene Programm ist quasi ein »Unterprozeß« des CLI-Prozesses. Vor Einsprung in den Programmcode werden noch einige Parameter übergeben: ein Zeiger auf die Kommandozeile im Register »A0«, die Länge der Zeile in »D0«. Der Stackpointer »A7« zeigt auf die Rücksprungadresse. Die Größe des Stackbereiches kann im Langwort darüber ausgelesen werden. Nach Rücksprung wird der Programmcode mittels UnLoadSeg() wieder aus dem Speicher entfernt und der Stackbereich freigegeben.

— Die zweite Möglichkeit, der Start über die Workbench, erfolgt auf etwas andere Weise. Der Workbench-Task lädt zunächst den Programmcode über LoadSeg() und startet einen eigenen Prozeß mit der Funktion CreateProc(), die auch für die Beschaffung des Stackspeichers sorgt. Der neue Prozeß erhält dann unmittelbar eine Nachricht über seinen Message-Port als Ersatz für die

Insider

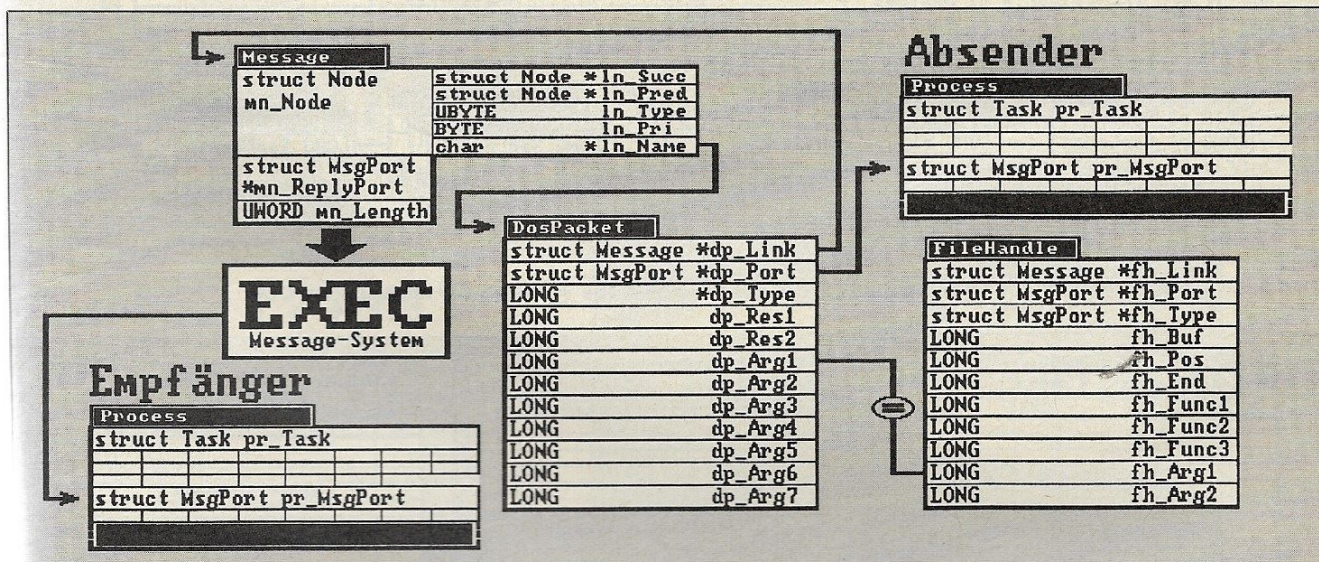


Bild 1. Das DOS verwendet das Exec-Message-System, um seine Kommunikation abzuwickeln

benannt oder gelöscht werden können. Die Funktionen Rename() und Delete() werden verwendet.

Schließlich erhalten wir über die Funktionen Input() und Output() bereits offene FileHandles für die umgeleitete Ein- und Ausgabe eines CLI-Prozesses. Wenn wir Dateien zur Bearbeitung öffnen oder schließen, erledigen dies die Funktionen Open() und Close() über die be-

gen auch von reinen Exec-Tasks aufrufen, beispielsweise macht die Workbench regen Gebrauch davon. Beendet wird ein Prozeß, wenn er die Funktion Exit() aufruft. Die Prozeß-Struktur und den Stackspeicher gibt Amiga-DOS dann wieder frei. Um die Rückgabe der Segmentliste müssen Sie sich allerdings selbst kümmern. Mit Hilfe der Funktion DeviceProc()

wie er vom Linker erzeugt wird. Es gibt nun zwei Möglichkeiten, das Programm zu starten.

— Die erste ist der sogenannte CLI-Start, dazu werden der Name des Programms und gegebenenfalls weitere Parameter in einer Kommandozeile eingegeben. Der CLI-Prozeß lädt nun den Programmcode über die Funktion LoadSeg() in den Speicher und stellt den erforder-

liche Kommandozeile. Sie gibt Auskunft über etwaige Project-Icons, die in Zusammenhang mit dem Programmstart angewählt wurden. Diese Nachricht vom Typ »WBStartup« kann ebensogut ignoriert werden, muß aber in jedem Fall mittels GetMessage() abgeholt werden und beantwortet werden. Bevor dies mit ReplyMsg() geschieht, muß unbedingt For-



bid() aufgerufen werden, sonst wird der Programmcode von der Workbench über UnLoadSeg() möglicherweise zu früh freigegeben, was unweigerlich zum Absturz führt. Das ist auch der Grund dafür, daß sich CLI-Befehle in der Regel nicht von der Workbench aus starten lassen. Ein weiterer Grund liegt darin, daß kein CLI-Fenster als Terminal zur Verfügung steht.

Ein- und Ausgabeoperationen laufen ins Leere, wenn vorher kein Fenster geöffnet wurde.

Mit der Besprechung der DOS-Prozesse schließt sich unser Kreis um alle Objekte, die von Amiga-DOS verwaltet werden. Thema unseres letzten Abschnitts soll die Beschreibung der Formate und der Organisation sein, mit der Amiga-DOS seine Objekte auf Massenspeicher ablegt. In der Regel sind dies die 3½-Zoll-Disketten-Laufwerke, aber ebenso gut können an ihre Stelle Festplattenpartitionen, RAM-Disks und in naher Zukunft wohl auch Laserdisks und CD-ROMs treten.

Zunächst müssen wir uns klarmachen, wie das Betriebssystem ein Massenspeicher-

medium über das zugehörige Exec-Device verwaltet. Alle Massenspeichermedien, ob Festplatten, Floppies oder RAM-Disks, bestehen zunächst aus einer bestimmten Anzahl von (gedachten) Spur-Stapeln, den sogenannten »Cylinders«. Ein Spur-Stapel zerfällt je nach Anzahl der Diskettenoberflächen in eine bestimmte Anzahl von Spuren, englisch »Tracks«. Da jede Oberfläche (»Surface«) von einem Schreib-/Lesekopf bedient wird, drückt man ihre Anzahl auch mit der Bezeichnung »Heads« aus. Als Beispiel soll hier eine Floppy-Disk dienen. Sie besteht aus 80 Cylindern, von Cylinder 0 bis 79. Da die Diskette über zwei Heads verfügt, teilen sich 80 Cylinder in 160 Tracks auf. Bis hierher

haben wir noch kein bestimmtes »Format« auf unserer Diskette, wir haben lediglich ihre hardwaremäßig festgelegte »Geometrie« beschrieben. Dies sind auch die Parameter, die für jede Einbindung eines neuen Disk-Devices in der »Mountlist« angegeben werden müssen. Zur »Formatierung«, die Voraussetzung für die Verwendung einer Diskette durch Amiga-DOS ist, werden die Tracks wiederum in kleinere Einheiten aufgeteilt, diese heißen »Blöcke« oder »Sectors«. Am Beispiel der Floppy-Disk besteht ein Track aus elf Blöcken. Ein Block ist die kleinste Einheit, aus der eine Diskette zusammengesetzt ist. In einem Block werden maximal 512 Byte Daten abgelegt.

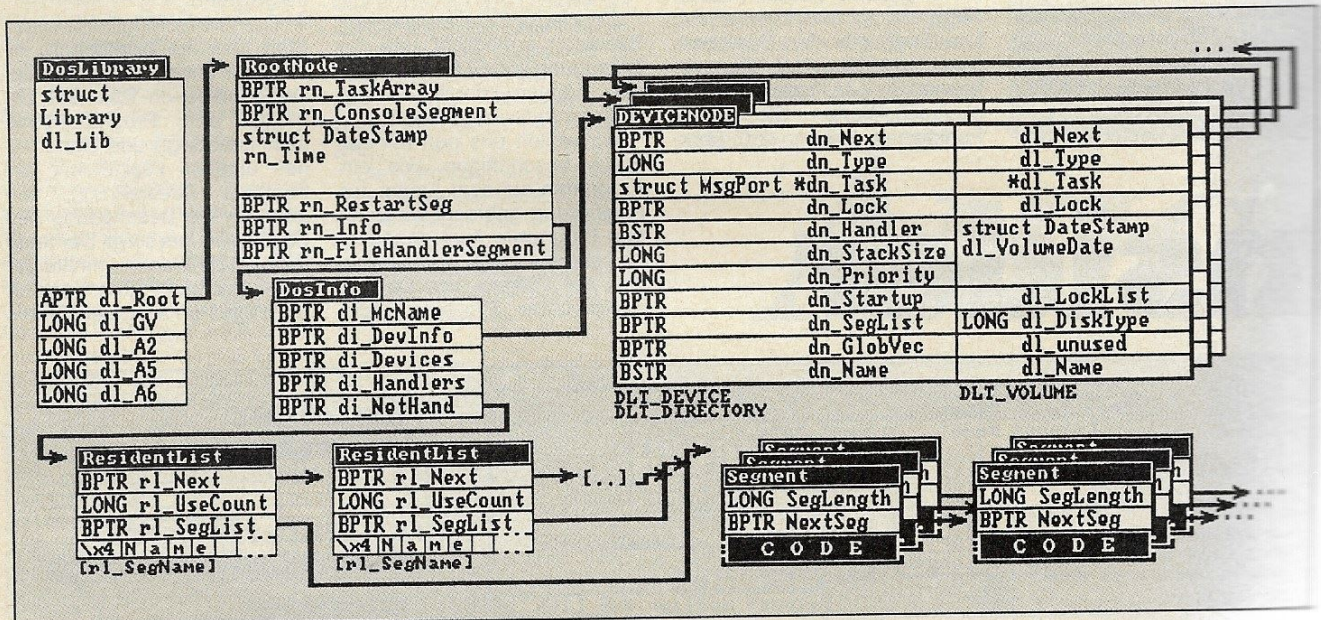


Bild 2. In der DOS-Library gibt es einen Eintrag, der auf die RootNode verweist, der Zugang zum DOS

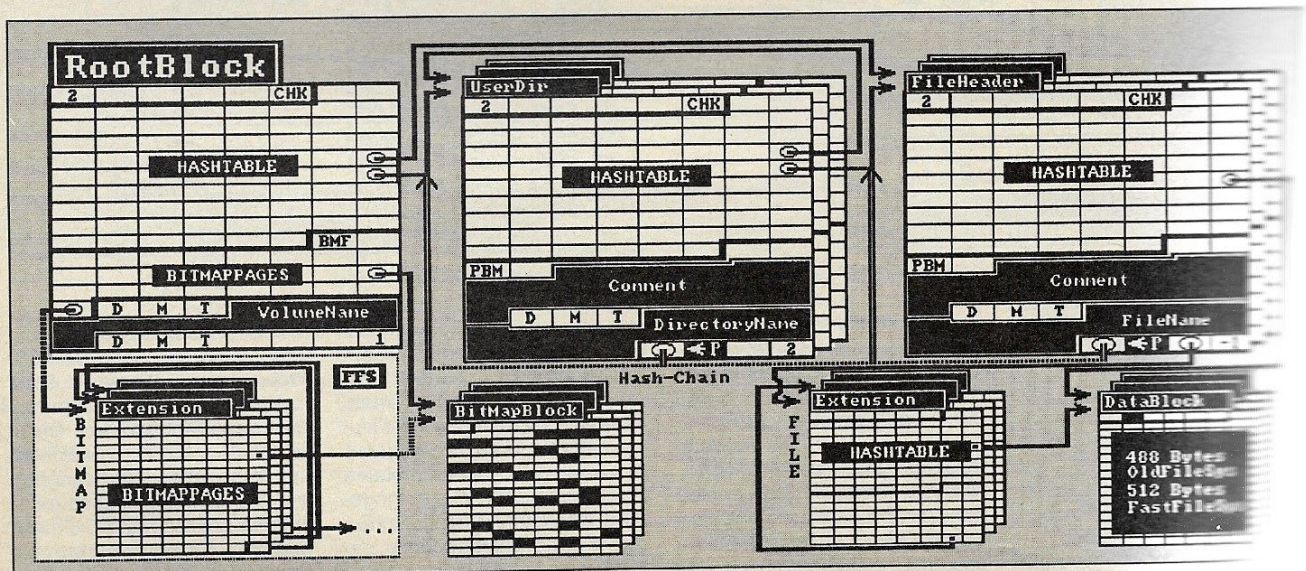


Bild 3. So verweist der Root-Block auf die User-Directories. Hashtable und Bitmappages sind auch enthalten.

Nach der Formatierung im Amiga-DOS-Format besteht eine Diskette also aus $80 \times 2 \times 11 = 1760$ Blöcken. Dies entspricht einer maximalen Speicherkapazität von $1760 \times 512 = 880 \times 1024 = 901\,120$ Byte oder 880 KByte.

Zum Ablegen seiner Objekte auf Diskettenblöcken hält Amiga-DOS für jedes Objekt einen eigenen Block-Typ bereit (Bild 3). Seit Erscheinen der Workbench V1.3 wird neben dem normalen Aufzeichnungsformat noch ein weiteres verwendet, das »Fast-File-System«. Dieses nutzt Geschwindigkeit und Speicherkapazität besser aus, kann zur Zeit aber nur für Festplatten und RAM-Disks verwendet werden. Für die Zukunft plant Commodore eine Erweiterung auf Disketten-Laufwerke. Wir werden bei der nachfolgenden Besprechung der Block-Formate auf die Unterschiede zwischen beiden Systemen eingehen.

Angeführt wird jede DOS-Diskette durch mindestens zwei »Boot-Blöcke«, die reserviert sind. Entscheidend ist hier das erste Langwort, in dem die Diskettenidentifikation abgelegt ist. »DOS\x00« steht dabei für

eine Standard-DOS-Diskette, »DOS\x01« dagegen für eine Fast-File-System-Diskette. Alle anderen Kennungen werden nicht als DOS-Diskette erkannt und können demzufolge von Amiga-DOS auch nicht beschrieben werden. In den Bootblöcken kann auch ein kurzes Startprogramm stehen, das mit dem CLI-Befehl INSTALL auf das Massenspeichermedium geschrieben wird. Es handelt sich in diesem Fall um eine »bootfähige« DOS-Diskette, d. h., sie wird von Amiga-DOS als Start- oder Workbenchdiskette akzeptiert.

DOS-Organisation

Zunächst gelangen wir zum Ur-Verzeichnis des Volumes, das durch den »Root-Block« repräsentiert wird. Dieser Block befindet sich als einziger an einer bestimmten Stelle auf der Diskette. Teilen Sie die maximale Anzahl der Blöcke durch zwei und Sie erhalten die gesuchte Blocknummer, auf einer Floppy-Disk ist es also der Block mit der Nummer 880. Dies stimmt aber nur für »nicht-bootfähige« Disketten in jedem Fall. Bei bootfähigen DOS-

Disketten enthält das dritte Langwort der Boot-Blöcke die Nummer des Root-Blocks, die in diesem Falle abweichen kann. Sie finden im Root-Block eine Menge Informationen über das Volume: seinen Namen, zwei Datumsstempel (DMT) für sein Erstellungsdatum und das Datum der letzten Änderung. Die Einträge »Bitmappages« sind Verweise auf Blöcke, in denen die Blockbelegung der gesamten Diskette festgehalten wird.

Dort ist im ersten Langwort eine Prüfsumme abgelegt, alle folgenden Langworte speichern in jedem ihrer 32 Bit den Status eines Diskettenblocks. Ist das Bit gesetzt, ist der Block noch frei; ein gelöscht Bit kennzeichnet dagegen einen bereits belegten Block. Bei der Zählung werden die reservierten Bootblöcke außer acht gelassen. Bit 31 des zweiten Langwortes steht also für Diskettenblock zwei, Bit 30 für drei usw. Die Tabelle »Bitmappages« enthält zunächst ein Flag (BMF), ob die Tabelle gültig ist. Danach können insgesamt 24 Zeiger auf Bitmapblöcke stehen, von denen jeder 4064 Diskettenblöcke verwal-

ten kann. Bei einer Floppy-Disk reicht natürlich ein Bitmapblock aus, aber Festplattenpartitionen benötigen in der Regel mehrere. Rechnen wir doch mal nach, wie groß eine Partition maximal sein kann: $24 \times 4064 = 97\,536$ Blöcke, das entspricht knapp 50 MByte. Nun gibt es aber durchaus noch größere Festplatten, und die ins Haus stehenden CD-ROMs können sogar mit Plattenkapazitäten im Gigabyte-Bereich protzen. Hier hilft uns ein weiterer Eintrag, der unmittelbar im Anschluß an die Tabelle folgt. Dieser zeigt auf einen »Bitmap-Extension«-Block. Über diesen können wir nun beliebig viele Bitmapblöcke ansprechen, denn alle Langworte dieses Blocks können auf neue Bitmapblöcke verweisen. Das letzte Langwort eines Extension-Blocks zeigt auf weitere Extension-Blocks oder ist Null. Beachten Sie bitte, daß diese Eigenschaft nur mit dem Fast-File-System nutzbar ist. Beim herkömmlichen System ist dies wegen eines Programmierfehlers nicht möglich.

Die Bitmap spielt für Amiga-DOS eine entscheidende Rolle. Sie wird daher bei jeder

Skyline • die ideale Festplattenlösung Autoboot-Filecard A2000

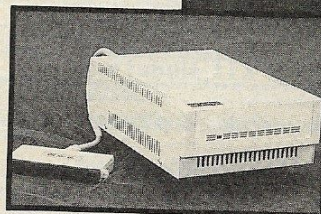
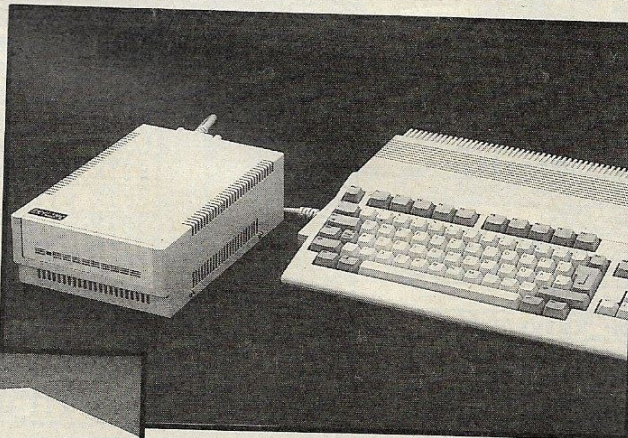
Erfolgreich getestet in Amiga 1/89,
Amiga Special sowie in Kickstart 1/89.
Für Amiga 500/1000/2000

Autoboot 1.2 + 1.3

Modul 149,-

20 MB nur 879,-

größere Platten 30/40/60 MB auch lieferbar.



- Zwei Harddisks werden unterstützt
- Schnellere Ladezeiten (V2.1 390 kByte/s)(40 MB)
- Deutlich höhere Schreibgeschwindigkeit bei gleichzeitig erhöhter Datensicherheit
- CheckDrive Funktion auch im Treiber
- Graphische Benutzeroberfläche

- Selbständige Installation
- Backup-Programm
- Park-Programm
- Disk Monitor
- Kopierprogramm
- Hilfreiche Utilities u. v. a.

- Automatischer Mountlisteditor

- Viele zusätzliche Utility Programme

SKYLINE-SOFT Michael Kuschel

Dieselstraße 4, 8044 Lohhof
Telefon 089/3 109496, Fax 089/3 171999

General distributor Österreich:

INTERCOMP, Heldendankstraße 24
A-6900 Bregenz, Telefon 055 74/27344



Schreiboperation vom »disk-validator«-Prozeß auf den neuesten Stand gebracht. Solange dieser nicht erfolgreich beendet wurde, sind alle weiteren Schreiboperationen auf das Volume verboten. Sollte er aus irgendeinem Grunde unterbrochen werden, ist die Bitmap ungültig und der Validator-Prozeß muß beim erneuten Einlegen des Volumes den gesamten File-System-Baum durchsuchen, um auf diesem Wege die belegten Blöcke zu ermitteln, bevor wieder Schreiboperationen stattfinden können. Je nach Größe und Zugriffszeit des Volumes kann das schon mal ein paar Minuten dauern. Wer hat in seinen ersten Amiga-Tagen nicht schon mal die Diskette zu früh rausgezogen, bevor die Laufwerkslampe verloschen war? Das Ergebnis war

anschaulich die Arbeitsweise der »Hash-Funktion« des Amiga-DOS. Jeder Eintrag der Hash-Tabelle kann auf einen weiteren Block zeigen, dies können sowohl »UserDir«-Blöcke wie auch »FileHeader«-Blöcke sein. Erstere stehen für Directories, letztere für Files, die sich auf dem Volume befinden. Die Hashtabelle besteht aus 72 Langworten, somit wäre es aber nicht möglich, mehr als 72 Einträge auf einem Volume zu verwalten. Man könnte natürlich ähnlich wie bei der Bitmap einen Extension-Block einführen, doch in diesem Falle wird ein anderer Weg beschritten. Amiga-DOS bestimmt über den Namen des Objekts seinen Platz im Hashtable. Dazu wird zunächst der »Hash-Value« nach der Rechenvorschrift im Flußdiagramm (Bild 4) berechnet. Ein Hash-Value wird relativ zum Blockbeginn berechnet und kann daher Werte von 6 bis 77 annehmen. So erhält beispielsweise das Directory mit dem Namen »L« einen Hash-Value von 23, also weist Langwort 23 des RootBlocks auf dieses Directory. Nehmen wir einmal an, auf der Volume existiert außerdem ein File mit dem Na-

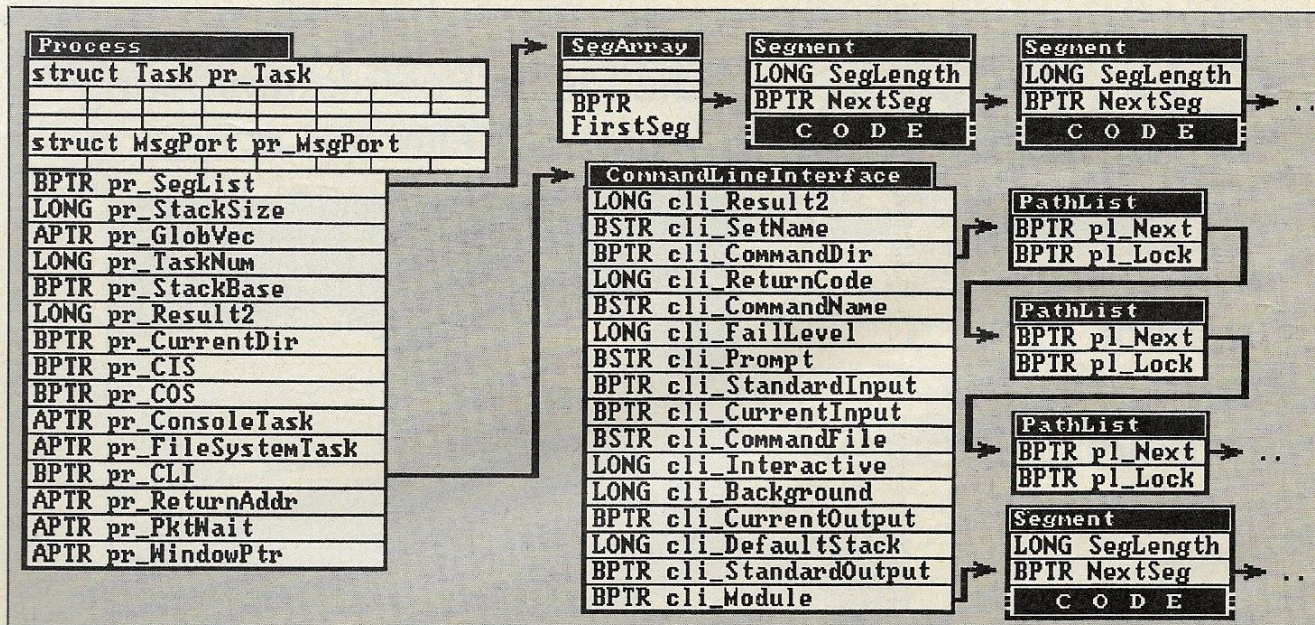
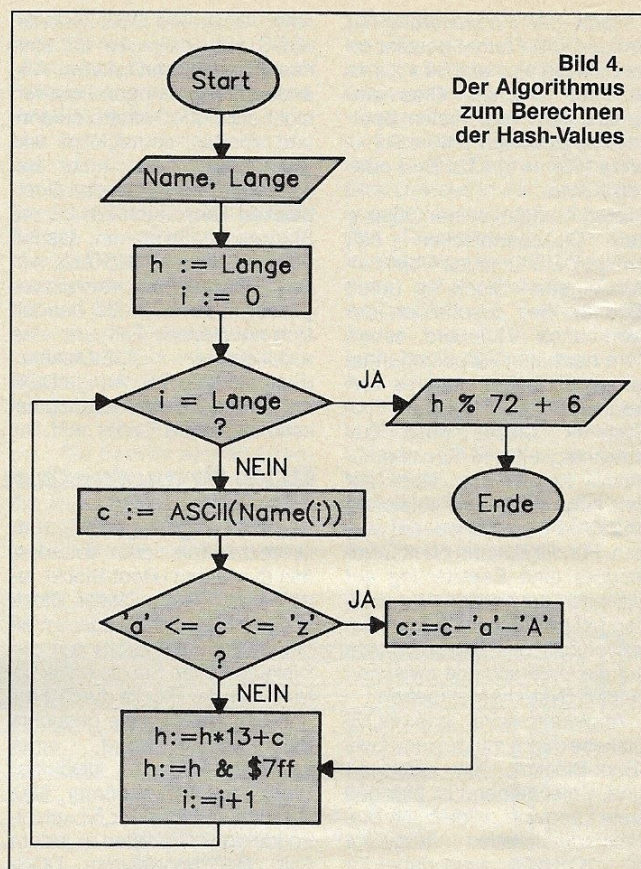


Bild 5. Ein DOS-Prozeß ist wie ein Hut auf einen Exec-Task gestülpt. Zusätzliche Informationen werden dort gespeichert.

(bestenfalls!), daß die zuletzt geschriebene Datei nicht mehr zu gebrauchen war.

Aber nun zurück zu unserem Root-Block. Eine weitere Tabelle mit 72 Einträgen weist auf unsere Dateien und Directories, die »HashTable«. »to hash« bedeutet übersetzt soviel wie »zerlegen« oder »zerhacken«. Dieses Wort charakterisiert recht

men »Shell«. Auch hier ergibt sich ein Hash-Value von 23. Da nun ein Langwort nicht auf zwei Objekte gleichzeitig zeigen kann, wird dieses File jetzt in der sogenannten »Hash-Chain« verkettet, d. h., ein bestimmtes Langwort im UserDir-Block des Directory »L« zeigt auf den File-Header des Files »Shell«. Alle weiteren Objekte mit gleichem

Hash-Value werden ebenfalls auf diese Weise verkettet. Sie werden es bereits ahnen: Mit dieser zunächst umständlich anmutenden Rechnerei werden gleich zwei Probleme auf einmal bewältigt. Zum einen sind der Anzahl von Directory-Einträgen keine Grenzen mehr gesetzt, zum anderen kann Amiga-DOS sehr schnell belie-

bige Objekte anhand ihres Namens auf einem Volume finden. Dazu braucht nur der Hash-Value berechnet und die Hash-Chain bis zum Ziel verfolgt zu werden. So ist der ganze umfangreiche Katalog an Directory-Einträgen durch die Hash-Funktion in Stücke »zerhackt«. Das Anzeigen des Directories dauert jedoch länger.

Wir hatten bereits die »UserDir«-Blöcke erwähnt und brauchen auch nicht mehr sehr viel dazu zu erläutern. Im Grunde ist der Aufbau der gleiche wie beim Root-Block. Anstelle der Bitmappages tritt hier allerdings ein Objekt-Kommentar und eine Protection-Bit-Maske (PBM). Außerdem enthält der Block den bereits erwähnten »HashEntry« in die Hash-Chain und einen Verweis zu seinem Ursprungs-Directory, dem »ParentDir«. Die Hashtable von UserDir-Blöcken kann wiederum Verweise auf weitere UserDrs und natürlich auch auf File-Headers enthalten. Dabei kommt die gleiche Hash-Berechnung zum Tragen wie beim Root-Block.

Als letztes Objekt bleiben noch die »FileHeader«-Blöcke übrig. Diese unterscheiden sich von den UserDir-Blöcken auch nur unwesentlich. Anstelle der Verweise auf weitere Objekte enthält die Hashtable nun endlich Verweise auf die »Data-Blocks«, wo wir unsere Daten finden. Auch hier stellt sich das Problem, daß ein File über mehr als 72 Datenblöcke verteilt

sein kann. Hier tritt dann ein weiterer Block in Aktion, der »FileExtension«-Block. Extension-Blöcke werden hier über das vorletzte Langwort verkettet. Bei diesen ist eigentlich nur die Hashtable von Bedeutung, die genau wie beim FileHeader-Block auf weitere Datenblöcke verweist.

Datenblöcke haben je nach Art des verwendeten File-Systems unterschiedliches Format. Beim »alten« File-System sind in den ersten sechs Langworten noch Verkettungsinformationen und Rückverweise auf den FileHeader abgelegt.

Fast-File-System

Als nutzbare Datenbytes bleiben daher nur ganze 488 Byte übrig. Anders verhält es sich beim Fast-File-System. Hier fehlt die Verkettungsinformation, und alle 512 Byte stehen der Datenspeicherung zur Verfügung. Damit wird die Diskettenkapazität noch um ca. 5 Prozent gesteigert. Dieses »zweischneidige Schwert« im Kampf um Speicherplatz kann sich

aber leider auch mal gegen uns selbst richten, nämlich wenn wir versehentlich gelöschte Daten wiederherstellen wollen. Die Verkettungsinformationen in den Datenblöcken waren nämlich ideal geeignet, eine verlorene Datei wiederaufzubauen, selbst wenn ihr FileHeader bereits überschrieben wurde. Beim Fast-File-System hat man aber in diesem Falle kaum noch eine Chance, da sich der Block nicht selbst identifizieren kann und man auf die Hash-Table angewiesen ist. Nicht nur aus diesem Grund sollte man besonders eine Sache niemals aus den Augen verlieren: Alle Daten, deren Reise wir nun von ihrem Ursprung durch alle DOS-Instanzen hindurch bis zu ihrem Ziel auf der Diskette verfolgt haben, schweben ständig in »Lebensgefahr«. Nichts ist so vergänglich wie eine Datei, von der keine Sicherheitskopie existiert, und nichts ist so frustrierend, wie die Arbeit von Tagen oder gar Wochen auf einen Schlag in die »ewigen elektromagnetischen Jagdgründe« entschwinden zu sehen. Sie können daher gar nicht oft ge-

nug daran denken, Sicherheitskopien von Ihren Disketten und regelmäßige Festplattenbackups anzufertigen.

An dieser Stelle wollen wir dann auch unsere Reise durch die Geheimnisse des Amiga-DOS beenden. Dieses Thema werden wir in unserem letzten Kursteil noch einmal aufgreifen, wenn es um die Libraries und Devices geht. Im nächsten Insider wird's erst einmal richtig bunt. Wir führen Sie in die Geheimnisse der Betriebssystemteile ein, über die die Verbindung zu der fantastischen Grafikhardware des Amiga hergestellt wird. Ein Beispielprogramm in C und Basic zeigt, wie man aus diesen Programmiersprachen auf die Besonderheiten der Amiga-Grafikhardware zugreifen kann. 784 der 4096 möglichen Farben des Amiga werden im HAM-Modus auf dem Bildschirm angezeigt. Über Copperlisten, View-Strukturen und Displaymodes können Sie auch etwas erfahren.

mi

(1) Amiga-DOS-Handbuch, Verlag Markt & Technik, ISBN 3-89090-465-3, 59 Mark



CPS Computertechnik GmbH
Hamburger Straße 283 • 3300 Braunschweig
Fax (05 31) 33 06 61 • BTX *20088 1490#

Ladenöffnungszeiten: Mo-Fr 9-13 und 14-18 Uhr
Sa 9-14 Uhr
Studentenrabatte geg. Nachweis a. Anfrage!

Tel. (05 31) 33 06 63

AMIGA

AMIGA 2000	1985,-
AMIGA 2000 + 20 MB Harddisk Autoboot	2740,-
inkl. Contr. 2090 A	2480,-
AMIGA 500	950,-
AMIGA 500 + 1084	1510,-
AMIGA 500 + 512 KB Speicher Erw. und Textverarbeitung DOCUMENTUM	1355,-
AMIGA 2000 + 20 MB Harddisk mit Autoboot und Word-Perfect 4.1 und Monitor 1084 S	3298,-

ERWEITERUNGEN

20 MB Harddisk A 2000 inkl. 2090 A Contr.	980,-
2 MB Erw. int. (A 2000) orig. CBM	1180,-
PC/XT Karte inkl. 5,25" Lw	899,-
+ MS-DOS + GW-Basic	2400,-
PC/AT Karte inkl. 5,25" Lw	330,-
+ MS-DOS + GW-Basic	

CPS AT I

80286-12 CPU 8/12 Mhz, 640 KB RAM bis 4 MB on board, 0-Wait, 1 ser/2 par/ Game-Port, Herc. komp. Grafikkarte, 1 Laufwerk TEAC 1,2 MB 5,25", Cherry MF II Tastatur

CPS weil Preis und Leistung stimmen!

Wir sind autorisierter Fachhändler für:



AMIGA-ZUBEHÖR

LW ext. 3,5" ohne Display	290,-
LW ext. 5,25" ohne Display	419,-
LW intern. 3,5" inkl. Einbausatz	220,-
Commodore 1084	630,-
Philips RGB Color CM 8833	630,-
EGA Standard	815,-
NEC MULTISYNC GS	630,-
NEC MULTISYNC II	1500,-
Mitsubishi 1481	1385,-

FARBFBÄNDER

Mindestabnahme 3 Stück	
STAR NL/NG/ND/NR-10, Stück	12,-
EPSON LX-800/LQ-500, Stück	8,-
PANASONIC KX-P, Stück	13,-
OKI ML 320, Stück	14,50
OKI ML 390, Stück	16,85
NEC 2200, Stück	13,50
NEC P 6 +P 7 +, Stück	15,30
STAR LC 10, Stück	10,-



DISKETTEN

NN 2DD 3,5" 10 Stck	18,-
NN 2DD 5,25" 10 Stck	9,-
Magix MF 2DD 3,5" 10 Stck	24,50
Select MF 2DD 3,5" 10 Stck	33,50
Select MD 2DD 5,25" 10 Stck	19,-
Fuji MD 2DD 5,25" 10 Stck	27,-

DRUCKER

OKI Microline 390	1448,-
NEC P 6 +	1798,-
NEC P 7 +	1985,-
STAR LC 10 Color	875,-
STAR LC 10, Centr.	599,-
NEC P 2200, Centr.	898,-
Mannesmann Tally mit Einzelblatteinzug	699,-
EPSON LX 800, Centr.	549,-
EPSON LQ 500, Centr.	992,-
EPSON LQ 1050, Centr.	1950,-
AMIGA Drucker-kabel	34,-
Centronics Drucker-kabel	18,80
Weitere Drucker auf Anfrage	

Wir liefern nur mit dt. Handbuch, Seriennummer und Herstellergarantie !!! Drucker-Grauporte mit engl. Handbuch, ohne Seriennummer, ohne Herstellergarantie sind bei uns ausgeschlossen.

Sämtliche Angebote freibleibend, Zwischenverkauf vorbehalten. Wir liefern an Nicht-Kaufleute nur per UPS-Nachnahme mit einem Rückgaberecht innerhalb von 10 Tagen (nur Hard-ware). Ins Ausland nur per Vorkasse. Fordern Sie unseren Gesamtkatalog gegen 3,-DM in Briefmarken. Es gelten unsere allgemeinen Geschäftsbedingungen.