Der Einstieg in die Shell (Folge 2)

Anwender, übernehmen Sie

von Hannes Rügheimer und Christian Spanik

Computer einschalten... Warten... Workbench da! Was passiert in den Sekunden dazwischen? Jede Menge. Wir zeigen es Ihnen und schlagen vor, wie Sie den Systemstart an Ihre Ansprüche anpassen.

Haben Sie sich schon mal gefragt, wie man die im Computer nach dem Einschalten ablaufenden Vorgänge beeinflussen kann? OK, für erste benommene Schritte nach dem stromlosen Tiefschlaf sind Startroutinen im Kickstart-ROM zuständig. Sie schalten die Bildschirmausgabe ein, prüfen, ob der RAM-Speicher ordnungsgemäß funktioniert oder legen dort Verwaltungsdaten ab, die das Betriebssystem benötigt. In diesen Ablauf können Sie zwar eingreifen, aber das macht nur bei wenigen Anwendungen Sinn.

Irgendein Systemprogramm sucht dann aber eine bestimmte Datei auf der Startdiskette bzw. Bootpartition der Festplatte und führt die darin befindlichen Shell-Anweisungen aus. Das ist unsere Chance. Mit der Shell kennen wir uns inzwischen ein wenig aus, und eine Textdatei werden wir doch wohl ändern können.

Die Datei heißt »Startup-sequence«, befindet sich im Verzeichnis bzw. auf dem logischen Datenträger S und ist ein klassischer Vertreter der sogenannten Befehlsfolgen (s. Seite 70). Bei der Namensgebung ist man sich nie einig geworden. Man hört und liest von Skript- und Batch-Dateien, von Sequences oder – spricht man Deutsch – von Befehls-, Stapeldateien oder auch -Sequenzen. Gemeint sind immer Textdateien, die Amiga-DOS- bzw. Shell-Befehle enthalten. EXECUTE, ein spezielles DOS-Programm lädt solche Dateien und führt die darin enthaltenen, programmierten Anweisungen der Reihe nach aus. Das schauen wir uns in der Praxis an.

Wissen Sie noch, wie man ein Shell-Fenster öffnet? Ein Doppelklick auf das entsprechende Piktogramm im Basis-Verzeichnis Ihrer Workbench-Diskette bzw. auf der System-Partition Ihrer Festplatte bringt Sie in diese andere Welt des Amiga. Am besten ziehen Sie das Shell-Fenster auf die volle Größe und geben ein:

```
ed S:startup-sequence
```

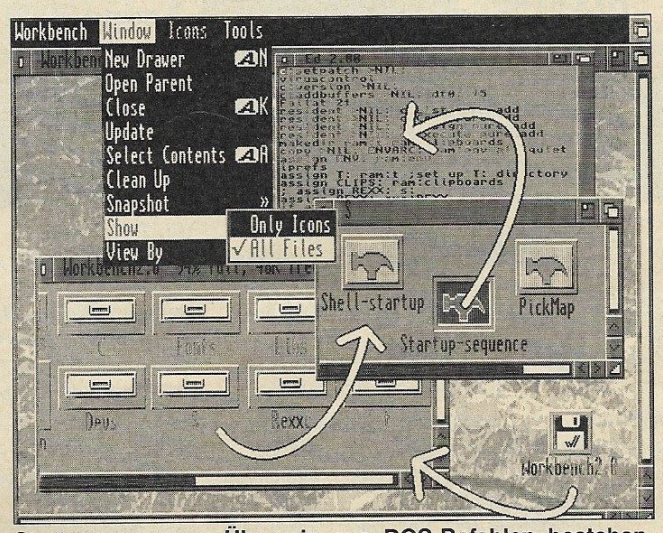
Beim »Ed« – mehr Anwendungsprogramm als DOS-Befehl – handelt sich um den zum Lieferumfang der Systemsoftware gehörigen Texteditor. Wenn Sie einen leistungsfähigeren oder komfortableren Editor wie etwa »Cygnus-Ed« besitzen, sollten Sie den an Stelle des Ed verwenden. Achtung: Skriptdateien müssen auf jeden Fall als ASCII-Text, also ohne Formatinformationen vorliegen, die Textverarbeitungen in der Regel mitspeichern.

In unserem Fall erscheint auf dem Bildschirm ein Fenster mit dem Titel »Ed 2.00« und dem Inhalt der Startup-sequence. Bringen Sie auch dieses Fenster mit der Maus auf maximale Größe.

Im Fenster des Ed sehen Sie nun eine Menge Befehle, und das ist auch der Grund dafür, warum der Amiga nach dem Starten relativ lang auf der Diskette bzw. Festplatte zu tun hat.

Noch eine Warnung, bevor wir Schritt für Schritt nachvollziehen, was die Startup-sequence bewirkt: Sie sollten die Datei vorerst nicht modifizieren, sonst besteht die Gefahr, daß Ihnen der Amiga beim Systemstart eine Bedienungsoberfläche präsentiert, mit der Sie nur eingeschränkt arbeiten können. Wir werden in diesem Text noch genügend Vorschläge für mögliche Änderungen und ihre Umsetzung machen.

Auf der Workbench Version 2.1 befindet sich eine andere Startup-sequence als auf der 2.04. Die entscheidenden Funktionsblöcke sind jedoch bis auf Feinheiten in beiden Versionen gleich.



Startup-sequence: Über ein aus DOS-Befehlen bestehendes Programm beeinflussen Sie den Systemstart

Am Anfang kann eine Zeile stehen, die mit einem Semikolon beginnt und Versionsangaben enthält. Angaben einer Skriptdatei hinter einem Semikolon sind Kommentare. Wenn Sie eine Anweisung nicht ausführen, aus Dokumentationsgründen aber im Skript lassen möchten, können Sie diese durch Semikolons »auskommentieren«.

Der erste Befehlsblock der Startup-sequence sieht so aus:

```
c:setpatch >NIL:
c:version >NIL:
c:addbuffers >NIL: DF0: 15
failat 21
```

SETPATCH bringt einige Systemfunktionen auf den aktuellen Stand. Was da bislang zusammengekommen ist, können Sie sich anschauen, wenn Sie diesen Befehl ohne den Zusatz »>NIL:« in einem Shell-Fenster oder über die Funktion »Workbench/Execute Command« (Befehl ausführen) aufrufen.

NIL ist ein DOS-Gerät [1] wie DF0, PAR oder RAM. Nach NIL geschickte Daten werden allerdings nicht gespeichert oder ausgegeben, sondern vernichtet. NIL ist damit der Mülleimer des DOS. Das Zeichen »>« davor ist ein Umleitungssymbol und bewirkt, daß alle Bildschirmausgaben auf das nachstehende Gerät bzw. die folgende Datei umgeleitet werden. Daten nach NIL umzuleiten bedeutet aber, Daten zu vernichten. Der einzige Nutzen dieser Maßnahme ist also, die Bildschirmausgabe von SETPATCH zu verhindern.

Genauso wird in der nächsten Zeile die Bildschirmausgabe von VERSION ins Nichts geschickt. Sinnvoll ist diese Zeile nur, weil neben der Anzeige der Versionsnummern von Kickstart und Workbench auch die Umgebungsvariablen $Kickstart und $Workbench aktualisiert werden. Darauf kommen wir noch zurück.

Der Befehl ADDBUFFERS richtet für das Startlaufwerk DF0 zusätzliche Datenpuffer ein, in diesem Fall 15 Stück. Jeder dieser Puffer ist 512 Byte groß, so daß 15 Puffer 7680 Byte Speicherplatz kosten. Je größer der gesamte Pufferbereich, desto mehr Daten können in einem Rutsch gelesen werden und desto mit fallen einige der relativ zeitaufwendigen Diskettenzugriffe weg. Auf Amigas mit wenig Speicher könnte es jedoch praktisch sein, die 7,5 KByte einzusparen. Dazu brauchen Sie die Zeile in der Startup-sequence nur mit einem vorangestellten Semikolon zu deaktivieren.

FAILAT stellt die Fehlerbruchschwelle, den »error level« ein. Tritt während der Ausführung eines Skripts ein Fehler auf, bricht EXECUTE normalerweise ab. In der Startup-sequence soll das allerdings nicht passieren, sonst findet sich der Anwender nach dem Systemstart in einem Shell-Fenster wieder, und von dem komfortablen Workbench wie

weit und breit nichts zu sehen. Mit FAILAT legen Sie fest, wie schwerwiegend ein Fehler sein muß, damit EXECUTE abbricht. In diesem Fall werden Fehler bis zur Stufe 20 zugelassen. Darunter fällt z.B. das Fehlen angeforderter Befehle/Programme, Dateien oder Speichermedien.

Der als nächstes (bei Workbench 2.1 etwas später) folgende Befehl

```
resident >NIL: C:execute pure
```

lädt den Befehl EXECUTE und verankert ihn so im Speicher, daß er auch nach dem Ablauf dort bleibt und beim nächsten Aufruf nicht erneut geladen werden muß. Da die Startup-sequence weitere Skripts aufruft, die EXECUTE ja ausführen muß, bewirkt der Befehl RESIDENT hier wieder eine geringe Ablaufbeschleunigung.

MAKEDIR legt nun auf der RAM-Disk einige Arbeitsverzeichnisse an: »t« dient für temporäre Dateien, »clipboards« für Daten, die mit »Cut, Copy & Paste« von einem Programm zum anderen befördert werden sollen. In »env« bzw. »env/sys« kommen die bereits erwähnten Umgebungsvariablen sowie die Einstellungen der Preferences-Editoren. Die nachfolgende Zeile

```
copy >NIL: ENVARC: ram:env all quie t
noreq
```

kopiert diese von Diskette oder Festplatte, dem »Env-Archive« (logischer Datenträger ENVARC) dorthin.

Bei all diesen Anweisungen sorgt übrigens wieder der Zusatz »>NIL:« dafür, daß eventuelle Bestätigungen oder Fehlermeldungen nicht auf dem Bildschirm erscheinen. Die beim Systemstart sichtbaren Meldungen sollen so auf wichtige, aussagekräftige Texte beschränkt werden. Andernfalls würde vielleicht eine Meldung inmitten all der sonst erscheinenden Status- und Standardausgaben nicht mehr wahrgenommen.

Die folgenden ASSIGNs (siehe erste Folge dieses Kurses) weisen nun bestimmten vom System verwendeten Verzeichnissen Namen wie ENV, T, CLIPS oder REXX zu. Unter Workbench 2.1 gibt es zusätzlich die logischen Datenträger PRINTERS, KEYMAPS und LOCALE. Sie werden benötigt für die unter dieser Systemsoftware etwas anders gelöste Auswahl von Druckertreibern und Tastaturbelegungen sowie für die erst ab 2.1 vorhandene Auswahl der Sprache für Systemmeldungen und -menüs.

Neu in der Startup-sequence 2.1 ist der Block

```
if not exists SYS:fonts
   assign FONTS:
endif
```

Systemabläufe automatisieren mit Skripts

Normalerweise erwartet das System Zeichensätze im Verzeichnis fonts auf SYS. SYS ist der Name des der Startdiskette oder Festplatten-Bootpartition zugeordneten logischen Datenträgers. Basiert das Betriebssystem 2.x aber auf Disketten, befindet sich fonts auf einer separaten Diskette. Über IF (Wenn...) programmieren Sie Blöcke, die nur ausgeführt werden, wenn bestimmte Bedingungen erfüllt sind. Die Bedingung folgt auf IF und lautet in diesem Fall »not exists SYS:fonts« (... es das Verzeichnis SYS:fonts nicht gibt).

Ist die Bedingung erfüllt, führt EXECUTE alle Anweisungen zwischen IF und ENDIF aus – und hebt in diesem Fall die Zuweisung »FONTS: sys:fonts« auf. Das führt dazu, daß der Amiga die gleichnamige Diskette anfordert, wenn irgendein Programm Daten vom (logischen) Datenträger »FONTS« benötigt. Wurde die Workbench hingegen auf Festplatte installiert, findet IF das gesuchte Verzeichnis – die Aufhebung unterbleibt also.

Die Konstruktion IF...ENDIF begegnet uns gleich wieder im nächsten größeren Block:

```
if exists SYS:monitors
   join >NIL: SYS:monitors/~(#?.in
fo). as T:mon-start
   execute T:mon-start
   delete >NIL: T:mon-start
endif
```

Auf der Workbench 2.1 ist das Verzeichnis »Monitors« im Verzeichnis Devs untergebracht, und die zweite Zeile dieses Blocks ist ein wenig anders aufgebaut. Die grundsätzliche Funktion ist jedoch bei beiden Versionen identisch: Die Monitor-Skripts, die Sie aus der Schublade MonitorStore in Monitors gezogen haben, werden (ohne die .info-Dateien) durch den Befehl JOIN (oder LIST unter 2.1) zu einer Skriptdatei namens »Mon-Start« (M unter 2.1) zusammengefaßt. Dieses von der Startup-sequence selbst erzeugte Skript führt EXECUTE danach aus. Dadurch sorgt das System dafür, daß die von den jeweiligen Monitoren (PAL-Monitor, NTSC-Monitor, VGA-Monitor, Multiscan, A 2024 und seit Version 2.1 noch spezielle 36- und 72-Hz-Modi der Multiscan-Monitore) unterstützten Grafikmodi dem System und damit den Anwendungsprogrammen bekannt sind. Mon-Start bzw. M werden nur temporär benötigt, deshalb auf T erzeugt und nach der Aktion mit DELETE wieder gelöscht.

Interessant in diesem Zusammenhang: Erstens sehen Sie, daß sich Skripts aus Skripts aufrufen lassen und gegebenenfalls sogar eigene Skriptdateien erzeugen. Nach Ausführung eines externen Skripts macht EXECUTE hinter dem Aufruf weiter. Und zweitens ist es offensichtlich über die Skript-Programmierung möglich, relativ komplexe Funktionen zu realisieren.

Der nächste Befehl, BINDDRIVERS, aktiviert die in der Schublade »Expansion« vorhandenen Treiberprogramme. Das gilt insbesondere für die »Janus.library«, die eine eventuell eingebaute MS-DOS-Brückenkarte unterstützt. Davon abgesehen ist die Methode, Treiberprogramme in Expansion zu legen, kaum mehr üblich. Zur Einbindung von Gerätetreibern dient in der Regel der Befehl MOUNT, der uns zumindest in der Startup-sequence 2.1 auch gleich in der nächsten Zeile begegnet. Mit

```
mount >NIL: DEVS:DOSDrivers/~(#? .i
nfo)
```

werden die in der Schublade »DEVS:DOSDrivers« enthaltenen Gerätetreiber (aber nicht die dazugehörigen .infos) eingebunden. Unter Workbench 2.1 können Sie Treiber von DOS-Geräten wie z. B. PIPE, AUX und RAD sowie PC0 und PC1 des mitgelieferten Hilfsprogramms »CrossDOS« (zum Lesen und Schreiben von MS-DOS-Disketten) durch Ablegen entsprechender Piktogramme in DOSDrivers installieren. Bei den vorherigen Systemversionen sind die Kenndaten jedes Gerätetreibers in die Datei DEVS:Mountlist aufzunehmen, und danach auch mit MOUNT einzubinden.

Über Umgebungsvariablen informieren sowohl das System, aber auch beliebige Programme andere Software über bestimmte Zustände bzw. geben Voreinstellungen bekannt.

```
setenv Workbench $Workbench
setenv Kickstart $Kickstart
```

z.B. übertragen die vom DOS-Befehl VERSION bisher nur lokal, also nur für diese Shell bzw. den Ablauf der Startup-sequence angelegten Umgebungsvariablen Workbench und Kickstart in globale Variablen. Die Startup-sequence 2.1 löscht die lokalen Variablen anschließend mit dem unter 2.1 hinzugekommenen Befehl UNSET. Globale Umgebungsvariablen sind übrigens Dateien auf dem logischen Datenträger ENV, deren Inhalt den Variablenwert darstellt.

Anschließend startet das Dienstprogramm »IPrefs«, das unauffällig im Hintergrund darauf lauert, daß jemand die Systemeinstellungen über die Preferences-Editoren ändert. Dann schickt IPrefs eine entsprechende Nachricht an das System, das alle Programme darüber informiert, die sich dafür interessieren.

ECHO in der nächsten Zeile der Startup-sequence bringt die Versionsnummern der installierten Kickstart und Workbench auf den Bildschirm. Als nächstes startet EXECUTE das Programm CONCLIP im Hintergrund, das dafür sorgt, daß Sie in Shell-Fenstern Textpassagen mit der Maus markieren, mit der Tastenkombination <Amiga_links_c> zwi-

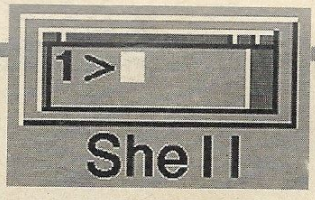
Kursübersicht

Die Dokumentation der Modelle Amiga 500, 600 und 2000 mit OS 2.0 enthält nur eine knappe Beschreibung der Shell, der textorientierten Bedienerschnittstelle des Computers. Viele Systemfunktionen sind aber nur über die Shell erreichbar. Der sichere Umgang damit erleichtert jedem Anwender die Installation neuer Hard- und Software. Dieser dreiteilige Kurs informiert Sie über wichtige Konzepte sowohl der Shell als auch des Betriebssystems.

Folge 1: Über »Execute Command« zur Shell; grundlegende Bedienung der Shell; Programme starten; Übergabeparameter nutzen; Zeichensatz ändern mit SETFONT; die Befehle CD, DIR, LIST und DELETE; Mülleimer löschen; logische Datenträger; der Befehl ASSIGN; Verzeichnishierarchien.

Folge 2: Suchpfade einrichten mit PATH; Skript- bzw. Batchprogrammierung; Modifikation von Startup-Sequence, User-Startup und Shell-Startup; die Skriptdateien PCD, SPAT und DPAT.

Folge 3: Goddies der Shell; die nichtflüchtige RAM-Disk RAD; andere Geräte (PIPE:, SPEAK:, PAR:, PRT:); der Befehl MOUNT; Schutzbits; Hard- und Softlinks mit MAKELINK; Tipparbeit sparen mit CONCLIP; die Befehle STATUS, ALIAS und RESIDENT.



schenspeichern (kopieren) und mit <Amiga links v> an der aktuellen Cursorposition wieder einfügen können. Bei der Workbench 2.04 ohne Schublade DOSDrivers« werden die Geräte SPEAK, AUX und PIPE anschließend direkt geMOUNTed.

Weiter geht es mit PATH. Der Befehl erweitert den »Suchpfad«, das ist eine Liste von Verzeichnissen, die das System nach den System- oder Anwendungsprogrammen durchsucht, deren Namen Sie in der Shell eingeben. Ohne PATH sucht Amiga-DOS nur im aktuellen Verzeichnis und auf dem logischen Datenträger C. PATH in der Startup-sequence erweitert die Liste um die Verzeichnisse Utilities, Rexxc, System, S, Prefs, WBStartup, Tools und Tools/Commodities.

Nach Einstellung des länderspezifischen Zeichensatzes folgt dieser Block:

```
if exists S:user-startup
   execute S:user-startup
endif
```

Hier prüft das System, ob es eine Skriptdatei »User-Startup« auf dem logischen Datenträger S gibt. Wenn ja, wird diese Datei mit EXECUTE ausgeführt. Nutzen Sie die User-Startup, um eigene Einstellungen und Zuweisungen für den Systemstart aufzunehmen. Dazu kommen wir gleich noch.

Bei der 2.1 wird das im Speicher verankerte EXECUTE mit RESIDENT..REMOVE wieder entfernt. LOADWB schließlich sorgt dafür, daß die eigentliche Workbench-Oberfläche erscheint. Sobald das erledigt ist, entfernt ENDCLI das beim Systemstart automatisch erschienene Shell-Fenster. Übrig bleibt wie gewohnt die grafische Benutzeroberfläche. Eine Shell erscheint erst wieder nach Anklicken des entsprechenden Piktogramms.

■ Nach dieser ausführlichen Erkundung der Startup-sequence haben Sie sicherlich schon eine bessere Vorstellung davon, was beim Systemstart Ihres Amiga alles passiert. Auch wenn Sie nun einige Ideen dafür haben, wie Sie die Startup-sequence ergänzen oder modifizieren könnten – lassen Sie es bitte. Die Startup-sequence sollte möglichst unangetastet bleiben. Trotzdem gibt es natürlich eine weitere Möglichkeit, eigene Befehle in die Startprozedur des Amiga einzufügen: die User-Startup.

Wenn Sie den Editor zum Ansehen des Skripts gestartet haben, sollten Sie diesen nun mit »Projekt/Quit« wieder beenden. Falls er Sie auffordert, den Verlust irgendwelcher Änderungen zu bestätigen, tun Sie das getrost mit der Taste <y>. Wenn es noch keine User-Startup auf S gibt, wird es Zeit, sie anzulegen. Nach

```
ed S:User-Startup
```

erscheint entweder dessen Inhalt oder die Meldung »Creating new file« (lege neue Datei an).

Besitzen Sie mehrere Diskettenlaufwerke, können Sie diese mit separaten ADDBUFFERS beschleunigen. Beispiel:

```
addbuffers >NIL DF1: 15
```

Möchten Sie in Zukunft nach dem Einschalten des Amiga von ihm begrüßt werden? Tragen Sie

```
ECHO "Hallo, lieber Anwender"
```

mit entsprechender Namensangabe ein.

Die geänderte Fassung der User-Startup speichern Sie mit »Project/Save«. Überprüfen Sie aber vorher, ob Ihre Eingaben frei von Tippfehlern sind.

Prädestiniert für die User-Startup sind weiterhin ASSIGN-Anweisungen, wie wir sie in der vorangegangenen Kursfolge vorgeführt haben. So könnte man den Zugriff auf Bilder im tief verschachtelten Unterverzeichnis »work:grafik/malen/dpaint/bilder« durch Zuordnung des logischen Datenträgers BILDER wesentlich erleichtern.

Die etwas andere Eingabeaufforderung

Wollen Sie, daß beim Systemstart automatisch das Uhrenprogramm »Clock« auf der Workbench erscheint? Tragen Sie in der User-Startup die folgende Zeile ein:

```
run >NIL: SYS:utilities/clock digit al
left 480 top 0
```

Denken Sie daran, daß Änderungen der User-Startup erst nach dem nächsten Systemstart wirken.

Neben der User-Startup bietet die »Shell-Startup« weitere Möglichkeiten, Arbeitsvoraussetzungen zu schaffen bzw. zu ändern. Die Shell-Startup wird bei jedem Start einer Shell ausgeführt.

```
ed S:Shell-Startup
```

bringt die Datei auf den Bildschirm. Sie sehen viele ALIAS und einen PROMPT. ALIAS verwaltet Abkürzungen für komplizierte Befehlssequenzen wie etwa CLEAR für »ECHO "*E[0;0H*E[J"«. Auch für oft benötigte Befehle wie DELETE oder RENAME können Sie selbstdefinierte Kürzel wie DEL und REN vereinbaren.

Den Prompt schließlich haben wir bereits in der letzten Kursfolge kennengelernt. Mit dem Befehl PROMPT ändern Sie dessen Zusammensetzung. Der Platzhalter »%N« steht dabei für die Nummer des aktuellen Shell-Fensters und »%S« für den Pfad des aktuellen Verzeichnisses.

Wir rufen z.B. unseren Editor mit dem Kürzel CED auf – in der Shell-Startup befindet sich die Anweisung

```
alias ced work:Programme/Editor/Cyg
nusEd
```

Hier noch ein paar Vorschläge für eine persönliche Eingabeaufforderung:

```
prompt "[%N] %S> "
prompt "Was wünschen Sie? > "
prompt "*E[2m[%N] %S>*E[0m "
```

Standardmäßig ist nach Aufruf eines Shell-Fensters über das Piktogramm das Basisverzeichnis der Systempartition (SYS) aktuelles Verzeichnis. Wenn Sie jedoch häufiger an anderer Stelle Ihre Shell-Arbeiten verrichten, fügen Sie in die Shell-Startup am besten einen CD-Befehl ein:

```
cd work:
```

Nach den entsprechenden Einfügungen speichern Sie die veränderte »Shell-Startup« mit dem Befehl »Project/Save« und beenden den Ed mit »Quit« – beim nächsten Aufruf eines Shell-Fensters sind die Einstellungen aktiv.

■ Zu den von Commodore im Verzeichnis S der Workbench gespeicherten Skripts gehört PCD (previous current directory), das den Befehl CD simuliert. Sie wechseln wie mit CD über

```
c:setpatch >NIL:
c:version >NIL:
addbuffers >NIL: df0: 15
Failat 21
Resident >NIL: C:Execute PURE ADD
makedir ram:T ram:Clipboards ram:env ram:env/sys
copy >NIL: ENVARC: ram:env all quiet noreq
assign ENV: ram:env
assign T: ram:t ;set up T: directory for scripts
assign CLIPS: ram:clipboards
assign REXX: s:
if exists sys:Monitors
    join >NIL: sys:monitors/~(#?.info) as t:mon-
start
    execute t:mon-start
    delete >NIL: t:mon-start
endif
BindDrivers
setenv Workbench $Workbench
setenv Kickstart $Kickstart
IPrefs
echo "Amiga Release 2.  Kickstart $Kickstart, Work-
bench $Workbench"
conclip
mount speak:
mount aux:
mount pipe:

path ram: c: sys:utilities sys:rexxc sys:system s:
sys:prefs sys:wbstartup add
if exists sys:tools
    path sys:tools add
    if exists sys:tools/commodities
        path sys:tools/commodities add
    endif
endif
; If this is the initial boot (i.e. keyboard env
variable is not set)
; then execute PickMap which will query for a keym-
ap and set the
; keyboard env variable.
;
; if keyboard env variable is set, set the keymap
if ${sys/keyboard} NOT EQ "*${sys/keyboard}"
    setmap ${sys/keyboard}
else
    PickMap sys: initial
endif
if exists s:user-startup
    execute s:user-startup
endif
LoadWB
endcli >NIL:
```

Startup 2.04: So beginnt das Betriebssystem Version 2.04

```
PCD <Verzeichnis>
```

aus dem aktuellen Verzeichnis in ein anderes, z.B. die RAM-Disk. Wenn Sie anschließend aber PCD ohne weitere Zusätze ausführen, stellt das Skript das vormals (previous) aktuelle Verzeichnis wieder ein. Diese Methode ist ideal für kurze Ausflüge in andere Verzeichnisse, von denen Sie später wieder an den Ausgangspunkt zurück möchten. Schauen Sie sich die Textdatei PCD einmal mit

```
type S:pcd
```

an. Interessant ist vor allem die erste Zeile:

```
.key dir
```

».key« durchsucht den im Shell-Fenster eingegebenen Aufruf des Skripts nach Zeichenfolgen hinter dem Skriptnamen, und ordnet diese den hinter .key angegebenen Platzhaltern zu. Über die Platzhalter bzw. Variablen greifen die Anweisungen des Skripts auf die Angaben – praktisch übergebene Parameter – der Befehlszeile zu. Damit die Platzhalter in den weiteren Anweisungen nicht mit Dateinamen verwechselt werden, gehören ihre Namen in spitze Klammern.

Nicht zuletzt als Beispiel dafür, was man mit Skripts alles machen kann, liefert Commodore außerdem die beiden Dateien SPAT und DPAT mit. SPAT etwa ermöglicht, Befehle mit Platzhaltern (Wildcards, Joker) zu verwenden, die dafür eigentlich gar nicht ausgelegt sind. Da ab Workbench 2.04 die meisten AmigaDOS-Befehle solche Platzhalter unterstützen (TYPE #?, COPY #? TO ..., DELETE #?), wird diese ab Workbench 1.3 vorhandene, und früher oft hilfreiche Lösung nur noch selten verwendet. DPAT hat im Prinzip dieselbe Funktion, ist aber für Befehle ausgelegt, die zwei Parameter benötigen (FILENOTE #? <Datei> oder PROTECT #? +rwed). Auch sie ist mittlerweile durch Verbesserungen der DOS-Befehle fast überflüssig. Für den Aufruf so manches PD-Programms sind sie aber weiter nützlich. Beispiel:

```
spat MapIcon #?.info
```

oder

```
spat MuchMore #?.doc
```

■ Bis jetzt haben wir nur vorhandene Skriptdateien untersucht. Es wird Zeit, ein eigenes Skript zu programmieren. Stellen Sie sich vor, Sie nutzen Ihren Amiga regelmäßig für die Arbeit mit dem DTP-Programm »Professional Page« und wünschen, daß das Programm direkt nach dem Einschalten so einfach wie möglich zu starten ist. Manchmal erledigen Sie aber auch andere Arbeiten auf dem Amiga, und dann soll der Computer nach dem Start die Workbench für den Aufruf anderer Programme zeigen. Mit DOS-Befehlen programmieren Sie sich eine kleine Menü-Auswahl.

Programmstart über die Tastatur

So ein Skript sollte zweckmäßigerweise in die User-Startup integriert werden und könnte etwa so aussehen:

```
run newshell
lab Menu
echo "Bitte wählen Sie:"
echo "1 = Workbench starten"
echo "2 = DPaint laden"
setenv >NIL: Auswahl ?
if $Auswahl eq "2"
   echo "DPaint wird geladen
   DPaint:DPaint
endif
if not $Auswahl eq "1"
   skip back Menu
endif
```

Auf LAB folgt ein sogenanntes Label, eine »Sprungmarke«. Damit kennzeichnen Sie eine bestimmte Zeile im Skript, die Sie später von anderer Stelle anspringen können. Die drei ECHOs danach bringen die Auswahlmöglichkeiten auf den Bildschirm.

Der SETENV ist schon trickreicher. Hier geht es darum, eine Tastatureingabe anzufordern. Da Amiga-DOS dafür eigentlich keinen Befehl besitzt, helfen wir uns mit SETENV, das für die direkte Zuweisung von Zahlen oder Begriffen an Umgebungsvariablen gedacht ist. Durch das Fragezeichen am Ende der Zeile bringt Amiga-DOS normalerweise die Hilfsinformationen dieses Befehls auf den Bildschirm. Diese werden wir allerdings nicht sehen, weil Sie mit der Umleitung >NIL: unterdrückt werden. Der Vorteil ist jedoch, daß Amiga-DOS im Anschluß an solche Hilfsausgaben die Eingabe der Parameter für den jeweiligen Befehl erlaubt – und damit haben wir genau, was wir brauchen: Der Umgebungsvariablen »Auswahl« wird die Tastatureingabe zugewiesen.

Anschließend müssen wir nur noch feststellen, was der Benutzer eingegeben hat. Um den Wert einer Umgebungsvariablen abzufragen, müssen Sie ihr ein Dollarzeichen voranstellen.

```
IF $Auswahl EQ "2"
```

Wir prüfen hier, ob die Variable $Auswahl den Wert 2 enthält. Trifft dies zu, wird der Teil innerhalb der IF...ENDIF-Klammer ausgeführt: DPaint wird geladen – falls es sich unter diesem Namen und an dieser Stelle auf Festplatte oder Diskette befindet.

War die Eingabe für $Auswahl nicht 2, macht EXECUTE beim nächsten IF...ENDIF weiter: Hier prüfen wir, ob die Eingabe 1 war. Trifft dies nicht zu, sollten wir den Anwender um eine neue Eingabe bitten, denn dann hat er irgendet- was anderes als 1 oder 2 einge- tippt. Daher prüfen wir nur kurz ob die Eingabe für $Auswah nicht gleich 1 (NOT $Auswahl EQ "1") war. Falls etwas anderes eingegeben wurde, lassen wir EXECUTE zurück zum Anfang springen. Solche Programmverzweigungen führt SKIP aus. Wenn Sie damit zu einem Label (LAB) verzweigen, das sich im Skript weiter oben befindet, müssen Sie zusätzlich den Parameter BACK angeben. In unserem Fall also SKIP BACK Menu.

Trifft auch die zweite Bedingung nicht zu, hat der Anwender wohl 1 eingegeben. In diesem Fall endet die Skriptausführung. Wir gehen ja davon aus, daß sie innerhalb der User-Startup abläuft. Daher wird in diesem Fall der Rest von User-Startup und anschließend der Rest der Startup-sequence abgearbeitet und damit die Workbench geladen. Und genau das wollen wir ja.

Sie können die Beispieldatei um weitere Menüpunkte erweitern. Dafür sind zunächst neue ECHO-Anweisungen mit zusätzlichen Optionen zu ergänzen:

```
echo "3 = Art Department laden"
```

Außerdem gehören dann hinter SETENV neue Abfrageblöcke:

```
if $Auswahl eq "3"
   echo "Art Department wird gela
den"
   "work:Art Department/ADPro"
endif
```

Möchten Sie lernen, wie man komplexe Skripts programmiert, sollten Sie nochmal einen Blick auf die von Commodore mitgelieferten Beispiele (PCD, SPAT, DPAT, PickMap...) werfen.

Literaturhinweis:
[1] Rügheimer/Spanik: Wir entrümpeln die Workbench; AMIGA-Magazin 12/92, Seite 17
[2] Peter Aurich: Amiga-DOS und die Shell; AMIGA-Magazin 9/91, Seite 76

```
; $VER: startup-sequence 38.22 (24.4.92)
C:SetPatch QUIET
Version >NIL:
AddBuffers >NIL: DF0: 15
FailAt 21
MakeDir RAM:T RAM:Clipboards RAM:ENV RAM:ENV/Sys
Copy >NIL: ENVARC: RAM:ENV ALL NOREQ
Assign >NIL: ENV: RAM:ENV
Assign >NIL: T: RAM:T
Assign >NIL: CLIPS: RAM:Clipboards
Assign >NIL: REXX: S:
Assign >NIL: PRINTERS: DEVS:Printers
Assign >NIL: KEYMAPS: DEVS:Keymaps
Assign >NIL: LOCALE: SYS:Locale
IF NOT EXISTS SYS:Fonts
  Assign FONTS:
EndIF
BindDrivers
Mount >NIL: DEVS:DOSDrivers/~(#?.info)
Resident >NIL: C:Execute PURE
IF EXISTS DEVS:Monitors
  List >NIL: DEVS:Monitors/~(#?.info) TO T:M LFORMAT
"DEVS:Monitors/%s"
  Execute T:M
  Delete >NIL: T:M
EndIF
SetEnv Workbench $Workbench
SetEnv Kickstart $Kickstart
UnSet Workbench
UnSet Kickstart
IPrefs
Echo "Amiga Release 2.1.  Kickstart $Kickstart,
Workbench $Workbench"
ConClip
Path >NIL: RAM: C: SYS:Utilities SYS:Rexxc SYS:System S: SYS:Prefs SYS:WBStartup SYS:Tools
SYS:Tools/Commodities
IF EXISTS S:User-Startup
  Execute S:User-Startup
EndIF
Resident Execute REMOVE
LoadWB
EndCLI >NIL:
```

Startup 2.1: So legt das Betriebssystem Version 2.1 los