Amiga-FAQ

Häufig gestellte Fragen und Antworten

10 October 1994

Ignaz Kellerer

Georg-Habel-Str. 11
81241 München (Deutschland)
Tel. (+49) 089 / 885147
Internet: kellerer@informatik.tu-muenchen.de

Dieses Dokument enthält einige häufig gestellte Fragen und versucht, Antworten zu geben. Seine Absicht ist es, neuen Benutzern zu helfen und die Zahl der News-Artikel zu reduzieren, die erfahrene Benutzer nicht mehr sehen wollen.

Der Autor gibt **absolut keine** Garantie, daß die hier gegebenen Antworten korrekt sind. Einige dieser Antworten sind von anderen Benutzern beigetragen worden und ich habe teilweise nicht einmal die Möglichkeit, auch nur die einfachsten Tests durchzuführen. Vorschläge, weitere Beiträge, neue Fragen und Antworten, Kritik und Beschimpfungen (oh, wie ich mein '`nil:`' liebe :-) sind aber sehr willkommen.

Bitte beachten Sie, daß viele Abschnitte bis jetzt noch fast völlig leer sind. Ich verstehe von den betreffenden Abschnitten nichts und habe nicht das Gefühl, als ob ich daran etwas ändern könnte. Damit sind Sie an der Reihe: Füllen Sie die Lücken und teilen Sie mir mit, was man hier noch aufnehmen kann!

1 Hardware

Dieses Kapitel enthält Fragen zur Amiga-Hardware.

1.1 Was sind 68EC020, 68EC030 und 68LC040?

Motorola, die Herstellerfirma der 680x0-Familie, bietet auch gestutzte Versionen ihrer Prozessoren an. Diese sind etwas preisgünstiger als die Originale, weshalb Commodore auch den 68EC020 in den A1200 und den 68EC030 in den A4000/030 einbaut. Dafür können sie aber auch etwas weniger.

Der Unterschied zwischen 68020 und 68EC020 ist, daß letzterer nur 16MB Speicher adressieren kann. (Das ist der Grund, warum der A1200 nur maximal 10MB RAM haben kann.) In den meisten Fällen sollte man den Unterschied aber gar nicht bemerken.

Das ist beim 68EC030 anders: Viele Benutzer werden feststellen, daß der 68030 eine MMU (Memory management unit) hat und der 68EC030 nicht. (Beim 68020 gibt es noch die Möglichkeit, eine externe MMU, den 68851 nachzurüsten.) Es gibt einige wichtige Programme, die eine MMU brauchen, z.B. Enforcer (ein Hilfsprogramm zur Fehlersuche), GigaMem (ein Programm zur Emulation von virtuellem RAM) oder alle aktuellen Unix-Versionen (siehe Abschnitt 6.1 [Unix], Seite 28). Wer eines dieser Programme verwenden will und keinen A4000 oder A3000 hat, braucht deshalb eine Prozessorkarte mit einem 68030 oder 68040.

Der 68LC040 ist ein 68040 ohne FPU. Siehe Abschnitt 1.2 [FPU], Seite 1.

1.2 Was ist ein mathematischer Coprozessor (FPU) ?

Die Prozessoren 68000 bis 68030 bieten ausschließlich Integer (= Ganzzahl) Arithmetik. Floating-Point (Fließkomma) Operationen werden über eine Befehlssequenz ausgeführt. Floating-Point Units (FPUs) bzw. mathematische Koprozessoren sind für diese Aufgaben optimiert.

Im wesentlichen kann man drei FPU-Typen unterscheiden: Den 68881, 68882 und die interne FPU des 68040. Aufgrund der Trennung von Arithmetikeinheit und Konvertierlogik (notwenig zur Konversion anliegender Zahlenformate in das prozessoreigene 80bit-Format) ist die 68882 FPU bis zu 1.5 mal schneller als die 68881. Die interne FPU des 68040 besitzt darüber hinaus eine dreistufige Pipeline, allerdings sind in ihr nicht alle Befehle der 68881/68882 realisiert. Die fehlenden (trigonometrischen) Befehle werden per Software emuliert (z.B. über die 68040.library).

Für viele Programme (Raytracing, DTP, Mathematik, TeX) existieren spezielle Coprozessorversionen. Je nach Auslastungsgrad kann damit eine Geschwindigkeitssteigerung bis zum Faktor 50 erreicht werden.

Michael Kaiser (kaiser@ira.uka.de)

1.3 Kann ich eine 3.5"-Festplatte im A1200 verwenden?

Viele Leute würden statt der im A1200 üblichen 2.5"-Festplatten lieber eine 3.5"-Festplatte verwenden, da diese viel billiger sind. Das ist auch möglich, allerdings braucht man ein spezielles Kabel, um die Festplatte an den eingebauten IDE-Controller anzuschließen. Außerdem sollte man isolierendes Material zwischen die Festplatte und die Platine schieben. Es gibt Berichte über thermische Probleme, aber ich habe nichts davon bemerkt.

Einige Händler bieten für ca. 40-50DM Kabel, Isoliermaterial und Informationen an, was sicherlich empfehlenswert ist. In den üblichen Magazinen sollte man diesbezügliche Anzeigen finden.

Thomas Schuh (thomas@stepout.tynet.sub.org)

2 Das Betriebssystem

Dieses Kapitel behandelt Fragen sowohl zum Betriebssystem Kickstart als auch zur Oberfläche Workbench.

2.1 Kann ich eine andere als die eingebaute Kickstart benutzen?

Zunächst ein paar Worte zur Legalität der Benutzung einer anderen Kickstart: Es ist **nicht** erlaubt, Kopien von Kickstarts fremder Computer zu erstellen und auf einem anderen Computer zu benutzen! (Es ist sogar fraglich, ob man das auf eigenen Maschinen darf, wenn man mehr als einen Amiga hat.)

Natürlich ist es möglich und für gewisse Personen (z.B. Developer, siehe Abschnitt 3.4 [Developer], Seite 8) auch erlaubt. Es gibt zwei verschiedene Möglichkeiten, eine Hardware- und eine Softwarelösung. Erstere ist, eine Platine in den Computer einzubauen, die sowohl die eigene als auch eine andere Kickstart aufnehmen kann, zwischen denen man dann umschalten kann.

Die Softwarelösung benötigt ein Programm (einen sogenannten Softkicker) und die Kopie der anderen Kickstart. Der Softkicker alloziert RAM, lädt die Kickstart-Kopie in dieses RAM und führt einen Reset aus. Natürlich fehlt dann nach dem Reboot unter der neuen Kickstart etwas RAM: 256KB weniger für Kickstart 1.2 oder 1.3 und 512KB für die neueren Kickstarts. Es gibt verschiedene Softkicker, von denen einige eine MMU benötigen (siehe Abschnitt 1.1 [MMU], Seite 1). Ich empfehle SKick 3.43 (Aminet, Directory '`util/boot`') empfehlen, das keine MMU braucht und viele verschiedene Kickstarts unterstützt. Die Kopie des ROM zu erzeugen ist mit einem Programm wie dem Folgendem möglich:

```
#include <stdio.h>

#define kickorig 0xf80000   /*  0xfc0000 für Kick 1.2 und 1.3   */
#define kicklen  0x080000   /*  0x040000 für Kick 1.2 und 1.3   */
```

```
void main(int argc, char*argv[])
{
    FILE *fh;

    if ((fh = fopen("kickstart.file", "w"))  !=  NULL) {
        result = fwrite(kickorig, kicklen, 1, fh);
    }
    fclose(fh);
}
```

2.2 Was entspricht unter AmigaDOS dem . (Aktuelles Directory)?

Unter AmigaDOS entspricht dem . (der das aktuelle Directory unter Unix und gewissen nicht-reentranten Interrupt-Handlern repräsentiert) der leere String, den man als `""` schreibt.

Beispiel:

```
COPY S:Startup-Sequence ""
```

Dies kopiert Ihre Startup-Sequence in das aktuelle Directory.

Es gibt verschiedene Programme, die AmigaDOS so patchen, daß es . und .. wie unter Unix akzeptiert, z.B. UnixDirs. (Aminet, '`os20/util/UnixDirsII.lha`' oder Fish-Disk 837)

Arno Eigenwillig (arno@yaps.dinoco.de)

2.3 Der Queue-Handler PIPE:

In AmigaOS 2.04 wurde ein neuer Handler eingeführt, der es erlaubt, Daten zwischen verschiedenen Programmen auszutauschen. Dieser Handler heißt '`L:Queue-Handler`', ist aber besser als '`PIPE:`' bekannt.

'`PIPE:`' implementiert eine echte, Unix-artige `Pipe`, mit der man die Standardausgabe eines Programms als Standardeingabe eines anderen Programms verwenden kann. Es ist auch möglich, mehrere Programme über mehrere Pipes gleichzeitig zu verketten. Pipes brauchen weniger RAM für temporäre Dateien, und der Zugriff ist schneller.

Allerdings unterscheidet sich '`PIPE:`' von Unix-Pipes in zwei wesentlichen Punkten:

1. Es handelt sich um ein Device; die Ein- und Ausgabe von Programmen muß also nicht unbedingt eine Datei sein, obwohl das meistens der Fall ist. Man kann auch '`PIPE:`' wie andere Devices verwenden, aber natürlich weder Directories lesen noch Seek-artige Zugriffe durchführen.
2. Es gibt keine Flush-Operation. Wenn nicht alle Daten gelesen wurden, die in '`PIPE:`' geschrieben wurden, dann bleiben sie stehen, bis sie evtl. von einem anderen Programm gelesen werden. Man muß also Pipes immer leeren, bevor man sie schließt.

3. Aus demselben Grund kann ein Programm blockiert werden, wenn es die interne Puffergröße überschreibt. Auch dies wird durch rechtzeitiges Leeren der Pipe vermieden.

Der Handlername von PIPE: ist vollständig '`PIPE:name/bufsize/bufnum`', wobei '`name`' den verwendeten Pipekanal identifiziert und eindeutig sein sollte. Durch verschiedene Namen kann man also gleichzeitig mehrere Pipe-Kanäle öffnen. Die optionalen Argumente `bufsize` und `bufnum` geben die Größe und Anzahl der verwendeten Datenpuffer an. Meist schreibt man einfach '`PIPE:name`', die Vorgabegröße ist dann 4096 Bytes und die Anzahl unbegrenzt.

Osma Ahvenlampi (Osma.Ahvenlampi@hut.fi)

2.3.1 Verwendung von PIPE: in einer AmigaShell

Zunächst muß '`PIPE:`' natürlich mit `Mount` angemeldet sein. Dies kann durch das Kommando

```
1> Mount PIPE:
```

in der Shell geschehen, ab AmigaOS 2.1 auch dadurch, daß man die Datei '`PIPE`' nach '`DEVS:DosDrivers`' schiebt.

In einem AmigaShell-Fenster kann man dann folgende Kommandos eingeben:

```
1> Run List SYS: >PIPE:Listoutput
1> More <PIPE:Listoutput
```

[1]

Diese beiden Kommandos erzeugen also zunächst eine Liste der Dateien in '`SYS:`' und geben diese dann mit Hilfe des More-Kommandos aus. Man könnte auch folgendes probieren:

```
1> Run List SYS: NOHEAD >PIPE:Listoutput
1> Run Sort PIPE:Listoutput PIPE:Sortedoutput
1> More <PIPE:Sortedoutput
```

Dies würde also die Liste vor der Ausgabe noch sortieren.

Beachten Sie die Verwendung von `Run` außer für das jeweils letzte Programm, durch die alle Programme gleichzeitig ablaufen. Man kann auch die Programme gleichzeitig in verschiedenen Shells ablaufen lassen.

Das Leeren der Pipe kann auch manuell geschehen, vorausgesetzt man kennt den Namen des verwendeten Kanals, indem man folgendes eingibt:

```
1> Type PIPE:name TO NIL:
```

2.3.2 Das Pipe-Kommando

Im vorigen Beispiel ist die Verwendung von Pipes recht kompliziert, vor allem verglichen mit Unix-Pipes. Es gibt aber eine bessere Möglichkeit.

[1] Dabei '`1>`' der Prompt der AmigaShell, die eine Eingabe erwartet. Die Nummer kann natürlich variieren.

Andy Finkel, der früher bei Commodore gearbeitet hat, hat auch ein Kommando `Pipe` geschrieben (was man nicht mit '`PIPE:`' verwechseln sollte!), das die Benutzung von Pipes stark vereinfacht. Unglücklicherweise wurde dieses Programm letzten Endes dann doch nicht in die offizielle Workbench aufgenommen, allerdings mit Billigung von Commodore veröffentlicht (Quellen: Fish-Disk 637, Aminet, '`util/cli/finkelshelltools.lha`'). Dieses Programm arbeitet auch unter OS3.1 noch problemlos.

Die Verwendung des Pipe-Kommandos ist einfach. Man übergibt die auszuführenden Programme als Argumente an Pipe, getrennt durch das Zeichen `|`, z.B.

```
1> Pipe List SYS: | More
    oder
1> Pipe List: SYS: NOHEAD | Sort IN: OUT: | More
```

Beachten Sie die Verwendung von '`IN:`' und '`OUT:`', die nötig sind, weil das Sort-Kommando nicht von der Standardeingabe lesen und nicht auf die Standardausgabe schreiben kann. Diese Devices werden durch das Kommandos Pipe simuliert. Durch die Environment-Variable *_pchar* kann man auch ein anderes Zeichen anstelle von `|` verwenden.

2.3.3 Das Pipe-Kommando in der AmigaShell

Es gibt ein sehr nützliches, allerdings undokumentiertes Feature der AmigaShell: Diese kennt das Pipe-Kommando! Ist die lokale Variable *_pchar* gesetzt, dann kann man Pipes sogar ohne Eingabe des Pipe-Kommandos benutzen. Die Shell erkennt das durch *_pchar* vorgegebene Zeichen und ruft für Kommandozeilen, die es enthalten, automatisch Pipe auf.

Am besten wird das folgende Kommando in '`s:Shell-Startup`' eingetragen:

```
1> Set _pchar "|"
```

[2] Man kann die vorigen Beispiele dann so ausführen:

```
1> List SYS: | More
    oder
1> List SYS: NOHEAD | Sort IN: OUT: | More
```

Dies entspricht also völlig den Unix-Pipes.

2.3.4 Die _mchar-Variable

Die lokale Variable _mchar legt das Zeichen fest, das als Kommandotrenner dient. Setzt man also

```
1> Set _mchar ";"
```

dann kann man in der Shell die Befehle hintereinanderfügen, durch den Strichpunkt getrennt.

[2] Die Anführungsstriche sind wesentlich! Wenn *_pchar* bereits gesetzt ist, wird sonst angenommen, es handele sich bereits um einen Aufruf von Pipe. Dies kann etwa dann geschehen, wenn aus einer Shell eine neue gestartet wird.

2.3.5 Bekannte Probleme

F: Ich bekomme die Fehlermeldung

```
PIPE: Unknown command
```

wenn ich das Kommando '`List SYS: | More`' ausführe, obwohl '`PIPE:`' mit `Mount` angemeldet ist. Was ist falsch?

A: Die Shell sucht nach dem Kommando `Pipe` in der Fehlermeldung, nicht nach dem '`PIPE:`'-Device. Dieses Kommando ist also nicht im Suchpfad (meist in '`C:`') installiert.

F: Ein Requester meldet

```
Please insert volume PIPE: in any drive
```

wenn ich eines der Kommandos aus den Beispielen ausführen will.

A: '`PIPE:`' ist nicht angemeldet. Dies kann mit '`Mount PIPE:`' geschehen.

F: Wenn ich eines der Beispiele ausführe, dann eröffnet das `More`-Kommando ein Fenster, aber dort erscheint nichts/ eine Fehlermeldung erscheint/ein Filerequester erscheint.

A: Sie verwenden ein anderes More-Kommando als das aus der Workbench. Verwenden Sie einen anderen Namen oder installieren Sie ein More, das Pipes unterstützt, z.B. das More von Commodore, Less oder Most.

3 Programmierung

In diesem Kapitel finden vermutlich nur Programmierer Interessantes.

3.1 Was ist die beste Dokumentation für Programmierer?

Die beste verfügbare Dokumentation sind sicherlich die RKM's (ROM Kernel Manuals, die schwarzen) von Commodore. Sie werden von Addison-Wesley veröffentlicht.

```
The Amiga ROM Kernel Manual:  Libraries, ISBN 0-201-56774-1
The Amiga ROM Kernel Manual:  Devices, ISBN 0-201-56775-X
The Amiga ROM Kernel Manual:  Includes and Autodocs, ISBN
                                                  0-201-56773-3
The Amiga Hardware Manual, ISBN 0-201-56776-8
The Amiga User Interface Style Guide, ISBN 0-201-57757-7
```

Vor allem die Libraries sind ein Muß. Weniger nützlich sind die Includes und Autodocs: Sie sind auf Diskette als Online-Hilfe sicher nützlicher. Siehe Abschnitt 3.3 [Include-Dateien], Seite 8.

AmigaDOS wird in diesen Büchern kaum behandelt. Die Autodocs geben einige Informationen, aber um tiefer einzusteigen braucht man das

```
The AmigaDOS Manual, 3rd Edition, ISBN 0-553-35403-5
```

ebenfalls von Commodore, das von Bantam Books herausgegeben wird.

Eine gute Wahl ist auch

`The Amiga Guru Book`

von Ralph Babel. Das Buch beginnt mit einem ca. 250-seitigen allgemeinen Überblick über verschiedenste Aspekte der Programmierung des Amiga. Für Anfänger dürften vor allem die Abschnitte über die Amiga-Datentypen sowie die Amiga-Includes und die amiga.lib interessant sein, aber auch Erfahrene finden hier mit Sicherheit noch Neues, was zum Teil nicht einmal in den RKM's enthalten ist. Den größten Teil des Buches nehmen aber 500 Seiten nur zu AmigaDOS ein. Dieser Teil ist meines Erachtens der wichtigste, weil das AmigaDOS-Manual der schlechteste Teil der offiziellen Dokumentation ist. Das Buch ist sehr dicht geschrieben und deshalb weniger leicht lesbar als die RKM's, aber ich empfehle es als Zusatz und vor allem anstelle des AmigaDOS-Manuals. (Allerdings ist das Guru-Buch kein Ersatz für die Libraries & Devices, die hier nicht behandelt werden.) Unglücklicherweise hat das Buch keine ISBN-Nummer und ist deshalb nur bei den folgenden Adressen erhältlich:

```
Hirsch & Wolf OHG; Mittelstrasse 33; D-56564 Neuwied; Germany
Voice: +49 (2631) 8399-0; Fax: +49 (2631) 8399-31
E-Mail: <hhhirsch@carla.adsp.sub.org> (fax preferred)
(Eurocard/Mastercard/VISA)

Periscope; Attn: Cody Lee; 1717 W Kirby Ave; Champaign, IL 61821, USA
Voice: +1 (217) 398 4237; Fax: +1 (217) 398 4238
E-Mail: <periscope@cei.com>

Someware; 27 rue Gabriel Peri; 59186 Anor; France
Voice: +33 27596000; Fax: +33 27595206
E-Mail: <didierj@swad.adsp.sub.org>
```

Für weitere Informationen empfehle ich auch die FAQ von Marc Atkins über Bücher zum Thema Amiga, die alle 4 Wochen in `comp.sys.amiga.misc` erscheint.

3.2 Was ist CATS?

Dies ist eine Abteilung von Commodore West Chester, die früher `Commodore Amiga Technical Support` hieß und später in `Commodore Application and Technical Support` umbenannt wurde. Die Mitglieder arbeiten unabhängig von der Entwicklungsabteilung, aber eng mit ihr zusammen, und versuchen, Entwicklern außerhalb von Commodore beim Erstellen guter Amiga-Anwendungen zu helfen, sei das eine Hard- oder Software. Dazu hat CATS eine Menge an Informationen und Tools gesammelt, auf Floppy, CD oder Papier. Ein großer Teil dieses Materials ist auch der Allgemeinheit, d. h. für Nicht-Developer zugänglich. Aber bitte CATS nicht mit einer Hotline verwechseln!

Amerikaner können das Material von

```
CATS - Developer Applications
Commodore
1200 Wilson Drive
West Chester, PA. 19380
```

bekommen, der Distributor für ganz Europa ist die

```
Fa. Hirsch & Wolf
Mittelstr. 33
56564 Neuwied
Tel. 02631/83990
```

Dr. Peter Kittel, peterk@cbmger.de.so.commodore.com

3.3 Wo bekomme ich die Amiga-Include-Dateien?

Der einzige legale Weg, die Include-Dateien und die AutoDocs zu bekommen (und Sie sollten sie bekommen, sie sind *sehr* nützlich!), ist, Developer zu werden (siehe siehe Abschnitt 3.4 [Developer], Seite 8) oder das sogenannte `NDU` (Native developers update kit, auch als `NDUK`, `NDK` bekannt oder `ADU` für Amiga Developer Update) bei CATS zu kaufen. Siehe Abschnitt 3.2 [CATS], Seite 7. Es kostet etwa 50DM, was sicher ein fairer Preis ist. Dabei handelt es sich um ein 5-Disketten-Set, das neben den aktuellen Includes und AutoDocs auch Tools für Programmierer, z.B. Enforcer, Mungwall, Sushi und die debug.libg (zum Debuggen) sowie anderes, z.B. CatComp (siehe Abschnitt 3.8 [Lokalisierung], Seite 13) oder Report (für Bug-Reports oder Vorschläge an Commodore) enthält. Die aktuelle Version ist 3.1 und für Programmierer beinahe obligatorisch.

Wem die Includes genügen, der kann diese über ein Update seines Compilers (nur kommerzielle Compiler) oder von den Fish-CDs bekommen. Siehe Abschnitt 8.6.2 [Fish-CD], Seite 37.

3.4 Wie werde ich Developer?

Um Developer zu werden, braucht man die `ADSP`-Antragsformulare (Amiga developer support program). Um diese zu bekommen, sollte man einen Brief an die lokale Commodore-Niederlassung schreiben und nach diesen Papieren fragen, in denen das weitere erklärt wird. In Deutschland ist die Adresse folgende:

```
Commodore
Lyoner Straße 38
60528 Frankfurt
```

Es gibt drei verschiedene Arten von Developern:

Registered

Hier bekommt man vor allem Zugang zum CBMNET (eine Art Commodore-internes Usenet), über das man direkt mit anderen Entwicklern auch von Commodore selber über seine Probleme diskutieren kann. Die Jahresgebühr beträgt 150 DM, die einmalige Aufnahmegebühr 50 DM.

Certified Dies ist die interessanteste Klasse: Man bekommt von der meisten Systemsoftware die aktuellen Beta-Versionen (z. B. Kickstart und Workbench) sowie die dazugehörigen Includes und AutoDocs. Certifieds bekommen aber nicht jede Beta und in der Regel auch keine Hardware-Beta. Man bezahlt 400 DM pro Jahr dafür und eine einmalige Aufnahmegebühr von 100 DM.

Commercial

Kommerzielle Entwickler bekommen im wesentlichen dieselben Informationen wie die "Zertifizierten", allerdings kompletter, sprich auch Beta-Hardware, und etwas früher. Dies hat auch seinen Preis: 700DM pro Jahr plus einmalige Aufnahmegebühr von 100DM.

Die Preise und die angebotenen Möglichkeiten können von Land zu Land verschieden sein, auch gibt es meines Wissens nicht in allen Ländern den "Registered". Ein guter Tip ist es, eine Gruppe zu bilden und damit die Kosten zu reduzieren.

Alle Entwickler müssen Non-disclosure agreements (NDA) unterzeichnen. Diese besagen, daß sie über die erhaltenen Informationen außerhalb von speziell dazu freigegebenen Orten oder Kanälen nicht einmal zu anderen Entwicklern sprechen dürfen, so lange sie nicht die explizite Erlaubnis dazu von Commodore haben.

3.5 Was für Compiler (und Assembler) gibt es?

Es gibt viele Programmiersprachen auf dem Amiga, kommerzielle Compiler ebenso wie frei kopierbare. Ich möchte nur diejenigen aufzählen, die mir bekannt sind oder die ich aus einem anderem Grund für erwähnenswert halte.

Assembler Alle C-Compiler beinhalten einen Assembler. Frei kopierbar und zuverlässig sind A68K und PhxAss. (Aminet, Directory '`dev/asm`' oder Fish-Disks 521 bzw. 906)

C

C++ Frei kopierbare C-Compiler sind der `gcc` (der sein eigenes Directory '`dev/gcc`' auf dem Aminet hat) und die Probeversion (mit der man aber schon eine ganze Menge anfangen kann) von `Dice` (per FTP von `ftp.uni-paderborn.de`, Directory '`/news/comp.binaries.amiga/volume91/languages`' oder auf Fish disk 491). Der große Vorteil von gcc ist, daß man ihn auf der ganzen Welt und auf nahezu jedem Computer findet. Ein weiterer Vorteil ist, daß er sogar einen C++-Compiler enthält! Aber er ist langsam und benötigt 4MB RAM oder mehr. Siehe Abschnitt 3.13 [Der GNU C Compiler], Seite 19. Siehe Abschnitt 8.9 [Mail-Listen], Seite 38.

Kommerzielle C-Compiler sind `Aztec-C`, `Dice` und `SAS-C`. `Aztec-C` wird jedoch leider nicht mehr weiterentwickelt. Was die kommerziellen Compiler auszeichnet, sind ihre hervorragenden Source-Level-Debugger, die den anderen fehlen.

SAS hat leider angekündigt, den Amiga-Compiler nicht weiter zu unterstützen. Verkauft wird er aber noch, und da er gegenwärtig noch voll aktuell ist und sogar einen Crosscompiler von C++ in C enthält (der vom Debugger unterstützt wird), ist er

meines Erachtens derzeit das beste Angebot, insbesondere zu dem äußerst günstigen Preis von 184.-DM für Studenten und Besitzer anderer Compiler. In Deutschland erhält man SAS/C bei

```
SAS Institute GmbH
Postfach 10 53 40
69043 Heidelberg
Deutschland

Telefon: 06221/4160
EMail: eurdoc2@vm.sas.com
```

Dice bietet wie SAS einen Sonderpreis für Schüler und Studenten von ca. 130.-DM. Der Compiler ist stabil und vor allem sehr schnell. Der größte Nachteil von Dice ist (verglichen mit den anderen kommerziellen Compilern) der Debugger, ein sogenannter Source-Line-Debugger: Dies bedeutet, daß man den Quelltext sieht und das Programm Schritt für Schritt abarbeiten kann, aber leider nur Speicher und nicht etwa bestimmte Variablen anzeigen kann. Informationen über Dice: info@oic.COM.

Comeau C++ ist ebenfalls ein Crosscompiler, was an und für sich kein Problem wäre. Aber Comeau C++ hat keinen integrierten C-Compiler, man braucht also zusätzlich SAS-C, Aztec-C oder Dice. Dafür ist er kompatibel zu AT&T cfront 3.0, unterstützt Exceptions und läuft wie `gcc` auf vielen verschiedenen Systemen. In Deutschland wird auch Maxxon C++ angeboten, über das ich nichts sagen kann. Beide Compiler sind kommerziell. Comeau's Adresse ist:

```
Comeau computing
91-34, 120th Street
Richmond Hill, NY, 11418-3214
USA

EMail: Greg Comeau, comeau@bix.com
```

Forth JForth soll eine exzellente Forth-Version sein. Unter anderem enthält es objektorientierte Erweiterungen, ein volles Amiga-Interface und einen Anwendungsgenerator. Es ist erhältlich von:

```
Delta Research
P.O. Box 151051
San Rafael, CA   94915-1051

Phone: (415) 453-4320
EMail: Phil Burk, phil@ntg.com
       Mike Haas, haas@starnine.com
```

Fortran (Seufz! Es gibt immer noch Leute, die es brauchen :-<) Frei kopierbar sind BCF (Fish disk 470) und f2c, der Fortran in C-Quelltext umwandelt. (Aminet, Directory '`/dev/misc`'). Ein kommerzieller Compiler ist von ABSoft erhältlich. Allerdings sind dies alles nur Fortran-77-Compiler, es gibt keine Fortran-90-Compiler auf dem Amiga.

Lisp Frei kopierbare Lisp-Interpreter sind XLisp (Fish-Disk 181) und OakLisp (Fish-Disks 519 und 520) und CLISP ('`/pub/lisp/clisp/binaries/amiga`' at '`ma2s2.mathematik.uni-karlsruhe.de`'). Auch Compiler gibt es: Gambit (Fish-Disks 764 und 765) sowie Scheme-to-C (Fish-Disks 556-558). Von Interesse ist

vielleicht eine Mail-Liste: Senden Sie dazu eine Mail mit dem Wort '`Subscribe`' an `amigalisp@contessa.phone.net`.

Prolog '`/dev/lang/UNSWProlog.lha`' und '`dev/lang/sbp3_1e`' auf dem Aminet sowie '`SBProlog`' auf der Fish-Disk 141 und '`SBProlog`' auf der Fish-Disk 145 sind frei kopierbare Prolog-Interpreter.

Modula-2 M2Amiga wird in Europa und Benchmark Modula-2 in den USA angeboten. Beide sollen sehr gut sein und sowohl über gute Source-Level-Debugger als auch eine umfangreiche Bibliothek verfügen. Besonders M2Amiga wird sehr gut unterstützt durch eine deutsche Benutzergruppe (AMOK), die z.B. eine eigene PD-Serie anbieten. Siehe Abschnitt 8.9 [Mail-Listen], Seite 38.

M2Amiga bekommt man bei

```
A+L AG
Daderiz 61
2540 Grenchen
Schweiz

Tel.: +41/65/52 03-11
Fax:              -79
```

und Benchmark Modula-2 ist erhältlich von:

```
Armadillo Computing
5225 Marymount Drive
Austin, Texas 78723
USA

Phone/FAX: 512/926-0360.
EMail: Jim Olinger, jolinger@bix.com
```

Oberon

Oberon-2 Es gibt zwei Oberon-2-Compiler für den Amiga: AmigaOberon ist wie M2Amiga von A+L und kommerziell. Der Compiler kommt mit einer integrierten Entwicklungsumgebung (incl. freikonfigurierbarem Editor) und umfangreicher Modulbibliothek. Library Linker zum einfachen Erzeugen von AmigaOS Shared Libraries sowie Run-time Source-Level-Debugger sind ebenfalls erhältlich.

Oberon-A ist ein Freeware-Compiler, allerdings erst in einer Beta-Version, insbesondere sind die Modulbibliotheken unvollständig. (Quelle: Aminet, Directory '`dev/obero`'). Siehe Abschnitt 8.9 [Mail-Listen], Seite 38.

Für beide Compiler gibt es Unmengen von Modulen und Bsp.-Sourcen auf AMOK-Disks.

Pascal Es gibt einen PD-Compiler namens PCQ (Aminet, Directory '`dev/lang`' oder Fish-Disk 511), der allerdings kein voller Pascal-Compiler ist und dem sehr wesentliche Dinge fehlen. P2C konvertiert Pascal in C und ist auf der Fish-Disk 341 zu finden. (Aminet: '`/dev/misc/p2c120.lha`') Ferner gibt es zwei kommerzielle Compiler namens HiSoft-Pascal (von der gleichnamigen Firma) und KickPascal von Maxxon. HiSoft und P2c behaupten, kompatibel zu Turbo-Pascal 5.0 zu sein. HiSoft hat außerdem einen guten Source-Level-Debugger.

3.6 Warum funktioniert keine Esc-Sequenz?

Viele Drucker kommen mit einem Handbuch, das erklärt, welche ESC-Sequenzen welche Funktion auslösen. Aber wenn man diese Sequenzen dann einmal ausprobiert, passieren oft merkwürdige Sachen, nämlich entweder gar nichts oder etwas ganz anderes, als geplant war. Und das hat auch einen Grund, nämlich die Amiga-Druckertreiber. Diese Treiber sind so gebaut, daß sie nur einen bestimmten Satz ANSI-Esc-Sequenzen verstehen, nicht die (verschiedenen) druckereigenen, von den verschiedenen Druckerherstellern definierten. Der Sinn dabei ist, daß jede Anwendung auf dem Amiga nur diesen einen Standardsatz an Sequenzen verwendet und so nicht zu wissen braucht, welcher Drucker tatsächlich angeschloßen ist. Der Druckertreiber übersetzt dann diese Standardsequenzen in die druckereigenen. Eine Liste der verfügbaren ANSI-Esc-Sequenzen findet sich im aktuellen Workbench-Handbuch (oder in älteren AmigaDOS-Handbüchern). Wenn Du nun eine Steuersequenz an den Drucker schicken willst, die es nicht als ANSI-Sequenz gibt, so hast Du zwei Möglichkeiten, dies doch zu erreichen:

1. Umgeh den Druckertreiber (der erfolglos versuchen würde, die Sequenz zu interpretieren oder zu übersetzen) und sende die Ausgabe **nur** für die Länge dieser Sequenz an 'PAR:' (bzw. 'SER:'). Dabei muß man die Druckerausgabekanäle umständlich oft umschalten, und man muß wissen, wo der Drucker angeschlossen ist ('PAR:' oder 'SER:').
2. Benutze eine spezielle ANSI-Sequenz, genau für diesen Fall gedacht:

   'Esc[<n>"<x>'

   wobei '<n>' die dezimal geschriebene Anzahl an Bytes in der Sequenz '<x>' ist, die gerade die spezielle Drucker-Sequenz enthält. Diese ANSI-Sequenz sagt dem Druckertreiber, daß er die nächsten '<n>' Bytes nicht interpretieren oder übersetzen soll.

Aber beide Methoden haben einen grossen Nachteil, wenn sie in einem Anwendungsprogramm verwendet werden: Man verliert die Druckerunabhängigkeit! Solange man sich an die ANSI-Sequenzen hält, kann man jeden Drucker der Welt ansteuern, solange es einen Amiga-Druckertreiber für ihn gibt. Wenn man anfängt, druckereigene Sequenzen zu verwenden, ist das Programm an diesen einen Druckertyp gebunden und mit keinem anderen benutzbar (oder man müßte einige dutzend neue Druckertreiber für dieses Programm erstellen).

Dr. Peter Kittel, peterk@cbmger.de.so.commodore.com

3.7 Kann ich AmigaBasic auf dem A1200 verwenden?

Letztens ging eine Kontroverse über AmigaBasic durch die Netze: Ich sagte, auf dem A1200 läuft es einigermaßen problemlos, jemand anders berichtete, daß es bei jedem kleinsten Fehler sofort komplett abstürzt, was ich nicht nachvollziehen konnte.

Jetzt kann ich es: Es liegt am Sound-Prefs-Editor. Wenn man in ihm die Sound-Ausgabe ganz abschaltet, kann man mit AmigaBasic arbeiten. Wenn hier ein Sound (z. B. Piepsen) angewählt ist, kollidiert das mit dem Sound, den AmigaBasic offensichtlich zu Fuß und nicht ganz korrekt selber erzeugen will, bumm.

Abhilfe also:

1. Im Sound-Editor den Ton abstellen.
2. Auf einem A4000 (oder einem A1200 mit Fast-RAM-Erweiterung[3]) muß man zusätzlich NoFastMem aktivieren.
3. Möglichst SUBs vermeiden und stattdessen herkömmliche GOSUBs benutzen, dann ist die Kompatibilität zu neueren Prozessoren höher.

Dr. Peter Kittel, peterk@cbmger.de.so.commodore.co

3.8 Wie lokalisiere ich mein Programm?

Nehmen wir an, wir wollen ein `HelloLocalWorld.c` schreiben. Das letztendliche Programm sieht dann ungefähr so aus:

```
#include "HelloLocalWorld_Cat.h"
#include <clib/exec_protos.h>

struct Library *LocaleBase;

void main(int argc, char *argv[])

{
    /* Öffne die locale.library. (Kein Abbruch, wenn sie nicht
       da ist, weil dann einfach die eingebauten Strings verwendet
       werden. Aus diesem Grund auch keine Verwendung des
       AutoOpening, auch wenn es der Compiler beherrscht.)
    */
    LocaleBase = OpenLibrary("locale.library", 38);
    OpenHelloLocalWorldCatalogs(NULL, NULL);

    printf(GetString(MSG_Hello));

    CloseHelloLocalWorldCatalog();
    if (LocaleBase) CloseLibrary(LocaleBase);
}
```

Die Funktion GetString prüft, ob die gewünschten Kataloge vorhanden sind und liefert einen Zeiger auf einen String, entweder den eingebauten oder den Katalogstring. (In unserem Fall den deutschen String.)

Der Hauptunterschied zum gewohnten HelloWorld.c ist also (abgesehen von der minimalen Initialisierung und dem Gegenstück am Programmende), Strings durch einen Funktionsaufruf zu ersetzen. Man braucht also eine Datei '`HelloLocalWorld_Cat.c`', die die Funktionen OpenHelloLocalWorld, GetString, CloseHelloLocalWorld_Cat.h und die eingebauten Strings enthält (dies könnte ein Array sein, das unter anderem

```
array[MSG_Hello] = "Hello, local world.\n";
```

[3] Nur bei einem zusätzlichen Prozessor

enthält) und ein Includefile '`HelloLocalWorld_Cat.h`', das die ID's wie MSG_Hello definiert. Es ist nicht nötig zu wissen, wie diese Dateien intern arbeiten, insbesondere benötigt man auch keine Kenntnis der `locale.library`!

Dazu gibt es verschiedene Kataloggeneratoren (im Folgenden KG), nämlich '`CatComp`' (nur für Developer), `KitCat` (nur deutsche Dokumentation, was hierzulande kein Problem ist), '`MakeCat`' (das ich nicht kenne) und `FlexCat` (das ich empfehle, einerseits, weil es sehr flexibel im erzeugten Source ist und z.B. Lokalisierung unter 2.0 sowie beliebige Programmiersprachen unterstützt, selbst Amiga-E, Cluster, Pascal, ... und andererseits, weil es von mir ist ;-)), die diese Dateien sowie die Kataloge erzeugen. (Der obige Quelltext könnte je nach KG leicht unterschiedlich aussehen.) Siehe Aminet, directory '`dev/misc`'.

Wie funktionieren diese KGs? Zunächst erzeugt man eine sogenannte `Katalogbeschreibung` (`Catalog description`), die so aussehen könnte:

```
; Mit einem Semikolon beginnende Zeilen sind Kommentare
# language english
; die Sprache der eingebauten Strings
# version 0
; die Katalogversion (0 = beliebig)
MSG_Hello (1/15/30)
Hello, local world
```

Jeder String wird durch zwei Zeilen wie die letzten beiden definiert: MSG_Hello ist die String-ID, (1/15/30) gibt den Wert der ID sowie die minimale und maximale Länge an. (Diese Argumente können auch weggelassen werden, in welchem Fall einfach die nächste freie ID verwendet wird.)

Nun schreiben wir das Programm. Sobald es fertig ist, wird mit dem KG eine sogenannte Katalogübersetzung (eine für jede andere Sprache als die eingebaute) erzeugt, die so aussehen könnte:

```
; Mit einem Semikolon beginnende Zeilen sind Kommentare
## language deutsch
; the catalog language
## version $VER: Deutsch.catalog 1.0 (22.12.93)
; the catalog files version string
MSG_Hello

; Hello, local world
```

Beachten Sie die leere Zeile nach der String-ID! (Die Argumente von ## language und ## version wären vielleicht leer.) Hier müssten jetzt die deutschen Strings eingesetzt werden. Mit dem KG wird daraus dann der eigentliche `Katalog` erzeugt. (Beachten Sie auch, daß hier die Angaben über String-ID und Stringlänge fehlen: Sie werden aus der Katalogbeschreibung übernommen.

Wenn das Programm verändert wird (neue Strings, andere Längen) und die Katalogbeschreibung sich damit ebenfalls ändert, dann kann der KG analog benutzt werden, um auch die Katalogübersetzung und damit den Katalog auf den neuesten Stand zu bringen.

3.9 Wie erhält man einen Zeiger auf das Fenster einer Konsole?

Die folgende Funktion liefert den Window-Zeiger eines CON-Fensters. Sie kann unter allen Versionen des Amiga-OS ausgeführt werden.

```
struct Window *getConWindowPtr(BPTR fh)
{
  struct Window *w;
  struct FileHandle *cfh;
  struct StandardPacket *sp;
  struct InfoData *id;
  struct MsgPort *mp;

  w = NULL;

  if ((cfh = BADDR(fh))->fh_Type != NULL) {
    if (sp = AllocMem(sizeof (struct StandardPacket),
                      MEMF_PUBLIC | MEMF_CLEAR)) {
      if (id = AllocMem(sizeof (struct InfoData),
                        MEMF_PUBLIC | MEMF_CLEAR)) {
        if (mp = CreatePort(NULL, 0)) {
          sp->sp_Msg.mn_Node.ln_Name = (char *) &sp->sp_Pkt;
          sp->sp_Pkt.dp_Link         = &sp->sp_Msg;
          sp->sp_Pkt.dp_Port         = mp;
          sp->sp_Pkt.dp_Type         = ACTION_DISK_INFO;
          sp->sp_Pkt.dp_Arg1         = MKBADDR(id);

          PutMsg(cfh->fh_Type, &sp->sp_Msg);
          (void) WaitPort(mp);
          (void) GetMsg(mp);

          if (sp->sp_Pkt.dp_Res1)
            w = (struct Window *) id->id_VolumeNode;

          DeletePort(mp);
        }
        FreeMem(id, sizeof (struct InfoData));
      }
      FreeMem(sp, sizeof (struct StandardPacket));
    }
  }

  return w;
}
```

Anmerkungen:

Auf ein CON-Fenster direkt zuzugreifen kann Konflikte mit Aktionen des CON-Handlers hervorrufen. Seien Sie vorsichtig!

Um den Window-Zeiger einer CLI-Konsole zu erhalten, übergibt man ein durch Open("*", MODE_OLDFILE) gewonnenes FileHandle an obige Funktion.

Das Ergebnis der obigen Funktion kann sehr wohl NULL sein, etwa im Falle eines AUX-Handlers oder falls ein AUTO-CON-Handler sein Fenster nicht öffnen kann.

Schickt man ein ACTION_DISK_INFO-Paket an einen AUTO-CON-Handler (2.0+), so verliert dessen Fenster seine speziellen AUTO-Eigenschaften (es kann also nicht mehr durch das Betätigen des Close-Gadgets geschlossen werden), weil der in id_VolumeNode gelieferte Window-Zeiger gültig bleiben muß.

Alles in allem: Verwenden Sie diese Funktion nicht. :-)

Weitere Informationen finden Sie auf den Seiten 273, 276, 435, 463, 485 und 629 im "Amiga Guru Book" (siehe Abschnitt 3.1 [Dokumentation], Seite 6).

Ralph Babel, rbabel@babylon.pfm-mainz.de

3.10 Was sind Pragmas?

Pragmas sind spezielle Anweisungen an den C-Compiler. Zwei Probleme entstehen bei der Verwendung von Pragmas:

1. Pragmas sind hochgradig compilerspezifisch. Nicht einmal die Amiga-Compiler haben dieselben Pragmas, selbst wenn damit das gleiche bewirkt wird.
2. Man kann sich nicht darauf verlassen, daß ein Compiler Pragmas ignoriert, die er nicht versteht. Dies gilt selbst dann, wenn man einen Ausdruck wie den folgenden verwendet:

```
#ifndef MY_COMPILER
#pragma DoAnything
#endif
```

Das letztere Problem läßt sich umgehen, indem man Pragmas wie folgt in eigene Include-Files setzt. (Das gleiche gilt übrigens auch für Präprozessor-Kommandos wie #asm (Aztec-C) oder #extern (C++).)

```
#ifndef MY_COMPUTER
#include <mypragmas.h>
#endif
```

Aber was machen Pragmas auf dem Amiga? Meistens werden sie verwendet, um dem Compiler mitzuteilen, wie gewisse Library-Funktionen aufgerufen werden. (Tatsächlich wird fast immer diese Verwendung gemeint, wenn Amiga-Besitzer über Pragmas sprechen.) Gewöhnliche C-Funktionen erwarten ihre Argumente auf dem Stack, Library-Funktionen dagegen in bestimmten Registern. Ferner erwarten sie den `Library-Base-Pointer` in Register a6. Betrachten wir eine Pragma-Anweisung von Aztec-C:

```
#pragma amicall(SysBase,0xd2,FreeMem(a1,d0))
```

Dies weist den Compiler an, das erste Argument in Register a1 und das zweite in d0 zu laden. Ferner wird der Inhalt der Variablen SysBase in Register a6 geladen. Maxon-Pragmas sehen genauso aus, Dice- und SAS-Pragmas sind allerdings etwas komplizierter:

```
#pragma libcall SysBase FreeMem d2 0902
```

Hier ist d2 (wie 0xd2 oben) der `Library-Vektor-Offset` (siehe nächstes Beispiel). Die letzte Ziffer ist die Zahl der Argumente, die davorstehende 0 ein Code für das Register mit dem

Ergebnis und die davor stehenden Ziffern sind Codes für die Register mit den Argumenten in verkehrter Reihenfolge. (Die Codes bedeuten 0=d0, 1=d1, .., 8=a0, 9=a1, a=a2, ..)

Ein Kommando wie '`FreeMem(fib,sizeof(*fib);`' würde ein Compiler nun in folgenden Code übersetzen:

```
move.l  _fib,a1
move.l  260,d1     ; sizeof(struct FileInfoBlock)
move.l  _SysBase,a6
jsr     -0xd2(a6)       ; 0xd2 = _LVOFreeMem
```

FreeMem in dieser Art aufzurufen ist kürzer und schneller als zunächst die Argumente auf den Stack zu legen und dann eine Funktion _FreeMem aufzurufen, die letzten Endes doch nur dasselbe tun und die Argumente vom Stack in dieselben Register laden würde.

Das Portierungsproblem der Pragmas umgeht man, indem man sie folgendermaßen in den eigenen Quelltext einbindet:

```
/* Lade den Funktionsprototyp. Dieser ist nicht vom */
/* verwendeten Compiler abhängig. */
#include <clib/exec_protos.h>

/* Pragmas sind vom Compiler abhängig, aber wenigstens */
/* die Namen der Dateien mit Pragmas sind relativ */
/* einheitlich. */
#ifdef AZTEC_C
#include <pragmas/exec_lib.h>
#endif
#if defined(__SASC) || defined(_DCC) || defined(__MAXON__)
#include <pragmas/exec_pragmas.h>
#endif
#ifdef __GNUC__
#include <inline/exec_lib.h>
#endif
```

Das obige Beispiel kann problemlos mit allen angegebenen Compilern verwendet werden und produziert optimalen Code.

Eine abschließende Frage bleibt allerdings: Wie bekommt man die Pragmas? Die meisten Compiler haben bereits fertige Pragmas im Lieferumfang. Allerdings hilft das nicht, wenn man z.B. eine neue Library benutzen möchte oder nur die Pragmas einer veralteten Version hat. In diesem Fall kann man die Pragmas selbst aus den sogenannten `FD`-Files erzeugen. Dazu haben die meisten Compiler ein Utility namens `fd2pragma`. (Das NDU hat z.B. ein Directory namens FD, in dem die FD-Files aller Libraries und Devices des OS enthalten sind. siehe Abschnitt 3.3 [Include-Dateien], Seite 8) Es gibt auch ein frei kopierbares fd2pragma, das Pragmas für Aztec, Dice, SAS und Maxon sowie LVO-Files für den Aztec-Assembler und eventuelle Tag-Versionen produziert. (Aminet, '`dev/misc/fd2pragma2_0.lha`' oder auf den Fish-CDs)

Für Pragmas unter dem gcc siehe Abschnitt 3.13.5 [Inline-Dateien], Seite 21.

3.11 Mein Compiler/Linker vermißt Symbole.

Zunächst sollte man sich versichern, daß die Funktion tatsächlich fehlt: Z.B. Floating-Point-Funktionen befinden sich in einer speziellen Link-Library, die erst mit der Option '`-lm`' eingebunden wird. Ferner kann es eine fehlende Variable sein: Wenn man z.B. ohne es zu bemerken eine Intuition-Funktion benutzt, dann wird der Linker über das Fehlen eines Symbols `IntuitionBase` klagen. In diesem Fall muß man also lediglich das Folgende irgendwo im globalen Teil seines Quelltextes einbauen:

```
struct Library *IntuitionBase;
```

(Und vergessen Sie nicht, die Library mit OpenLibrary() zu eröffnen und mit CloseLibrary() zu schließen!) :-)

Allerdings könnte die Funktion natürlich tatsächlich fehlen. Wenn man zum Beispiel nur die Version 2.0 der amiga.lib hat, dann fehlen etwa die Locale-Funktionen oder die Memory-Pool-Funktionen, obwohl sie prinzipiell verwendbar sind.[4] Die einfachste (und beste) Lösung ist, das sogenannte NDU zu kaufen. Siehe Abschnitt 3.3 [Include-Dateien], Seite 8. Wer nicht solange warten möchte, für den ist die Frage, welche Art von Funktion in seiner Link-Library fehlt:

Einfache Amiga-Library-Funktionen (z.B. '`exec/AllocPooled`' oder '`locale/OpenCatalogA`') kann man mit Pragmas aufrufen und braucht dann lediglich Informationen über die Aufrufkonventionen (Library-Base, Library-Vektor-Offset und Argumentregister)

Tag-Funktionen sind meistens einfach Zwischenfunktionen, die ihre Argumente auf dem Stack erwarten und dann die eigentliche Library-Funktion aufrufen. Wenn man z.B. die Funktion '`dos/AllocDosObject`', die eine Konstante und einen Zeiger auf ein Array als Argumente erwartet, nach der obigen Methode konstruiert hat, dann hat man damit auch ihre Stack-Variante! Dazu erzeugt man einfach die folgende Funktion:

```
#include <clib/dos_protos.h>
#include <pragmas/dos_pragmas.h> /*  Evtl. anderer Name */

void *AllocDosObjectTags(ULONG objtype, Tag tag1, ...)
{ return(AllocDosObject(objtype, (struct TagItem *) &tag1);
}
```

Mit fd2pragma kann das auch automatisch geschehen. Siehe Abschnitt 3.10 [Pragmas], Seite 16.

Einige Funktionen bleiben aber noch übrig: Amiga.lib enthält nämlich auch Funktionen, die für sich selbst interessant sind, z.B. die BOOPSI-Funktionen ('`DoMethod`', '`DoSuperMethod`'), die Memory-Pool-Funktionen ('`LibAllocPooled`', '`LibCreatePool`', die Ersatz für entsprechende 3.0-Funktionen sind) oder '`HookEntry`', das sehr hilfreich bei der Programmierung von Hooks ist. Diese kann man nur durch entsprechende, disassemblierte und neu assemblierte oder compilierter Versionen ersetzen. Im AmigaFAQ-Archiv sind einige dieser Funktionen bereits enthalten.

[4] Dieses Problem betrifft vor allem Besitzer von Aztec-C, das seither nicht weiter unterstützt wird und von Dice, der manchmal etwas unvollständig ist. Ich besitze beide ..

3.12 Wie erfahre ich, was für Funktionen es gibt?

Wenn Sie sich über den Namen einer für einen bestimmten Zweck geeigneten Funktion im Unklaren sind, dann gibt es folgende Möglichkeiten:

In den Autodocs der verschiedenen Libraries findet man am Anfang eine alphabetisch sortierte Tabelle aller Funktionen, die die betreffende Library bietet. Im Hauptteil findet man dann eine detaillierte Spezifikation aller Funktionen. Siehe Abschnitt 3.3 [Include-Dateien], Seite 8.

Die '`.FD`'-Dateien bieten eine sehr kompakte Übersicht über die Funktionen der diversen Libraries, sogar mit einer kurzen Angabe der Funktionsargumente. Wenn man schon grob weiß, wonach man sucht (z.B. nur die grobe Angabe der Argumente benötigt), dann findet man hier alle gewünschten Informationen. Siehe Abschnitt 3.10 [Pragmas], Seite 16.

Dr. Peter Kittel, peterk@cbmger.de.so.commodore.com

3.13 Der GNU C Compiler: Allgemeine Informationen und Installation

Dieser Abschnitt enthält Informationen über den Amiga-Port des GNU C Compilers generell und die Installation insbesondere.

3.13.1 Aktuelle Version

Die aktuelle gcc-Version (im Folgenden als <gcc-aktuell> bezeichnet) ist 2.5.8. Die Version 2.6 ist in Arbeit und wird unter anderem eine neue Version der beliebten ixemul.library (die Unix-Systemaufrufe auf dem Amiga emuliert) sowie eine absolut neue C-Library enthalten, die die ixemul.library nicht mehr benötigt.

3.13.2 Hardwareanforderungen

Jeder Amiga (vom A1000 bis zum A4000/40) ist prinzipiell in der Lage, die GNU-Utilities für AmigaDOS zu benutzen. Allerdings braucht man wenigstens 4MB RAM, um kleinere oder mittlere Projekte zu übersetzen, für größere (z.B. für gcc selbst) entsprechend mehr. Gigamem arbeitet mit gcc zusammen, es *könnte* also evtl. auch mit weniger gehen. Dazu braucht man aber eine MMU. Siehe Abschnitt 1.1 [68EC0xx], Seite 1.

Eine volle Installation mitsamt C++ und Objective-C, inline-Dateien und Commodore Includes (siehe Abschnitt 3.3 [Include-Dateien], Seite 8) benötigt ca. 20 MB Platz auf der Festplatte.

Gcc arbeitet auch unter Kickstart 1.2/1.3, die volle Funktionalität ist aber erst ab Kickstart 2.x+ vorhanden. Eine schnelle CPU (z.B. 68030@25MHz oder besser) ist ebenfalls sinnvoll.

3.13.3 Wer hat es gemacht?

Gcc und damit zusammenhängende Software wurde von den folgenden Personen auf den Amiga portiert (siehe Abschnitt 3.13.8 [Hilfe], Seite 23):

```
Gcc v2.2.2:   Markus Wild
Gcc v2.3.3:   Markus Wild
Gcc v2.4.5:   Philippe Brand, Lars Hecking, Fred Fish
Gcc v2.5.0 und später: Philippe Brand, Fred Fish, Leonard Norrgard

Ixemul.library:    Markus Wild, Leonard Norrgard, R. Luebbert
Libnix:    Matthias Fleischer, Gunther Nikl
        Gerlib:            Gerhard Müller
```

3.13.4 Wo finde ich die gcc-Quelltexte?

Alle gcc-Quelltexte und alle Binärdateien findet man auf:

1. Aminet (wuarchive.wustl.edu und Mirror wie ftp.luth.se) in /pub/aminet/dev/gcc (siehe Abschnitt 8.3 [FTP], Seite 33)
2. Ramses The Amiga Flying BBS:

```
+33-1-60037015  HST Dual v32 terbo 4800-21600
        +33-1-60037713  SupraFax v32bis    4800-14400
        +33-1-60037716  Tornado v22bis     1200-2400
```

   in Topic '`Development`', Area '`Gcc`' (are 156).

Den originalen GNU-Quelltext bekommt man

1. von denselben FTP-Servern wie die Binaries
2. gnu.prep.ai.mit.edu (18.71.0.38) in '`/pub/gnu`'
3. Ramses The Amiga Flying BBS in Topic '`AmigaUnix/Unix/Linux/NetBSD`', Area '`Gnu Source Code`'

Diese Archive sollten alles Nötige enthalten, mit Ausnahme der Quelltexte der ixemul.library. Diese erhält man ebenfalls auf dem Aminet, Directory '`dev/gcc`'. (Zur Zeit ist die Version 40 der ixemul.library in Arbeit. Die Quelltexte werden am gleichen Ort sein.)

Durch Richard Stallman, Free Software Foundation, ist festgelegt:

"The GPL says that any distribution of binaries must contain either the source code or a written offer to supply source code (see the GPL for details of what is required)."

Übersetzt: Die GPL (GNU General Public License) legt fest, daß jede Distribution entweder den Quelltext oder das schriftliche (oder geschriebene?) Angebot, diese zu liefern, enthalten muß. (Näheres siehe in der GPL.)

3.13.5 Inline-Dateien

Die Inline-Dateien für den gcc kann man aus den originalen FD-Dateien von Commodore wie folgt erzeugen (Siehe Abschnitt 3.3 [Include-Dateien], Seite 8, siehe Abschnitt 3.10 [Pragmas], Seite 16):

```
CLI> Assign INCLUDE: GCC:os-include
CLI> Assign FD: INCLUDE:fd
CLI> Makedir INCLUDE:inline
CLI> cd USR:bin/geninline
CLI> gen31
```

Dies sollte alle Inline-Dateien in '`GCC:os-include/inline`' erzeugen. Falls man die 2.0- bzw. 3.0-Includes besitzt, sollte man gen20 bzw. gen30 benutzen. Inline-Dateien für OS3.1 (rev 40.13) sind in <gcc-aktuell> enthalten. Siehe Abschnitt 3.13.1 [Aktuelle Version], Seite 19.[5]

Auch mit dem Programm fd2inline kann man Inline-Dateien erzeugen:

```
CLI> fd2inline <fd_file> <proto_file>
```

3.13.6 Wie konvertiere ich die Amiga-Libraries für den gcc?

Linklibraries im Amigastil wie amiga.lib können von gcc nicht ohne weiteres verwendet werden. Man muß sie zunächst mit hunk2gcc, einem Programm von Markus Wild, übersetzen.

Dazu braucht man zunächst eine aktuelle Kopie der amiga.lib (aus dem NDU, siehe Abschnitt 3.3 [Include-Dateien], Seite 8). Nachdem man sich ein Directory für die konvertierten Dateien erstellt hat, wechselt man mit cd in dieses Directory und gibt das Folgende ein:

```
hunk2gcc amiga.lib [..further libs if you like..]
```

Dies erzeugt ein Objektfile wie a.out für jedes einzelne Modul, das die Library enthält. Diese Module muß man anschließend in einer gcc-Library zusammenfassen:

```
ar qc libamiga.a obj.*
ranlib libamiga.a
```

Das Programm ranlib fügt eine Symboltabelle in die Library ein. (Dadurch wird der Zugriff auf die Library sehr viel schneller.)

Solange man keine reinen AmigaDOS-Funktionen verwendet, kann man auch eine Pseudo-Library erzeugen:

```
cat "int dummy;" >dummy.c
gcc -c dummy.c
ar crv libamiga.a dummy.o
mv libamiga.a gcc:lib
```

[5] Perl Skripte haben Schwierigkeiten, die Include-Dateien von AmigaDOS korrekt zu behandeln. Hier wäre etwas freiwillige Arbeit nötig ...

3.13.7 Wie installiere ich den gcc?

Zunächst eine wichtige Bemerkung von Philippe Brand: Die meist gestellte Frage im Zusammenhang mit gcc-2.5.8 ist

"Warum kann ich keine C++-Programme übersetzen?"

Das liegt daran, daß ich einen Fehler bei der gcc-Distribution gemacht habe. Anstelle der 2.5.8-Versionen der C++- und Objective-C-Compiler habe ich die Version 2.5.7 verwendet. Um dies zu beheben, gibt es zwei Möglichkeiten:

'`lib/gcc-lib/amigados/2.5.7`' umbenennen in '`lib/gcc-lib/amigados/2.5.8`'

Die Datei '`dev/gcc/gcc258ud.lha`' von Aminet laden. Diese enthält die korrekten Versionen.

Auf dem Aminet befindet sich auch die Datei '`dev/gcc/gcc258ud1.lha`', die einige korrigierte Libraries, Includes usw. enthält ...

Und vergessen Sie nicht, die Datei '`gcc258-readme.lha`' ebenfalls von Aminet zu kopieren.

1. Falls dies die erste gcc-Installation ist:

   ```
   CLI> cd Disk_mit_viel_freiem_Platz:
   CLI> lha x gcc258-x.lha
   ```

   Dabei steht x für die Zahlen 1 bis 4, evtl. auch 5 bis 6, falls Sie auch die Quelltexte installieren wollen.

   Von der Workbench aus wird nun das Programm GCC-Install gestartet. Dies ist das von Vielen gewünschte Installer-Skript für den gcc. Man muss also nur die Archivdateien entpacken, der Rest (Erzeugen von Umgebungsvariablen, Assigns, Kopieren der Shared-Libraries usw.) wird vom Installationsskript erledigt.

   Bei dieser Gelegenheit:

   Installer and Installer project icon © Copyright 1991-93 Commodore-Amiga, Inc. All Rights Reserved. Reproduced and distributed under license from Commodore.

2. Falls bereits eine ältere gcc-Version vorhanden ist: Löschen Sie einfach das alte gcc-Directory und führen Sie dann eine Neuinstallation durch. Es hat sich zu viel geändert. (Ein Backup vorher ist aber sicher keine schlechte Idee.)

Bitte verwenden Sie **unbedingt** die ixemul.library in der Revision 45 und nicht die Revision 47! Letztere ignoriert '`ENV:`'.

Um gcc zu verwenden, brauchen Sie einen Stack von wenigstens 50000 Bytes. Dies sollte für die meisten Anwendungen ausreichen. Wollen Sie allerdings gcc selbst übersetzen, dann sollten Sie wenigstens 250000 Bytes rechnen, für noch komplexere Dinge entsprechend mehr. Auch ar und ranlib brauchen einen großen Stack, wenigstens 200000 bis 300000 Bytes, abhängig von der Größe der Library.

3.13.8 Wichtige Informationsquellen

Der aktuelle Betreuer des Amiga-Ports von gcc ist:

```
Philippe BRAND
Fidonet: Ramses The Amiga Flying BBS 2:320/104.21
Email:   phb@colombo.telesys-innov.fr (ONLY for personnal email).
Ftp:     colombo.telesys-innov.fr:/pub/amigados-gnu
         or /pub/incoming/uploads for uploads.
```

Es gibt auch eine Mail-Liste in Finnland. Siehe Abschnitt 8.9 [Mail-Listen], Seite 38. Philippe Brand wird im Normalfall Fragen an diese Liste weiterleiten.

4 Anwendungen

Dieses Kapitel enthält Informationen zu verschiedenen Anwendungsbereichen.

4.1 Text-Editoren

Editoren sind Programme, die die Eingabe von unformatiertem Text ermöglichen. Dies ist meist Text, der anschließend durch den Computer verarbeitet wird. Programmierer benutzen etwa Editoren zur Eingabe der Quelltexte. Unter Unix beginnt Textverarbeitung meist mit einem Editor, dessen Ausgabe dann mit einer Textbearbeitungssprache (z.B. Abschnitt 4.4 [LaTeX], Seite 25) in gutaussehende Dokumente umgewandelt wird, da unter Unix Textverarbeitungsprogramme kaum verbreitet sind.

Kommerzielle Produkte
: CygnusEd Professional und TurboText scheinen hier die Nase vorn zu haben. Auf den Fish-Disks gibt es Dutzende von Shareware-Editoren. Eine Demoversion von TurboText ist auf Fish-Disk 445 und eine allerdings sehr alte von CygnusEd auf Fish-Disk 95. Die folgenden Editoren sind frei kopierbar.

Emacs
: Gnu Emacs (Aminet, Directory '`util/gnu`') kommt von Unix und ist möglicherweise der König der Editoren. Er ist riesig (über 1MB), enthält eine unglaubliche Vielfalt von Funktionen (z.B. ein eigenes Fenstersystem und sogar ein Spiel :-)) und beliebig erweiterbar: Emacs enthält einen eigenen Lisp-Interpreter! Andererseits ist er dadurch schwer erlernbar und so beliebig konfigurierbar, daß man mit dem Emacs eines anderen oft nicht mehr umgehen kann...

Vi
: Einfacher, aber weniger konfigurierbar und nicht erweiterbar ist Vi. Er ist vor allem bei Unix-Benutzern beliebt, weil es ihn auf *jeder* Unix-Maschine gibt. Die Wahl zwischen beiden ist eine Sache der persönlichen Vorliebe, kann einen allerdings fürs Leben kennzeichnen. Vim und Stevie (Fish-Disk 591 bzw. 256 und Aminet, Directory '`util/edit`') sind Vi-Versionen für den Amiga.

DME
: Viele Amiga-Programmierer mögen DME. Er ist schnell und voll konfigurierbar: Menüs und Tastenbelegung sind beliebig. Er ist auch leichter erlernbar als Emacs

und Vi. Es gibt drei verschiedene Versionen: AmokEd, DME und XDME. Die Wahl ist Geschmacksache. (Oberon- und Modula-Programmierer bevorzugen häufig den AmokEd, weil der in Oberon geschrieben ist und die Fehlermeldungen von AmigaOberon direkt anzeigen kann, C-Programmierer dagegen stehen öfter auf letzterem.) Quellen: Aminet, Directory '`util/edit`', Fish-Disk 776 (XDME) und 749 (AmokEd), AMOK 90.

4.2 Was für Textverarbeitungsprogramme gibt es?

Textverarbeitungsprogramme benötigt man, um Briefe, Artikel oder ähnliches zu schreiben. Dies ist natürlich eine der wichtigsten Anwendungen eines Computers und eine für Sie geeignete Wahl bestimmt häufig, wie gut Sie mit Ihrem Computer überhaupt zurechtkommen. Textverarbeitungsprogramme besitzen teilweise Fähigkeiten, die sie sogar für DTP (siehe Abschnitt 4.3 [DTP], Seite 24) geeignet machen. Für einfacher strukturierte Texte, etwa Quelltexte benützt man allerdings meist besser einen Editor. Siehe Abschnitt 4.1 [Editoren], Seite 23.

Man unterscheidet Wysiwyg-Programme (What you see is what you get) und Seitenbearbeitungssprachen. Wysiwyg-Programme sollten schnell, komfortabel und einfach erlernbar sein. Die meisten Anwender ziehen sie vor. Die Alternative funktioniert ähnlich wie ein Compiler: Man verarbeitet Text-Dateien mit einem Programm, das das Layout übernimmt und eine weitere Datei erzeugt, die dann auf dem Bildschirm angezeigt oder auf einem Drucker ausgegeben werden kann. LaTeX ist eines dieser Programme. Siehe Abschnitt 4.4 [TeX], Seite 25. Lout ist ein weiteres solches Programm, schneller und einfacher als LaTeX und mit vollständiger Dokumentation versehen, aber natürlich wesentlich weniger verbreitet. Lout erzeugt PostScript-Dateien. (Ob dies ein Vor- oder Nachteil ist, muß jeder selbst beurteilen. ;-)) Siehe Abschnitt 4.5 [PostScript], Seite 26. Beide Programme sind frei kopierbar.

Wysiwyg-Programme gibt es viele, allerdings alle kommerziell: Final Copy II, Wordworth, Word Perfect, AmiWrite, Beckertext II, Maxon Word und viele andere. Zur Wahl kann ich nur eines sagen: Lassen Sie sich viel Zeit bei der Auswahl.

4.3 Desktop Publishing

Die Stärke dieser Programme ist die beliebige Anordnung und Gestaltung von Text- oder Grafikelementen. Bei Textverarbeitungen sind diese Fähigkeiten meist unterentwickelt. Umgekehrt fehlen DTP-Programmen häufig wichtige Funktionen zur Manipulation des Textes selbst. Allerdings kommen die besten Textverarbeitungen sicher nahe an den DTP Bereich (etwa Microsoft Word auf dem Mac oder PC) und umgekehrt ist Framemaker (UNIX, Mac, DOS) ein Beispiel eines DTP-Programms, das die wichtigsten Textverarbeitungsfunktionen enthält. Auf dem Amiga gibt es leider weder in der einen noch in der anderen Richtung hervorragende Beispiele, wenn auch die wichtigsten Textverarbeitungen schon einiges leisten. Wenn man nicht gerade regelmäßig kleinere Zeitungen herausbringt, dann wird in den meisten Fällen ein Textverarbeitungsprogramm genügen. Und wer umgekehrt komplexere Funktionen wie mathemati-

sche Formeln, umfangreiche Indexe und Referenztabellen benötigt, der wird womöglich auf eine Seitenbearbeitungssprache angewiesen sein. Siehe Abschnitt 4.2 [Textverarbeitungen], Seite 24.

Es gibt derzeit nur kommerzielle DTP-Systeme, vor allem ProPage und PageStream, die in den letzten Jahren wohl gegenseitig Hase und Igel miteinander gespielt haben. Derzeit scheint PageStream 3.0 vorne zu liegen. Beide Programme liegen im Preis um 300$, in den USA gibt es wohl auch Studentenermäßigung (um 40%). Eine ausführlichere Beschreibung wäre willkommen.

4.4 Was ist TeX und wo bekomme ich es?

TEX ist eines der mächtigsten Textverarbeitungssysteme, die es gibt. Es kann praktisch beliebige mathematische Formeln ebenso darstellen wie komplexe Tabellen, Funktionen plotten (mit Hilfsprogrammen), Indexe oder Inhaltsverzeichnisse erzeugen und viele andere Dinge. Sein größter Vorteil ist, daß es frei kopierbar ist (TeX selbst, nicht unbedingt die Drucker- und Bildschirmtreiber) und sich dadurch auf der ganzen Welt und auf praktisch jeder Computerfamilie verbreitet hat. Sein größter Nachteil ist, daß es sehr unhandlich und schwer zu erlernen ist (Es arbeitet ähnlich wie ein Compiler) und keinerlei Wysiwyg-Fähigkeiten hat. Aber viele Leute mögen es. (Dieser Text ist übrigens mit TeX erstellt worden. 8-)) Siehe Abschnitt 4.2 [Textverarbeitungen], Seite 24.

Es gibt im wesentlichen zwei Implementationen auf dem Amiga. Amiga-TEX von Thomas Rockicki und Radical Eye Software ist kommerziell und kostet etwa 200-300$. Es soll wirklich exzellent sein und seine Besitzer schwören darauf.

Ich persönlich empfehle PasTEX, eine frei kopierbare Version. Es gibt immer wieder Leute, die über Installationsprobleme klagen (besonders, was das Laden und die automatische Erzeugung von Fonts angeht), aber ich kenne niemanden, der unzufrieden ist, nachdem es erst einmal installiert ist. (Ein Freund mit TEX-Kenntnissen kann enorm hilfreich sein.) Wenn man sich an die Vorgaben des Installationsskripts hält, sollte es eigentlich keine Probleme geben. Wesentlich ist allerdings, daß man alles Nötige installiert und dazu braucht man:

- 5 Disketten mit dem eigentlichen TEX-Compiler
- 2 Disketten mit MetaFont
- Nichts weiter

Viele Leute fragen nach Fonts. Diese sind Teil des MetaFont-Paketes und werden automatisch erzeugt, wenn sie gebraucht werden, vorausgesetzt natürlich, daß TEX so wie in der Dokumentation beschrieben installiert ist. (Das Erzeugen der Fonts kann am Anfang natürlich dauern, bis die wichtigsten erst mal da sind.) Bitte beachten Sie, daß die PasTeX-Disketten mit Zoom gepackt sind. (siehe Abschnitt 7.2 [Endungen], Seite 30) PasTEX bekommt man per FTP z.B. bei `ftp.uni-passau.de` im Directory '`/pub/amiga/tex/PasTeX1.3`' und durch die Fish-CDs.

4.5 Gibt es PostScript-Interpreter auf dem Amiga?

PostScript ist eine Programmiersprache für das Ausdrucken von ganzen Seiten. Apple hat PostScript populär gemacht, indem sie es in ihre Drucker eingebaut haben. Inzwischen gibt es viele Programme, die ihre Ausdrucke als PostScript-Quelltexte machen können. Bis vor kurzem brauchte man für PostScript einen relativ teuren Laserdrucker. Inzwischen gibt es aber auch Programme, die PostScript-Dateien auf anderen Druckern ausgeben können.

Einer der Vorteile von PostScript ist, daß es unabhängig von der gewählten Auflösung ist. Man kann also die Vorteile der möglichen Auflösung auf dem Drucker ausnutzen und trotzdem dieselbe Datei auf dem Bildschirm mit seiner meist schlechteren Auflösung darstellen.

Es gibt zwei frei kopierbare PostScript-Interpreter, Post und GhostScript. Post ist eine shared-Library mit Programmen zur Ausgabe auf Drucker und Bildschirm. Man kann damit beispielsweise in AmigaTeX PostScript-Dateien als Bilder in TEX-Dateien einbauen. Siehe Abschnitt 4.4 [TeX], Seite 25. GhostScript besteht ebenfalls aus zwei Programmen, GhostScript selbst (der eigentliche Interpreter) und GhostView, der Benutzeroberfläche. Quellen: Aminet (Directories '`text/print`' und '`text/dtp`'), Fish Disk 669.

5 Grafik

Grafik ist eine der Stärken des Amiga. Warum hat es hier so wenig Fragen und Antworten?

5.1 Was heißt Chunky- und Planar-Display?

Einfach gesagt stehen die Bezeichnungen `chunky` und `planar` (Kürzel für `bitplanar`) für verschiedene Arten, graphische Daten im RAM des Computers zu speichern. Sie sind einfach zu verstehen, aber vielleicht etwas schwierig zu erklären.

Die Anzeige eines Computers besteht aus einem Netz von Pixeln. Jedes Pixel kann man sich als eine Zahl denken, die für die Farbnummer des Pixels steht. Hier ist zum Beispiel eine ganz einfache Anzeige mit 4 Farben:

```
00302132
```

Der Amiga speichert dies im `bitplanaren` Modus, d.h. es werden verschiedene sogenannte Bitplanes verwendet, in denen zu jedem Pixel genau ein Bit gehört. Für eine Zahl zwischen 0 und 3 brauchen wir 2 Bits, also auch zwei Bitplanes, die dann so aussehen:

```
00100110    Dies ist Bitplane 0
00101011    Dies ist Bitplane 1
--------    Nun addieren wir sie, wobei wir die zweite mit 2
00302132    multiplizieren
```

Das ist also die gewünschte Grafik. Nun gäbe es aber natürlich auch eine andere Möglichkeit: Wir könnten die jeweils 2 Bits direkt hintereinander anordnen in sogenannten Chunks:

```
00 00 11 00 01 10 11 01 = 00302132
```

Dies ist das Prinzip des Chunky-Modus. Man kann im allgemeinen kaum sagen, daß eine dieser beiden Methoden besser oder schlechter ist. Allerdings haben natürlich beide ihre Vor- und Nachteile:

Zunächst hat vielleicht jeder schon einmal gesehen, daß auf dem Amiga beim Scrollen von farbigem Text ein gewisses Flackern entsteht. Genauer gesagt ändern sich kurz die Farben. Was dabei passiert, ist, daß der Computer Bitplanes verschiebt, gleichzeitig aber dieselben Daten für die Anzeige verwendet werden. Wenn etwa gerade Bitplane 0 verschoben wurde, aber Bitplane 1 noch nicht verschoben ist, so hätten wir kurzfristig im obigen Beispiel die folgende Anzeige:

```
01001100    Dies ist Bitplane 0 (nach links verschoben)
00101011    Dies ist Bitplane 1
--------    Nun addieren wir sie wieder
01203122
```

Sobald die zweite Bitplane ebenfalls verschoben ist, stimmt wieder alles, aber kurzfristig entsteht dabei eben jenes Flackern. Bei einer Chunky-Anzeige dagegen wäre eben nur ein Teil des Bildschirms schon verschoben und ein anderer Teil noch nicht.

Umgekehrt ist es im Chunky-Modus schlecht möglich, mit beliebiger Anzahl von Farben zu arbeiten: Da ein Byte 8 Bits hat, gehören etwa bei 4 Farben zu jedem Byte 4 Pixel. Man muß also stets erst berechnen an welcher Stelle des Bytes die Informationen zu einem bestimmten Pixel sitzen. Das ist aufwendig. Noch schlimmer wird die Sache bei 8 Farben: Da beginnen die Pixel noch nicht mal an der gleichen Stelle. Das ist sehr umständlich und rechenzeitaufwendig. In der Praxis gibt es daher Chunky-Anzeigen nur im 8-Bit-Modus (256 Farben) und im 24-Bit-Modus (16 Millionen Farben). Es ist allerdings möglich, daß die Anwender da gar nicht so unglücklich darüber sind...

5.2 Was ist Doublebuffering?

Bei animierter Grafik entsteht das Problem, daß gleichzeitig die Daten verändert und angezeigt werden. Dabei kommt es dann unweigerlich zu einem gewissen Flackern. (siehe Abschnitt 5.1 [Chunky vs. Planar], Seite 26)

Die Lösung des Problems ist es, quasi zwei Bildschirme zu benutzen: Der eine wird immer angezeigt. Gleichzeitig wird auf dem anderen Bildschirm, der nicht angezeigt wird, das neue Bild gezeichnet. Es gibt kein Flackern, da die Grafik-Hardware nur auf das RAM des ersten Bildschirms zugreift und der Prozessor nur auf das des zweiten. Dann wird umgeschaltet und der zweite Bildschirm angezeigt. Auf dem ersten kann jetzt gezeichnet werden.

5.3 Was für Monitore arbeiten am Amiga 1200 oder 4000?

Monitore kann man klassifizieren nach der horizontalen Frequenz, die sie für ihre Anzeige benötigen. Fernseher und Commodore's 1084 benötigen z.B. etwa 15 kHz, VGA und SVGA benötigen mindestens etwa 30 kHz. Multisync-Monitore können verschiedene Frequenzen darstellen.

Man kann also jeden dieser Monitore am A1200 verwenden, **aber**:

- Mit einem gewöhnlichen VGA/SVGA-Monitor kann man nur einige Anzeige-Modi (DblPal, DblNTSC und/oder Productivity, d.h. (320|640) x (256|512|1024) Pixel) verwenden. Dies ist großartig für die Workbench und die meisten ernsthaften Anwendungen, aber manche grafikorientierten Programme, vor allem Spiele laufen fast sicher nicht: Sie benutzen nämlich die Preferences nicht, übernehmen statt dessen einfach die Maschine und gehen von einem 15 kHz Monitor aus. Ferner kann man das Bootmenü nicht verwenden: Das arbeitet nämlich ebenfalls nur mit 15 kHz. Schließlich haben VGA-Monitore keine Lautsprecher und die VGA-artigen Modi unterstützen keine Genlocks.
- Das größte Problem eines 15 kHz-Monitors ist das ständige Flickern im Interlace-Modus. Ein kleiner Tip ist hier die Verwendung von NTSC anstelle von PAL. Dies erhöht die Refresh-Rate von 25 Hz auf 30 Hz, kostet allerdings einige darstellbare Zeilen. (Maximal möglich sind 482.) Die NTSC- und PAL-Modi sind aber nicht so schlecht, wie die Leute denken: Bei einem Monitor mit viel Phosphor (der also lange nachglüht) ist PAL-Laced einigermaßen akzeptabel und gibt eine Auflösung von 1448x566 in SuperHiRes. Mehr ist auf keinem AGA-Amiga möglich.
- Die beste Lösung sind die Multisync-Monitore, vorausgesetzt sie unterützen die Bereiche von 15-31 kHz Horizontal- und 50-72 Vertikalfrequenz. Die neuen 1940- und 1942-Monitore von Commodore sind nicht schlecht, allerdings etwas umständlich zu handhaben: Bei jedem Umschalten des Anzeige-Modus muß man nämlich auch die horizontale/vertikale Größe und den Offset von Hand einstellen.[6] Geeignet sind z.B. auch der Mitsubishi EUM 1491 oder der EIZO 9060S.

Für einen VGA/SVGA- oder Multisync-Monitor braucht man ein Kabel, das etwa 30 DM kostet.

6 Emulationen

Was? Der Amiga ist nicht gut genug? Sie wollen noch eine andere Maschine? Also gut, dann schauen Sie hier...

6.1 Kann ich meinen Amiga unter Unix benutzen?

Es gibt derzeit drei Unix-Versionen für den Amiga. Alle benötigen mindestens einen 68030 wegen der MMU (siehe Abschnitt 1.1 [MMU], Seite 1), 68040-Unterstützung ist erst in Arbeit. Alle scheinen Probleme mit manchen Harddisk-Controllern zu haben, man sollte sich deshalb anhand der Dokumentation informieren, ob sie auf der eigenen Maschine überhaupt laufen. Unix benötigt natürlich eine ungeheure Menge von Resourcen, 8Mb RAM und eine 150Mb-Partition für Unix dürften eher die Untergrenze sein.

[6] Es gibt einen Patch, der dies auch ohne manuelle Eingriffe möglich macht. Siehe '`os30/util/Monitor30Patch.lha`' auf dem Aminet.

1. Commodore hat in der Vergangenheit ein System-V-Unix angeboten. Es enthielt TCP/IP, X11 und andere Software und hat eigentlich einen guten Eindruck gemacht, aber war teuer und wird vor allem nicht mehr weiterentwickelt. Commodore-Unix benötigt einen Streamer, denn es wird auf solchen Bändern ausgeliefert.
2. Ein Linux-Port wird vorbereitet, allerdings gibt es derzeit nur den Kernel, das heißt die untersten Funktionen des Betriebssystems. Für Spezialisten könnte es interessant sein, kann aber wohl nicht empfohlen werden. Linux ist frei kopierbar. Quellen: `ftp.uni-paderborn.de` oder `ftp.uni-erlangen.de`, beide Directory '`/pub/Linux/MIRROR.tsx-11/680x0`'.
3. NetBSD ist ebenfalls frei kopierbar. Wie Linux ist es noch nicht fertig, aber macht derzeit gute Fortschritte. Beispielsweise läuft wohl schon die meiste GNU-Software, vor allem Emacs und gcc. Es ist auf jeden Fall einen Blick wert. Quellen: `ftp.uni-erlangen.de`, Directory '`/pub/amiga/unix/NetBSD-regensburg`'. Siehe Abschnitt 8.6.2 [Fish-CD], Seite 37.

6.2 Ist es möglich, den Amiga als X11-Terminal zu benutzen?

Es gibt zwei Softwarepakete, die das möglich machen:

GfxBase bietet ein kommerzielles Paket an. Es enthält verschiedene Window-Manager und Clients. Eine Demo-Version ist auf dem Aminet. ('`gfx/x11/GfxBase-X11-Demo.lha`')

Frei kopierbar ist DaggeX. Allerdings ist dieses möglicherweise noch nicht ganz fertig, zumindest wird die Versionsnummer mit 0.91 angegeben. Zu finden ist es ebenfalls auf dem Aminet. ('`gfx/x11/DaggeX-0.91.lha`' und '`gfx/x11/twm_930531.lha`')

6.3 Wie kann ich MS-Dos-Programme starten?

PC-Task ist ein softwaremäßiger IBM-Emulator für alle Amiga-Rechner. Die momentane Version 2.03 bietet die Emulation von 8086 mit MDA/CGA/EGA/VGA, seriellem und parallelem Port, Maus, zwei Festplatten und zwei Diskettenlaufwerke. Eine Demoversion liegt auf dem Aminet in '`/pub/aminet/misc/emu/PCTaskDemo203a.lzh`'. Chris Hames (bytey@melbourne.dialix.oz.au) (pctask@quasar.dialix.oz.au)

7 Verschiedenes

Dieses Kapitel enthält Dinge, die nicht in die anderen passen.

7.1 Gibt es eine Unix-Version von LhA?

Ja. Siehe Abschnitt 7.2 [Endungen], Seite 30.

7.2 Was sind Dateien, die mit ... enden?

Die meisten dieser Endungen besagen, daß die betreffende Datei komprimiert (gepackt) ist oder ein Archiv aus mehreren Dateien ist oder beides. (Einige Programme können sogar ganze Disketten archivieren.) Übliche Endungen und die dazugehörigen Programme sind:

.sfx Gepackte Archive, die in ein ausführbares Programm eingebunden sind; dieses wird einfach aufgerufen, um sich selbst zu entpacken. (sfx = self extract)

.lha

.lzh Gepackte Archive; empfohlen: LhA ('`util/arc/LhA_e138.run`' auf Aminet oder Fish-Disk 715) oder Lx ('`util/arc/lx100.lha`' auf Aminet), es gibt auch eine Unix-Version ('`misc/unix/lha-1.00.tar.Z`')

.dms Mit DMS komprimierte Disketten; Quellen: Aminet ('`util/arc/dms111.sfx`') oder Fish-Disk 406

.zom Mit Zoom komprimierte Disketten; Quellen: Aminet ('`util/arc/Zoom_5.4.lha`' oder Fish-Disk 682); eine ältere Version findet man auf Fish-Disk 459, diese ist möglicherweise für das Entpacken von PasTEX nötig

.zoo Komprimierte Archive; empfohlen: Zoo ('`util/arc/zpp2-10.lzh`' auf Aminet oder Fish-Disk 527)

.Z

.z

.gz Komprimierte Dateien; empfohlen: Gzip ('`util/pack/gzip124x.lha`' auf dem Aminet). diese Dateien sind meist Unix-Dateien

.tar Acrhive; empfohlen: tar ('`util/arc/tar.lha`' oder '`util/arc/gtar10.lha`' auf Aminet oder Fish-Disk 445), ebenfalls meist Unix-Dateien. Man findet häufig `.tar.Z` oder `.tar.gz`.

.arj Komprimierte Archive; empfohlen unarj ('`util/arc/unarj-0.5.lha`' auf Aminet)

.zip Komprimierte Archive; empfohlen UnZip ('`util/arc/unzip-5.1.lha`' auf Aminet). Dies sind meist MS-Dos-Archive.

7.3 Gibt es ein Programm wie Stacker, um die Hard-Disk zu packen?

XFH ist eine gute Möglichkeit. Es arbeitet als Handler und benützt die XPK-Libraries, man kann also zwischen verschiedenen (und in Zukunft vielleicht noch weiteren) Komprimiermodi wählen. (`NUKE` ist eine gute Wahl. Der einzige Nachteil ist, daß die Größe von Dateien durch das verfügbare RAM beschränkt ist, unter 2MBytes RAM kann man Probleme bekommen.)

Eine andere Möglichkeit ist EPU. Es ist Shareware und sollte dasselbe wie XFH bieten und außerdem ohne die Probleme mit der Dateigröße.

Quellen: Aminet, Directory '`util/pack`' und Fish-Disk 754 (XFH) sowie 858 (EPU).

7.4 Wo bekomme ich Fish-Disk xxx?

Einige FTP-Server haben genügend Platz oder ein CD-Rom gemounted und haben alle Fish-Disks online verfügbar:

```
ftp.isca.uiowa.edu      (USA, directory ‘/amiga/fx/fxxx’)
ftp.hawaii.edu          (USA, directory ‘/pub/amiga/fish’)
ftp.dfv.rwth-aachen.de  (Germany, directory ‘/pub/amiga/fish’)
ftp.funet.fi            (Finland, directory ‘/pub/amiga/fish’)
```

Beachten Sie bitte, daß die CD-Roms nicht immer gemounted sind. Siehe Abschnitt 8.3 [FTP], Seite 33.

Eine andere Möglichkeit wäre, Ihren örtlichen PD-Händler zu fragen. :-)

7.5 Wie füllt man die Tintenkartuschen der HPDeskjet-Drucker nach?

Dies ist eigentlich keine Amiga-spezifische Frage, taucht aber in den Amiga-Newsgruppen so hartnäckig und regelmäßig auf, daß sie auch in dieser FAQ beantwortet wird.

Die Drucker der Deskjet-Serie von HP besitzen einen in den Druckkopf integrierten Tinten-Vorratsbehälter. Wenn dieser Behälter leer ist, muß normalerweise der komplette Druckkopf ausgetauscht werden. Doch es geht auch billiger: Man kann den Tintenbehälter mit etwas Übung problemlos mehrfach nachfüllen.

Man benötigt dazu eine Einwegspritze mit passender Nadel, etwas schwarze Füllfederhalter-Tinte (z.B. Pelikan 4001) sowie Isopropanol (a.k.a. Isopropylalkohol, erhältlich in jeder gutsortierten Apotheke).

Als erstes gibt man in das neu gekaufte Tintenfäßchen einige Tropfen Isopropanol (ca. 0,5-1,0 ml pro 30 ml Tinte). Für eine Füllung des Druckkopfes zieht man dann etwa 10-15 ml der Tintenmischung in die Spritze, sticht mit der Nadel in die Öffnung im oberen (grünen) Teil des Druckkopfs und spritzt dann die Tinte langsam und vorsichtig in den Druckkopf. Achten Sie darauf, daß Sie die Nadel nicht ganz "bis zum Anschlag" einstechen, während des Einspritzens muß die verdrängte Luft noch durch die Öffnung entweichen können, sonst tritt die Tinte eventuell durch die Düsen am unteren Ende des Druckkopfs aus.

Die Dosierung des Isopropanols ist etwas kitzlig; zuviel fördert das Verlaufen der Tinte auf dem Papier und führt zu einem unsauberen Schriftbild, zuwenig führt eventuell zu verstopften Düsen am Druckkopf.

Es gibt mittlerweile von verschiedenen Herstellern auch sogenannte Nachfüllkits, die passende Spritzen und fertig vorbereitete Tintenmischungen enthalten. Preislich liegen diese "Fertiglösungen" etwa in der Mitte zwischen der Füllertinte und einem neuen Druckkopf.

Die neuen Spezialdruckköpfe mit doppeltem Volumen lassen sich übrigens angeblich nicht mehr nachfüllen - es wäre schön, wenn das jemand mal aus eigener Erfahrung bestätigen oder verneinen könnte...?

Jürgen Weinelt, jow@rz.uni-wuerzburg.de

7.6 Was ist MUI und wo bekomme ich es?

MUI besteht aus einer Reihe von shared-Libraries, die eine sehr komfortable graphische Benutzeroberfläche (GUI = graphical user interface) ermöglichen. Die generelle Idee von MUI ist es, den Programmierer nur die logische Struktur des GUI festlegen zu lassen. Das konkrete Aussehen (Fonts, Fenstergroesse, Fenster auf Workbench, eigenem oder öffentlichem Screen usw.) wird durch den Benutzer bestimmt. Für den Programmierer ist MUI erheblich einfacher und umfangreicher als die `gadtools.library`. Andererseits sind mit MUI erzeugte GUI's langsamer als die mit der gadtools.library erzeugten, vor allem auf alten 68000er-Maschinen.

MUI besteht aus zwei Archiven, eines für Programmierer und eines für normale Benutzer. Quellen: Aminet, '`dev/misc`'.

8 Software-Quellen und andere Informationen?

Drei Fragen entstehen in diesem Zusammenhang: Was für Programme gibt es überhaupt, wo und wie bekomme ich sie und wie kann ich sie dann nach Hause bringen?

8.1 Dateien und Datenbanken zur frei kopierbaren Software

Natürlich muß man wissen, wo man welche Software überhaupt findet. Viele wichtige Dinge sind bereits angegeben worden, wie ich hoffe. Weitere Informationen liefern:

AmigaSciUnixSchool
: ist eine Software-Liste im Ascii-Format. Sie wird monatlich in den Newsgroups `comp.sys.amiga.applications`, `comp.unix.amiga` und `news.answers` gepostet. (Aminet: '`text/doc/AmigaSciUnixSchool-4.01`'). Sie behandelt alles, was auch hier angegeben wurde und vieles mehr, z.B. GNU-Software, Libraries (Link-Libraries und shared Libraries), Shells, Unix-Kommandos, wissenschaftliche Software und vieles mehr.

FishCon
: sind die gesammelten Inhaltsverzeichnisse der Fish-Disketten im Ascii-Format. (Aminet: '`fish/doc/fishcon-???.lzh`')

FishXref
: ist ein Kreuzreferenzverzeichnis der FishCon-Dateien, ebenfalls im Ascii-Format ('`fish/doc/fishxref-???.lzh`' auf Aminet)

KingFisher
: Eine Fish-Disk-Datenbank, getrennt in Programm (Fish-Disk 863 oder Aminet, '`fish/doc/Kingfisher1_30.lha`') und Datendatei '`fish/doc/KFData850.lha`', die die Suche nach Namen oder Kontext ermöglichen.

8.2 Eine Sammlung von Testberichten

`Comp.sys.amiga.reviews` ist eine moderierte Newsgruppe, in der ausschließlich Testberichte über Soft- und Hardware, Bücher und alles mögliche Andere, den Amiga Betreffendes veröffentlicht werden. Es ist immer eine gute Idee, hier nachzuschauen, wenn man an etwas Bestimmtem interessiert ist. Natürlich findet man in der eigentlichen Newsgruppe nur die jeweils neuesten Berichte, aber die älteren werden archiviert und sind per FTP erhältlich bei `math.uh.edu`, Directory '`/pub/Amiga/comp.sys.amiga.reviews`' oder auf den Fish-CD's.

8.3 Empfangen von Dateien von einem FTP-Server

Software zu laden ist einfach, wenn man Zugang zum Internet mit einem Programm namens FTP (File Transfer Program) hat. Unix-Computer haben häufig beides.

FTP erlaubt Zugriffe auf andere Maschinen zum Speichern und/oder Laden von Dateien. Natürlich braucht man eine Zugangsberechtigung auf der anderen Maschine, aber viele Maschinen erlauben Zugang für jeden, wenn man sich als Benutzer `ftp` oder `anonymous` anmeldet und als Paßwort die eigene Mailadresse angibt. Für Amiga-Besitzer sind die wichtigsten FTP-Server die Aminet-Server, die sich gegenseitig Dateien übertragen und so im wesentlichen dieselben Dateien anbieten. Aminet-Server sind

```
USA (MO)     ftp.wustl.edu          128.252.135.4
USA (CA)     ftp.cdrom.com          192.153.46.2
USA (TX)     ftp.etsu.edu           192.43.199.20
Scandinavia  ftp.luth.se            130.240.18.2
Germany      ftp.uni-kl.de          131.246.9.95
Germany      ftp.uni-erlangen.de    131.188.1.43
Germany      ftp.cs.tu-berlin.de    130.149.17.7
Germany      ftp.uni-paderborn.de   131.234.2.32
Germany      ftp.uni-oldenburg.de   134.106.40.9
Germany      ftp.coli.uni-sb.de     134.96.68.11
Switzerland  ftp.eunet.ch           146.228.10.16
Switzerland  litamiga.epfl.ch       128.178.151.32
UK           ftp.doc.ic.ac.uk       146.169.2.1
```

Alle diese Server haben ein Directory '`/pub/aminet`', wo man massig Software findet. Bitte benutzen Sie einen Server in Ihrer Nähe! Einige andere wichtige Server sind

```
ftp.funet.fi             (Finnland)
ftp.isca.uiowa.edu       (USA)
ftp.hawaii.edu           (USA)
ftp.cso.uiuc.edu         (USA)
ftp.dfv.rwth-aachen.de   (Deutschland)
```

Grind, Aachen und Erlangen haben z.B. die komplette Fish-Disk-Serie parat! Siehe Abschnitt 7.4 [Fish-Disk xxx], Seite 31.

Um sich mit einem Server in Verbindung zu setzen (z.B. `ftp.uni-erlangen.de`), gibt man ein:

```
ftp ftp.uni-erlangen.de
```

Der Server antwortet mit der Aufforderung, den Benutzernamen einzugeben. Als Benutzernamen gibt man

```
ftp
```

ein. Nun wird man nach einem Paßwort gefragt. Hier sollte man seine Mailadresse (wenn man eine hat, sonst einfach ftp) eingeben.

Nun ist man mit dem Server verbunden und kann eine Reihe von Kommandos ausführen. Die wichtigsten sind:

? Gibt einen Hilfstext aus. Man kann auch **? Kommando** eingeben, um Hilfe zu einem bestimmten Kommando zu verlangen.

bin Informiert FTP, daß man binäre Dateien transportieren will. Es ist immer eine gute Idee, dies als allererstes Kommando einzugeben! Ohne dieses Kommando können empfangene Dateien verändert und damit nutzlos sein.

get <Datei>
Lädt die angegebene Datei vom Server. Auf den meisten Unix-Maschinen kann man auch '`get file.txt -`' oder '`get file.txt |more`' eingeben, um sich die angegebene Datei auf den Bildschirm ausgeben zu lassen. (Achtung: Hier darf **kein** Blank zwischen `|` und dem Wort more sein!)

mget <pat>
Lädt die angegebenen Dateien. Im Unterschied zu get dürfen hier auch Unix-Wildcards (* oder ?) verwendet werden.

put <file>
mput <pat>
Wie get und mget, aber es werden Dateien *zum* Server geschoben. Dies ist meist nur in speziellen Directories mit Namen wie '`incoming`' oder '`new`' erlaubt. Man kann dort Dateien plazieren, die man auf dem Aminet frei zugänglich machen will.

cd <dir> Wie das übliche cd. Die Kommandos get, mget, put, mput, dir und ls beziehen sich auf das angegebene Directory.

dir [<dir>]

ls [<dir>] Wie '`list`' und '`dir`' auf dem Amiga. Beachten Sie allerdings, daß FTP-dir dem Amiga-list entspricht.

bye Verläßt das FTP-Programm.

Wenn man FTP das erste Mal benutzt hat, wird man feststellen, daß immer die gleichen Schritte ausgeführt werden:

1. Benutzernamen eingeben (meist ftp)
2. Paßwort eingeben (meist die Mailadresse)
3. bin eingeben
4. In ein bestimmtes Directory wechseln (meist '`/pub/aminet/...`')

Dies kann man automatisieren. Dazu braucht man eine Datei namens '`.netrc`' in seinem Home-Directory. Diese muß unbedingt nur für Sie selbst lesbar sein, FTP akzeptiert sie sonst nicht! (Dies erreichen Sie unter Unix mit dem Kommando '`chmod go-rwx .netrc`'.) Die .netrc-Datei enthält für eine Reihe von Servern je einen Eintrag, die durch Leerzeilen getrennt werden. Ein typischer Eintrag sieht etwa so aus:

```
machine ftp.uni-erlangen.de
login ftp
password <Ihre Mailaddresse> oder <ftp>
macdef init
    bin
    cd pub/aminet
```

Auf einigen Computern ist auch der Servername '`default`' erlaubt, der für alle anderen Maschinen außer den Angegebenen gilt.

8.4 Empfangen von Dateien von einem Mail-Server

Eine andere Möglichkeit, Dateien zu empfangen, sind die Mail-Server. Dazu braucht man die Möglichkeit, an Internet-Adressen Mail zu verschicken und zu empfangen. Es funktioniert, indem man an den Server eine Mail schickt, in der man ihm sagt, was man haben möchte. Die Dateien werden dann ebenfalls als Mail geschickt, allerdings kodiert. Man braucht ein Programm namens `uudecode`, um sie zu dekodieren.

Die wichtigsten Mail-Server sind:

```
ftpmail@decwrl.dec.com
mailserver@nic.funet.fi
ftp-mailer@ftp.informatik.tu-muenchen.de
mrcserv@janus.mtroyal.ab.ca
mail-server@ftp.cs.tu-berlin.de
mail-server@rtfm.mit.edu
```

Die an einen Server zu schickende Mail darf eine Reihe von Kommandos enthalten. Die wichtigsten sind:

Help Veranlaßt den Server, einem einen Hilfstext zu mailen, in dem eine umfangreiche Anleitung enthalten ist.

Limit <Anzahl>
Gibt an, daß eine einzelne Mail höchstens <Anzahl> KByte lang sein darf. Größere Dateien werden in mehrere kleinere aufgeteilt, die als separate Mails verschickt werden. Beachten Sie, daß einzelne Mails durch die Deodierung und den Mailheader auch etwas länger sein können!

Cwd <dir>
Wie der cd-Befehl; das angegebene Directory wird von den Kommandos send und dir benutzt.

Index liefert eine Liste von Dateien und/oder Directories, die der Server anbietet. Diese Liste kann **sehr** lang werden! (Berlin z.B. 1 MByte)

Index <item>
liefert eine Liste von Dateien, deren Namen <item> enthält.

Dir [<dir>]
liefert eine Liste von Dateien und Directories im angegebenen Directory

Send <file1> <file2> ... <fileN>
Liefert einem die angegebenen Dateien

Begin Veranlaßt den Server, alle Zeilen oberhalb zu ignorieren.

End Wie Begin, aber für die unten folgenden Zeilen. (Eine Signatur zum Beispiel!)

Eine typische Mail an einen Mail-Server sieht also so aus:

```
BEGIN
CD /pub/aminet/util/arc
SEND LhA_e138.run
END
```

8.5 Empfangen von Dateien von einer Mailbox

Man kann sich auch Daten von einem der vielen BBS (`Bulletin Board System`) holen. Zumeist werden diese von privaten Anwendern betrieben, und deshalb gibt es fast überall eine oder mehrere BBS, die man zum Ortstarif erreichen kann.

Mailboxen bieten zumeist Möglichkeiten zum Meinungs- und Datenaustausch unter allen Teilnehmern, sie stellen eine Reihe von Programmen zum `Saugen` zur Verfügung, und sie bieten oft noch einige andere Serviceleistungen an.

Hier folgt eine Liste aller mir bekannten Amiga-Mailboxen. Bitte schreibt mir die Namen und alle Telefonnummern der Euch bekannten Amiga-Mailboxen sowie (wo nötig) eine kleine Bemerkung, danke.

Mailbox	Vorwahl	Nummer 1	Nummer 2	Nummer 3	Bemerkung
IMAGINE *	de-089	6892721			
AMIGA WORK *	de-089	6256183	6256159		
COMCOR *	de-089	7141035			Computer Corner BBS
FORTRESS	de-089	8915316	8110130		
Black Empire	de-089	472396	6885313		
AMIGA WORK II	de-089	6258696			
KUCKUCKSNEST	de-089	183000			
ERESSEA	de-089	6888534			
MAGIC	de-08121	45578			
NAMELESS	de-08285	1008	1630		Burtenbach
NATHAN	de-08191	65542			Landsberg
STAR BBS	de-08232	6077			Schwabmünchen
Shannara	de-09931	72923			Plattling

wobei `de` hier für Deutschland steht.

8.6 Die Fish-PD-Serie

Eine sehr gute Quelle ist die Fish-PD-Serie. Man muß zwischen Disketten und CDs unterscheiden.

8.6.1 Die Amiga-Library-Disks

Fred Fish hat in der Mitte der Achtziger begonnen, frei kopierbare Software zu sammeln. Es gibt inzwischen 1000 Disketten und viele sehr gute Software darauf. Viele Amiga-Händler bieten sie an und die meisten Amiga-Zeitschriften enthalten Anbieter, die sie für ca. 3 DM pro Diskette mit der Post versenden. Die Disketten werden nicht mehr von Fred Fish selbst fortgeführt. Es gibt einen Drittanbieter, der die neu erscheinende Software auf den Fresh-Fish-CDs weiterhin in Diskettenform anbietet.

Es gibt Dinge, die auf den Fish-Disketten, nicht aber auf dem Aminet verfügbar sind. Trotzdem ist es möglich, sie per FTP zu bekommen. Siehe Abschnitt 7.4 [Fish-Disk xxx], Seite 31.

8.6.2 Die Fresh-Fish-Serie

Fred Fish bietet weiterhin frei kopierbare Software an, allerdings jetzt auf CD. Es gibt zwei verschiedene Arten:

1. Die monatlichen CDs sind drei Teile unterteilt:
    1. Neues Material, sowohl auf den Disketten erscheinende als auch dort nicht erscheinende Software. Auf der ersten CD sind das etwa 84Mb.
    2. Nützliche, bereits installierte Software, die direkt von der CD benutzt werden können und damit Platz auf der Festplatte sparen. (GNU Emacs, Gnu C, GNU C++, Amiga E, PasTeX, AmigaGuide, Installer, 2.0 und 3.0-Includes, verschiedene Archivierprogramme, das AmiCDROM Filesystem, GNU und BSD-Utilities...) Dieser Teil macht etwa 150Mb auf der ersten Disk aus.
    3. Älteres Material, das bereits früher erschienen ist. (Ungefähr 404Mb auf der ersten CD, entsprechend Fish-Disk 600-910.)
2. Die zweite Sorte enthält im wesentlichen mehr Software, dafür aber in gepacktem Format. (Diese CD's sind speziell für Mailboxen gedacht, die dadurch diese Software zum Downloaden anbieten.)

Ich empfehle vor allem die erste Sorte. Sie kosten etwa 20$ plus 5$ Versandkosten und können bei der folgenden Adresse bestellt werden:

```
Amiga Library Services
610 N. Alma School Road, Suite 18
Chandler, AZ 85224-3687
U.S.A.

Phone/FAX: (602) 917-0917
```

Als beste Zahlungsweise haben sich angeblich Kreditkarten bewährt. Allerdings bin ich überfragt, welche von Fred akzeptiert werden.

8.7 Wie kann ich MS-Dos-Disketten lesen und schreiben?

Für Besitzer von Workbench 2.1 oder höher ist das kein Problem: Das Programm CrossDos ist da Teil der Workbench. Man muß lediglich '`pc0:`' mounten, indem man die Datei '`Sys:Storage/DOSDrivers/pc0:`' startet oder nach '`Devs:Storage/DOSDrivers`' verschiebt. MS-Dos-Disketten in '`df0:`' können nun ganz normal behandelt werden, indem man jeweils das Wort '`df0:`' durch '`pc0:`' ersetzt. Z.B. kann man das Directory mit '`dir pc0:`' anzeigen.

Alle anderen benötigen ein Programm namens `msh` (Aminet, Directory '`misc/emu`' oder Fish-Disk 382). Nachdem man die Datei '`devs:MountList`' wie in der Dokumentation angegeben verändert hat, muß man lediglich im CLI das Kommando '`mount msh:`' eingeben und kann dann wie bei CrossDos damit verfahren, wobei man natürlich jeweils '`msh:`' anstelle von '`pc0:`' angeben muß.

8.8 Wie transportiere ich sehr große Dateien

Es gibt einige Archive, die zu groß sind, um auf eine Diskette passen. (Das gcc-Archiv ist z.B. 3,5 MByte groß.) Um diese zu transportieren, benötigt man ein Programm, das sie in kleinere Teile aufteilt, die dann auf verschiedenen Dateien transportiert werden. Ich empfehle Martin Schlodders `Splitter`. (Aminet, '`util/misc/splitter_121.lha`') Das Archiv enthält Binaries für den Amiga und MS-DOS und der Quelltext sollte ohne Probleme auf jedem Unix-Rechner zu compilieren sein.

8.9 Diskussionen über Mail

Eine Mail-Liste ist ein Server, der es ermöglicht, sich mit anderen Leuten über ein bestimmtes Thema via Mail zu unterhalten. Der Server unterhält eine Liste interessierter Teilnehmer, die am Thema interessiert sind. Z.B. geht es bei der gcc-Mailliste um Bugs, neue Features und andere Probleme rund um den gcc. (siehe Abschnitt 3.5 [Compiler], Seite 9) Wenn ein Teilnehmer eine Mail an den Server schickt, dann wird sie vom Server an alle anderen Teilnehmer weitergeleitet.

Man meldet sich als Teilnehmer an, indem man eine Mail an den Server schickt, die z.B. das Wort '`Subscribe`' enthält. Umgekehrt kann man sich auf ähnliche Weise abmelden, wenn man an der Mail-Liste nicht mehr länger interessiert ist.

Unglücklicherweise erwarten die Server zum Teil eine recht unterschiedliche Syntax bei der Anmeldung. Allerdings unterstützen sie alle eine Hilfefunktion: Schickt man eine Mail mit dem Wort '`Help`' an den Server, so antwortet dieser mit einer ausführlichen Beschreibung.

Einige interessante Mail-Listen sind:

Topic	Server

Amok	listserv@amokle.stgt.sub.org
Dice	dice-request@hactar.hanse.de
Gcc	listserv@lists.funet.fi
Lisp	amigalisp@contessa.phone.net
Mui	mui-request@taloa.unice.fr
Oberon-A	oberon-a-request@wossname.apana.org.au

8.10 Andere FAQ's

Amiga related books FAQ

Enthält eine Liste von Büchern zum Amiga, komplett mit Kurzbesprechungen, Preisen und Herkunftsangaben. Newsgruppen: comp.sys.amiga.misc, comp.sys.amiga.introduction, comp.sys.amiga.programmer (monatlich) Ftp: rtfm.mit.edu, pub/usenet/comp.sys.amiga.misc Betreuer: Marc Atkins, atkin@cs.umass.edu

AmiTCP/IP FAQ

Dies ist für Anwender von AmiTCP/IP, einer Reihe von Programmen, die das Einbinden eines Amigas in ein TCP/IP-Netzwerk ermöglichen. (Die meisten bekannten Netze, das Internet z.B., benutzen TCP/IP.) Newsgruppen: comp.sys.amiga.misc, comp.sys.amiga.datacomm, comp.sys.amiga.networking (zweiwöchentlich) Ftp: rtfm.mit.edu, pub/usenet/comp.sys.amiga.networking Betreuer: Neil J. McRae (atcpfaq@domino.demon.co.uk)

Amiga Networking FAQ

Im Gegensatz zur AmiTCP/IP-FAQ möchte diese alle Aspekte des Netzwerkelns abdecken, z.B. auch Envoy. Newsgruppen: comp.sys.amiga.datacomm, comp.sys.amiga.hardware Ftp: rtfm.mit.edu, pub/usenet/comp.sys.amiga.networking Betreuer: Richard Norman (norman@afas.msfc.nasa.gov)

Point Manager FAQ

Netzwerke scheinen wirklich kompliziert zu sein: Dies ist die dritte FAQ zum Thema, diesmal zu einem speziellen FidoNet-Client (einem sogenannten Point), dem Point Manager. Newsgruppen: comp.sys.amiga.datacomm Ftp: rtfm.mit.edu, pub/usenet/comp.sys.amiga.datacomm Betreuer: Eric Krieger (pm_faq@quasar.hacktic.nl)

All about FTP

Erklärt den Umgang mit dem Filetransferprogramm FTP. Siehe Abschnitt 8.3 [FTP], Seite 33. Newsgruppen: comp.sys.amiga.misc (Monatlich) Ftp: Aminet, info/start Betreuer: Urban Dominik Müller (umueller@amiga.icu.net.ch)

Das Amiga-FAQ-Archiv

Die Amiga-FAQ ist in verschiedenen Formaten erhältlich: Im Ascii-Format (wie sie in den Netzen gepostet wird), im AmigaGuide-Format (wie sie auf einem Amiga wohl am praktischsten ist) und im dvi-Format zum Ausdrucken. Weiter gibt es einige Dinge, die nützlich oder interessant sein könnten, die aber nicht in den Text eingebunden werden konnten:

```
txt/amiga.history       Zur Geschichte des Amiga
txt/story.txt           Die Commodore-Story (oder: Die Tramiel-
                        Story ;-)
txt/amiga.newsgroups    Übersicht über comp.sys.amiga.*
txt/amiga.sites         Liste von FTP-Servern
txt/AmigaOverview.tex   Übersicht über die Amiga-Soft- und Hardware
txt/Hardware.tips       Für Hardware-Bastler
txt/Nullmodem.txt       Anleitung zum Bau eines Nullmodems
src/JWSplit.c           Der Quelltext eines Dateisplitters
src/JWJoin.c            Das Gegenstück zu JWSplit.c
src/addtoc.c            Fügt ein Inhaltsverzeichnis zu mit texinfo
                        erzeugten doc's bei (dieser Text verwendet es)
```

Um diese öffentlich zugänglich zu machen, habe ich mich dazu entschlossen, diese in einem Archiv zu sammeln. Es heißt '`AmigaFAQxxxxxg.lha`', wobei xxxxxx das Datum der letzten Version ist. Sie finden es per FTP im Aminet, Directory '`docs/misc`'.

Beiträge zur Amiga-FAQ

Die FAQ kann nicht nützlich sein und nicht weiterentwickelt werden ohne Ihre Hilfe. Vorschläge, Beiträge, neue Fragen und Antworten, Kritik, alles ist willkommen.

Bitte beachten Sie, daß einige sehr wesentliche Themen bis jetzt unterrepräsentiert sind oder gänzlich fehlen: Nichts über Animation, Sound, Grafikkarten. (Alles über die *endgültige* Merlin-Software ...) Dies sind einige der Stärken des Amiga! Aber ich verstehe nichts davon :-(

Also nehmen Sie ihre Tastatur (Ihren Bleistift? Na gut, wenn's sein muß...) und schreiben Sie an:


```
      Ignaz Kellerer
      Georg-Habel-Str. 11
81241 München (Deutschland)
      Tel. (+49) 089 / 885147

      Internet: kellerer@informatik.tu-muenchen.de
```


Danksagungen

Meinen Dank an:

Reinhard Spisser and Sebastiano Vigna
: für die Amiga-Version von TexInfo. Dieser Text wurde damit erstellt.

The Free Software Foundation
: für die Originalversion von TexInfo und viele andere hervorragende Programme.

Dylan McNamee
: für die Abschnitte über Editoren, Textverarbeitungen, DTP und PostScript.

Joseph Luk
für die Abschnitte über Chunky/Planar und Double-Buffering

Urban Dominik Müller
für die FAQ über FTP- und Mail-Server

Lars Hecking (lhecking@nmrc.ucc.ie)
Philippe Brand (phb@colombo.telesys-innov.fr)
für den Abschnitt über gcc

Jochen Wiedmann (zrawi01@decap2.zdv.uni-tuebingen.de)
für die Zusammenstellung und das Posten der Amiga FAQ bis Juli 1994.

Index

G

H

I

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Table of Contents