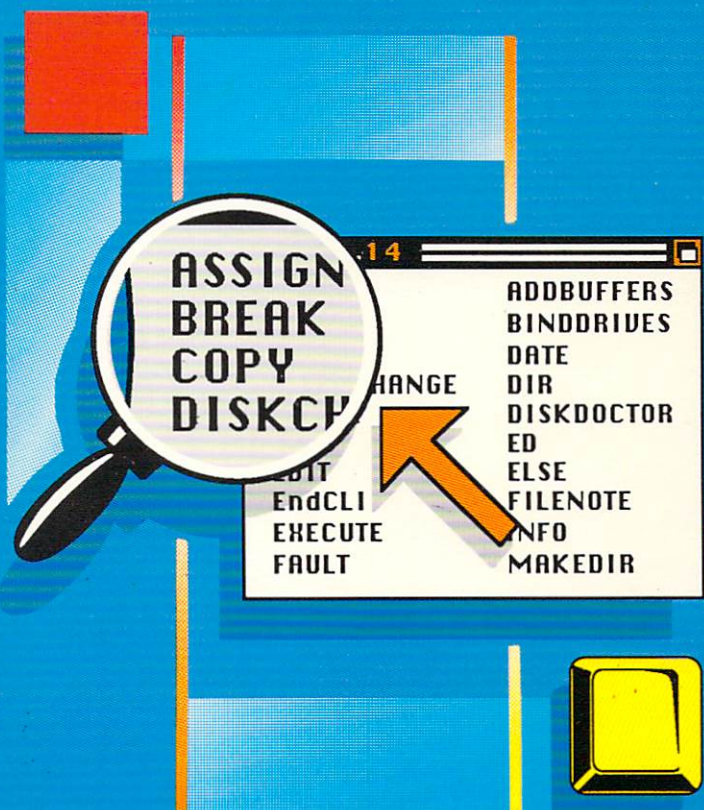



AmigaDOS[®]

Quick Reference



Abacus 
A Data Becker Book

Includes
WORKBENCH
1.3

AmigaDOS *Quick* *Reference*

Hannes Rügheimer
Christian Spanik

Abacus 

A Data Becker Book

First Printing, December 1988

Printed in U.S.A.

Copyright © 1987, 1988

Data Becker GmbH

Merowingerstraße 30

4000 Düsseldorf, West Germany

Abacus

5370 52nd Street

Grand Rapids, MI 49508

Copyright © 1988

This book is copyrighted. No part of this book may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, electronic, mechanical, photocopying, recording or otherwise, without the prior written permission of Abacus Software or Data Becker GmbH.

Every effort has been made to ensure complete and accurate information concerning the material presented in this book. However, Abacus Software can neither guarantee, nor be held legally responsible for, any mistakes in printing or faulty instructions contained in this book. The authors always appreciate receiving notice of any errors or misprints.

Amiga 500, Amiga 1000, Amiga 2000, AmigaDOS and Amiga are trademarks or registered trademarks of Commodore-Amiga, Inc.

ISBN 1-55755-049-2

Table of Contents

•	Introduction	1
•	About this Guide	2
•	General Information	5
•	Commands	10
•	Error Messages	90
•	CLI Shortcuts	94
•	ASCII Table	96
•	Escape Sequences	98
•	Memory Map	101
•	Guru Meditation Codes	102
•	Quick Index	106
•	Subject Index	108
•	Index	111

Introduction

The **AmigaDOS Quick Reference** is for the intermediate and advanced level AmigaDOS user. However, it is not an introductory guide to AmigaDOS. We assume that you have a general knowledge of the Amiga and experience in working with AmigaDOS.

AmigaDOS is an exceptionally powerful disk operating system used on the Amiga family of computers. Although many Amiga users may be content to perform disk maintenance chores from the Workbench, AmigaDOS is much more flexible than those commands available from the Workbench.

AmigaDOS features a large command set. These commands are difficult for most users to memorize, though. The **AmigaDOS Quick Reference** lets you find AmigaDOS commands quickly and easily.

About this Guide

This guide has three major sections: Introduction, General Information, Commands and three Indexes.

The Commands section describes all AmigaDOS commands in alphabetical order followed by their syntax and description.

There are three indexes included in this guide for easy reference. For example, if you know the name of the command, but are unsure of its usage and syntax, refer to the *Quick Index*. Each command is listed in alphabetical order, and refers you to the page on which it can be found.

If you are looking for a command to perform a certain task and you are not sure of its name or syntax, refer to the Subject Index. There you will find each command in alphabetical order according to task, with a brief description and the page on which it can be found.

For general information refer to the General Index.

The characters in the command descriptions need some explanation. The following is a sample command:

```
DELETE Name [ Name2 ... Name10] [ALL] [QUIET|Q]
```

Command words are displayed in courier type like this: Delete.

Values, names, etc. are written in uppercase and lowercase letters. Name and Name2 are names, in this case filenames, that can be entered after the command.

The brackets [] are optional and need not be entered. They can be used to handle additional possibilities and to make it easier to read.

The character | stands for "or". In the above example the word QUIET or the letter Q can be entered.

... stands for further parameters. For example, with Delete you can enter up to 10 different names here.

General Information

Disk drive labels, devices (logical devices), etc.

DF0:, DF1: Disk drives (DF stands for "drive floppy").

DF2:, DF3: The four drives that AmigaDOS can access have numbers 0 through 3.

DH0:,
DH1:, ...: Partitions on any Amiga ready hard disk (over the SCSI controller).

JH0:,
JH1:, ...: Amiga partitions on a PC ready hard disk.

RAM: RAM disk of the Amiga. Contains only enough memory needed for the contents.

PRT: Addresses the printer set in Preferences. Command characters (see appendix "Standard printer command characters") are converted by the printer driver.

PAR: Parallel interface. Used mostly for the printer. No command character conversion is done.

SER: Serial interface with the parameters set in Preferences. No command character conversion is done when a printer is connected with this interface.

CON: Console device. Mostly an input/output window for AmigaDOS. After CON: the following window data can be entered:

"CON:X-startposition/Y-startposition/Width/Height/[Window-Name]"

The descriptions for positions, width and height are pixels.
<Ctrl><\> closes a Con: window.

RAW: Can open a window like a Con: device:

"RAW:X-startposition/Y-startposition/Width/Height/[Window-Name]"

Input in RAW: is passed on to AmigaDOS. The window can no longer be closed because the <Ctrl><\> code does not work.

- NIL:** Pseudo device that takes input and does nothing with it. Data sent to NIL: simply disappears. It can be used to dispose of unwanted output.
- *** Not a true device. The asterisk stands for the current window. It can be used to direct output in or out of the actual window.
- SYS:** Stands for the current Workbench disk.
- C:** AmigaDOS expects the DOS commands to be in this directory.
- S:** Script files (also called sequences) should be here. The `Execute` command looks first in the current directory and then in the S: directory. The startup sequence is also expected to be here.
- T:** Directory for temporary help files, etc.
- L:** Libraries that AmigaDOS itself uses are stored here.
- LIBS:** Libraries that belong to the operating system are in this directory.
- DEVS:** The driver programs for the AmigaDOS devices are found in the Devs: directory. The Keymaps, Printers and Clipboards directory (for Clipboard contents) are found here. `Preferences` saves the valid entries in the file `System-configuration`.
- FONTS:** The different character sets for the Amiga are in this directory.
- System:** AmigaDOS commands that can be started as programs from the Workbench surface should be stored here.
- Utilities:** The utility programs Notepad and Calculator are located here.
- Expansion:** All driver programs for hardware expansion are here. They should be connected to the system by using the `BindDrivers` command.

Construction of Filenames

A valid AmigaDOS filename consist of the following elements:

[Diskname|Drive|Logical Device]: [Directoryname/] [Directoryname/.../Directoryname/] Filename

Filenames in AmigaDOS can be up to 30 characters long. The characters (:) and (/) cannot be in the filename itself. The colon (:) marks the disk name, the slash (/) separates individual directory names. The filename must be in quotation marks if it contains characters like spaces, parentheses () or semicolons (;).

A disk can be addressed either with its name (e.g., "A2000 WB 1.2D"), the drive containing it (DF0:), or with a logical device name (SYS:).

Any number of subdirectories can be stored within each other. AmigaDOS knows the structures of files and directories on the disk.

Wildcard Characters

To declare multiple files at the same time, AmigaDOS offers a very efficient system of *patterns* or *wildcard characters*.

Not all AmigaDOS commands understand the wildcard characters. Here is an overview of the commands which can use the wildcards in their parameters:

Copy, Delete, Dir, List, Search

Wildcard descriptions can be made up of the following elements:

- ? Any character (except a null string)
- # The following section can occur once or more
- % A null string
- . The following characters are part of the filename; the wildcard stops here

-
- | Logical OR. One of two given wildcard combinations should be satisfied
 - () Parentheses join wildcard combinations to another unit

Some uses and typical combinations:

- #? Every file (any number of occurrences of every character in the filename)
- #?.info All files that end with .info
- (?|%)test Filenames like Test, ATest and MTest. The first character is missing or is one character long
- ?est Filenames like Test, Vest, Nest, etc. The first letter can be anything
- h#(a|e)llo Filenames like Hallo, Hello, Heeeello

The Argument Template

Most AmigaDOS commands have a argument template (help option). If you enter the command name followed by a space and a question mark, the command displays a line with all of the arguments.

After entering:

```
list ?
```

the following argument template appears on the screen:

```
DIR, P=PAT/K, KEYS/S, DATES/S, NODATES/S, TO/K, S/K, SINCE /K, UPTO/K, QUICK/S:
```

AmigaDOS gives the arguments and qualifiers and waits for input. Now you can enter the parameters that would normally follow the command.

The three descriptions /A, /K or /S usually appear behind the options. They have the following meanings:

- /A** The parameter must be entered without any conditions
/K The given option must be entered with the same command word. In the above example P=PAT/K means that: If after List a wildcard character is given (Pat stands for Pattern),

LIST PAT Wildcard character

or

LIST P Wildcard character

must be entered.

- /S** The parameter is optional. It stands alone if given. The above example displays the statement Quick/S:. When Quick is given, the syntax reads:

LIST QUICK

The argument template has a big advantage for users with only one disk drive: Commands that were loaded from one disk but should be used on another disk can be placed in memory in a short time.

Place the Workbench disk in the drive and enter the name of the command followed by a space and a question mark. For example:

dir ?

The Dir command is now loaded and displays:

DIR, OPT/K:

Now you can change the disk. Insert the disk that contains the directory you want to see. Dir always waits for input. Enter:

df0: opt a

The directory of the disk in drive df0: appears.

Sample AmigaDOS Command Page

Here's a quick explanation of the format this guide uses to describe AmigaDOS commands, descriptions and syntax.

Command



SetClock

Syntax: SETCLOCK OPT LOAD|SAVE

Syntax

This command is of interest only to those Amiga owners who have a built-in real time clock. AmigaDOS manages its own time which is completely independent of the time of the real time clock.

Description

The concept "time" includes the clock time and the date. The real time clock can be read or set with SetClock.

SetClock Opt Load reads the time from the real time clock and transfers it to AmigaDOS. The AmigaDOS time is then set according to the real time clock. When this command is executed in the startup sequence, AmigaDOS recognizes the real time clock as the correct time. This spares you from having to enter the correct time after every system start.

Use the Date command to enter the current clock time and the current date from AmigaDOS.

Note: When the real time clock is first put into operation, the SetClock Opt Load command may display the message:

<invalid> <invalid> <invalid>

The real time clock contains some data that cannot be interpreted as the clock time and date. After the first SetClock Opt Save this message doesn't appear any more. SetClock gives the message:

Internal clock not functioning

if the real time clock is defective or not built-in.

**Workbench
1.3 version**

Command Under Workbench 1.3

No changes have been made.

Commands

;

Syntax: [Command]; [Comments]

The semicolon separates comments from AmigaDOS commands. AmigaDOS ignores any characters behind the semicolon. The most frequent use of the semicolon is to comment lines in script files. The semicolon can be placed at the beginning of the line without a command being given first.

For example:

; The next lines show contents of the dir directory

< >

Syntax: Command >Filename|Device[Options]

The < and > symbols allow the input or output of a command to be redirected. A file or a logical device can be given after > and the output is then sent there.

A file or logical device name can be given after < and input is then taken from this for the command.

For example:

```
dir >ram:Temp opt a
dir <con:0/0/300/50/ opt i
```


AddBuffers

Syntax: ADDBUFFERS [DRIVE] Drive [BUFFERS] Number

AmigaDOS automatically reserves a buffer for each connected drive. For multiple readings of data, the data is already located in the buffer.

AddBuffers makes it possible to increase the size of a drive buffer. This decreases access time.

This increase is only noticeable through the disk drive. No speed improvement is achieved when using a hard drive or RAM disk. The value "Number" gives the buffer 512 byte blocks.

Note: You shouldn't distribute buffer memory too liberally because each buffer block reduces the size of the system memory. There is no way to recover the buffer memory after a reset. Values in the range of 25 to 30 per drive are a good compromise.

Command Under Workbench 1.3

No changes have been made.

Alias

(Workbench 1.3 only)

Alias allows you to assign multiple command parameters a single user-defined command word.

Enter the following sample line in the Shell:

```
Alias Ramdir Dir Ram:
```

Now the new command **Ramdir** can be used. You see, it is very simple. First the command word (**Alias**), then the label of the new command (**Ramdir**), and then what the command does (**Dir Ram:**).

Which **Alias** commands you can use and how they are built can only be determined by entering **Alias** and then pressing the <Return> key. Here are examples of **Alias**.

1. Let's say you'd like to change the disk drive, but without much typing. Type in the following lines:

```
alias 0 cd df0:
alias 1 cd df1:
alias 2 cd df2:
alias r cd ram:
.
.
.
```

After entering these lines they are in position to change the drive to 0 or 1 or 2 or r. You can do so with other drives as well (example: dh0:). You can also change the directory in a given drive. To change the C: directory to drive df1:, you must enter the 1 c command.

2. Delete the contents of the **Shell** window.

```
Alias CLS echo "*ec"
```

Enter this line in the **Shell**. Then enter **CLS**. The contents of the window are erased and the prompt appears in the top of the window. This is accomplished by using the **Echo** command and an **Escape** sequence. The sequence is located inside the quotation marks. The first two characters inform the computer that an **Escape** sequence will follow. The **c** erases the screen. You will find more **Escape** sequences in the section named **CLI Shortcuts** (see page 94). The prompt stands in the second line, not the first. To get it to appear in the first line after the screen is erased you need to use a little trick. First, you must change the prompt character a little with the following line:

```
Prompt "*e[11y*e[33m*e[1m*e[3m*s*e[0m*n*e[t"
```

After you enter the line and press the <Return> key the result is displayed. The directory is displayed in italics and is colored

orange. In addition, the input takes place in the next line. These appearances cause the different Escape sequences in the Prompt command.

Now enter the new CLS command:

```
Alias CLSC echo "*ec*[2y*[2t"
```

When you erase the screen with CLSC, the prompt appears in the top line of the window.

3. Because we have talked about the Escape sequences in commands before this, we would like to give you some more uses. The first example does not use the Alias command but is still rather interesting.

```
echo "*e[1m*[3mThis is bold and italic.*e[0m"
echo "*e[32m*[43mThis is black text and orange
background.*e[0m"
echo "*e[7m*[4mThis is inverted and
underlined.*e[0m"
```

You notice the changes affect not only the output. The Alias command can bring about many other changes.

```
Alias Prom1 Prompt "*e[32m*[43m%s> "
Alias Prom2 Prompt "*e[42m*[31m%s> "
Alias Prom3 Prompt "*e[41m*[33m%s> "
Alias Prom4 Prompt "%nter Task*n%s*n- "
```

In normal mode you can enter echo "*ec".

4. The following example sends a file to the printer, leaving the computer for other tasks.

```
Alias Print run copy to prt: [] clone
```

The brackets serve as placeholders for the parameters that are entered with Print. The file must be given what you want it to print out. For example:

Print `sys:s/startup` sequence

5. There is a command that makes a file "invisible" so that it is not displayed when a `Dir` or `List` command is executed.

```
Alias Hide protect [] +h
```

A file can be covered up by entering the `Hide` command followed by the filename. With the `Protect` command the `H` status bit of the file is set. The computer must use `Kickstart 1.3` to use this command. This `H` bit is disregarded when `Kickstart 1.2` is used.

6. Like in 5, the status bit `S` can be set. The following line must be entered:

```
Alias SBit protect {} +s
```

Then you can execute a script file with the `SBit` filename without using the `Execute` command.

7. There is an `Alias` command already in your `Shell-Startup` file. It is the `xCopy` command. The same principle can be used for deleting.

```
Alias xDelete delete [] all
```

After entering this line, the contents of the directory can be erased. Enter a `xDelete` and a directory in the `Shell`.

8. The `CLI/Shell` commands can be abbreviated with the `Alias` command.

```
Alias c copy
Alias p path
Alias d dir
Alias ex execute
Alias d delete
```

```
Alias t type
Alias r rename
Alias e echo
.
.
.
```

The individual commands can be used after entering the lines with the respective abbreviations.

Ask

Syntax: Ask "Question"

With the Ask command it's possible for AmigaDOS to ask questions of the user from inside of script files. Earlier versions of Workbench 1.2 may not have this command.

Enter a question text in quotation marks after Ask. This text is displayed on the screen when the script file is executed.

The user can choose between two answers: Y for Yes and N for No. When another answer is given, Ask repeats the question and waits for a Yes or No answer. Only the first character of the input is relevant. Yabadabadoo would be read as Y.

The answer Y has an error number 5. The answer N has error number 0.

Note: If you want to leave the script file with the answer Y, set the error limit beforehand with Failat at 5. Then after entering Y the number is increased to 10.

By using If Warn the input Y or N can be executed from the program without interrupting your script file. Compare the Failat command and If [Warn/Error/Fail].

For example (the following script files must be entered with ED and started with Execute):

Break a script file by resetting the error limit:

```
failat 5
ask "Would you like to stop the script file (Y/N)"
echo "The script file continues to execute."
```

Different reactions to the input using If Warn:

```
ask "Enter a Y or N please"
if warn
    echo "You have entered Y"
else
    echo "You have entered N"
endif
```

Command Under Workbench 1.3

No changes have been made.

Assign

```
Syntax:  ASSIGN [LIST]
           ASSIGN [NAME] Logicaldevice
           ASSIGN [NAME] Logicaldevice [DIR] Directory
```

The Assign command assigns a logical device name to a corresponding directory. When the given logical device already exists, your assignment is ignored, otherwise Assign creates a new logical device with the given name. This assignment is used to abbreviate long pathnames. When you enter:

```
assign p: sys:Preferences
```

Preferences can be called by entering:

```
p:
```

The most frequent use of Assign is to transfer system directories. Directories like C:, Fonts:, Devs:, Libs:, etc. can be placed on other drives.

Assign with a device name but without a directory statement erases the assignment for the logical device.

Note: You should usually erase only the logical devices that you have assigned yourself. When you tamper with the assignments for system directories, AmigaDOS is then unable to find DOS commands, libraries or fonts.

Remember that the assignments are lost next time the computer is turned off or reset. **Assign** commands that should always be valid must be put in the startup sequence or another script file that can be started with **Execute**.

Assign List and **Assign** without parameters give an overview of all the current assignments on the screen:

Volumes:

RAM Disk [Mounted]

A2000 WB 1.2D [Mounted]

Directories:

Utilities A2000 WB 1.2D:Utilities

System A2000 WB 1.2D:System

sys A2000 WB 1.2D:

fonts A2000 WB 1.2D:fonts

devs A2000 WB 1.2D:devs

libs A2000 WB 1.2D:libs

l A2000 WB 1.2D:l

t A2000 WB 1.2D:t

s A2000 WB 1.2D:s

c A2000 WB 1.2D:c

Devices:

Df2 DF1 DF0 PRT PAR

SER RAW CON RAM

Disk names that AmigaDOS knows are listed under **Volumes:**. The disks that are in the drives at the present time are marked with [Mounted].

The Directories: heading is the current assignment list. The assignments for the system directories are located here. The devices that are connected at the time are given under Devices:.

The following syntax is also possible. This creates a new assignment and then gives the current Assign overview:

```
assign Texts: ram:Texts list
```

Command under Workbench 1.3

The Assign command can be used to test whether a certain device exists. The device name and list must be added to the command to do this. When the given device is present, the respective entry is shown and the error status is set at zero.

When the device is not present, the Assign command returns the error status 5.

Avail

(Workbench 1.3 only)

Syntax: AVAIL

Until now the statement in the Workbench title bar kept you informed of the working memory. Thanks to the new Avail command this is no longer necessary. Avail reads more, as the following example output demonstrates:

Type	Available	In-Use	Maximum	Largest
chip	77472	445760	523232	42712
fast	226200	290688	516888	219008
total	303672	736448	1040120	219008

Descriptions for the chip memory, the fast memory (only present in memory expansions) and for the entire memory region (Chip mem and Fast mem) are made in each column. The amount of memory not in use is displayed under "Available" and the size of the memory being used is shown under "In-Use". Both values add up to the value found under "Maximum".

A program can be loaded into small memory sections but these memory sections must have the smallest allowable size. When a program won't load anymore even though it must still be in memory, there is a good possibility that it's broken up in working memory.

BindDrivers

Syntax: BINDDRIVERS

This command makes it possible to add driver programs to the system. The driver programs are found in the system directory "SYS:Expansion". Most drivers are command programs for expansion hardware. Most of the drivers are written so that they can only be active when the corresponding hardware is present in the system.

Frequently, the expansion must also be connected with the Mount command.

Two typical driver programs are "hddisk" (stands for the Amiga ready hard disk) and "janus.library" (stands for a PC or AT card in the Amiga 2000).

Command Under Workbench 1.3

No changes have been made.

Break

Syntax: BREAK [TASK] Tasknumber [ALL]
BREAK [TASK] Tasknumber [C] [D] [E] [F]

The execution of a single command can be stopped by pressing <Ctrl><C>. To interrupt a script file, the key combination <Ctrl><D> is used. <Ctrl><E> and <Ctrl><F> can be used to interrupt certain tasks.

-
- <Ctrl><C>** The execution of the current command is stopped. The next command is started from the script file.
- <Ctrl><D>** The script file is interrupted after the conclusion of the current command.
- <Ctrl><E>** These are not normally used from AmigaDOS.
- <Ctrl><F>** They can be defined in individual tasks as further break criteria.

By using the **Break** command it is possible to break out of a task from the CLI. You must know the number of the task in question. Further information is found under the **Status** command.

The use of the **Break** command is equivalent to clicking in the window of the given task and entering either **<Ctrl><C>**, **<Ctrl><D>**, **<Ctrl><E>** or **<Ctrl><F>**.

The **All** argument means that four interrupt codes are sent to the task. The statement of one or more letters from the group **C, D, E** and **F** is equal to the interrupt codes that are created through the **<Ctrl>** combinations.

The task number with no further arguments is the same as transmitting the **<Ctrl><C>** code.

Command Under Workbench 1.3

No changes have been made.

Cd

Syntax: CD [DIR] Directory
CD

The **Cd** command changes the current directory and/or drive or displays the current directory. When no other directory is given with the command, AmigaDOS works with the current directory.

When no parameters are entered with the command, the name of this directory is displayed. Otherwise, if the name of a directory is entered after **Cd**, that directory becomes the current directory.

The given directory must be in the main directory or the path name given to the main directory (example: **Cd:Devs/Printers**).

With:

cd /

you reach every directory above the current directory. The **Cd** command causes the error message "Can't find /" if you are already in the main directory.

The command:

cd :

brings you directly to the main directory of the current disk, and:

cd

alone gives the name of the current directory.

Command Under Workbench 1.3

No changes have been made.

ChangeTaskPri

Syntax: CHANGETASKPRI [PRI] Priority

The **ChangeTaskPri** command can change the priority of the current CLI task.

The Amiga assigns every task a priority when managing its multitasking. The tasks with higher priority receive more computing time than the tasks with lower priorities.

The values for the task priorities can be between -128 and +127. CLI tasks are automatically given a priority of 0.

Because the tasks of the operating system also have priorities, you should set your priorities between the values -5 and +5. Using different values could interfere with the operating system and disrupt system execution.

Command Under Workbench 1.3

Syntax: changetaskpri PRI/A, PROCESS/K

The Process/k argument is a new feature. It's possible to change the priority of any process with it. The respective process number must be entered after the keyword Process. This number can be obtained with the help of the Status command.

Copy

Syntax: COPY [FROM] Sourcefile [TO] Targetfile [ALL]
[QUIET]

The Copy command or group of files make a copy of a file or the entire contents of a directory. These copies can be on the same disk or on another drive.

The All argument means that the subdirectories and their contents are also copied. The directory is automatically placed in the target directory.

Quiet suppresses the control listing of the individual files while other files are being copied.

Note: Although AmigaDOS does not distinguish between upper and lowercase letters, the target files should be written with either upper or lowercase letters. AmigaDOS takes exactly what you write for the new file. This is shown with each Dir or List.

When a file or directory is copied and a target name isn't given, AmigaDOS automatically gives the copy the same name as the original.

AmigaDOS recognizes whether the given name is a file or a directory.

Attention: When the name of a file that already exists is given as the target file, AmigaDOS overwrites the old file without any warning.

When a directory name is given as the source file, the name of an existing directory must be given as the target directory.

When a source directory isn't given and only a target directory is entered, `Copy` copies the contents of the main directory. The name behind `TO` is always interpreted as a directory name in this case.

The complete contents of a drive can be copied to another drive:

```
copy df0: to df1: all
copy df0: to jh0: all
```

Note: We recommend that you use the `DiskCopy` command for copying an entire disk. It does the same task only faster. However, copying with `Copy All` arranges the files on the disk so that they can be accessed faster later.

The wildcard character can be used in the name of the source file. You can find more information on wildcards in the section `Wildcard Characters` (see page 6).

Here are two alternate forms of the `Copy` command:

```
copy sys:s/Startup-sequence to prt:
copy sys:s/Startup-sequence to par:
```

or

`copy sys:s/Startup sequence to ser:`

prints the contents of a startup sequence to the connected printer.

The second alternate form copies keyboard input into a file or to a logical device. When an asterisk is given as the source file, AmigaDOS copies everything you type to the given target file. The key combination <Ctrl><\> must be pressed to end this function.

`copy * to ram:Test`

writes all keyboard input to the file "Test" on the RAM disk.

`copy * to prt:`

`copy * to par:`

or

`copy * to ser:`

writes the input line by line (after each <Return>) to the printer.

`copy * to con:1/1/639/50/`

writes all of the keyboard input line by line in the new window. Before this, the old CLI has to be activated because after the Con: window appears it is activated.

To combine several source files in a new target file, use the `Join` command.

Command Under Workbench 1.3

Syntax: `copy FROM, TO/A, ALL/S, QUIET/S, BUF=BUFFER/k, CLONE/S, DATE/S, NOPRO/S, COM/S`

When copying a drawer and the target drive doesn't contain the directory, the command does not stop any more, but creates a directory of the same name. The drawer is copied into this directory.

The new `Copy` command also allows the output of an entire directory to the printer. The output may be distorted if the directory contains only true ASCII data files.

Now we come to the added arguments that are allowed in the command line:

`Buffer` or `Buf` allow the size of the buffer memory for the copy operation to be set up to 512 bytes by the user.

The four arguments `Clone`, `Date`, `NoPro` and `Com` stand for the additional information that should be given to the copy. The additional information, that DOS prepares for all files and directories, are statements about the time period in which the file was accessed and other status information that falls under the task of the `Protect` command. Up to 80 characters of comments can be added to a file.

For the `Copy` command you must know that with `Clone` all statements, with `Date` the date, and with `Com` the comments are all given to the copy. The `NoPro` argument functions exactly the opposite: It provides the copy with the old status information.

Date

Syntax: `DATE [TIME] Time [DATE] Date`
 `DATE [TO|VER] Filename`

`Date` can indicate the system date or enable you to change it.

Note: AmigaDOS time and a battery-powered realtime clock may be different. To load realtime into the AmigaDOS system time, use the `SetClock` command.

Every time a disk is accessed, the current date is placed on disk. When you do not make a statement for the date and don't load a real time, AmigaDOS automatically uses the most recent date that it finds on the current disk.

When a time and/or date is entered after `Date`, it becomes the system time for AmigaDOS.

The time is entered in HH:MM:SS format. HH stands for hours between 0 and 24, MM stands for minutes between 0 and 59, and SS stands for seconds between 0 and 59. When one of these values is only one digit it must be accompanied by a 0. For example, 8 should be entered as 08. AmigaDOS sets the seconds at 00 if none are entered.

The date is entered in DD-MMM-YY form. DD stands for the day from 01 to 28, 29, 30 or 31 (depending on the month and year). Don't forget the necessary zero. MMM is an abbreviation of the month.

Jan (January)	Jul (July)
Feb (February)	Aug (August)
Mar (March)	Sep (September)
Apr (April)	Oct (October)
May (May)	Nov (November)
Jun (June)	Dec (December)

Numbers are not accepted for the month input.

The year is given as two numbers. Values between 78 and 99 are interpreted by AmigaDOS as years in our century (1978-1999). Numbers between 00-46 are in the 21st century (2000-2046). Values between 47 and 77 are invalid in AmigaDOS. The earliest possible date is 01-Jan-78.

After the date and time are entered they are checked for validity. Incorrect times or dates produce an error message.

When AmigaDOS already knows the day, you can enter a weekday. The date is set at the next current date of the given weekday. The weekday must be written out completely.

You can also use the words `tomorrow` and `yesterday` to set the date one day before or after the current date.

Note: For date input, AmigaDOS uses the hyphen (-) and the colon (:) for the time. It is important that you don't use any division signs (/).

The time and date can be set with the **Preferences** program. This information is then transferred to AmigaDOS.

The **Date** command without arguments displays the current setting.

Enter the **To** argument after **Date** and a file or device name and AmigaDOS sends the output of time and date to the given name. For example, the time and date can be given to the printer or written in a file.

Command under Workbench 1.3

The new **Date** command accepts a single digit as well as a double digit statement of the date.

Delete

Syntax: DELETE Name [Name2...Name10] [ALL] [QUIET|Q]

Files or directories can be erased with the **Delete** command. You have the option of entering up to 10 filenames, pathnames or directories. A message appears if the given file cannot be erased for some reason (it may be under use for another task, protected by the **Protect** command, or simply does not exist).

A directory must be empty before it can be erased. The **All** argument must be entered if you want to erase a directory that is not empty. All of the files and subdirectories of the given directory are erased in this case.

When more than a file is erased, **Delete** gives a list of all the files. To suppress the list, enter the **Quiet** argument. For your convenience **Q** can be used as an abbreviation for **Quiet**.

Wildcards can be used in the name of the file to be deleted. Delete erases all files that meet the search criteria.

Command Under Workbench 1.3

The syntax of this command has not changed.

Unlike the old version of this command, the new command doesn't stop when it doesn't find an entry. If the command:

```
delete df0:test1 df0:test2 df0:test3
```

is entered, the file test3 is erased even if a file named test2 is not found on the disk. The old command would have erased test1 and then given an error message.

D i r

Syntax: DIR [Directory] [OPT A|I|AI|D|DA]

Dir displays a directory in alphabetical order.

When a directory is entered, the contents of this directory are displayed. The directory must be found in the main directory or the main directory will receive an exact directory description.

The OPT A argument displays the contents and subdirectories of the directory.

OPT D displays the directories, not the files.

OPT I changes the main directory to interactive mode. A question mark appears after each directory entry in this mode. Here the following input is possible:

- <Return> Goes to next entry.
- ? Indicates the allowed options.
- B A directory surface like above. In the directory that was actually called, this option stops the interactive mode.

-
- E** Enters a directory, displays the directory contents.
DEL Erases the indicated entries. Can only be used on directories when they are empty.
T Displays the file contents on the screen. <Ctrl><C> stops this output.

Note: When the output of binary files is done in the alternative character set, entering <Ctrl><O> brings back the old character set.

- Q** Ends the interactive mode. <Ctrl><C> can also be entered instead of Q.

Combinations of the individual modes are also possible.

Note: More possibilities for indicating the directories exist with the `List` command.

Command Under Workbench 1.3

Syntax: `dir DIR,OPT/K,ALL/S,DIRS/S,INTER/S`

The new command has the possibility of replacing the options A through I with the keywords all through inter. Entering "dir inter" works the same as entering "dir opt i".

By using the Dirs argument, it is possible to display only the directory names. In the interactive mode (dir opt i or dir inter) there is a new command called Com. Using this command it is possible to start any CLI command directly or as a new process (with Run). The output of the function remains in the same position.

DiskChange

Syntax: `DISKCHANGE Drive`

This command informs AmigaDOS of a disk change in the given drive.

The 5.25" disk drives must use the `DiskChange` command to inform the Amiga of a disk swap.

Command Under Workbench 1.3

No changes have been made.

DiskCopy

Syntax: DISKCOPY [FROM] Sourcedrive TO Targetdrive
[NAME Diskname]

This command copies the contents of one disk to another. One or two drives can be used. When copying with only one drive, the source and target disks must be swapped several times. AmigaDOS informs you when to change them.

Note: It is assumed that the source and target drives have the same size, number of tracks and same number of sectors. You can also work with different data media (3.5" disks, 5.25" disks, hard disk partitions).

The target disk does not have to be formatted to copy to it. This takes place automatically during the copy process.

The source drive is either a device name (`df0:`, etc.) or the name of a disk.

The target drive is either a device name or a disk name (in case the target disk has been formatted under AmigaDOS already).

Attention: The old contents of the target disk are erased when copying. To avoid losing any data, the source disk should be write-protected.

The target disk can be given another name by using the `Name` argument. Otherwise the target disk will have the same name as the source disk.

The copy process can be interrupted using <Ctrl><C>. The information on the target disk is then lost.

Note: This command is found in the System: directory on the Workbench disk. When the search path is not in this path, AmigaDOS cannot find the DiskCopy command.

No disks can be copied from the Workbench surface if the file SYS:System/DiskCopy does not exist or cannot be found by AmigaDOS.

DiskCopy must have access to the Libs:Icon.Library file.

Use the Copy command to copy individual files.

Command Under Workbench 1.3

No changes have been made.

DiskDoctor

Syntax: DISKDOCTOR [DRIVE] Drive

DiskDoctor tries to restore a disk that has encountered a read error. It tries to fix the structure of the damaged disk.

By using the DiskDoctor it is possible to resurrect lost files as long as the new files have been written to the disk in the meantime.

Attention: Before working with the DiskDoctor you should make a copy of the damaged disk with the DiskCopy command. Other utilities may help later if the DiskDoctor does not work.

Before it starts to repair a disk, the DiskDoctor asks if the files that can no longer be repaired should be erased.

After the repairs the individual files should be copied to an error free disk using Copy.

Note: In case the DiskDoctor can no longer determine which directory a file belongs to, this file is assigned to the main directory. Then the problem of the same names arises. To solve this problem, the file must be erased with Delete after copying it to another disk. The second file of the same name remains and can be copied with another name or to another directory.

Command Under Workbench 1.3

The DiskDoctor can also be used on the RAM disk RAD.

The use of the DiskDoctor has not changed. However, the functioning of the program has been improved somewhat.

DJMount

Syntax: DJMOUNT

The DJMount command connects all of the partitions of a Janus hard disk into the system. This operation is usually found in the startup sequence.

After that the device names JH0:, JH1:, ... are ready for use and AmigaDOS has access to these partitions. On a PC hard disk, these are controlled over a PC or AT card or with the Sidecar.

Note: To install one or more Janus hard disk partitions you need the MS-DOS program "ADisk". The individual partitions must be formatted from the Amiga with DFormat.

The driver Expansion:HDDISK must be connected using BindDrivers to use the Janus hard disk.

D P F o r m a t

Syntax: DIFORMAT DRIVE Harddisk Drive NAME Diskname
[NOICONS]

This command functions exactly like **F o r m a t** except that it is used to format hard disk partitions. It isn't suitable for formatting floppy disks.

For the hard disk drive, give the device name of the hard disk (for example JH0:, JH1: or DH0:).

Behind **Name** is the name that the formatted partition should receive. The **NoIcons** argument suppresses the trash can icon on the formatted partition.

E c h o

Syntax: ECHO Text

The **Echo** command places output text on the screen. The main use for this command is in script files. Directing the output with the **>** character enables the output to go to other devices.

Text can be placed in quotation marks. The quotation marks at the beginning and end of the text are not printed. Quotation marks in the middle of the text produce an error message.

The ***** character produces special functions when placed in text:

- *e** is equivalent to **<Escape>** and can be used to execute escape sequences
- *n** forces a line feed
- **** prints the ***** character
- *"** prints the **"** character

The ***** functions are only possible when text appears in quotation marks.

Command under Workbench 1.3**Syntax:** echo , NOLINE/S

The NoLine argument suppresses the line feed that usually follows the output of the Echo command.

ED**Syntax:** ED [FROM] Filename [[SIZE] Memory]

ED is the screen oriented text editor from AmigaDOS. It works in a separate window that is independent of the CLI.

When the Filename argument already exists on disk, ED loads its contents. In case it does not exist, the file is created in ED.

Memory prepares the work section of ED. ED uses 40K if no code exists there. The size can be increased if large files need a large work area. Enter a value for "Memory" that is somewhat larger than the number of bytes in the file.

ED offers the following possibilities:

<Cursor keys>	Moves the cursor left, right, up or down
<Backspace>	Erases the character to the left of the cursor
<Ctrl><H>	Erases the character to the left of the cursor
	Erases the character under the cursor
<Return>	Inserts a line feed at the current cursor position
<Ctrl><M>	Inserts a line feed at the current cursor position
<Tab> or <Ctrl><I>	Moves the cursor to a tab position on the right. No characters are inserted in the text
<Ctrl><A>	Inserts a line
<Ctrl>	Erases a line
<Ctrl><D>	Scrolls down
<Ctrl><E>	Cursor alternates between first and last character on the screen
<Ctrl><F>	Changes the characters under the cursor from lower to uppercase or vice versa and moves the cursor one position to the right

<Ctrl><G>	Repeats the <Esc> sequence entered last
<Ctrl><O>	Erases all of the characters up to the next word
<Ctrl><R>	Jumps to the next word on the left
<Ctrl><T>	Jumps to the next word on the right
<Ctrl><U>	Scrolls up
<Ctrl><V>	New text construction in the editor window
<Ctrl><Y>	Erases everything from the cursor position to the end of the line
<Ctrl><[>	Same as <Esc>
<Ctrl><]>	Cursor alternates between the beginning of the line and the end of the line

The following Esc commands must be closed or confirmed with <Return>.

<Esc> A /Text/	Inserts text in the current line
<Esc> B	Jumps to the end of the file
<Esc> BE	Marks the end of the block
<Esc> BF /Text/	Searches for Text from the current cursor location to the beginning of the file
<Esc> BS	Marks the beginning of the block
<Esc> CE	Sends cursor to the end of the line
<Esc> CL	Moves cursor one position to the left
<Esc> CR	Moves cursor one position to the right
<Esc> D	Erases the current line
<Esc> DB	Erases the marked block
<Esc> DC	Erases the character under the cursor
<Esc> E /Search_ Text/Replace_ Text/	Replaces Search_ Text with Replace_ Text
<Esc> EQ /Search_ Text/Replace_ Text/	Replaces Search_ Text with Replace_ Text but first asks whether the user wants it replaced
<Esc> EX	Increases right margin
<Esc> F /Text/	Searches for Text in the direction of the end of the text
<Esc> I /Text/	Inserts Text before the active line
<Esc> IB	Inserts a copy of the marked block
<Esc> IF /Filename/	Inserts the given file

<Esc> J	Erases the line feed at the end of the current line
<Esc> LC	Differentiates between upper and lower case letters when searching for text
<Esc> M x	Positions text cursor in the first column of line x
<Esc> N	Positions text cursor in the first column of the next line
<Esc> P	Positions text cursor in the first column of the previous line
<Esc> Q	Quits ED without saving the last changes
<Esc> RP	Repeats the line until an error is encountered
<Esc> S	Inserts a line feed at the current cursor position
<Esc> SA/Filename/	Saves the text in the file last used or in the given file
<Esc> SB	Highlights the marked block
<Esc> SH	Indicates current window status
<Esc> SL x	Sets left margin at x characters
<Esc> SR x	Sets right margin at x characters
<Esc> ST x	Sets tab position at x characters
<Esc> T	Jumps to top of file
<Esc> U	Undoes the changes made to the current line
<Esc> UC	Terminates the differentiation between lower and uppercase letters when searching for text
<Esc> WB /Filename/	Saves marked block in the given file
<Esc> X	Saves text in the last file used and exits ED

Edit

Syntax: EDIT [FROM] Filename1 [[TO] Filename2]
[[WITH] Filename3] [[VER] Filename4] [OPT P
Lines [W Characters]]

There is another editor besides the screen oriented ED: Edit, the line editor. Edit does not work in its own window but rather in the CLI window from which it was called.

Edit reads the lines to work with from the file Filename1. This file must already exist. The edited version of this file is saved to file Filename2.

Note: In all cases the Filename1 file remains unchanged. When a To file isn't given, Edit creates a temporary work file. When it's done working it saves the old file as ":T/edit-backup" and the temporary file receives the name Filename1. It automatically uses the name ":T/edit-backup" because the T: directory is always on disk.

When a With Filename3 file is given, the editing can be completely automatic. This file can contain any editor commands that can work through the lines.

Edit sends insertions and information to the Ver file Filename4.

The OPT P argument (lines) sets the number of lines that Edit can have resident in memory. This number has nothing to do with the number of lines the file may hold because Edit can load them from disk. The standard for this value is 40.

OPT W (characters) is the number of characters per line. The standard value here is 120. The following options are available when using Edit:

Ending

- STOP** Edit is stopped, the file remains unchanged.
Q Leaves Edit and saves the file.
W Jumps to the end of the file, saves and exits Edit.

File commands

- C /Filename/** Makes the given file the With file: processes Edit commands from the Filename file
CF /Filename/ The given file is closed
FROM Original source file is chosen
FROM /Filename/ Chooses the given file as current source file
SHD Indicates the current position from Edit
SHG Indicates the current global command (GA, GB, GE)
TO Original To file is chosen again
TO /Filename/ Selects the given file as the To file

Positioning, display

- ?** Displays the current line
! Displays the current line with all of the hidden characters
< Text pointer one character left
> Text pointer one character right
Characters at the current pointer position are erased
\$ Characters at the current pointer position are changed to lowercase letters and the pointer is moved over one character
% Characters at the current pointer position are changed to uppercase letters and the pointer is moved over one character
- Characters at the current pointer position are changed to spaces and the pointer is moved over one character
=n The current line is given the line number n
I Inserts the new text before the current line

Note: Input from the keyboard can be halted with <Ctrl><C><Return>.

In Inserts the new text before line n
I[n]/Filename/ Inserts the contents of the given file before the current line or line number n

Note: When * is entered for the line number, the last line that was worked on is used.

Mn Positions the line pointer at line n
M+ Positions the line pointer at the first line in the buffer
M- Positions the line pointer at the last line in the buffer
N Positions the line pointer at the next line in the buffer
P Positions the line pointer at the previous line in the buffer
PB /Text/ Positions the text pointer one character before Text in the current line
PA /Text/ Positions the text pointer one character after Text in the current line
PR Positions the text pointer at the beginning of the line
R [Line1 [Line2]] Replaces the current line or Line1 with Line2 through input from the keyboard
R [Line1 [Line2]] Replaces the current line or /Filename/ Line1 with Line2 using the contents of the given file
REWIND Positions the line pointer at the first line of the file you're working with
T Displays all lines until the end of the file
Tn Displays n lines
TLn Displays n lines with the previously used line numbers
TN Displays the contents of the text buffer
TP Positions to the beginning of the text buffer and displays the complete contents of the buffer
TR+ The following spaces are important
TR- The following spaces are ignored

V+	Line display on
V-	Line display off
<u>Text processing</u>	
,	Repeats the last search/replace command (A, AP, B, BP, E, EP)
A /Text1/Text2/	Inserts Text2 behind Text1
AP /Text1/Text2/	Inserts Text2 behind Text1 and positions the text pointer behind Text2
B /Text1/Text2/	Inserts Text2 before Text1
BF /Searchtext/	Searches backwards from the current line for Searchtext
BP /Text1/Text2/	Inserts Text2 before Text1 and positions the text pointer behind Text2
CL	Combines the current line with the next line
CL /Text/	Combines the current line with the next line and inserts Text between them
CG	Turns off all global commands (GA, GB, GE)
CGn	Turns off global command number n (GA, GB, GE)
DF /Searchtext/	Searches forward from the current to the end of the file for Searchtext and erases all lines in between
DFA /Text/	Erases the rest of the lines after Text
DFB /Text/	Erase the rest of the lines beginning with the first character of Text
DG	Temporarily turns off all global variables (GA, GB, GE)
DGn	Temporarily turns off global variable (GA, GB, GE) number n. The number appears in the SHG list
DTA /Text/	Erases from the beginning of the line to the last character of Text
DTB /Text/	Erases from the beginning of the line to the first character of Text

E /Searchtext/Replacetext/	Replaces Searchtext with Replacetext in the current line
EG	Reactivates the temporarily disabled global commands (GA, GB, GE)
EGn	Reactivates the temporarily disabled global command (GA, GB, GE) number n. The number appears in the SHG list
EP /Searchtext/Replacetext/	Replaces Searchtext with Replacetext in the current line and positions the text pointer behind Replacetext
F /Searchtext/Replacetext/	Searches from the current line to the end of the file for Searchtext and positions the text pointer behind Replacetext
GA /Text1/Text2/	Searches through all of the lines in the direction of the beginning of the file and inserts Text2 behind Text1 in every line where it occurs
GB /Text1/Text2/	Searches through all of the lines in the direction of the end of the file and inserts Text2 behind Text1 in every line where it occurs
GE /Searchtext/Replacetext/	Searches through all of the lines in the direction of the end of the file and replaces Searchtext with Replacetext in every line where it occurs
Hn	The From file is only read up to the given line. When * is given for n, the whole file can be processed again
SA /Text/	Splits the current line after Text
SB /Text/	Splits the current line before Text
Z Characters	The given characters are used as the new end of input characters (standard: /Z)

Else

Refer to IF ... ELSE ... ENDIF.

EndCLI

Syntax: ENDCLI

EndCLI closes the CLI in which this command is entered. The CLI window disappears from the screen.

If the closed CLI was started from another CLI, the first CLI is activated again. The Workbench is again activated if this was the last CLI and it was started from the Workbench interface.

Attention: Be careful when closing CLI windows that were automatically started when booting the system. There is no Workbench surface in the background when a LoadWB isn't entered (or processed in the startup sequence). The Amiga can no longer be of use. The only possible option is reset.

Note: When one or more tasks that were started from the closed window are active, the window doesn't disappear from the screen. AmigaDOS isn't informed of any further input. This also applies to tasks that were started with Run in the present version of AmigaDOS.

Command Under Workbench 1.3

Some Shell systems have an Endshell command, while some don't.

EndIf

Refer to IF ... ELSE ... ENDIF.

Execute

Syntax: EXECUTE Script-file [Argument1] [Argument2]
[...]

The **Execute** command starts script files. Script files must be stored in the form of text files and can be entered with the editor ED. **Execute** can be given with a string of arguments when calling a script file. A call looks something like this:

```
execute ram:Testfile Hannes Christian Rainer
```

The arguments must be processed from the called script file. The following commands can be used:

.KEY or .K After this expression enter the name of variables in succession, that are evaluated after **Execute**. The variable names are designated by two characters inside of the script file. Normally the characters **<** and **>** are used. In the script file it could also read:

```
.KEY <Name1> <Name2> <Name3>
```

.DOT Character The point in the dot commands can be replaced by some another character

.BRA Character The **<** character, which is used to designate the variables in the script file, can be replaced by another character

.KET Character The **>** character, which is used to designate the variables in the script file, can be replaced by another character

.DOL or .DOLLAR A substitute occupation can be given to each variable when no corresponding variable can be transferred with **Execute**. The **\$** character designates the substitution. It can be replaced by any other character. An example for replacing the dollar sign would be:

```
<Name1$Silvia>
```

- .DEF** Variable contents The variable <Variable> is assigned the value <Contents>. This assignment is used in all circumstances
- .<Spaces>** Comment Comment line in the script file

Command under Workbench 1.3

Script files are called by using **Execute**. By using the **S** status flag (Script flag), it is also possible to start a script file by just entering the filename. The Script flag must be set beforehand by using the **Protect** command.

FailAt

Syntax: FAILAT [Errorlimit]

FailAt establishes the error limit at which a script file is stopped. The standard setting for this error limit is 10. When **FailAt** is entered without parameters, the current error limit is given on the screen.

When an AmigaDOS command encounters an error, the command sends an error code back to AmigaDOS. Every code greater than 0 indicates an error. The error codes are organized in such a way that few bad errors have low codes and serious errors have higher codes.

Note: The error codes have nothing to do with the error numbers that are found in the "Error Messages" chapter or with the **Fault** command.

<u>Error code</u>	<u>Degree of the Error</u>
5	Warning, advice
10	Normal error, the section can be repaired
20	Difficult error, cannot be repaired most of the time

Usually script files are stopped with error codes higher than 10. The limit can be raised (to 20 for example) or lowered (to 5).

Note: After the completion of a script file, AmigaDOS automatically sets the error limit back to 10.

Command Under Workbench 1.3

No changes have been made.

Fault

<i>Syntax:</i> FAULT Errornumber [, Errornumber, ...]

Every error that can be encountered in AmigaDOS has its own number. The error message for any of these numbers can be displayed on the screen by using the **Fault** command. Up to 10 numbers can be entered after the **Fault** command.

Note: The **Fault** command is used from the operating system and AmigaDOS. The **Fault** command makes it possible to display the error message in the top line of the Workbench surface or by using the **Why** command. When AmigaDOS cannot find "C:Fault", the error can only be given in the form of error numbers.

Command Under Workbench 1.3

No changes have been made.

FF	(Workbench 1.3 only)
-----------	----------------------

FF accesses a program named *FastFonts*. It was invented by the company Microsmiths and has been on the market for some time. *FastFonts* accelerates the output of text on the Amiga. The output increases in speed by a maximum of 20%.

FastFonts is activated by entering **ff -0**. The message:

FastFonts V1.1 Copyright - 1987 by C.Heath of Microsmiths, Inc
Turning on FastText

appears. The command is usually found in the startup sequence of a boot disk. Here the message is suppressed by sending it to the Nil device with: `ff > nil: -0`.

The `-n` command can be entered if the normal output mode is needed for some reason. The starting message also appears. The message "Turning on FastText" does not appear.

FileNote

Syntax: FILENOTE [FILE] Filename [COMMENT] Remark

FileNote can be used to give the given file a remark text. This remark is shown in the main directory of **List**.

The remark can be up to 80 characters long. It must be in quotation marks if it contains any spaces. To erase a remark add the text "" to the command.

Note: The remarks are not copied with **Copy**. They remain in the main directory when the file is overwritten with completely new contents. Renaming doesn't change the comments.

Command Under Workbench 1.3

No changes have been made.

Format

Syntax: FORMAT DRIVE Drive NAME Diskname [NOICONS]

Format is used to format a new disk in AmigaDOS. The label for the drive of the disk to format is entered after **Format Drive**. A name for the newly formatted disk must be entered after **Name**. This name can be up to 30 characters long.

The `NoIcons` argument prevents the new disk from having a trash can directory and trashcan icon. The trashcan icon is copied from “SYS:Trashcan.info” and the trashcan directory is created if this option is not given.

Attention: All information on the disk in question is lost when it is formatted.

Note: The `Format` command is found in the `System:` directory. It can only be used from the CLI when it runs with the indicated search path. When AmigaDOS can not find “SYS:System/Format”, formatting a disk from the Workbench is not possible.

Use the `DPFormat` command to format hard disk partitions.

Command under Workbench 1.3

Syntax: `format DRIVE <disk> NAME <name> [NOICONS]
[QUICK] [FFS] [NOFFS]`

The three arguments `Quick`, `FFS` and `NOFFS` are new. Adding `Quick` speeds up the operation so that it takes only a few seconds because only the tracks that contain the root block and the boot blocks are formatted. This argument works only on disks that are already formatted and should be erased completely. A complete format of a disk without the `Quick` argument takes about two minutes.

The `FFS` and `NOFFS` arguments are connected. They create the desired file system for the single partitions when formatting a hard disk. Adding `FFS` puts the new and faster `FastFileSystem` into use. The slower `FileSystem` is used if `NOFFS` is entered.

Getenv

Syntax: getenv NAME/A

This command makes it possible to use environment variables. The environment handler is still missing. The handler can be simulated by the RAM disk, but full use is not yet realized.

IconX

(Workbench 1.3 only)

This command makes it possible to call a script file from the Workbench by double clicking on it. The following must be done:

- Create a Project icon with the help of the icon editor on the Extras disk. A Project icon is received when the CLI icon is loaded from the Workbench, modified, and then saved under the name of the script file. The Frame and Save options of the icon editor make it possible to save smaller icons by enclosing the desired picture with the mouse.
- Then open the disk drawer with the new icon, click on the icon, and choose the Info item from the Workbench menu. The C:XI**co**n command must be entered in the Default Tool field. Save the In**fo** window, and then the script file can be called by double clicking on it. Descriptions about the window size for the output of the script files can be made in the Tool Type filed in the Info screen. For example:

```
TOOL TYPE WINDOW=CON:0/0/400/100/Batch_window
```

The window can be made to stay open after processing the script file by entering the following:

```
TOOL TYPE DELAY=1000
```

The delay time must be given in 1/50 seconds.

If...Else...EndIF

Syntax: IF [NOT] [WARN|ERROR] | [Text1 EQ
Text2] | [Exists Filename]
ELSE
ENDIF

The structures that are possible with the commands **If**, **Else** and **EndIF** can be programmed into script files. The direct input of this command sequence is useless.

AmigaDOS checks the condition after the **If** command. When the condition is met, AmigaDOS executes all of the commands between the **If** line and the next **Else** or **EndIF** line. When an **Else** is encountered and the condition is met, AmigaDOS jumps to the next **EndIF**.

AmigaDOS jumps to the next **Else** or **EndIF** if the condition is not met. The following possibilities exist for the conditions:

NOT	Returns the logical value of the condition
WARN	Is satisfied when the error code of the last processed command is greater than or equal to 5
ERROR	Is satisfied when the error code of the last processed command is greater than or equal to 10. The error limit must be raised over 10 with FailAt
FAIL	Is satisfied when the error code of the last processed command is greater than or equal to 20. The error limit must be raised over 20 with FailAt
EQ	Compares two texts with each other. Input can be examined this way. Lower and uppercase letters are ignored in this comparison. The presence and absence of input can be detected with:

```
IF Text EQ ""
```

EXISTS	Checks if the given file exists. The condition is met if it does
---------------	--

Simple Yes/No input can be checked with the Ask command. An interlocking of If/Else/EndIF is possible:

```
IF ((Condition))
  ((DOS-Commands))
ELSE IF ((Condition))
  ((DOS-Commands))
ELSE
  ((DOS-Commands))
ENDIF
ENDIF
```

The jump commands can also be used in script files. They are described later.

Command Under Workbench 1.3

No changes have been made.

Info

Syntax: INFO

The Info command gives information about all of the connected drives. After entering Info the following may appear for example:

Mounted disks:

Unit	Size	Used	Free	Full	Errs	Status	Name
JH0:	20M	39805	1877	95%	0	Read/Write	Harddisk
DF2:	No disk present						
DF1:	880K	1714	44	97%	0	Read/Write	BASICDisk
DF0:	880K	1664	94	94%	0	Rad/Write	A2000 WB 1.2D
RAM:	1K	14	0	100%	0	Read/Write	

Volumes Available:

A2000 WB 1.2D [Mounted]
 RAM disk [Mounted]
 Harddisk [Mounted]
 BASICDisk [Mounted]

The individual columns in the Mounted Disks: list have the following meaning:

Unit	The device name of the respective drive
Size	Size of the memory medium in Kbytes or Mbytes
Used	Number of blocks used (each 512 bytes). Each 3.5" disk in AmigaDOS has 1758 blocks (880K) for use
Free	Number of free blocks
Full	Percentage of the memory medium used
Errs	Number of disk errors
Status	Read/Write = Read and Write possible Read Only = Write protect is active
Name	Name of the data medium (disk, hard disk, etc)

Note: Due to the memory management of the RAM disk, RAM: is always 100% used. One block is needed for the management information.

The disk names that AmigaDOS knows at the time are listed under the **Volumes available:** list. Disks that are in the drive are designated with [Mounted]. These statements can also be seen under the **Assign** command.

Command Under Workbench 1.3

Syntax: info DEVICE

The new **Info** command can include the device description. You can receive only information about the specified device. A re-working of the table output is worthwhile when dealing with longer device names. A new tab function in the **Info** command makes this new organization possible.

InitPrinter

Syntax: INITPRINTER

InitPrinter is not an actual AmigaDOS command. It can be entered in the CLI like an AmigaDOS command.

Its job is to send a initialize command to the printer selected in **Preferences**. **InitPrinter** knows which command characters to use for initialization from the selected printer driver.

When you switch your printer on and off between printing operations, you should initialize it using `InitPrinter`. It can cause problems if you don't do this because the printer can be in another operating mode after turning it on.

Install

Syntax: `INSTALL [DRIVE] Drive`

The disk in the drive is made bootable by using `Install`. This means that you can start your Amiga with this disk. The disk is also accepted as the Workbench disk and is handled as `SYS:` from AmigaDOS.

You should prepare a startup sequence in the `S:` directory of this disk. The AmigaDOS commands that are found in the `C:` directory must be present on the disk.

What goes on the rest of the disk depends on which programs work from this disk. Most programs require the directories `Devs:`, `Libs:` and `L:`.

`Fonts:`, `System:` and `Expansion:` must also be present.

Command Under Workbench 1.3

Syntax: `install DRIVE/A,NOBOOT/S,CHECK/S`

The `NoBoot` and `Check` arguments are new here. The `NoBoot` argument removes disk installation. The `Check` argument can be used to see if a disk is bootable or if the boot block has been damaged. The `Check` argument returns the following message for non-bootable disks:

No bootblock installed

When it handles a boot disk that has no damage in the boot block, the message reads:

Appears to be normal V1.2/V1.3 bootblock

The Install command gives the following message:

May not be standard V1.2/V1.3 bootblock
if something is not normal.

Join

Syntax: JOIN Filename1 [Filename2 ... Filename15] AS
Newfilename

The contents of multiple files can be connected in one target file by using Join. Up to 15 source files can be given.

The source files are copied one after another into the new file. The building of the new file is halted if one of the source files is not found. The file then contains all of the files that were copied up to that point.

The Join command is only useful for ASCII text files.

Note: The original files remain unchanged. The target file Newfilename cannot be given the name of any other files.

Another use for the Join command is to display file contents on the screen or on a printer:

```
join test as *
join test1 test2 test3 as prt:
```

Command Under Workbench 1.3

Syntax: joinAS=TO/K

The Join command now understands TO as well as AS.

Lab

Syntax: LAB [Label]

Individual program sections can be labeled within a script file. A label is designated by a Lab command.

Any word can be a label. Spaces are not allowed.

When just a Lab command is given and no label, AmigaDOS interprets this as a mark for the last Skip command.

Command Under Workbench 1.3

No changes have been made.

List

Syntax: LIST [[DIR] Filename|Directory] [P|PAT]
 Wildcard-Pattern [KEYS] [DATES|NODATES] [TO
 Filename] [S Searchtext] [SINCE Date] [UPTO
 Date] [QUICK]

List works exactly like Dir for displaying the main directory of a disk. List doesn't give an alphabetical list. The filenames are given as they occur on the disk.

List gives more information than just directories. List without parameters displays the following information about a file:

```
Expansion   Dir      rwd     05-Jul-87   12:16:24
:Here we have defined a remark
CLI         2356    rwd     05-Jul-87   09:20:52
```

From left to right:

First the name of an entry appears. Dir appears in the next column if the entry is a directory. The size in bytes would be given here if it were a file. "Empty" appears for empty files.

The third column contains the protection status of the file.

r	read	e	execute
w	write	d	delete

When the flags are set, the respective action is allowed. Missing flags mean that operation is forbidden. More information can be found under the `Protect` command.

At the end is the date and time when the file was created or last changed. When remarks are defined for a file, these appear in the second line under the entry.

The following arguments can be given with the `List` command:

- | | |
|---------------------------------------|---|
| <code>[Dir] Filename Directory</code> | Gives information about the filename. When the name stands for a directory, <code>Dir</code> gives the contents of this directory. The name of a disk, a drive, or a logical device can also be given here. When a name isn't given, <code>List</code> shows the contents of the current directory. |
| <code>PPAT Wildcard-Pattern</code> | Only the entry that fits the given wildcard pattern is shown. More information about the wildcard characters can be found in the <code>General Information</code> section. |
| <code>KEYS</code> | The address of the start block of an entry is displayed. This is important when working with a disk monitor. |
| <code>DATES</code> | Normally the date of the current week is designated with <code>Yesterday</code> , <code>Monday</code> etc. The <code>Dates</code> option provides the date in <code>DD-MMM-YY</code> format. |
| <code>NODATES</code> | Suppresses the display of the date and time. |
| <code>TO Filename</code> | Directs the output of the directory to the given file. |

S Searchtext	An extension of the P PAT option. Only the names of the files that contain the given Searchtext are displayed. The Searchtext must be in quotation marks if it contains spaces or special characters.
SINCE Date	Shows only the files that were created or changed since a given date. The date is either in DD- MMM-YY format or an input such as Yesterday, Today, Tomorrow, Monday, Tuesday, etc. More information about date format is given under the Date command.
UPTO Date	Shows only the files that were created or changed before a given date. The valid formats for the date are given under the Since argument.
QUICK	Only the entry name is shown. All other infor- mation is suppressed.

Command Under Workbench 1.3

Syntax: list DIR, P=PAT/K, KEYS/S, DATES/S, NODATES/S,
TO/K, S/K, SINCE/K, UPTO/K, QUICK/S, BLOCK/S, NOHE
AD/S, FILES/SDIRS/S, LFORMAT/K

There are some very useful options:

When the List command is called with the addition of Block, the size of the files is not displayed in bytes but in disk blocks.

The NoHead argument suppresses the output of the directory names and the current date. This statement always appears when the List command is entered with a directory name (for example list df0:). In addition, NoHead prevents the output of the closing message (xx files - yy directories - zz blocks used).

With the addition of Dirs, you only need to know the names of the subdirectories to glance through the List command. It is even possible to view only the names of the data files. The argument for this is called Files.

It is possible to format the output of the `List` command using the `LFormat` argument so that it can be used directly as text for a script file. The output of the filenames is followed by the `Quick` and `NoHead` argument. The format for the output is given after the `LFormat` argument.

```
list df0: LFORMAT="..."
```

The statement that makes up the `List` line must be given inside quotation marks. Any text can be here. The character combination `%s` has a special meaning:

When the combination `%s` arises in the quotation marks, the filename is inserted at that point.

The character combination `%s` is allowed to appear more than once. The filename appears in both places if two `%s` are used. When three are used, the last show the filename while the first displays the path of the current directory.

Four `%s` alternate between path description and filename.

The `List` command has still more uses.

The wildcard characters are much more flexible. It is possible to use the wildcards with the path description.

The output of the status information has gone through another change. In addition to the already existing four flags `rwed`, Workbench 1.3 adds four more flags. They are identified using the letters `H` (Hidden), `S` (Script), `P` (Pure) and `A` (Archive). The flags are spoken about in detail under the `Protect` command.

LoadWB

Syntax: `LOADWB`

The `LoadWB` command activates the graphic user interface of the Workbench. The Workbench interface is reset and started if it

is already active. The Workbench screen is rebuilt and the Amiga shows an icon for each AmigaDOS compatible disk.

Note: LoadWB must access the library "Libs:icon.library". Otherwise the system is disrupted.

Lock

(Workbench 1.3 only)

Syntax: DRIVE/A,ON/S,OFF/S,PASSKEY

This new command functions in conjunction with a hard disk whose partitions run under the new FastFilingSystem (FFS) of Workbench 1.3. Lock has the capability of write protecting any partitions. Such a partition behaves exactly like a disk on which the write protect clips are in the write protect position.

The Lock command has the additional feature of being able to secure the write protect condition with a password. Then removing the write protection is only possible when you know the password:

```
lock dh1: on beethoven
```

Every write attempt to partition dj1: is greeted with the message "Volume xxx is write protected" (xxx stands for the name of the partition dh1:).

Write access can be restored by entering the command lock dh1: off beethoven.

MakeDir

Syntax: MAKEDIR Directory

MakeDir creates a new directory. The directory is placed in the current directory.

No entries with the same name as the directory are allowed to be placed in the directory once it is created.

Note: MakeDir makes only one new directory every time it is called. Directories that you describe in an eventual pathname must already exist.

Directories can be erased with Delete.

Command Under Workbench 1.3

No changes have been made.

Mount

Syntax: MOUNT [DEVICE] Devicename

Mount connects a new device in AmigaDOS. This is only for devices that are commanded from their own driver programs or that are not recognized in the Auto-configuration when booting.

Mount must have more information about the device to put in the "Devs:Mountlist". An entry in this file might look like this:

```
DF3: Device = trackdisk.device
      Unit = 4
      Flags = 1
      Surfaces = 2
      BlocksPerTrack = 11
      Reserved = 2
      PreAlloc = 11
      Interleave = 0
      LowCyl = 0 ; HighCyl = 79
      Buffers = 5
      BufMemType = 3
      #
```

A 5.25" drive would be connected as DF3: if this entry existed and Mount DF3: was entered.

Note: The DJMount command is used to connect the Amiga partition of a Janus hard disk.

Also read the description for the BindDrivers command.

Command Under Workbench 1.3

Syntax: DEVICE/A, FROM/K

The MountList can receive any name that follows the keyword FROM:

```
mount df2: FROM devs:devicelist_1
```

The Mount command searches in the Devs: directory for the file devicelist_1.

NewCLI

Syntax: NEWCLI [CON:x-start/y-start/width/height/
[Title]] [FROM script file]

NewCLI creates a new CLI and brings that window on the screen. The new CLI is automatically activated. The current main directory and prompt characters are taken from the old CLI. The old CLI remains available for input. NewCLI must not be called with Run.

AmigaDOS creates the new CLI window with a standard size and places it at a standard position on the screen. The message:

```
NewCLI task n
```

appears in the first line of the new window. n stands for the number of the CLI task. AmigaDOS can manage up to 20 tasks at one time.

Note: The execution speed of each task depends on the number of tasks. To give individual tasks more computing time, use the ChangeTaskPri command.

You can enter the dimensions of the **CLI** window after **New-CLI** by defining a **Con:** window. The single values are as follows:

- x-start** x coordinate of the upper left hand window corner (from 0 to 639 on the normal Workbench screen).
- y-start** y coordinate of the upper left hand window corner (from 0 to 255 on the normal Workbench screen).
- width** width of the window in pixels (from 0 to 640 on the normal Workbench screen).
- height** height of the window in pixels (from 0 to 256 on the normal Workbench screen).
- Title** A title can be entered for the **Con:** window. The entire **Con:** expression must be entered in quotation marks if the title contains spaces. For example:

```
NEWCLI "CON:0/0/630/100/Our new CLI window"
```

Note: For tasks that need a **CLI** window but should not disturb the screen, a 1x1 pixel mini-window can be created:

```
NewCLI CON:1/199/1/1/
```

This window can be made larger. The following command enlarges the window:

```
NewCLI CON:1/199/56/21/
```

A script file can be given after the **From** argument. This script file can be automatically executed after calling the new **CLI**.

Command Under Workbench 1.3

In the **C:** directory on the new Workbench disk there is a new command called **NewShell**. This command creates a window port to DOS that has some advantages over the **CLI**.

Many of these additions can only be used when the **Shell** segment is resident in the work memory of the Amiga before calling the **NewShell** command. The command reads:

```
resident CLI 1:Shell-Seg SYSTEM pure
```

This command is automatically executed when the computer is first turned on so that you don't have to bother with it. The **Shell** window has the following qualities:

Resident commands are supported

DOS can load most CLI commands into working memory.

Synonyms can be used for all commands

Shell commands can be renamed or abbreviated using the **Alias** command:

```
alias desiredname originalname
```

Desiredname stands for any character string (without spaces) that can be used to call that command. The **originalname** is the name of the command that should be executed by using the new name. When the **Shell** finds a name at the beginning of a line for which such a relationship exists, this name is replaced by the related command.

Output of the current directory paths

In the new **Shell**, the prompt stands for the current directory path. This informs at which branch of the directory tree you stand. The last part of the prompt can be eliminated by entering **CD**. The making of your own prompt is handled under the **Prompt** command.

Direct calling possibilities of a script file

You can call script files from the **Shell** without using an **Execute** command, if the script flag is set in the script file (see **Protect**).

NewShell

(Workbench 1.3)

Syntax: Window, From

NewShell makes it possible to open another window for entering DOS commands. The NewCLI command did this in older Workbench versions. A Shell has the following advantages over the CLI:

The input line can be edited with cursor keys.

The Shell uses the NewCon: device for input and output. This new window interface is responsible for many of the new Shell features. NewCon: allows the use of the cursor keys to edit entered text. The cursor can be placed anywhere on the input line by using the left and right arrow keys. The <Delete> (Del) and <Backspace> (←) keys function as usual. Additional text is entered from the current cursor position. When <Return> is pressed, the Shell accepts the entered line.

The following additional key combinations are accepted:

<Shift><Cursor left> (or <Ctrl><A>)

Places the cursor at the beginning of the line.

<Shift><Cursor right> (or <Ctrl><Z>)

Places the cursor at the end of the line.

<Ctrl><K>

Erases the text from the cursor to the end of the line.

<Ctrl><U>

Erases the text to the left of the cursor.

<Ctrl><W>

Places the cursor at the next tab position.

<Ctrl><X>

Erases the entire line.

Commands already entered can be recalled with the cursor keys

This new feature is also due to the NewCon: device. Every command entered is stored in a 2K buffer. The up arrow redisplay the last command.

When you are looking for a special command that was entered a short time ago, the NewCon: device can help: Enter the first letter of the command and press the <Shift> and <↑> keys at the same time. The command that starts with that letter is searched for.

Pressing the <Shift> and <↓> keys brings you to the end of the buffer.

Control codes are handled neutrally

When a control code is entered in the Shell (for example <Ctrl><L>), it is not shown. Instead, the current cursor position is only a space. The control code is still present and operates normally.

A Startup script file is executed

Every time the NewShell command is called, a script file with the name Shell-Startup is automatically called. This file is found in the S: directory of the Workbench disk. Here the appearance of the Shell prompt can be stored. The Shell functions are only useful when the Shell segment is integrated into the operating system before the Shell is called. The command that is needed here reads resident CLI 1:Shell-Seg System. These commands are usually found in the startup sequence of the Workbench disk so that entering them manually is not necessary.

Resident commands can be called

Programs can only be called from a Shell with the help of the Resident command which is present in the working memory of the Amiga. These commands must be ready for the user and cannot be loaded from the disk drive.

Program names can be shortened

Shell commands can be renamed or abbreviated using the Alias command:

```
alias ex execute
```

After entering this line the Execute command can be called under the name ex. The Alias command assigns the first character string (ex) the same text as the rest of the line. In this case the rest of the line only consists of the word execute. Entire command sequences can be abbreviated into single words:

```
alias st up ed s:startup sequence
```

Now just entering the command St to load the startup sequence into ED.

When you don't want to enter the Alias command after each new Shell is opened, it can be placed in the script file Shell-startup. As already said, this script file is automatically executed with each NewShell.

The Alias command without the additional arguments lists the existing name assignments.

NoFastMem

Syntax: NOFASTMEM

NoFastMem is not an actual AmigaDOS command. NoFastMem can be entered as a DOS command in the CLI.

Many programs that were developed on the Amiga 1000, mainly games or old software, do not run on Amigas with a memory configuration of 512K. In many cases they can be run when the `FastMem` is turned off.

This is done with `NoFastMem`. The program occupies memory areas not used by `FastMem`. Another start from `NoFastMem` frees up the memory areas again.

Note: The task that is created using `NoFastMem` ends when the memory is made free again. You should always start `NoFastMem` with:

```
run NoFastMem
```

to keep the current CLI free. In this case `NoFastMem` can also be stopped with:

```
break ((Tasknumber)) all
```

`((Tasknumber))` stands for the number of the CLI task that `NoFastMem` is processing. This is given after calling it with `Run` or can be determined with `Status`. No more CLIs must be blocked for the second call.

Path

```
Syntax:  PATH [ADD] Directory1 [Directory2 ...  
           Directory10] | [SHOW] | [RESET]
```

AmigaDOS works by searching for a given CLI command with a search path. A command is looked for first in the current directory. AmigaDOS searches in directory C: next.

The search path can be enlarged, announced or changed. Commands from other directories can be given if the path is expanded.

Ten directories, logical devices or disk names can be given after **Path**. The number of elements of the search path can be up to ten, then only multiple **Path** commands can be entered after each other.

Note: Very large search paths use up a lot of time waiting for AmigaDOS to search through all of the directories.

Path Reset erases all of the sections of the search path and reduces the elements to the current directory and directory **C:**. When directory names follow **Path Reset**, these directories are inserted in the new search path.

Path with one or more directory names inserts the given elements into the search path.

Path without arguments or **Path Show** displays the current composition of the search path.

Note: There is the following difference between **Path** without parameters and **Path Show**: **Path** alone checks the reading of the search path for the individual entries. When the search path runs over a disk that is not in the drive, the Amiga responds with the corresponding requester. **Path Show** does not make this check.

The current Workbench disk automatically places the following search path in the startup sequence:

```
Current directory
SYS:System
SYS:Utilities
SYS:PC (only for A2000)
RAM:
C:
```

Command Under Workbench 1.3

The function of the **Path** command has not changed, although the order of the search operation has. When no special path has been given to a command, DOS first searches in the resident

commands. Only when the command is not found in the resident commands does the search operation continue as the `Path` command describes.

Prompt

Syntax: `PROMPT [Prompt-Text]`

`Prompt` can change the prompt text in the current CLI. The standard prompt in most versions of AmigaDOS is `1>` or `n>`, where `n` is the number of the CLI task.

You can make any text be the prompt text. Text with spaces must be entered in quotation marks. The maximum length for the text is 59 characters.

The character combination `%n` stands for the number of the actual CLI task. After entering:

```
prompt "Task %n >"
```

AmigaDOS answers with:

```
Task 1 >
```

`Prompt` without arguments activates a standard prompt that simply consists of the `>` character.

Command under Workbench 1.3

The new `Prompt` command makes it possible to have the current directory as part of the prompt text. The control characters `%s` are responsible for this and are used in the same manner as the control characters `%n`.

Protect

Syntax: PROTECT [FILE] Filename [FLAGS] Protection flags

AmigaDOS manages a sequence of flags for each file. These flags permit or inhibit certain actions. `Protect` can change these flags.

Filename is the name of the file in which the flags should be changed. The following flags are available in AmigaDOS:

R	Read	-the file can be read
W	Write	-the file can be changed
E	Execute	-the file can be executed as a program
D	Delete	-the file can be deleted

Note: AmigaDOS manages a fifth internal flag called the Archived flag (A). The `Protect` command cannot process this flag. Every time a file is accessed for writing, the A flag is erased. An example: To see if a program should be protected from disk or hard disk data operations, AmigaDOS checks to see whether the last version was changed or not.

When a file is created, AmigaDOS sets all four flags so all of the operations are permitted.

The protection flags that are given behind `Protect` remain set for the processed file.

The input:

```
protect Test rwe
```

sets everything but the D flag - the file cannot be deleted.

When no flags are given behind `Protect`, all of the actions are prevented.

The condition of the protection flags can be seen using the `List` command. Flags not set are replaced with a - in the list.

Attention: In the current version of AmigaDOS only the `D` flags work correctly. All other flags can be set any way you want but AmigaDOS pays no attention. In all practicality, `Protect` works like a delete protect for a file. At the same time write access is not possible on deleted protected files. An operational `D` flag also has the same effect as the `W` flag. Files cannot be protected from reading and execution.

When a directory is delete protected using `Protect`, only the directory is protected and not its contents. `Delete ...` All first notices the protection when the entire contents of the directory are already erased.

Note: `Protect` accepts no wildcard characters in the file name. You must enter each file to be protected separately.

Command Under Workbench 1.3

Syntax: `Protect FILE/A,FLAGS,ADD/S,SUB/S`

There are four new flags with Workbench 1.3:

H (Hidden)
S (Script)
P (Pure)
A (Archived)

When using the combination Workbench 1.3/Kickstart 1.2 you have access to the `P` and `S` flags.

A Hidden flag suppresses the entry of the respective files in the directory. Using this method, long directories can be much clearer.

The **Script** flag deals with script files. When the **Script** flag is set, the script file can be started from a **Shell**. It isn't necessary to enter the **Execute** command before the script file any more.

A **Pure** flag means that the associated program can be loaded with the help of the **Resident** command. By doing this it is always ready for the user and it also does not have to be loaded from the drive any more.

The **Archive** flag prevents the possibility of copying files under **Kickstart 1.3**. The **Copy** command only copies files that do not have the **Archive** flag set. A file with a set **A** flag is recognized as being archived. When the file is written to, the **Archive** flag is automatically erased. The **Copy** command is signalled that the file has been written to and a new **Archive** flag must be set to protect it.

The **Add** and **Sub** arguments have been added to the **Protect** command. With these it is possible to set single flags and then take them away. The description of the complete status words is no longer necessary. A single flag can be set using **Add** and then it can be taken away using **Sub**.

The **Add** and **Sub** arguments can be replaced by plus and minus signs.

Quit

Syntax: QUIT [Error code]

Quit is used exclusively in script files to end the file. When you add an error code, the message:

```
quit failed returncode ((Error code))
```

appears in the **CLI** window. When testing the script file you can assign different error codes to the different break conditions.

When appending a script file to another script file the error code can be useful for transferring information about the severity of the error. See also `If [Warn|Error|Fail]`.

Note: The output of the error code only functions when the code is greater than the error limit. The error limit must be lowered at the beginning of the script file if you want to use a code under ten.

Relabel

Syntax: RELABEL [DRIVE] Drive [NAME] Diskname

Disks, hard disk partitions and similar data carriers can be renamed with `Relabel`. This command has the equivalent function of the `Rename` item from the `Workbench` menu when used on a disk icon.

`Drive` is the device name of the storage medium to be renamed. For disk drives, the disk in the drive is the one to be renamed; and for hard drive partitions, the partition is renamed.

Note: Instead of the device name, you can give the name of a disk or partition (for example "A2000 WB 1.2D:"), or the name of a logical device that is on the disk (for example, `SYS:`).

The data carrier encountered first is renamed if there is more than one disk or partition with the same name.

Attention: When you give a logical device that isn't displayed on a disk or partition (like `C:` or `Devs:`), the disk or partition where the logical device is found is renamed. Devices like `Prt:`, `Con:`, `Nil:`, etc. cannot be renamed.

The `Diskname` argument can be up to 30 characters long. It must be in quotation marks if it contains spaces.

Note: When formatting a disk on the Workbench interface the standard name "Empty" is given. When formatting with the `Format` or `DPFormat` command in the CLI, the name must be given without conditions. Both name assignments can be changed with `Relabel`.

Command Under Workbench 1.3

No changes have been made.

Remrad	(Workbench 1.3 only)
---------------	----------------------

<i>Syntax:</i> <code>remrad</code>

The contents of the reset-resistant RAM disk `RAD:` can be erased using the `Remrad` (Remove Recoverable Ram Disk) command. The RAM disk then takes up a relatively small section of memory. When the computer is re-started, this memory is returned to the system.

Rename

<i>Syntax:</i> <code>RENAME [FROM] Filename [TO AS] Newfilename</code>
--

`Rename` renames the given file or directory. The name must be in quotation marks if it contains spaces. An error message appears if the file or directory `Newfilename` already exists.

Renaming a directory does not have the same effect on its contents. A special feature of the `Rename` command is the ability to remove a file from one directory and place it in another.

`rename :Text/Letter as :Data/Letter`

Takes the file `Letter` from the `Text:` directory and puts it in the `Data:` directory.

Using this method you can move directories from one place on a disk to another place:

```
rename :Text/Private as :Data/Private
```

If **Private:** is a directory, the entire directory along with its contents and subdirectories is moved from the **Text:** directory into the **Data:** directory.

Note: Rename can only work with files or directories within one disk. Otherwise you'll receive the error message "Rename across devices attempted".

Command Under Workbench 1.3

There are no changes to the Rename command. It is no longer possible to have two files in the RAM disk with the same name due to a better RAM handler.

Resident

(Workbench 1.3 only)

Syntax: resident NAME,FILE,DELETE/S,ADD/S,REPLACE/S,
PURE/S,SYSTEM/S

Resident makes it possible for the user to load his most frequently used commands into the work memory. Then the command is present and there is no need to load it.

Before the **Resident** command existed, the important commands were copied into the RAM disk and DOS was informed by means of the **Path** command to look in the RAM disk before it accessed the Workbench. This method functioned very well except for one large disadvantage: When a command in the RAM disk was called, it still had to be loaded just like from the disk drive. This is a very inefficient use of memory because the command is then present in two locations. Each new call of the program copied another command into RAM.

Commands that are loaded using **Resident** are only in the work memory once. When it is called for a second time from a second CLI, the program code only finds a location in RAM.

You can easily understand that a CLI command must meet some requirements so that this procedure can function:

1. The command must be "re-executable". This means that you must be able to use it from more than one CLI. Example: In the CLI window the directory of drive 0 is listed while the Dir command is being used in CLI window 2 for drive 1. All programs, with very few exceptions, are "re-executable" on the Amiga.
2. The commands must be "re-entrant". As described above, the program code of a resident command is only found in one location when the command is executed in several places at the same time. The feature that makes a re-entrant command so good is the use of local variables that must be replaced with every call of the program.

When the command is entered without parameters, a list of the commands that are resident at the present appears. The resident system segments are also shown. For example:

<u>Name</u>	<u>UseCount</u>
CD	1
Dir	1
Execute	1
CLI	SYSTEM
Filehandler	SYSTEM
Restart	SYSTEM
CLI	SYSTEM

Information about how active the respective command is at the time is given under UseCount. This statement is usually a 1. A 1 also means that the command is not being used at the time. System segments are listed as System.

In the syntax list of the Resident command Name and File stand for the exact path of the command or segment that should become resident. For example:

```
resident c:dir
```

The `Dir` command is in the `Shell` ready for use. The path for the command you want made resident must be specified.

When you use the `Resident` command in a file where the `Pure` flag is not set, the error message:

```
Pure bit not set
Cannot load xxx
(yyy stands for the filename)
```

appears. When `Pure` is not set, a file can be loaded using `Resident` by adding `Pure`. The message "Pure bit not set" is displayed in this case. The `Pure` option should be used with caution because programs where the `Pure` flag is not set are not usually re-entrant.

The `Remove` option serves to eliminate an entry from the list of resident files. For example:

```
resident execute remove
```

The `Execute` command is removed and its memory location is made available.

The `UseCount` value of a system segment is set at -1. Because an entry can only be removed, when `UseCount` is at one a segment cannot be erased using `Remove`.

The `Add` argument makes it possible to make more commands or segments resident with the same name. It can only call the last entered command from the `Shell`.

`Replace` allows any command (or segment) to be replaced with a command (or segment) already in the list. For example, if the `Execute` command is resident and you enter:

```
resident execute c/date replace
```

then the `Date` command can be called using the input from `Execute`.

Run

Syntax: RUN Command [+ <Return> Command + <Return>...]

Run starts a given command or program as a background task. The CLI from which the command or program was started remains free for further use. Because the background task does not have its own window, the eventual return message is given in the CLI window from which it was started.

Note: When one or more tasks are started from one CLI, the Amiga waits until all of the tasks are finished before closing the window. After an EndCLI the window remains on the screen, but it cannot be used for entering more CLI commands.

You can enter more commands after Run. They are executed one after another. The individual commands can be separated by the + character and entering <Return>. No more characters are allowed to be in the line after the + (including spaces). Immediately after this the <Return> must be pressed.

Multiple AmigaDOS commands can be executed one after another in the background:

```
run copy Text to ram:Temp +<Return>
copy ram:Temp to prt: +<Return>
delete ram:Temp +<Return>
echo "That was it." <Return>
```

This way you can make sure that multiple commands are executed one after another:

```
run :Preferences +<Return>
:Clock +<Return>
df0:DPaint <Return>
```

As soon as one program ends or exits, the Amiga automatically starts the next one.

Note: When a command or program is stopped due to an error, the following sections are not executed.

Command Under Workbench 1.3

It should be possible to leave the Shell that was used to start a process by using EndCLI, but also without closing the window that is eventually used for output.

The following command creates a background process that should write the entire contents of the disk in drive zero in a file named List:

```
run >List dir df0: opt a
```

It should be theoretically possible to leave the Shell using EndCLI and close the window while the Dir command continues to work. Unfortunately it doesn't work that way because the device receives an End of File command character. The number of the process (for example CLI [2]) is given to the device.

Our example file displays the process number instead of the disk directory if the command type List is used.

S a y

Syntax: SAY [-m|-f] [-r|-n] [-s Speech speed] [-p Speech frequency] [-x filename] | [Text]

Say can be started from the Workbench as a program and can also be used from the CLI as an AmigaDOS command.

With this command the Amiga can speak a given text (Say Hello) or can read the contents of a text file. To read from a file contents the -x option follows the name of the corresponding file (Say -x:Startup-sequence).

Attention: It does not make sense to read the contents of a file that isn't in ASCII format (for example, program files). In most cases, Say recognizes that it is not a text file and ignores the

speech command. If the Say command executes anyway, the Amiga responds with gibberish.

The following arguments allow control of voice parameters:

- m male voice
- f female voice
- r robot monotone
- n natural intonation
- s statement of speech speed between 40 (slow) and 400 (fast)
- p statement of speech frequency between 65 (deep) and 320 (high)

When Say is given without arguments, the Say program is started as if it was called from the Workbench interface. In this case two windows appear. The options for the speech commands can be entered in the top of each window, and the text to be spoken can be entered in the bottom. The program can be ended by pressing <Ctrl><\> or entering an empty line in the input window.

Note: Say is not found on the Workbench disk. It is kept on the ExtrasD disk. When you have a hard disk you can simply copy the Say program into the System drawer. When you only have one drive, you can place the search path over the ExtrasD disk using Path.

Owners of the Amiga 1000 and 500 can find the command in the System drawer.

Search

Syntax: SEARCH [[FROM] Filename] [SEARCH] Searchtext
[ALL]

Search searches through the given file for the search text and displays all of the lines in which the text was found on the screen.

When wildcards are used in the filename or if a directory is used, **Search** looks through all files. The names of the files that are searched are displayed on the screen.

The **All** argument means that all subdirectories and their files are searched through. When you want to search through all of the files in the current directory, you can leave the statement of the filenames off. In this case the **Search** command option is brought into action:

```
search search ((Search text))
```

The search text must be in quotation marks when it contains spaces.

AmigaDOS does not differentiate between lower and uppercase letters when searching.

When the search text is found in a file, AmigaDOS displays the line (with line number) in which the text occurs.

When searching through a program file or data file that is not in ASCII format, it can happen that the line is too long for **Search**. The command then reads characters until it runs into a line feed code.

When a line has more than 205 characters, **Search** stops reading and prints the message:

```
Line ((Line number)) truncated
```

Then the search is continued in the next line.

<Ctrl><D> Stops the search in the current file. **Search** then continues in the next file.

<Ctrl><C> Completely ends the search.

SetClock

Syntax: SETCLOCK OPT LOAD|SAVE

This command is of interest only to those Amiga owners who have a built-in real time clock. AmigaDOS manages its own time which is completely independent of the time of the real time clock.

The concept "time" includes the clock time and the date. The real time clock can be read or set with SetClock.

SetClock Opt Load reads the time from the real time clock and transfers it to AmigaDOS. The AmigaDOS time is then set according to the real time clock. When this command is executed in the startup sequence, AmigaDOS recognizes the real time clock as the correct time. This spares you from having to enter the correct time after every system start.

Use the Date command to enter the current clock time and the current date from AmigaDOS.

Note: When the real time clock is first put into operation, the SetClock Opt Load command may display the message:

<invalid> <invalid> <invalid>

The real time clock contains some data that cannot be interpreted as the clock time and date. After the first SetClock Opt Save this message doesn't appear any more. SetClock gives the message:

Internal clock not functioning

if the real time clock is defective or not built-in.

Command Under Workbench 1.3

No changes have been made.

SetDate

Syntax: SETDATE [FILE] Filename [DATE] Date [[TIME]
Clock time]

When AmigaDOS accesses a file for writing, it places the current date and time in the file. This time description can be changed with the SetDate command.

All formats for the date and time that are accepted for the Date command are accepted here.

Clock time can be given as HH:MM:SS, and the date can be entered in the form DD-*MMM*-YYY or input like Yesterday, Tomorrow, Monday.

Attention: When you enter a date but no time, AmigaDOS sets the time at 00:00:00. The old time is overwritten.

The date and time when AmigaDOS last modified a file can be seen with the List command.

Command Under Workbench 1.3

No changes have been made.

Setenv

(Workbench 1.3 only)

Syntax: setenv NAME/A,String

This command makes it possible to use environment variables. The environment handler is still missing. The handler can be simulated by the RAM disk, but full use is not yet realized.

SetMap

Syntax: SETMAP Keyboard driver

A national keyboard setting can be inserted using SetMap.

The American keyboard setup is the default on the Amiga. This assignment is part of the Kickstart ROM. The drivers for all other settings must be inserted with SetMap.

AmigaDOS searches for the given keyboard driver in the Devs: Keymaps directory. The following keyboard drivers are found there:

cdn	Canadian
ch1	Swiss (french)
ch2	Swiss (german)
d	German
dk	Danish
e	Spanish
f	French
gb	British
i	Italian
is	Island
n	Norwegian
s	Swedish
usa0	USA keyboard setup from Workbench 1.1 (Amiga 1000)
usa1	USA standard setup (in Kickstart ROM)
usa2	Dvorak keyboard

SetPatch

(Workbench 1.3 only)

Syntax: Syntax : setpatch

This command is found in the startup sequence on the new Workbench disk. It modifies the Kernal so that a Guru meditation does not follow a Recoverable Alert. SetPatch is a background process started using Run. It can be created with the help of the Status command. Some versions of Workbench 1.3 may have this command under the name SetAlert.

Skip

Syntax: SKIP [Label]

Skip is a jump command in script files. **Label** is a jump mark that must be defined with the **Lab** command. AmigaDOS continues processing the script file in the line after the **Lab** command.

When no label is given after **Skip**, **Skip** jumps to the next **Lab** command. When **Skip** cannot find the given label, the script file is interrupted with the following error message:

Label "((Label))" not found by Skip

Jumps in script files are only allowed in the direction of the end of the file. **Skip** can only jump over a row of lines and commands. A jump back to a line that has already been processed does not work. No loops can be programmed in script files.

Command Under Workbench 1.3

No changes have been made.

Sort

Syntax: SORT [FROM] Filename [TO] Sortfile [COLSTART
Start column]

Sort sorts the text in text files according to lines. **Filename** is an ASCII file which should be sorted. The result of the sort operation is saved in the **Sortfile**.

Note: Similar to the **Search** command, **Sort** reads characters from a file until it finds a line feed code. It doesn't make sense to sort files that aren't in ASCII format.

Normally, each line is sorted according to the first column (the first character). Another start column can be given using `Colstart`. The all characters before this column are ignored.

The lines are written in alphabetical order in the `Sortfile`. Lower and uppercase letters are not differentiated between when sorting.

Attention: `Sort` can only sort files with a maximum of 200 lines. When a larger file is given, you can run into memory problems. In this case you should enlarge the stack memory of the current CLI with the `Stack` command.

Command Under Workbench 1.3

No changes have been made.

Stack

Syntax: `STACK [Stack memory]`

Each task has its own stack memory. Intermediate values and so forth are stored in this memory area. Tasks that are started from a CLI get a standard 4000 byte stack memory.

The size of the stack memory for the current CLI can be displayed or changed with the `Stack` command. `Stack` without parameters displays the size of the stack memory at the time.

Attention: When the stack memory for a task is not sufficient, a system interrupt occurs. You have the option of changing the stack memory area. Under no circumstances should the stack memory be set below 4000 bytes.

When more CLIs or other tasks are set up from one CLI, they are given the same size stack memory as the current CLI.

Command Under Workbench 1.3

No changes have been made.

Status

Syntax: STATUS [PROCESS] Task number [FULL] [TCB]
[CLI|ALL]

Status displays an overview of the active CLI tasks on the screen. **Status** without arguments displays a list of currently running CLI tasks and the commands or programs in process.

You can also give a certain task number and the information about this task will be displayed. The TCB argument stands for Task Control Block and displays the following information about a task:

Task 1: stk 3200, gv 150, pri 0

“stk” displays the task memory, “gv” shows the size of the Global Vector Table, and “pri” is the priority of the given task. Also read the description of the **Stack** and **ChangeTaskpri** commands.

Note: An error in the current version of the **Status** command causes an incorrect representation of negative task priorities. When the task priority on the list is greater than +128, you must subtract 256 from the priority.

Status Full displays the data of the Task Control Blocks and the command being processed for each task.

The CLI or All arguments work exactly like **Status Full** when placed behind TCB.

Command Under Workbench 1.3

Syntax: PROCESS, FULL/S, TCB/S, CLI=ALL/S, COM=COMMAND/K

The new **Status** command handles negative priorities correctly. In addition, the new argument COM=COMMAND/K is introduced. With its help you can determine if a certain program is found under the current process. The **Status** command must be given COM (or COMMAND) and the name of the process.

Type

Syntax: TYPE [FROM] Filename [[TO] Filename2] [OPT N|H]

Type allows you to display the contents of a file. Normally the output goes to the screen.

The name of a file or a logical device (like PRT:) can be given after TO. The output is then directed to this device or file. This results in the file either being copied or displayed.

OPT N causes each line to be printed with a line number. This option is only used for ASCII files.

OPT H prints the contents of the file in hexadecimal form. This mode is especially useful for displaying program files that are not in ASCII format.

Note: The display of the file contents can be stopped with <Ctrl><C>. Pressing any key halts the display until <Backspace>, <Return> or <Ctrl><X> is entered.

Command Under Workbench 1.3

Syntax: type FROM/A, TO, OPT/K, HEX/S, NUMBER/S

A difference from the old Type command is that the options OPT H and OPT N can now be activated by Hex and Number. For example:

```
type s:startup-sequence number
```

replaces

```
type s:startup-sequence opt n
```

Version

Syntax: VERSION

Version displays the current version number of the Kickstart and Workbench.

The ROM version of Kickstart 1.2, which is built-in to the Amiga 500 and 2000, has the version number 33.180. Kickstart versions that are loaded from disk for the Amiga 1000 can have different version numbers. 33.180 is the most recent.

The Workbench should have the version number 33.56. 33.53 and 33.48 are also common, but they are already being outdated. The output of the **Version** command can look like the following:

```
Kickstart version 33.180. Workbench version 33.56
```

Command Under Workbench 1.3

No changes have been made.

Wait

Syntax: WAIT [n [SEC|SECS] [MIN|MINS]]
WAIT UNTIL Time

The **Wait** command either pauses for a certain time or waits until a certain time.

This function is interesting for script files and also after calling a task with **Run**. In both cases, situations can be encountered where one task must wait for another.

The wait time can be entered in *n* seconds or *n* minutes. The value is interpreted as seconds if no statement is given.

Wait without parameters waits for 1 second. When the **Until** option or a time is given after **Wait**, AmigaDOS waits until this time is reached.

Note: Compare this to the **Date** and **SetClock** commands.

Wait can be used as an alarm clock or reminder: Let the Amiga wait in a background task until a certain time and then start a program such as **Clock**.

Command Under Workbench 1.3

No changes have been made.

Why

Syntax: WHY

When a command does not function and does not give an extensive error message, you can ask AmigaDOS why nothing happened. The **Why** command displays the encountered error in a plain text message.

This function only works when **Why** is entered immediately after the command that caused the error.

Note: **Why** can only give a plain text message when AmigaDOS can find the **Fault** command. **Why** can only give an error number if it cannot find the **Fault** command.

Please read the description of **Fault**. This command allows you to translate error numbers into plain text messages.

Command Under Workbench 1.3

No changes have been made.

Error Messages

This section lists AmigaDOS error messages by number.

103 insufficient free store

AmigaDOS can't load the program due to insufficient memory. End any other tasks or close any other open CLI windows.

105 task table full

AmigaDOS can only manage 20 CLI tasks at once. As soon as the internal task table is full, no more CLIs can be opened.

120 argument line invalid or too long

This error message appears if an AmigaDOS command has a problem with the given parameters.

121 file is not an object module

Only program files can be started by directly entering their names (e.g., script files must be started with Execute).

122 invalid resident library during load

A problem occurs when opening or loading a library.

202 object in use

This message prevents file writing or directory deletion while another task accesses the file or directory.

203 object already exists

A given name already exists and cannot be erased.

204 directory not found

AmigaDOS cannot find a given directory.

205 object not found

A file or directory cannot be found from AmigaDOS.

206 invalid window description

The syntax was incorrect when opening a window. Check the window coordinates and syntax (e.g., con:0/0/635/100/).

209 packet request type unknown

A driver cannot fulfill a desired access. This only occurs because of programming errors.

211 invalid object lock

A programming error created an invalid lock code.

212 object not of required type

Confusion between files and directories causes this error.

213 disk not validated

The disk in the drive is probably damaged.

214 disk write-protected

It is not possible to write to this disk. The write protect clips are probably in the wrong position.

215 rename across devices attempted

Rename cannot rename from one disk to another.

216 directory not empty

Trying to erase a directory that is not empty causes this error.

218 device (or volume) not mounted

AmigaDOS cannot find the requested disk.

219 seek failure

A false argument was given when calling the Seek function.

220 comment too big

File comments added to a file with FileNote cannot be longer than 80 characters.

221 disk full

No memory available on the given disk for the desired action.

222 file is protected from deletion

The file is probably protected from deletion with Protect.

223 file is write protected

224 file is read protected

Both of these commands react to the protection flags set using Protect. These two error message are not used because the present version of DOS only supports the D flag.

225 not a valid DOS disk

Either the disk structure of the disk is completely destroyed or it was not formatted under AmigaDOS.

226 no disk in drive

There is no disk in the requested drive at the time.

232 no more entries in directory

This programming error informs you that the access of the ExNext routine in a directory cannot find any more entries.

CLI Shortcuts

The CLI and Shell commands include <Ctrl> and <Esc> command sequences that can be entered from the keyboard. The command characters can also be used in script files through the Echo command. The Escape sequence appears in quotation marks, beginning with an asterisk acting as the <Esc> key (e.g., echo "*ec" clears the screen). You can change the type style, enter a color, move the cursor and more by entering these codes in a CLI window.

<Esc> c	Clear screen and disable all special modes
<Esc> [0m	Disable all special modes (normal characters)
<Esc> [1m	Bold type
<Esc> [2m	Black type (color number 2)
<Esc> [3m	Italic type
<Esc> [30m	Blue type (color number 0)
<Esc> [31m	White type (color number 1)
<Esc> [32m	Black type (color number 2)
<Esc> [33m	Orange type (color number 3)
<Esc> [4m	Underlining
<Esc> [40m	Blue background (color number 0)
<Esc> [41m	White background (color number 1)
<Esc> [42m	Black background (color number 2)
<Esc> [43m	Orange background (color number 3)
<Esc> [7m	Inverse presentation (normally blue on white)
<Esc> [8m	Blue type, invisible (or color number 0)
<Esc> [nu	Width of CLI window in characters (n)
<Esc> [nx	Left margin of the CLI window in pixels (n)
<Esc> [ny	Distance of window from top in pixels (n)
<Esc> [nt	Number of lines in CLI window (n)

-
- <Ctrl><H>** Deletes the last character entered
or **<Backspace>**
- <Ctrl><I>** Moves the cursor to a tab position to the right
or **<Tab>** (default 5 characters)
- <Ctrl><J>** Enters line feed without executing the entered
command. This allows multiple command entry.
Pressing the **<Return>** key executes all
commands in sequence
- <Ctrl><K>** Moves the cursor to a position as above. The
text that is there cannot be changed
- <Ctrl><L>** Clears the screen
- <Ctrl><M>** Ends the line and executes the entered
or **<Return>** commands
- <Ctrl><N>** Enables the Alt character set. Only special
characters are printed
- <Ctrl><O>** Disables the Alt character set and returns to the
normal character set
- <Ctrl><X>** Deletes the current line
- <Ctrl><\\>** Signals the end of a file in AmigaDOS. Also
ends input in Con: windows

ASCII Table

0	[CTRL]-[@]	32	!	64	@	96	'
1	[CTRL]-[A]	33	!	65	A	97	a
2	[CTRL]-[B]	34	"	66	B	98	b
3	[CTRL]-[C] (Break)	35	#	67	C	99	c
4	[CTRL]-[D]	36	\$	68	D	100	d
5	[CTRL]-[E]	37	%	69	E	101	e
6	[CTRL]-[F]	38	&	70	F	102	f
7	[CTRL]-[G] (Beep)	39	'	71	G	103	g
8	[CTRL]-[H] <Backspace>	40	(72	H	104	h
9	[CTRL]-[I] <TAB>	41)	73	I	105	i
10	[CTRL]-[J] (Linefeed)	42	*	74	J	106	j
11	[CTRL]-[K]	43	+	75	K	107	k
12	[CTRL]-[L] 	44	,	76	L	108	l
13	[CTRL]-[M] <Return>	45	-	77	M	109	m
14	[CTRL]-[N]	46	.	78	N	110	n
15	[CTRL]-[O]	47	/	79	O	111	o
16	[CTRL]-[P]	48	0	80	P	112	p
17	[CTRL]-[Q]	49	1	81	Q	113	q
18	[CTRL]-[R]	50	2	82	R	114	r
19	[CTRL]-[S]	51	3	83	S	115	s
20	[CTRL]-[T]	52	4	84	T	116	t
21	[CTRL]-[U]	53	5	85	U	117	u
22	[CTRL]-[V]	54	6	86	V	118	v
23	[CTRL]-[W]	55	7	87	W	119	w
24	[CTRL]-[X]	56	8	88	X	120	x
25	[CTRL]-[Y]	57	9	89	Y	121	y
26	[CTRL]-[Z]	58	:	90	Z	122	z
27	[CTRL]-[[<ESC>	59	;	91	[123	{
28	[CTRL]-[\ <Crsr up>	60	<	92	\	124	
29	[CTRL]-[] <Crsr down>	61	=	93]	125	}
30	[CTRL]-[^ <Crsr right>	62	>	94	^	126	~
31	[CTRL]-[_ <Crsr left>	63	?	95	_	127	ÿ

128	□		160	ì	192	À	224	à
129	□	<F1>	161	í	193	Á	225	á
130	□	<F2>	162	î	194	Â	226	â
131	□	<F3>	163	ï	195	Ã	227	ã
132	□	<F4>	164	ÿ	196	Ä	228	ä
133	□	<F5>	165	ÿ	197	Å	229	å
134	□	<F6>	166	ÿ	198	Æ	230	æ
135	□	<F7>	167	ÿ	199	Ç	231	ç
136	□	<F8>	168	ÿ	200	È	232	è
137	□	<F9>	169	ÿ	201	É	233	é
138	□	<F10>	170	ÿ	202	Ê	234	ê
139	□	<HELP>	171	ÿ	203	Ë	235	ë
140	□		172	ÿ	204	Ì	236	ì
141	□		173	ÿ	205	Í	237	í
142	□		174	ÿ	206	Î	238	î
143	□		175	ÿ	207	Ï	239	ï
144	□		176	ÿ	208	Ð	240	ð
145	□		177	ÿ	209	Ñ	241	ñ
146	□		178	ÿ	210	Ò	242	ò
147	□		179	ÿ	211	Ó	243	ó
148	□		180	ÿ	212	Ô	244	ô
149	□		181	ÿ	213	Õ	245	õ
150	□		182	ÿ	214	Ö	246	ö
151	□		183	ÿ	215	Ø	247	ø
152	□		184	ÿ	216	Ù	248	ù
153	□		185	ÿ	217	Ú	249	ú
154	□		186	ÿ	218	Û	250	û
155	□		187	ÿ	219	Ü	251	ü
156	□		188	ÿ	220	Ý	252	ý
157	□		189	ÿ	221	Þ	253	þ
158	□		190	ÿ	222	ß	254	ÿ
159	□		191	ÿ	223		255	ÿ

Escape Sequences

The following printer Escape sequences are translated using the printer drivers included in Preferences.

<u>Escape sequence</u>	<u>Meaning</u>
<Esc>c	Initialize (reset) printer
<Esc>#1	Disable all other modes
<Esc>D	Line feed
<Esc>E	Line feed + carriage return
<Esc>M	One line up
<Esc>[0m	Normal characters
<Esc>[1m	Bold on
<Esc>[22m	Bold off
<Esc>[3m	Italics on
<Esc>[23m	Italics off
<Esc>[4m	Underlining on
<Esc>[24m	Underlining off
<Esc>[xm	Colors (x=30 to 39 [foreground] or 40 to 49 [background])
<Esc>[0w	Normal text size
<Esc>[2w	Elite on
<Esc>[1w	Elite off
<Esc>[4w	Condensed type on
<Esc>[3w	Condensed type off
<Esc>[6w	Enlarged type on
<Esc>[5w	Enlarged type off
<Esc>[2"z	NLQ on
<Esc>[1"z	NLQ off
<Esc>[4"z	Double strike on
<Esc>[3"z	Double strike off

<Esc>[6"z	Shadow type on
<Esc>[5"z	Shadow type off
<Esc>[2v	Superscript on
<Esc>[1v	Superscript off
<Esc>[4v	Subscript on
<Esc>[3v	Subscript off
<Esc>[0v	Back to normal type
<Esc>[2p	Proportional type on
<Esc>[1p	Proportional type off
<Esc>[0p	Delete proportional spacing
<Esc>[xE	Proportional spacing = x
<Esc>[5F	Left justify
<Esc>[7F	Right justify
<Esc>[6F	Set block
<Esc>[0F	Set block off
<Esc>[3F	Justify letter width
<Esc>[1F	Center justify
<Esc>[0z	Line dimension 1/8 inch
<Esc>[1z	Line dimension 1/6 inch
<Esc>[xt	Page length set at x lines
<Esc>[xq	Perforation jumps to x lines
<Esc>[0q	Perforation jumping off
<Esc>(B	American character set
<Esc>(R	French character set
<Esc>(K	German character set
<Esc>(A	English character set
<Esc>(E	Danish character set (Nr.1)
<Esc>(H	Swedish character set
<Esc>(Y	Italian character set
<Esc>(Z	Spanish character set
<Esc>(J	Japanese character set
<Esc>(6	Norwegian character set
<Esc>(C	Danish character set (Nr.2)

<Esc>#9	Set left margin
<Esc>#0	Set right margin
<Esc>#8	Set header
<Esc>#2	Set footer
<Esc>#3	Delete margins
<Esc>[xyr	Header x lines from top; footer y lines from bottom
<Esc>[xys	Set left margin (x) and right margin (y)
<Esc>H	Set horizontal tab
<Esc>J	Set vertical tab
<Esc>[0g	Delete horizontal tab
<Esc>[3g	Delete all horizontal tabs
<Esc>[1g	Delete vertical tab
<Esc>[4g	Delete all vertical tabs
<Esc>#4	Delete all tabs
<Esc>#5	Set standard tabs

Memory Map

\$FFFFFF	KickStart ROM (256K) (Amiga 500 and 2000)
\$FC0000	Address copy of KickStart ROM (256K)
\$F80000	
\$F00000	Expansion slots
\$E80000	
\$E00000	Custom chip register
\$DF0000	Battery-backed realtime clock (Amiga 500 and 2000)
\$DC0000	
\$C80000	CPU RAM (512K) (Amiga 500 and 2000)
\$C00000	CIA B and
\$B00000	CIA A
\$A00000	FastRAM (up to 8 megabytes)
\$200000	Address copy of chip RAM
\$080000	Chip RAM (512K)
\$000000	

Guru Meditation Codes

Guru Meditations supply information about system crashes.

Guru Meditations return two eight-digit numbers. The first number gives detailed error information in the following format:

System ID code	Error class	Error number
XX	XX	XXXX

The second eight-digit number gives the starting address of the task that started the interrupt.

System ID codes

00	CPU trap
01	Exec library
02	Graphics library
03	Layers library
04	Intuition library
05	Math library
06	CList library
07	DOS library
08	RAM library
09	Icon library
0A	Expansion library
10	Audio device
11	Console device
12	GamePort device
13	Keyboard device
14	Trackdisk device
15	Timer device
20	CIA resource
21	Disk resource
22	Misc resource

30	Bootstrap
31	Workbench
32	Diskcopy

Error classes

01	Insufficient memory
02	MakeLibrary error
03	OpenLibrary error
04	OpenDevice error
05	OpenResource error
06	I/O error
07	No signal

Special guru meditation codes

Note: When a system ID code begins with a number greater than or equal to 8, the error is non-recoverable. Subtract 8 from the first digit to get the true system ID code.

CPU traps

00000002	Bus error
00000003	Address error
00000004	Illegal instruction
00000005	Divide by zero
00000006	CHK instruction
00000007	TRAPV instruction
00000008	Privilege violation
00000009	Trace
0000000A	Opcode 1010
0000000B	Opcode 1111

Exec library

81000001	Error in 68000 exception vector checksum
81000002	Error in ExecBase checksum
81000003	Error in a Library checksum
81000004	Insufficient memory for MakeLibrary
81000005	Memory list scrambled
81000006	No free memory for interrupt server
81000007	Problem with InitAPtr

8100008 Semaphore scrambled
8100009 Double call from free
810000A "Bogus Exception"

Graphics library

8201001 Insufficient memory for Copper display list
8201002 Insufficient memory for Copper command list
8201003 Copper list overflow
8201004 "Copper Intermediate" list overflow
8201005 Insufficient memory for header of Copper list
8201006 Memory absence at Long Frame
8201007 Memory absence at Short Frame
8201008 Insufficient memory for Flood Fill
8201009 Insufficient memory for TmpRas
820100A Insufficient memory for BltBitMap
820100B "Region Memory"

Layers library

8301001 No available memory for layers

Intuition library

8400001 Gadget type unknown
8401002 Insufficient memory to add port
8401003 Insufficient memory for Item Plane Alloc
8401004 Insufficient memory for Sub Alloc
8401005 Insufficient memory for Plane Alloc
8400006 Original coordinate smaller than RelZero
8401007 Insufficient memory to open screen
8400008 Insufficient memory for Raster Alloc
8401009 Unknown type at Open Sys Screen
810100A Insufficient memory for gadgets
810100B Insufficient memory for window
810000C Faulty return code encountered in Intuition
810000D IDCMP sent a faulty message
840000E Answer was incomprehensible
840000F Error when opening Console device

DOS library

07010001	Memory problem at startup
07000002	Problem with EndTask
07000003	Problem with Qpkt
07000004	Receiver packet not expected
07000005	Problem with FreeVec
07000006	Error in DiskBlock sequence
07000007	Faulty bitmap
07000008	Key already erased
07000009	Checksum false
0700000A	Diskette error
0700000B	Incorrect value for key
0700000C	Problem at overlay

RAM library

08000001	Faulty Segment-List
----------	---------------------

Expansion library

0A000001	Problem at Expansion Free
----------	---------------------------

Trackdisk device

14000001	Seek error at calibrate
14000002	Error at timer delay

Timer device

15000001	Incorrect request
15000002	Incorrect transfer

Disk resource

21000001	Get drive has prepared the diskette
21000002	Interrupt: no active drive

Bootstrap

30000001	Boot code error
----------	-----------------

Quick Index

<code>;</code>	10	<code>FailAt</code>	44
<code>< ></code>	10	<code>Fault</code>	45
<code>AddBuffers</code>	11	<code>FF (1.3 only)</code>	45
<code>Alias (1.3 only)</code>	11	<code>FileNote</code>	46
<code>Ask</code>	15	<code>Format</code>	46
<code>Assign</code>	16	<code>Getenv</code>	48
<code>Avail (1.3 only)</code>	18	<code>If...Else...EndIF</code>	49
<code>BindDrivers</code>	19	<code>Info</code>	50
<code>Break</code>	19	<code>InitPrinter</code>	51
<code>Cd</code>	20	<code>Install</code>	52
<code>ChangeTaskPri</code>	21	<code>Join</code>	53
<code>Copy</code>	22	<code>Lab</code>	54
<code>Date</code>	25	<code>List</code>	54
<code>Delete</code>	27	<code>LoadWB</code>	57
<code>Dir</code>	28	<code>Lock (1.3 only)</code>	58
<code>DiskChange</code>	29	<code>MakeDir</code>	58
<code>DiskCopy</code>	30	<code>Mount</code>	59
<code>DiskDoctor</code>	31	<code>NewCLI</code>	60
<code>DJMount</code>	32	<code>NewShell</code>	63
<code>DPFormat</code>	33	<code>NoFastMem</code>	65
<code>Echo</code>	33	<code>Path</code>	66
<code>ED</code>	34	<code>Prompt</code>	68
<code>Edit</code>	37	<code>Protect</code>	69
<code>Else</code>	42,49	<code>Quit</code>	71
<code>EndCLI</code>	42		
<code>EndIf</code>	42,49		
<code>Execute</code>	43		

Relabel	72
Remrad (1.3 only)	73
Rename	73
Resident (1.3 only)	74
Run	77
Say	78
Search	79
SetClock	81
SetDate	82
Setenv (1.3 only)	82
SetMap	82
SetPatch (1.3 only)	83
Skip	84
Sort	84
Stack	85
Status	86
Type	87
Version	88
Wait	88
Why	89

Subject Index

CLI management

Abbreviate commands	Alias (1.3)	11
Access speech device	Say	78
Alter priority of task	ChangeTaskPri	21
Change prompt parameters	Prompt	68
Define keyboard codes	SetMap	82
Display/change stack size	Stack	85
Guru-proof recoverable alert	SetPatch (1.3)	83
Invoke/change break key	Break	19
KickStart/Workbench version	Version	88
Open new CLI window	NewCLI	60
Open new Shell window	NewShell (1.3)	63
Time delay	Wait	88

Disk management

Add disk buffers	AddBuffers	11
Change disk name	Relabel	72
Create hard disk partitions	DPFormat	33
Duplicate disk	DiskCopy	30
Format floppy disk	Format	46
Make bootable floppy disk	Install	52
Remove reset-proof RAM disk	Remrad (1.3)	73
Repair disk structure	DiskDoctor	31
Indicate changed disk	DiskChange	29
Write protect partition	Lock	58

File management

Change directory	Cd	20
Change file protection status	Protect	69
Concatenate files	Join	53
Copy files/directories	Copy	22
Create directory	MakeDir	58
Delete files/empty directories	Delete	27
Display directory	Dir	28
Display reason for error	Why	89
Display text file contents	Type	87
Insert 80-column comment	FileNote	46
List directory in detail	List	54
Rename file/directory	Rename	73
Search files for text	Search	79
Sort text file	Sort	84

Script files

Conditional jump	If/Else/EndIf	49
Determines error limit	FailAt	44
Display data	Echo	33
Exit CLI	EndCLI	42
Exit script file	Quit	71
Exit Shell	EndShell	42
Icon script file access	IconX (1.3)	48
Invoke line editor	Edit	37
Invoke screen editor	Ed	34
Jump to label	Skip	84
Label definition	Lab	54
Run script file	Execute	43
Start comments	;	10
Wait for y/n user response	Ask	15

System management

Add device drivers	BindDrivers	19
Add/display device	Mount	59
Assign DOS path	Path	66
Assign logical device name	Assign	16
Create resident CLI commands	Resident	74
Disable fast memory	NoFastMem	65
Display active tasks	Status	86
Display available memory	Avail	18
Display drive information	Info	50
Display error number	Fault	45
Display/change date	Date	25
Enable/disable FastFonts	FF	45
Execute one or more tasks	Run	77
Get environment variables	Getenv	48
Initialize printer	InitPrinter	51
Load Workbench	LoadWB	57
Mount Janus hard disk partitions	DJMount	32
Redirect input/output	< >	10
Set file date	SetDate	82
Set file time	SetClock	81
Use environment variables	Setenv	82

Index

; (Commentary)	10
< (Redirection)	10
> (Redirection)	10
AddBuffers	11
Adding device drivers	19
Alias	11
Argument template	7
ASCII table	96-97
Ask	15
Assign	16
Avail	18
BindDrivers	19
Break	19
Cd	20
ChangeTaskPri	21
CLI	60,94
Comments	10,46
CON:	4
Concatenation	53
Copy	22
Date	25
Delete	27
Devices	4-5
Dir	28
DiskChange	29
DiskCopy	30
DiskDoctor	31
DJMount	32

DPFormat	33
Echo	33
ED	34
Edit	37
Else	42,49
EndCLI	42
EndIF	42,49
EndShell	42
Environment variables	48,82
Error limit	44
Error messages	90-93
Escape code sequences	98-100
Execute	43
FailAt	44
FastFonts	45
Fault	45
FF	45
FileNote	46
Format	46
Getenv	48
Guru meditations	83,102
Hard disk	4,32,33
Icon access	48
IconX	48
If/Else/EndIF	49
Info	50
InitPrinter	51
Install	52
Janus hard disk	32
Join	53
Lab	54
Labels	54
List	54

Abacus	Index
LoadWB	57
Lock	58
MakeDir	58
Memory map	101
Mount	59
Multitasking	21,77
NewCLI	60
NewShell	63
NIL:	5
NoFastMem	65
PAR:	4
Partitions	33
Path	66
Prompt	68
Protect	69
PRT:	4
Quit	71
RAD:	73
RAM:	4
RAM disk	4
RAW:	4-5
Redirection	10
Relabel	72
Remrad	73
Rename	73
Renaming disks	72
Renaming files	73
Resident	74
Run	77
Say	78
Screen editor	34-36
Script files	34,71
Search	79
SER:	4

SetClock	81
SetDate	82
Setenv	82
SetMap	82
SetPatch	83
Shell	63
Skip	84
Sort	84
Speech control	78
Stack	85
Status	86
Time	25
Type	87
Version	88
Wait	88
Why	89
Wildcards	6-7
Workbench	57

New Books for the **AMIGA**™



AMIGA BASIC- Inside and Out

AMIGA BASIC—Inside and Out is the definitive step-by-step guide to programming the AMIGA in BASIC. This huge volume should be within every AMIGA user's reach. Every AMIGA BASIC command is fully described and detailed. In addition, **AMIGA BASIC—Inside and Out** is loaded with real working programs.

Topics include:

- Video titling for high quality object animation
- Bar and pie charts
- Windows
- Pull down menus
- Mouse commands
- Statistics
- Sequential and relative files
- Speech and sound synthesis

Plus much, much more. Over 550 pages of vital information are contained in this book. Available late first quarter 1987.

Suggested retail price: \$24.95



AMIGA Tricks & Tips

AMIGA Tricks & Tips follows a tradition established by our other Tricks & Tips books for Commodore computers. It's another solid collection of diverse and useful programming techniques written for everyone who uses the AMIGA. This easy to understand source book details applications for the stunning processing power of the AMIGA.

Topics include:

- Displaying 64 colors on screen simultaneously
- Accessing libraries from BASIC
- Creating custom character sets
- Using Amiga DOS, graphics

In addition, **AMIGA Tricks & Tips** presents dozens of tips on windows, programming aids, the AMIGA's speech synthesis and musical capabilities, covers important 68000 memory locations, and much more.

AMIGA Tricks & Tips is available second quarter 1987.

Suggested retail price: \$19.95



Optional program diskettes are available for our AMIGA books

Suggested retail price: \$14.95



New Books for the AMIGA™



Amiga for Beginners

A perfect introductory book of you're a new or prospective Amiga owner. **Amiga for Beginners** introduces you to Intuition (the Amiga's graphic interface), the mouse, windows, the versatile CLI. This first volume in our Amiga series explains every practical aspect of the Amiga in plain English. Includes clear, step-by-step instructions for common Amiga tasks. Amiga for Beginners is all the info you need to get up and running.

Topics include:

- Unpacking and connecting the Amiga components
- Starting up your Amiga
- Customizing the Workbench
- Exploring the Extras disk
- Taking your first steps in the AmigaBASIC programming language
- AmigaDOS functions
- Using the CLI to perform "housekeeping" chores
- First Aid, Keyword, Technical appendixes
- Glossary



Suggested retail price: \$16.95

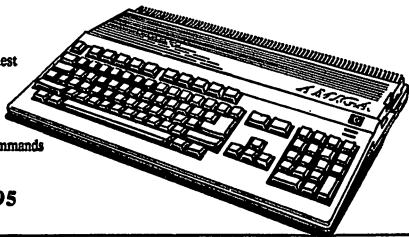


No optional disk available

Amiga System Programmer's Guide

Amiga System Programmer's Guide has a wealth of information about what goes on inside the Amiga. Whether you want to know about the Amiga kernel or DOS commands, Amiga System Programmer's Guide has the information you need, explained in a manner that you can easily understand. Just a few of the things you will find inside:

- EXEC Structure
- Multitasking functions
- I/O management through devices and I/O request
- Interrupts and resource management
- RESET and its operation
- DOS libraries
- Disk management
- Detailed information about the CLI and its commands
- Much more—over 600 pages worth



Suggested retail price: \$34.95



Optional program diskettes are available for our AMIGA books
Suggested retail price: \$14.95



New Books for the AMIGA™



Amiga Machine Language

The practical guide for learning how to program your Amiga in ultrafast machine language. Used in conjunction with our AssemPro Amiga software package, **Amiga Machine Language** is a comprehensive introduction to 68000 assembler/machine language programming. Topics include:

- 68000 microprocessor architecture
- 68000 address modes and instruction set
- Accessing RAM, operating system and multitasking capabilities
- Details the powerful Amiga libraries for using AmigaDOS
- Speech and sound facilities from machine language
- Many useful programs listed and explained



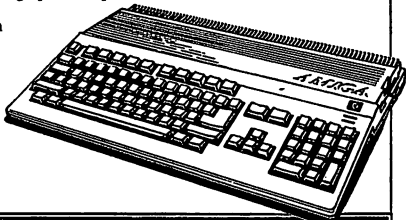
Suggested retail price: \$19.95

AmigaDOS Inside & Out

AmigaDOS covers the insides of AmigaDOS from the internal design up to practical applications. There is also a detailed reference section which helps you find information in a flash, both alphabetically and in command groups.

Topics include:

- 68000 microprocessor architecture
- AmigaDOS - Tasks and handling
- Detailed explanations of CLI commands and their functions
- DOS editors ED and EDIT
- Operating notes about the CLI (wildcards, shortening input and output)
- Amiga devices and how the CLI uses them
- Batch files - what they are and how to write them
- Changing the startup sequence
- AmigaDOS and multitasking
- Writing your own CLI commands
- Reference to the CLI, ED- and EDIT commands
- Resetting priorities - the TaskPri command
- Protecting your Amiga from unauthorized use



Suggested retail price: \$19.95



Optional program diskettes are available for our **AMIGA** books

Suggested retail price: \$14.95



Selected Abacus Products for the *Amiga* computers

AssemPro

Machine Language Development System for the Amiga

Bridge the gap between slow higher-level languages and *ultra-fast* machine language programming: AssemPro *Amiga* unlocks the full power of the AMIGA's 68000 processor. It's a complete developer's kit for rapidly developing machine language/assembler programs on your Amiga. AssemPro has everything you need to write professional-quality programs "down to the bare metal": editor, debugger, disassembler & reassembler.

Yet AssemPro isn't just for the 68000 experts. AssemPro is easy to use. You select options from dropdown menus or with shortcut keys, which makes your program development a much simpler process. With the optional Abacus book *Amiga Machine Language* (see page 3), AssemPro is the perfect introduction to Amiga machine language development and programming.

AssemPro also has the professional features that advanced programmers look for. Lots of "extras" eliminate the most tedious, repetitious and time-consuming m/l programming tasks. Like syntax error search/replace functions to speed program alterations and debugging. And you can compile to memory for lightning speed. The comprehensive tutorial and manual have the detailed information you need for fast, effective programming.

AssemPro *Amiga* offers more professional features, speed, sheer power, and ease of operation than any assembler package we've seen for the money. Test drive your AssemPro Amiga with the security of the Abacus 30-day MoneyBack Guarantee. Available January 1988.

Suggested retail price: **\$99.95**



Features

- Integrated Editor, Debugger, Disassembler and Reassembler
- Large operating system library
- Runs under CLI and Workbench
- Produces either PC-relocatable or absolute code
- Create custom macros for nearly any parameter (of different types)
- Error search and replace functions
- Cross-reference list
- Menu-controlled conditional and repeated assembly
- Full 32-bit arithmetic
- Advanced debugger with 68020 single-step emulation
- Written completely in machine language for ultra-fast operation
- Runs on any Amiga with 512K or more and Kickstart version 1.2
- Not copy protected

Machine language programming requires a solid understanding of the AMIGA's hardware and operating system. We do not recommend this package to beginning Amiga programmers

P

PROFESSIONAL *DataRetrieve*

File your other databases away!

Professional DataRetrieve, for the Amiga 500/1000/2000, is a friendly easy-to-operate professional level data management package with the features of a relational data base system.

Professional DataRetrieve has complete relational data management capabilities. Define relationships between different files (one to one, one to many, many to many). Change relations without file reorganization.

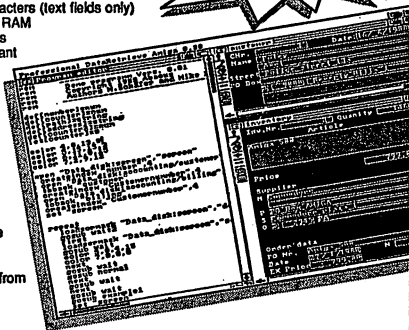
Professional DataRetrieve includes an extensive programming language which includes more than 200 BASIC-like commands and functions and integrated program editor. Design custom user interfaces with pulldown menus, icon selection, window activation and more.

Professional DataRetrieve can perform calculations and searches using complex mathematical comparisons using over 80 functions and constants.

Professional DataRetrieve's features:

- Up to 8 files can be edited simultaneously
- Maximum size of a data field 32,000 characters (text fields only)
- Maximum number of data fields limited by RAM
- Maximum record size of 64,000 characters
- Maximum number of records disk dependant (2,000,000,000 maximum)
- Up to 80 Index fields per file
- Up to 6 field types - Text, Date, Time, Numeric, IFF, Choice
- Unlimited number of searches and sub-range criteria
- Integrated list editor and full-page printer mask editor
- Index accuracy selectable from 1-999 characters
- Multiple file masks on-screen
- Easily create/edit on-screen masks for one or many files
- User-programmable pulldown menus
- Operate the program from the mouse or from the keyboard
- Calculation fields, Date fields
- IFF Graphics supported
- Mass-storage-oriented file organization
- Not Copy Protected, no dongle: can be installed on your hard drive

\$295⁰⁰



Abacus 

5370 52nd St. SE Grand Rapids MI 49508 - Order Toll Free! 800-451-4319

Selected Abacus Products for the *Amiga* computers

BeckerText

Powerful Word Processing Package for the Amiga

BeckerText *Amiga* is more than just a word processor. BeckerText *Amiga* gives you all of the easy-to-use features found in our TextPro *Amiga*, plus it lets you do a whole lot more. You can merge sophisticated IFF-graphics anywhere in your document. You can hyphenate, create indexes and generate a table of contents for your documents, automatically. And what you see on the BeckerText screen is what you get when you print the document—real WYSIWYG formatting on your Amiga.

But BeckerText gives you still more: it lets you perform calculations of numerical data within your documents, using flexible templates to add, subtract, multiply and divide up to five columns of numbers on a page. BeckerText can also display and print multiple columns of text, up to five columns per page, for professional-looking newsletters, presentations, reports, etc. Its expandable built-in spell checker eliminates those distracting typographical errors.

BeckerText works with most popular dot-matrix and letter-quality printers, and even the latest laser printers for typeset-quality output. Includes comprehensive tutorial and manual.

BeckerText gives you the power and flexibility that you need to produce the professional-quality documents that you demand.

When you need more from your word processor than just word processing, you need BeckerText *Amiga*.

Discover the power of BeckerText. Available February 1988.

Suggested retail price: **\$150.00**



Features

- Select options from pulldown menus or handy shortcut keys
- Fast, true WYSIWYG formatting
- Bold, italic, underline, superscript and subscript characters
- Automatic wordwrap and page numbering
- Sophisticated tab and indent options, with centering and margin justification
- Move, Copy, Delete, Search and Replace
- Automatic hyphenation, with automatic table of contents and index generation
- Write up to 999 characters per line with horizontal scrolling feature
- Check spelling as you write or interactively proof document; add to dictionary
- Performs calculations within your documents—calculate in columns with flexible templates
- Customize 30 function keys to store often-used text and macro commands
- Merge IFF graphics into documents
- Includes *BTSnap* program for converting text blocks to IFF graphics
- C-source mode for quick and easy C language program editing
- Print up to 5 columns on a single page
- Adapts to virtually any dot-matrix, letter-quality or laser printer
- Comprehensive tutorial and manual
- Not copy protected

Selected Abacus Products for the *Amiga* computers

DataRetrieve

A Powerful Database Manager for the Amiga

Imagine a powerful database for your Amiga: one that's fast, has a huge data capacity, yet is easy to work with.

Now think *DataRetrieve Amiga*. It works the same way as your Amiga—graphic and intuitive, with no obscure commands. You quickly set up your data files using convenient on-screen templates called masks. Select commands from the pulldown menus or time-saving shortcut keys. Customize the masks with different text fonts, styles, colors, sizes and graphics. If you have any questions, Help screens are available at the touch of a button. And *DataRetrieve's* 128-page manual is clear and comprehensive.

DataRetrieve is easy to use—but it also has professional features for your most demanding database applications. Password security for your data. Sophisticated indexing with variable precision. Full Search and Select functions. File sizes, data sets and data fields limited only by your memory and disk storage space. Customize up to 20 function keys to store macro commands and often-used text. For optimum access speed, *DataRetrieve* takes advantage of the Amiga's multi-tasking.

You can exchange data with *TextPro Amiga*, *BeckerText Amiga* and other packages to easily produce form letters, mailing labels, index cards, bulletins, etc. *DataRetrieve* prints data reports to most dot-matrix & letter-quality printers.

DataRetrieve is the perfect database for your Amiga. Get this proven system today with the assurance of the Abacus 30-day MoneyBack Guarantee.

Suggested retail price:

\$79.95



Features

- Select commands and options from the pulldown menus or shortcut keys
 - Enter data into convenient screenmasks
 - Enhance screen masks with different text styles, fonts, colors, graphics, etc.
 - Work with 8 databases concurrently
 - Define different field types: text, date, time, numeric & selection
 - Customize 20 function keys to store macro commands and text
 - Specify up to 80 index fields for *superfast* access to your data
 - Perform simple or complex data searches
 - Create subsets of a larger database for even faster operation
 - Exchange data with other packages: form letters, mailing lists etc.
 - Produce custom printer forms: index cards, labels, Rolodex® cards, etc. Adapts to most dot-matrix & letter-quality printers
 - Protect your data with passwords
 - Get Help from online screens
 - Not copy protected
- Max. file size
• Max. data record size
• Max. data set
• Max. no. of data fields
• Max. field size

*Limited only
by your memory
and disk space*

Selected Abacus Products for the *Amiga* computers

TextPro

The Ideal Word Processing Package for the Amiga

TextPro Amiga is a full-function word processing package that shares the true spirit of the Amiga: easy to use, fast and powerful—with a surprising number of "extra" features.

You can write your first **TextPro** documents without even reading the manual. Select options from the dropdown menus with your mouse, or use the time-saving shortcut keys to edit, format and print your documents.

Yet **TextPro** is much more than a beginner's package. It has the professional features you need for all of your printed documents. Fast formatting on the screen: bold, italic, underline, etc. Centering and margin justification. Page headers and footers. Automatic hyphenation of text. You can customize the **TextPro** keyboard and function keys to suit your own style. Even merge IFF-format graphics right into your documents. **TextPro** includes *BTSnap*, a utility for saving IFF graphics that you can use in your graphics programs. This package can also convert and print other popular word processor files.

TextPro is output-oriented. This means you can print your documents to exact specifications—and get top performance out of your dot-matrix or letter quality printer. (Printer drivers included on diskette let you customize **TextPro** to virtually any printer on the market). The complete tutorial and manual shows you how it's all done, step by step.

TextPro sets a new standard for word processors in its price range. Easy to use, packed with advanced features—it's the ideal package for all of your wordprocessing needs. Backed by the Abacus 30-day MoneyBack Guarantee.

Suggested retail price:

\$79.95



Features

- Fast editing and formatting on screen
- Display bold, italic, underline, superscript and subscript characters
- Select options from dropdown menus or handy shortcut keys
- Automatic wordwrap & page numbering
- Sophisticated tab and indent options, with centering & margin justification
- Move, Copy, Delete, Search & Replace options
- Automatic hyphenation
- Customize up to 30 function keys to store often-used text, macro commands
- Merge IFF format graphics into your documents
- Includes *BTSnap* program for saving IFF graphics from any program
- Load & save files through RS-232 port
- Flexible, *ultrafast* printer output—printer drivers for most popular dot-matrix & letter quality printers included
- Comprehensive tutorial and manual
- Not copy protected

COMPUTER VIRUSES

a high-tech disease

Computer VIRUSES, A High-Tech Disease describes the relatively new phenomena among personal computer users, one that has potential to destroy large amounts of data stored in PC systems. Simply put, this book explains what a virus is, how it works and what can be done to protect your PC against destruction.

Computer VIRUSES, A High Tech Disease starts with a short history of computer viruses and will describe how a virus can quietly take hold of a PC. It will give you lots of information on the creation and removal of computer viruses. For the curious, there are several rudimentary programs which demonstrate some of the ways in which a virus infects a PC.

Computer VIRUSES, A High Tech Disease even presents techniques on inoculating the PC from a virus. Whether you want to know a little or a lot about viruses, you'll find what you need in this book. 288 pages, \$18.95

Written by Ralf Burger

Published by Abacus Software Inc.

About the author: Ralf Burger is a system engineer who has spent many years experimenting with virus programs and locating them in computer systems.

Topics Include:

- What are computer viruses
- A short history of viruses
- Definition of a virus
- How self-manipulating programs work
- Design and function of viral programs
- Sample listings in BASIC, Pascal and machine language
- Viruses and batch file
- Examples of viral software manipulation
- Protection options for the user
- What to do when you're infected
- Protection viruses and strategies
- A virus recognition program
- Virus proof operating systems

Contact Abacus For More Information!

Abacus 

5370 52nd Street • Grand Rapids, MI 49508 • (616) 698-0330

How to Order

Abacus  5370 52nd Street SE Grand Rapids, MI 49508

All of our Amiga products—application and language software, and our Amiga Reference Library—are available at more than 2000 dealers in the U.S. and Canada. To find out the location of the Abacus dealer nearest you, call:

Toll Free 1-800-451-4319

8:30 am-8:00 pm Eastern Standard Time

Or order from Abacus directly by phone with your credit card. We accept Mastercard, Visa and American Express.

Every one of our software packages is backed by the Abacus 30-Day Guarantee—if for any reason you're not satisfied by the software purchased directly from us, simply return the product for a full refund of the purchase price.

Order Blank

Name: _____
 Address: _____
 City _____ State _____ Zip _____ Country _____
 Phone: _____

Qty	Name of product	Price
Mich. residents add 4% sales tax Shipping/Handling charge (Foreign Orders \$12 per item)		
Check/Money order TOTAL enclosed		

Credit Card# _____

Expiration date

Send your completed order blank to:

Abacus Software, Inc.
 5370 52nd St. S.E.
 Grand Rapids, MI 49508

Your order will be shipped within 24 hours of our receiving it



For extra-fast 24-hour shipment service, order by phone with your credit card.

AmigaDOS[®]

Quick Reference

Instant Information...at your Fingertips!

AmigaDOS Quick Reference Guide is an easy-to-use reference tool for beginners and advanced programmers alike. You can quickly find commands for your Amiga by using the three handy indexes designed with the user in mind. All commands are in alphabetical order for easy reference. The most useful information you need fast can be found—including:

- All AmigaDOS commands described **including Workbench 1.3**
- Command syntax and arguments described with examples
- CLI shortcuts
- CTRL sequences
- ESCape sequences
- Amiga ASCII table
- Guru Meditation Codes
- Error messages with their corresponding numbers

Three indexes for instant information at your fingertips! The AmigaDOS Quick Reference Guide is an indispensable tool you'll want to keep close to your Amiga.

ISBN 1-55755-049-2