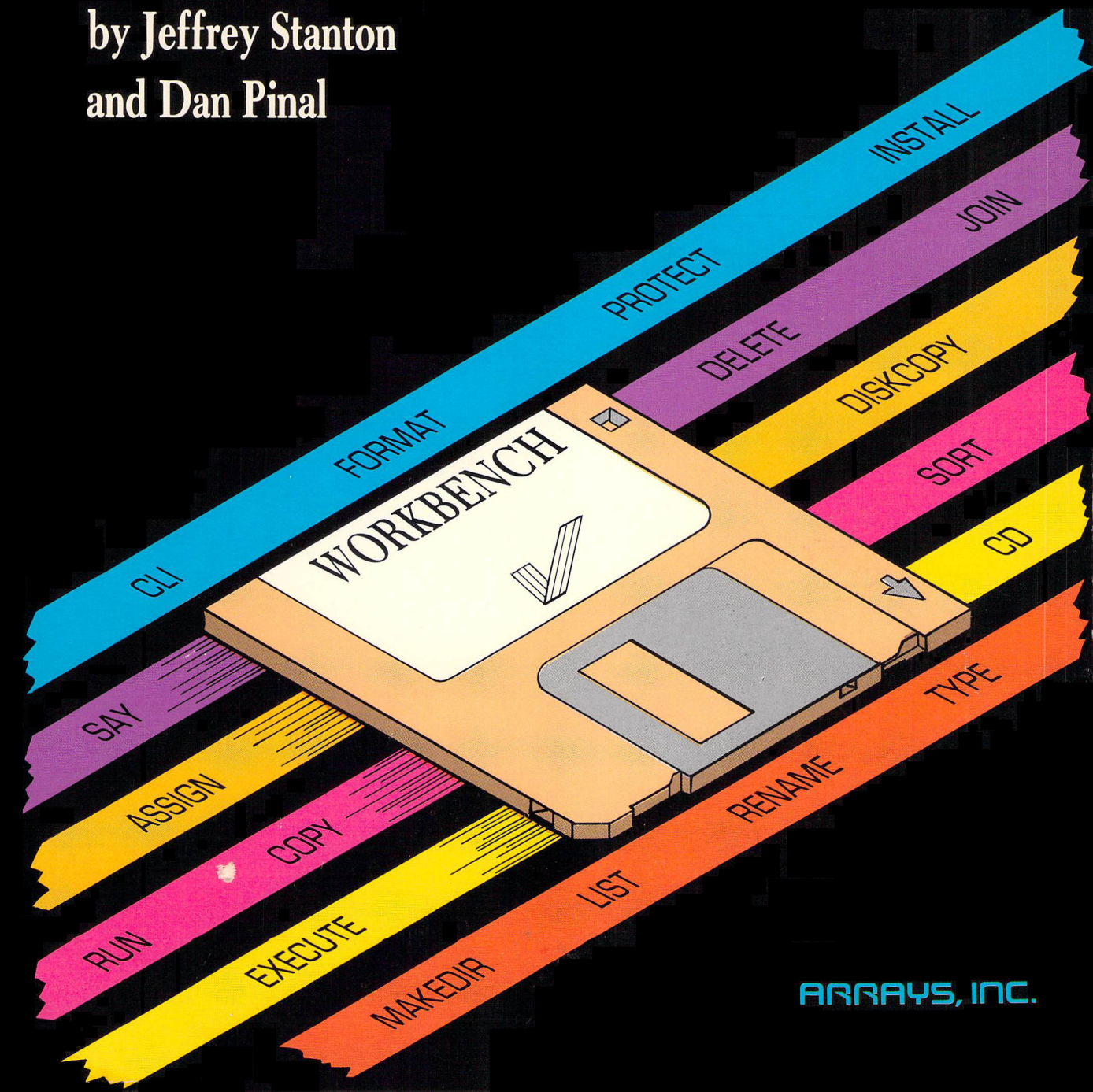
Mastering AmigaDOS

by Jeffrey Stanton
and Dan Pinal



ARRAYS, INC.

Mastering AmigaDOS

by
Jeffrey Stanton
and
Dan Pinal

ARRAYS, INC.
6711 Valjean Avenue
Van Nuys, California 91406

Library of Congress Cataloging-in-Publication Data

Stanton, Jeffrey.
Mastering AmigaDOS.

Includes index.
1. Amiga (Computer)--Programming. 2. AmigaDOS (Computer operating system) I. Pinal, Dan. II. Title.
QA76.8.A177S73 1986 005.4'465 86-10881
ISBN 0-912003-55-3 (pbk.)

Editor
Mia McCroskey

10 9 8 7 6 5 4 3 2 1

ISBN 0-912003-55-3

TABLE OF CONTENTS

INTRODUCTION

WORKBENCH is the user's interface to Amiga's multi-level, multi-tasking operating system called Intuition. It is an icon-based, mouse-driven windowing environment that serves as a command interface to a disk operating system called AmigaDOS. It is capable of running applications programs and performing disk-based functions like deleting files or copying disks and files. Its advantage over the usual command-driven operating system is that it frees the novice user from the drudgery of learning dozens of DOS commands with their often complicated syntax.

AmigaDOS, which is attached to Amiga's Intuition operating system, is a package of subroutines designed to handle graphics, sound, and input/output functions. It enables the programmer to deal with the machine at the utility level rather than the hardware level. AmigaDOS itself provides all the disk operating system functions for the computer, including managing, opening, accessing, updating, and closing files; supporting named devices, allocating memory, and buffering direct memory access (DMA) for the disk drives.

The computer's designers thoughtfully included a keyboard operated Command Line Interface (CLI) in AmigaDOS. A CLI is a traditional command-oriented operating system interface much like MS-DOS or CP/M, only more powerful. You can type in commands at the screen prompt to copy, delete, or rename files, list directories, or load and run files. A CLI can even execute simple programs called batch files. Since the computer is multi-tasking, and the CLI is a task, more than one CLI can operate at one time, each in its own window. Thus, it is possible, for example, to have one CLI request the directory while a second CLI in a separate window sends the contents of a disk file to your printer.

AmigaDOS is certainly powerful, but most of its unique features are hidden by Workbench's user interface. We have to thank Metacomco, the developers of AmigaDOS, for insisting that the underlying CLI be always available as a programmer's interface for more experienced users. They realized that the computer would attract two kinds of users: those who care nothing about

programming and only wish to run applications such as a word processor or music program; and serious programmers who want to get into the guts of the machine and push it to its limits. They provided minimal documentation for the casual users, and stacks of complex documentation for program developers who write in languages like C and 68000 assembly code. What they didn't realize is that a substantial number of users are innately curious about their Amigas and wish to play with them as they did with earlier eight-bit computers. This manual is intended for those users. It is a comprehensive guide to the Workbench and AmigaDOS.

Chapter 1

WORKBENCH AND AMIGADOS

Similarity between Workbench's Icon Filing System and AmigaDOS

Amiga's Workbench is a windowing environment that displays the disk directory in a visual form represented by icons. When you double-click the Workbench disk icon at the upper right corner of the screen via the left mouse button (Select Button), a window named Workbench opens showing various icons like Demos, System, Utilities, Empty, Preferences, Clock, and Trashcan. Some of these icons, like Clock and Preferences, are programs or "Tools," while others like System and Utilities are called "Drawers" because they contain other programs. For example if you double-click the left mouse button over the System icon, another window opens at the bottom right corner of the screen. This window, labeled System, shows a Diskcopy icon, an Icon Editor icon, and another named Initialize. These are programs, or Tools. The CLI icon also appears here if it is enabled in the Preferences program (we'll get to that later).

The way these Workbench icons are arranged, by Drawers and Tools in the initial Workbench window, with additional Tools within a Drawer, corresponds directly to the way AmigaDOS stores files in its directory. The initial group of icons in the Workbench window is called the "root" directory. Each of these Tools and Drawers is stored as a file in the root directory. For example, if you did a directory from the CLI you would obtain the following:

Trashcan(dir)	.info
c(dir)	clock.info
Demos(dir)	Disk.info
System(dir)	Preferences
l(dir)	System.info
devs(dir)	Utilities.info
s(dir)	Clock
t(dir)	Demos.info
fonts(dir)	Empty.info
libs(dir)	Preferences.info
empty(dir)	Trashcan.info
Utilities(dir)	

Figure 1—Sample Directory

You find three types of entries. The plain looking files like Clock and Preferences are tools, or programs, that can be run. Files like System.info contain the icon picture data and its location within the window. Names with a "(dir)" attached to them represent directories containing other files and/or directories. AmigaDOS file structure allows a root directory and multiple sub-directories. Sub-directories can contain further sub-directories or just files. This method allows you to group files by subject, user, or some other personalized system.

If you do a directory of System(dir) in the root directory you find the following:

.info	CLI
CLI.info	Diskcopy
Diskcopy.info	IconEd
IconEd.info	Initialize
Initialize.info	

Figure 2—Sample Sub-directory

This sub-directory contains no other directories, just programs and icon information for the System window.

However, if you do a directory of Fonts(dir) you'll find a combination of fonts and more sub-directories. These additional sub-directories contain files of the available font sizes.

ruby(dir)	ruby.font
opal(dir)	opal.font
sapphire(dir)	sapphire.font
diamond(dir)	diamond.font
garnet(dir)	garnet.font
emerald(dir)	emerald.font

Figure 3—Font Sub-directory

For example: ruby(dir) contains files named "12" and "8." There is no limit to the depth that you can "nest" directories.

Here's a more practical example to explain AmigaDOS's tree structured files. Let's assume that you have a disk named "Empty" that you initialized using the Initialize icon in Workbench's Systems drawer. That disk has a root directory that contains a file called Trashcan(dir). When you save something else to this disk it will automatically be placed in the root directory, at the same level with Trashcan(dir). Say you operate a small business with two very different products and you use a word processor to create letters and invoices for your customers. You create a sub-directory for each of your products in the root directory. Each sub-directory is again divided into letters and invoices, and these in turn are filed by company-date. The figure below shows the structure:

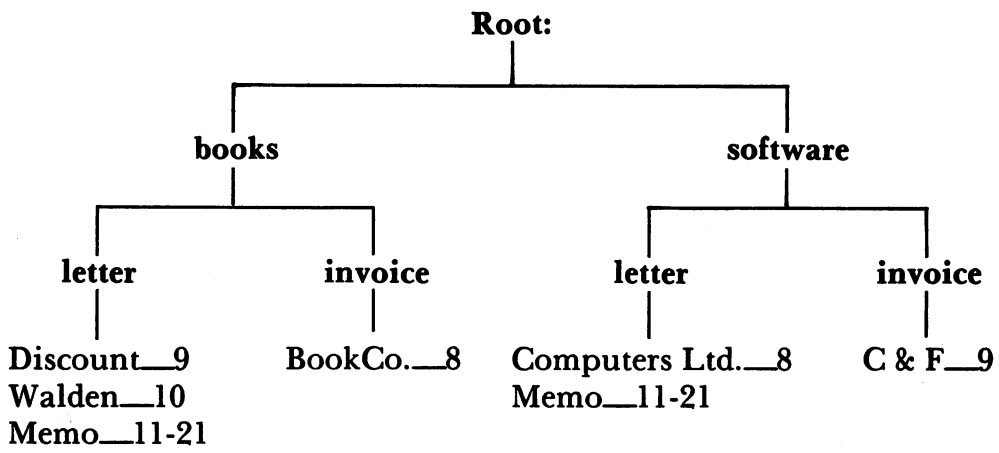


Figure 4—Tree Structured Directory

Directories in unstructured DOS filing systems of older eight-bit computers generally list files in the order they were saved (assuming you haven't replaced or deleted a file name). If you are looking for a file you may have to browse through a list of twenty to forty unrelated files in no particular order. Using the tree-structured directory under AmigaDOS to organize data on disk can save considerable time and keep you from accidentally accessing the wrong file. You know automatically that a letter called "Memo_11-21" in the sub-directory books/letters is not the same as "Memo_11-21" in a sub-directory software/letter. The (_) underline character is used to avoid leaving a space within a filename.

When accessing sub-directories, use a slash (/) to separate each directory name from the next sub-directory file name. For example, to see the list of letters under the sub-directory Books, type "dir books/letters". We will cover this in more detail later in the book in the section on the CLI beginning on page 19.

WORKBENCH OPERATIONS

WORKBENCH'S windowing environment, designed to simplify all disk interactive operations, makes it easy for the user to copy files or disks, obtain information about file size and type, rename or discard files, and, most importantly, to run programs (tools). Most tools provide menus from which you can choose things to do. Replacing the Title bar, these appear along the top of the screen when you press the right hand mouse button (the Menu Button). Workbench, itself a tool, has three menus along its Menu Bar: Workbench, Disk, and Special. To choose one of these, hold down the Menu Button, and move the pointer to it and a drop down menu appears with selectable items. Choose one of the items by pointing to it so that it is highlighted, then releasing the mouse button. The Workbench categories and their associated menus are listed below:

Workbench	**Disk**	**Special**
Open	Empty Trash	Clean Up
Close	Initialize	Last Error
Duplicate		Redraw
Rename		Snapshot
Info		Version
Discard		

Figure 5—The Workbench Menus

Notice that on the screen some of the menu items appear as nondistinct ghost items. This means that the item is temporarily unavailable and can't be selected. Sometimes it is a safety precaution to prevent you from doing something potentially destructive, while at other times it indicates that some other action has to be performed first.

When this ghosting appears on the title bar in a window it means that the window is inactive. While all of the windows in a screen can display information, only one window at a time can accept information from the user. This window is called the "selected" window. You can make any window the selected window by clicking the left mouse button (Select Button) after moving the cursor to any visible point in that window.

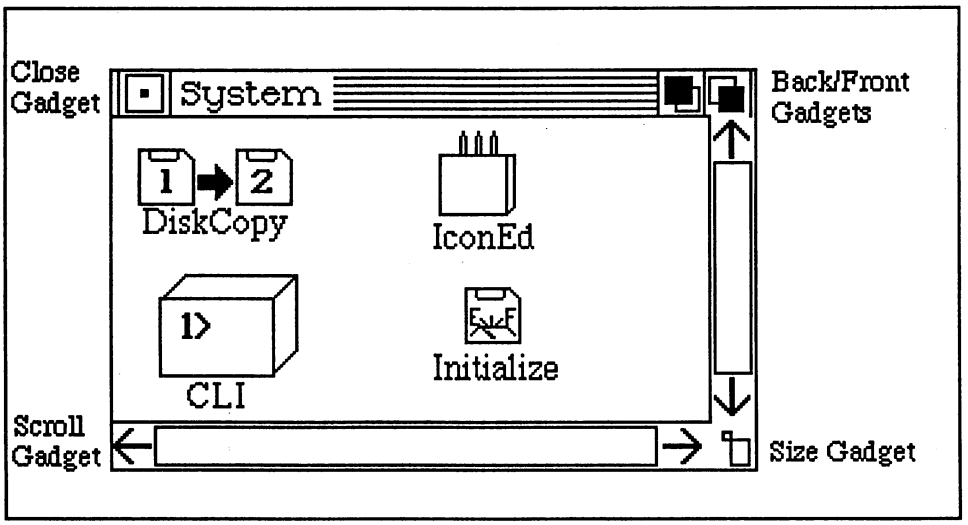


Figure 6—The Window Gadgets

Windows always appear within screens and can't be moved to other screens. However, windows can be dragged or moved anywhere within a screen by pointing in the window's Title Bar (Drag Bar), then holding down the left mouse button and moving the mouse to the desired position (this action is called "dragging").

The small boxes at the corners of the window are called "gadgets." Each window can contain all, some, or none of the gadgets described below. The one at the upper left corner is called the "close gadget." The one at the lower right corner is called the "sizing gadget," and the two at the upper right corner are called the "back gadget" and the

"front gadget," respectively. If you have several windows that overlap on the screen, you can establish which window is displayed in front of which by selecting the appropriate gadget.

The sizing gadget in the lower right corner controls the size of the window. You can stretch or shrink the window by dragging the gadget. If a window becomes too small to display all of the information it contains, you can scroll the image within the window by moving the scroll bar arrows at either end of the scroll bars on the bottom and right sides of the window. Scroll one pixel at a time by pressing the Shift key while clicking on one of the scroll arrows with the mouse. The scroll box changes size according to how much can appear in the window. If the screen is as wide as or wider than necessary, the scroll box fills entirely, while if it is only half the size that is normally needed, half of the box is filled between the scroll arrows.

Usually, you close a window with the mouse by selecting the close gadget at the upper left corner of the window. However, many mouse operations like Open and Close can also be selected from the Menu Bar. Others, like Rename and Empty Trash, can only be accessed from the menus on the Menu Bar.

Duplicating, Copying, and Initializing Disks

It is often necessary and always wise to make backup copies of your disks. You can either duplicate or copy a disk using one or two disk drives. If you plan to store anything on a new disk you must first initialize it. Initializing a disk formats it into eighty tracks—forty on each side. This process also creates a root directory and places a Trashcan file on the disk. You don't need to initialize a disk when you make a copy or a duplicate because the new disk is initialized as you make your copy. *Remember that initializing a disk erases all previous information stored on the disk.*

Disk operations are generally easier with two drives because the computer reads data from the source disk in one drive into memory, then writes it out to the target disk in the other drive. Disks generally store more bytes than internal memory. Amiga disks store 880K bytes of data, while you only have about 400K of free memory on a 512K Amiga (much less with a 256K machine). Therefore, copying with a single drive takes several disk swaps. Worse yet, Amiga doesn't store

all of its DOS commands in memory so it has to get the diskcopy program off the Workbench disk. (That's why you see the prompt: "Please insert volume Workbench" in the middle of a disk copying session.) Unfortunately, even when copying several disks you don't save any time since Workbench always purges the copy program from memory after it has finished copying.

When you're initializing a disk with just one drive, put the disk that you want to initialize into the internal drive. If you have two drives, put it in the external drive. If it is a brand new disk, the disk icon will say either "df0:bad" or "df1:bad" depending on which drive it is in. If it is a used disk it probably has a volume title which will appear on the screen. The internal drive is always 0 and the external drive is 1. You select the disk to be initialized by pointing to it and clicking the left mouse button. This sets the icon to black. With the right mouse button pressed, select Disk from the Menu Bar. Notice that Initialize is no longer a ghost item on the menu. Select it. If you have only one drive, you'll see a system request in the box in the upper left corner asking you to replace Workbench in drive zero. Do so, and Amiga loads the diskcopy program. Then it asks you to replace the disk to be initialized in drive zero. Once again it asks if it's okay to proceed, giving you a chance to back out if you've accidentally inserted a disk you don't really want to initialize. If you choose to proceed, the drive spins for a minute or so, then the disk icon label is changed to "Empty." Double-click on that icon and you'll see that the disk's window contains only a Trashcan.

If you have two disk drives, leave the Workbench disk in the internal drive so you won't be prompted to swap disks. You aren't likely to accidentally initialize your Workbench disk, but it is always safer to write protect it. Slide the tab in the write protect window at the top of the disk so that you see light through the window.

Duplicating a disk is very similar to initializing. You select the icon for the disk to be duplicated, then choose Duplicate from the Workbench menu. Follow the prompts in the request box. Do *not* select the Diskcopy icon from the Systems window.

For example, to copy your Kickstart disk, put it in the internal drive and click its icon to black. Select Duplicate from the Workbench menu. Users of single drive systems are then asked to replace the Workbench disk in the internal drive. (When you choose Duplicate, the Amiga uses only one disk drive even you have two or

more.) A second message then appears stating that it will take three disk swaps (for a 512K Amiga) or six disk swaps (for a 256K Amiga). Select Continue to proceed and remove the Workbench disk. Then follow the prompts, first putting the source disk in the drive, then the destination disk, etc.

Copying disks is much easier with two drives. You just drag the source disk icon on top of the destination disk icon by holding down the left mouse button. The system will prompt you if it needs the Workbench disk to get the diskcopy program. Afterwards, it will instruct you to put the source disk in one drive and the destination disk in the other. You can copy from drive 0 to drive 1 or from 1 to 0.

> **NOTE:** Always put the Workbench/System disk back in the internal drive (DF0:), even if the request states "Any drive." In general, you should put disks back into the same drive where AmigaDOS originally recognized them.

Moving a Tool, Project or Drawer

It is quite simple to rearrange files on a disk. As we explained earlier, files are stored in either the root directory or one of various sub-directories (called Drawers). Files are usually represented by icons within the Workbench window. When you first open the Workbench window you will see a drawer named Empty. If you double-click that icon, a new window named Empty appears with nothing in it. If you open the Systems window, assuming you previously enabled the CLI in the Preferences program, you could drag that icon into Empty's window by holding down the left mouse button. The computer moves the CLI icon and its associated files into the empty drawer.

Perhaps you don't want the Empty drawer on the disk anymore. You need only drag the Empty drawer icon into the Trashcan. *Putting something into the Trashcan doesn't erase it from the disk until you select Empty Trashcan from the Disk menu.* At any time prior to that you can retrieve files and icons that you mistakenly discarded by double-clicking the left mouse button on the Trashcan icon and dragging the items back to whatever drawer or window they are to be stored in. Once you select Empty Trashcan from the Disk menu, all is lost because at that time Amiga physically deletes those

files from the disk. You can also physically delete a file by selecting its icon then selecting "Discard" from the Workbench menu.

Moving a Tool, Project or Drawer to Another Disk

When you want to copy a file or group of files to a new disk, you need only open the source disk's window and the destination disk's window and drag one icon at a time from one to the other. If you are using one drive, the requester will tell you which disk to put in when. Sometimes copying requires more than one pass. Two drive systems require no prompts. The process is nearly automatic. For example the Amiga dealer demo disk contains four Drawers: Graphics, Animations, Pictures, and Sounds. You can copy just the Animations Drawer containing both "Boing" and "Robo-City" by dragging the Animations Drawer from the Demos window to the destination disk window named Empty, your freshly initialized disk. (Of course, you can also copy a drawer onto a disk that already contains other files.)

Renaming Disks and Files

Sometimes you need to rename a file or disk. When you copy a disk using the Discopy icon, the copy is named "Copy of *<filename>*". You might prefer to give the copy a name of its own using the Rename command in the Workbench menus. Select the disk icon (it will turn black), then select Rename from the Workbench menu. A box will appear with the old name. Click in the box with the pointer and type the new name in the box. The old name will scroll right without being erased. Delete the old name with the delete key, which deletes characters under the cursor. Renaming drawers or projects is a similar procedure. Just select the proper icon, choose Rename, and enter the new name.

Info

Info, the last item on the Workbench menu, gives information about any highlighted disk, Drawer, Tool, or Project. For example, if you highlight Preferences by single clicking the mouse button over its icon, then select Info, you will see the following:

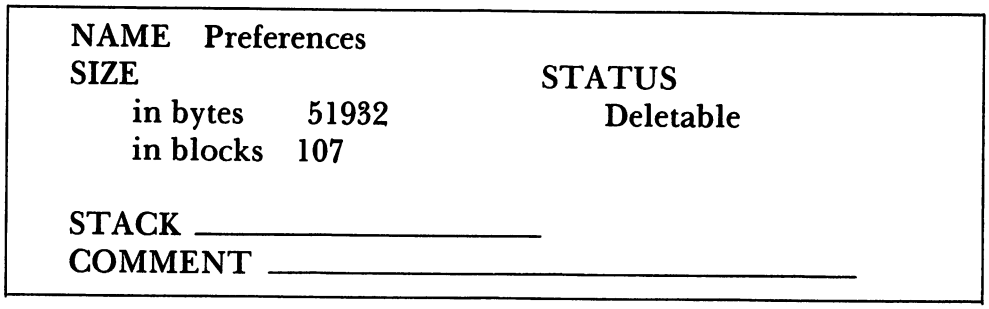


Figure 7—Info about "Preferences"

Special Menu Items

You can perform four other tasks from the Workbench special menu. These include displaying the last error message, straightening up the Workbench icons, redrawing the display, and displaying the version of AmigaDOS you are using.

When you assemble a new disk by copying Drawers and Projects from one or more other disks, the icons tend to be cluttered or overlapping, particularly if you didn't place them carefully when you dragged them into the window. You can use the Clean Up and Snapshot commands to organize them.

Arrange the icons in the window in a pattern you would like to be permanent. You can also select Clean Up and let the Amiga rearrange the icons as it sees fit. Resize the window, if necessary, by dragging the size gadget. When you are satisfied, select all of the icons in the window by holding down the Shift key and clicking on each one. Finally, select Snapshot from the Special menu. This actually writes the information about icon placement and window size to the disk, so each time you load it you'll see a nice, neat window. If you rearrange your icons, but don't select Snapshot, the next time you load the disk everything will be back to the original, messy arrangement.

This Clean Up-Snapshot combination is also quite useful in rearranging disk icons that sometimes appear on top of each other when using multi-disk drive systems. This happens all too often because manufacturers designed the disk to operate in drive 0 and some users insist on using the disk from their external drive. This can be fixed by dragging the overlapping disk icon to a position directly below. Again you will need to select Snapshot to actually save the new position.

Choosing Clean Up from the menu is fairly similar except that the computer straightens up your messy icons in the order that it prefers. Again you will have to select all the icons in the window and use Snapshot if the change is to become permanent.

Redraw is seldom needed. It is used to have Workbench redraw the screen when a program leaves garbage on the screen.

Version (Workbench 1.1 or later) displays the current revision of AmigaDOS.

Chapter 2

COMMAND LINE INTERFACE (CLI)

AmigaDOS's Command Line Interface (CLI) is a traditional command-oriented interface. It, like other operating systems such as MS-DOS and CP/M, can read command names and execute them provided that they are entered with the proper syntax. While AmigaDOS doesn't forgive, it does give you the proper command format if you make a mistake. All commands and user programs run under a particular CLI. Each operates as a separate task, so more than one command-driven operation system is running at one time. The first CLI gives a 1> prompt. The 1> prompt represents the task number assigned to the CLI window rather than the current disk drive number, as is common in operating systems for other computers.

You can start additional CLI tasks by typing "NEWCLI." The second CLI appears in its own window with the prompt "2>." While both can display output simultaneously, only one CLI window at a time can accept input. Select the active window by pointing to it with the mouse then clicking the mouse button. The title bars in any inactive CLI windows appear ghosted. While it is possible to open as many as twenty CLIs in a 512K Amiga before encountering an out-of-memory message, it is probably impractical to have more than three CLI windows open at one time.

CLI tasks can be closed when you are finished with them by entering "ENDCLI" while the appropriate window is active. For example, you can close the second CLI while leaving the first and third still operating. Then, if you request a new CLI in the third window, the second CLI window is reopened. AmigaDOS first checks the number of CLI tasks running before creating an additional one. Finally, if all CLI tasks, including the first, are closed, AmigaDOS reverts to Workbench; that is, assuming Workbench is running beneath the CLI. If not, the computer will hang.

A terminal handler that is line oriented rather than screen oriented processes and directs your keyboard entered DOS commands. The

line length can be up to 255 characters. Mistakes can be corrected by pressing the Backspace key for as many characters as you wish to delete. Holding down the CTRL key while you press X (CTRL-X) rubs out the entire line.

AmigaDOS is careful not to intermix output with your input. Essentially it waits for you to finish a line by pressing Return. Once it's satisfied that you are finished, it displays any output that it was holding back. If you find strange characters appearing on the screen as you type, you may have accidentally typed a CTRL-O. This instructs the console device to display the alternative character set. Just type CTRL-N and the rest of the characters will appear correctly. Last, but not least, AmigaDOS doesn't care if you type commands, arguments, and filenames in uppercase or lowercase. It will find the file regardless of the combination of cases you type. in

Unlike many other operating systems, AmigaDOS commands are not resident in memory. They are loaded from disk only when they are called and then immediately purged from memory. AmigaDOS always looks for commands first from the current directory, then the C sub-directory on the SYS: (startup) disk.

Disk and Device Names

AmigaDOS uses device names to determine the physical flow of data to and from input/output devices. These devices can be disk drives, serial and parallel ports, the keyboard, or the screen. In the case of disk drives, the internal drive is referred to as DF0: and the first external disk drive as DF1:. Two additional drives DF2: and DF3: can be daisy-chained to the system. The colon following the drive name is very important; it tells AmigaDOS that it is a device name rather than a file name.

The system startup disk has a special device name: "SYS:". This disk contains all of the DOS commands in the C sub-directory. You can obtain a complete command list by typing DIR SYS:C. While the system startup disk usually resides in the internal drive DF0:, SYS: refers to the startup disk, not a drive.

Disks can be accessed by name rather than by drive number. In any DOS command, disk names, like device names, must end with a colon. If the disk is not present in any drive, AmigaDOS prompts you

to insert it. These names, like those displayed beneath the disk icon in the Workbench screen, can be up to thirty characters long.

Owners of single-drive systems will find keeping track of disk names to be essential when requesting directories. The DIR program has to be loaded first from the Workbench disk. This means removing the disk to be cataloged, then reinserting the SYS: disk. Unfortunately, without a disk name AmigaDOS will just give a directory of the Workbench disk without prompting you to replace the disk to be cataloged. However, if you include the disk name, the computer will prompt you to insert the correct disk. For example, you can obtain a directory of your Microsoft BASIC disk with DIR EXTRAS:. You can also type DIR ?, and after the directory program is loaded from the Workbench disk, swap the disk to be cataloged.

Other Device Names

AmigaDOS has device names other than those for its disk drives. SER: is the serial port, PAR: is the parallel port, PRT: is the printer driver through either the parallel or serial port, NIL: is a dummy device, CON: usually refers to the current screen, RAM: is a filing system in memory, and RAW: is for sending raw data out a port.

AmigaDOS throws away output written to NIL:—a function which you may find surprisingly useful. You can browse through a file using the editor and have AmigaDOS throw it away as you finish with it by typing:

EDIT Data1 To NIL:

The device PRT: is the printer you choose in Preferences. There you specify whether your printer is connected through the serial or parallel port. The command sequence

COPY Data1 To PRT:

prints the file "Data1" no matter where the printer is connected.

PRT: translates every linefeed character to a carriage return plus a linefeed. Some printers require output without translation. Use PRT:RAW instead of PRT: to send the file untranslated.

PAR: is sometimes the only way to output to parallel printers not listed in the Preferences program. Although the "Generic" printer choice in Workbench 1.1 for standard Centronics interfaced printers is reported to solve most problems, some owners may still need to use PAR:. Our older Epson MX-80 with Graphtrax works fine using the

PAR:. Commands like "COPY Data1 To PAR:" work fine. Incidentally, if the LLIST command to list Microsoft BASIC programs fails, use

SAVE "PAR:",a

Use the SER: device to send contents of a file out the serial port. This can be to a serial printer or some other device, such as a modem. Since the device only copies in multiples of 400 bytes at a time, output will appear at the device in chunks.

AmigaDOS supports multiple windows through the device CON:. You can make a new window by specifying it. The format is as follows:

CON x/y/width/height/[title]

where "x" and "y" are coordinates, "width" and "height" are integers defining the width and height of the new window, and "title," is optional, is a string. The title appears on the window's title bar. You must use all of the slashes, including the last one. Titles can be up to thirty characters. Remember, if the title includes spaces, it must be enclosed in double quotes, like this:

"CON: 25/15/200/90/My window"

Which creates this window:

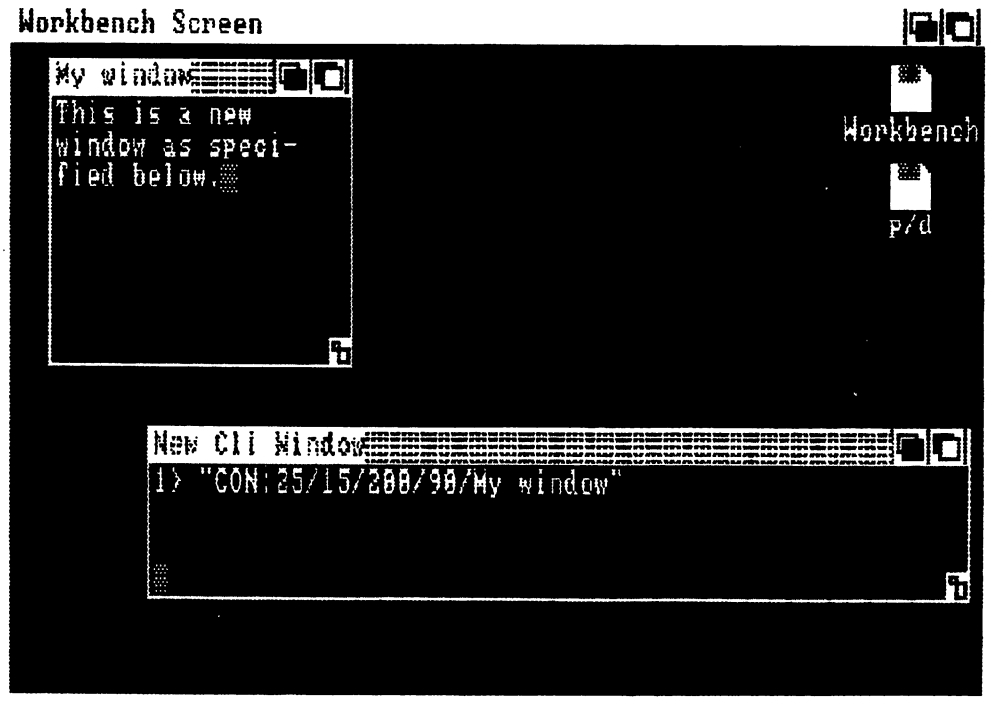


Figure 8—Window specified in the command above

Use an asterisk (*) to refer to the current *window*. (The asterisk is not used as a wildcard as in many other operating systems.) You can use "*" to copy from the current window to another window, for example:

COPY * TO "CON: 20/30/200/100/SMALL BOX"

AmigaDOS finishes copying when it comes to the end of the file. You can stop copying by hitting CTRL-\ (backslash).

Filenames

Filenames can be up to thirty characters long and be in any combination of uppercase and lowercase letters. They can include any printed character except slash (/) and colon (:) which are forbidden. This means that you can include space ( ), equals (=), plus (+) double quote ("), and all special characters recognized by the CLI, within a file name. However, if you use any of these special characters, you must enclose the *entire* filename within double quotes. This means that the filename "My text" (including the space) would have to be enclosed in double quotes in order for the CLI to accept it. Rather than using spaces to make your file names easier to read, use an underline (_) character like this: "My_text."

To use a double quote within a filename, type an asterisk (*) immediately before it.

To get this: **Memo"23** you must type this: **Memo*"23**

To include an asterisk in a filename you must type another asterisk before the CLI will accept it (so two asterisks = one asterisk). Our advice is to avoid these two characters within filenames. Also, spaces before and after the filenames can be confusing.

The asterisk is not used as a universal wildcard in AmigaDOS commands, as it is in older eight-bit DOSs. An asterisk by itself in AmigaDOS indicates the keyboard and current window. To copy a filename to the screen you just type:

COPY FILENAME TO *

Directories

Earlier we explained how disk files are arranged in directories and sub-directories on the disk. But to access or run a file, you must either

be in the correct directory or specify the file fully by including the chain of directories leading to it (the path). For instance, using the example of the files for the small business charted on page 9, you can access the file "Memo_11-21" in the Books sub-directory by specifying the path "Books/letter/memo_11-21". (Note that it doesn't matter if you type names with or without capital letters). Each of the directory names is separated from the next directory or filename by a slash (/).

Setting the Current Directory

Naturally, AmigaDOS provides a shortcut to save you typing these cumbersome filename descriptions. You can change the current directory through a CD (Change Directory) command. If you make "Books" your current directory, you could access that same letter by simply typing "letter/memo_11-21." Notice that you still have to type the name of the sub-directories along the path from the current directory. Since you can make any directory the current directory, you could save even more typing by making "letter" the current directory.

It is possible to reach any file on the disk without returning to the root directory by typing a colon (:) at the beginning of the file description. Thus you could reach the other memo that you wrote that same day for your software business by specifying ":software/letter/memo_11-21". The colon specifies the root directory, so AmigaDOS returns to the root directory, then follows the rest of the path as indicated to the file you want.

There is another shortcut to reaching files in different directories without actually changing to them. A slash (/) before the file name sends AmigaDOS to the directory immediately above the current directory. Thus it is possible to reach the invoice in the Books sub-directory by specifying "/invoice/bookco_8". Two slashes before the filename would take you back two directory levels.

The current directory is also associated with the "current drive" (the drive where you find the directory). These drives are numbered DF0 through DF3. In addition, the logical device SYS: is assigned to the disk from which you started the system. You can use this name in place of that device name (like DF0:).

Recall that prefacing a file description with a colon (:) identifies the root directory of the current drive. You can specify another drive

by preceding the colon with the drive name. For example, "DF1:books/invoice". You can also gain access to a particular file on a particular disk by specifying the disk's "volume name," instead of the device name. For example, "Workbench: System/diskcopy" will access the file "diskcopy" on the Workbench disk.

The device name, unlike a volume name, is not really part of the file name. You can always read a file created on one drive on any other drive. If you created a file known as "DF0:book/invoice/memo_11-21" and then moved its disk to drive DF1:, AmigaDOS could still find the file, reading it as "DF1:book/invoice/memo_11-21".

Left of the colon	Right of the colon	Right of the slash
Device name	Directory name	Sub-directory name
or	or	or
Volume name	Filename	Filename

Figure 9—The structure of file descriptions and directory paths

Rerouting Input and Output

Use the greater than and less than symbols (> and <) to direct the output and input of a command. The direction of the symbol indicates the direction of information flow. You usually use these symbols to change where a command reads input or writes output. Instead of sending output of a command to the current window, you can write output to a file by including the > symbol followed by the filename. Similarly, if you type the < symbol before a filename, the command reads from that file instead of from the keyboard. For example;

Inventory < March1

tells Inventory to accept input from March1 instead of the keyboard.

LIST > temp
SORT temp to *

produces a sorted list of files and displays them on the screen.

DIR > PRT: DF0: OPT A

lists the entire contents of the disk in drive 0 to the printer. This includes both the root directory and all sub-directories. If you do not include "OPT A," AmigaDOS will only list the root directory.

Using Directory Conventions and Logical Devices

AmigaDOS supports and uses a number of logical devices to find the files that your programs occasionally require.

SYS: Represents the SYStem root directory. At startup, AmigaDOS assigns SYS: to the root directory name of the disk in DF0:. If, for example, the disk in drive DF0: has the volume name "Bootdisk," then AmigaDOS assigns SYS: to Bootdisk. After this assignment, any program that refers to SYS: uses that disk's root directory.

C: represents the Commands directory. When you type a command to the CLI like "Dir," AmigaDOS first searches for that command in the current directory. If the system can't find the command there, it looks for "C:Dir". If you have assigned "C:" to another directory (for example "Bootdisk:c"), AmigaDOS reads and executes from "Bootdisk:c/DIR".

The Commands directory can be customized by either renaming the commands or adding commands. You can put any program you like into C: and AmigaDOS will treat it as a CLI command.

L: represents the Library directory. The directory keeps the overlays for large commands and non-resident parts of the operating system, like Disk-Validator, Port-Handler and Ram-Handler. Ram-Handler manages the RAM disk. It is loaded into memory the first time you copy a file to RAM:. Disk-Validator verifies the integrity of the disks that you insert. It also resets the system date to the latest date of creation of a file on the disk. AmigaDOS requires the Library directory to operate.

S: represents the Sequence library. Sequence files contain command sequences that the EXECUTE command searches for and uses. EXECUTE first looks for the sequence file in the current directory. If it can't find it there, it looks in the directory to which you have assigned S: ("Bootdisk:s").

LIBS: AmigaDOS searches the LIBS: device to find disk-based system libraries. Some libraries are loaded from the Kickstart disk into write-protected memory. Other libraries are loaded from the LIBS: directory into read/write memory as needed. Examples are translator.library, which translates English text into phonemes for the speech synthesizer, and info.library, which manages the Workbench info command.

DEVS: is the Device for Open Device Calls. AmigaDOS loads Input/Output device drivers into RAM when you send data to a modem or printer. The parallel.device and serial.device handle the parallel and serial ports. The narrator.device synthesizes speech while the clipboard.device takes care of the clipboard, used to exchange data among applications.

FONTS: Open Fonts looks here for your loadable fonts if they are not already loaded in memory. Like libraries and I/O devices, fonts are loaded into memory when needed. However, an operating system bug can make the system think a font is in memory when it is not.

Amiga fonts are named after gems. The Ruby font is used by Workbench. CLI uses the Topaz font, which is loaded in with the ROM kernal from the Kickstart disk. You can view all of the available font styles with the Notepad tool in the Utilities drawer of the Workbench disk.

> **NOTE:** Sometimes you will find files in the "T:" directory. Many programs like editors place their temporary work files in this directory. This is the first place to look for files that can be deleted when disk space gets cramped.

Creating a RAM disk

The device RAM: creates files in memory. RAM: implements a filing system in memory (a "RAM disk") that supports any of the normal filing commands. For instance, if you copy all of the system commands on to the RAM disk, you could do directories and copy files or disks without constantly being prompted to insert your Workbench disk. This is especially helpful if you only have one disk drive. However, you do trade extra memory for this convenience, as the RAM disk requires about 128K if you copy the entire command library. To do this, type the following commands:

```
MAKEDIR ram:c
Copy sys:c TO ram:c
Assign C: RAM:C
```

You could then type **DIR RAM: OPT A** to look at the output. It would include the directory "c" (DIR lists this as c(dir)).

Let's assume that you want to copy a half dozen disks and you only have one disk drive. Copy just the Diskcopy program into RAM with the following sequence:

```
makedir ram:c
copy sys:c/cd to ram:c
copy sys:c/diskcopy to ram:c/diskcopy
copy sys:c/assign to ram:c
assign c:ram:c
cd ram:
```

This would leave you more free memory and require fewer disk swaps for each copy. Of course, only the commands copied to RAM:c are available until you return to normal with:

```
assign c:sys:c
```

You can create a much more useful RAM disk by transferring the most used AmigaDOS commands to RAM:. The following RAM disk will require 48K and may require as many as four disk swaps with single disk drive systems when copying disks. If you delete the LIST command, you can save 8K and definitely duplicate disks with only three disk swaps.

Although you can type this in manually each time it is needed, it would be more practical to create an EXECUTE file by typing this in with ED, the full screen editor (see Chapter Five).

```
makedir ram:c
cd c:
copy cd to ram:c
copy dir to ram:c
copy copy to ram:c
copy diskcopy to ram:c
copy format to ram:c
copy info to ram:c
copy rename to ram:c
copy list to ram:c
copy execute to ram:c
copy delete to ram:c
copy assign to ram:c
copy echo to ram:c
assign c: to ram:c
echo "Command library in RAMdisk."
```

Figure 10—RAM disk Execute file

Running Commands In the Background

AmigaDOS can run one or more commands in the background. For each background command, create a new CLI window with the RUN command. The new CLI has a higher number prompt (2>) and a slightly lower priority. You can move the windows around so that you can see both CLIs simultaneously, which is very useful. For example, you can examine the contents of your directory and send a copy of a text file to the printer at the same time:

```
RUN TYPE textfile to PRT:
LIST
```

RUN creates a new CLI and carries out the printing task while your directory files are listed to the original CLI window. One RUN command can carry out several commands sequentially. It is terminated by a carriage return at the end of the line, but you can insert a plus sign (+) at the end of the line just before the carriage return to indicate that the RUN command line continues on the next line.

You can instead use the NEWCLI command to create another CLI window. You can move the windows around so you can use both windows simultaneously.

Activating the Command Line Interface (CLI)

There are three methods of activating the CLI (Command Line Interface). The first is by double-clicking on the CLI icon in the Systems Window of your Workbench. The second, which is effective but not recommended, is to interrupt the startup sequence on the Workbench just as text appears on the screen. You can interrupt the command sequence that loads Workbench by hitting CTRL-D. The third and recommended method is by modifying the startup sequence on the Workbench disk so that you start directly in the CLI mode rather than in Workbench's windowing environment. If you do this, we suggest that you use a copy of your Workbench disk and follow the steps carefully. This operation will also give you a chance to become familiar with the screen editor called ED.

First get the Workbench running through the usual icon oriented system. Double-click the Systems drawer to reveal the CLI icon in that window. If there is no CLI icon, return to Preferences and turn it

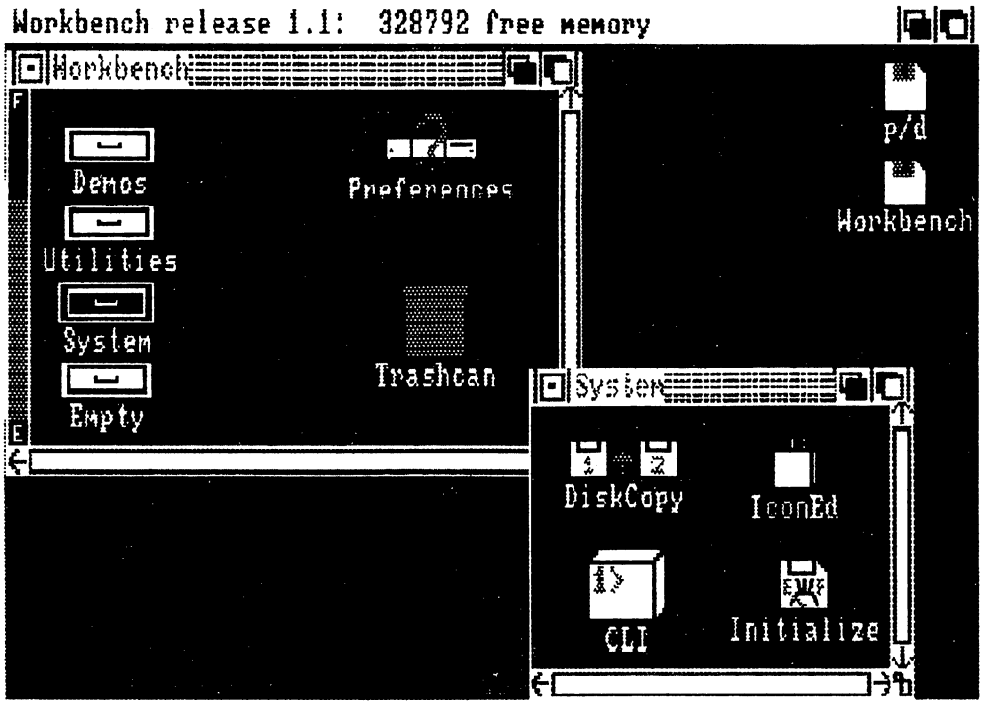


Figure 11—The Workbench and Systems Drawer

on. Remember to save it when you exit Preferences. Now double-click the CLI icon to open the CLI window. The cursor, just after the 1> prompt, is awaiting your command. Type:

ed s/startup-sequence.

This instructs the screen editor to edit a file called "startup-sequence" in a directory called "s." You will see the following:

```
echo "Workbench disk. Version 1.1"
echo " "
echo "Use Preferences tool to set date"
echo " "
LoadWb
endcli>nil:
```

You need to delete these last two lines: the next to last line loads Workbench, and the last line ends the CLI. You are in the immediate command mode when you enter the editor. Use the Down Arrow key to position the cursor at the beginning of the "LoadWb" line. You could use the Delete key to delete the characters on the line, but it is faster if you use CTRL-B to delete the entire line. Notice that the line

below moves upward. You could also delete the line with CTRL-Y. The difference is that CTRL-Y just deletes the characters, not the carriage return, leaving a blank line.

When you are finished, hit the ESC key to enter the extended command mode. An asterisk appears at the bottom of the screen with a cursor beside it. Type an "X" if you are ready to save the new text file on disk. Hit Return. Your text is now saved. You can always exit the editor without saving text by typing a "Q" and a carriage return. The editor will ask you to confirm your intention. If you type nothing and press Return, you will re-enter the editor.

Once you have changed this startup sequence, the CLI mode comes up automatically when you insert your Workbench disk.

Many users also modify their computer's system parameters like screen colors, shape of cursor, etc. in the Preferences program on their Workbench disk. These parameters are stored in a file in the "Devs" directory called "system-configuration." If you have several different start-up disks, you can save time by copying this file on to all of them.

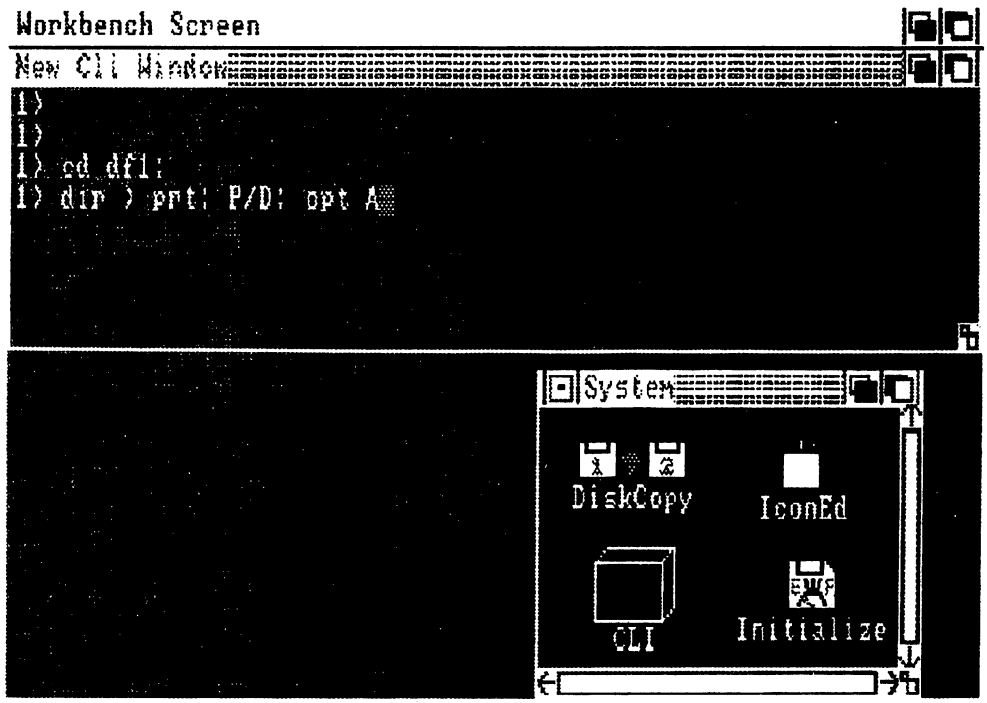


Figure 12—Inputting commands in the activated CLI window.

AmigaDOS Command Introduction

Beginners will find that the best way to master AmigaDOS is through practice. Don't just read the material in the next few sections, follow the examples by typing them in. While many of the CLI commands are explained in detail in the reference section of this book, less than two dozen of them are really useful to the novice or intermediate level user. They are listed in Figure 13.

Command	Purpose
DISKCOPY	Copy a disk
FORMAT	Format a new disk
INSTALL	Create a CLI disk
RELABEL	Rename a disk
INFO	Get information about a file system
DIR	Look at a disk directory
DIR OPT A	Look at all of the files on a disk
LIST	Get information about files
CD	Change the current directory
MAKEDIR	Create a new directory
ASSIGN	Tell AmigaDOS where to look for things
COPY	Copy files
DELETE	Delete a file
PROTECT	Prevent a file from being deleted
RENAME	Rename a file
TYPE	Type a text file to the screen command
DATE	Set the date and time
SAY	Convert text to speech
NEWCLI	Open a new CLI window
ENDCLI	Close an existing CLI window
< and >	Redirect the input or output of a command.

Figure 13—AmigaDOS commands beginners need most

CLI Practice Session

The following session should familiarize you with all of the AmigaDOS commands listed in Figure 13. First, we assume you are still in the CLI mode after modifying the system startup sequence on your Workbench disk, or that you can enter the CLI mode from the CLI icon in the Workbench window. All of the commands you must enter are indented. Type them exactly as shown, being careful to leave spaces where indicated. For clarity here, we printed everything in uppercase, but you may use either upper- or lowercase, AmigaDOS doesn't care.

CHANGING DIRECTORIES OR DEVICES—The CD command tells the system which directory or device to make current. This is important when you wish to move from one directory or sub-directory to another without typing a long pathname. It is also necessary when you change disk drives or use a RAM disk. If you are using a two drive system with Workbench or another systems disk in the internal drive, you must always tell AmigaDOS you wish to access the disk in DF1 by typing:

CD DF1:

Remember to leave a space between CD and DF1:.

This also applies whenever you change disks in a drive, regardless of whether you are currently using that drive. AmigaDOS must be told that you have switched disks.

LOOKING AT THE DIRECTORY—One of the first things that you might want to do with your Amiga is catalog the Workbench disk. Type:

DIR

You can stop items from scrolling off the top of the CLI window by pressing the space bar or Tab key. Either the Return or Backspace key will restart the listing. The right mouse button will also suspend the scrolling. If your CLI window isn't full screen, now would be a good time to stretch it by dragging the bottom left gadget.

You can also catalog your Workbench disk by specifying the drive number DF0: or its device name, SYS:.

DIR DF0:

or

DIR SYS:

Notice that there are a few files, like Preferences, but most of the catalog shows sub-directories. Request a directory of the sub-directory called "fonts(dir)" by typing:

DIR FONTS

Or change the current directory from the root directory to the fonts sub-directory and then display the directory:

CD FONTS
DIR

Most disks, including your Workbench disk, have a hierarchical tree shaped chain of directories and sub-directories one or more layers deep, all branching from the root directory (top level). If you wish to return to the root directory, type:

CD:

The colon (:) after the CD command returns you to the root directory. Typing CD by itself just shows you the name of the current directory.

Now let's see the directory of "devices," which is in the root directory. Type:

DIR DEVS/PRINTERS

Remember to use slashes (/) to separate each directory, sub-directory, and file name. You may use a maximum of 100 characters in a directory description (pathname). To be safe, pathnames should contain no more than sixty characters. Remember, you can shorten the pathname by using the CD command to switch to one of the sub-directories along the pathname.

DIRECTORY OF ALL FILES ON DISK—To display a catalog of all disk files on the screen, type:

DIR OPT A

to output the catalog to a disk file:

DIR > DISKDIR OPT A

Note that the > symbol in the line redirects the output to the disk file. The file name *must* come before the list of options for the DIR command. Now you can use ED, the CLI screen editor, to view the file. Type:

ED DISKDIR

DIRECTORIES OF OTHER DISKS—Directories of disks other than the system-startup disk can be obtained easily. In a single drive system, put Workbench in the drive and type:

DIR?

AmigaDOS responds with:

DIR,OPT/K:

Since AmigaDOS has already loaded the DIR command from your CLI disk, you can replace the Workbench disk with the one you wish to catalog. Then press Return to obtain the directory.

Owners of two-drive systems should change directories to the external drive and then request the directory:

CD DF1:
DIR

DIRECTORY BY DISKNAME—You can also catalog a disk by diskname. Place your "Extras" disk (the one that contains Microsoft BASIC) in the external drive if you have two drives, otherwise just keep it handy. Now type:

DIR EXTRAS:

If you have two drives, AmigaDOS will automatically shift to the second drive to obtain the catalog of its root directory. Single drive owners, however, will be prompted to insert a disk with Volume name "Extras." Once the Workbench is replaced by the "Extras" disk, the catalog will appear. Note that unless you have created a RAM disk for the AmigaDOS commands, you can't avoid the disk swap as the Workbench must be in drive DF0 to access the DIR command. The directory is as follows:

```
Trashcan(dir)
t(dir)
BasicDemos(dir)
Tfiles(dir)
.info
Amiga Tutor.info        Amiga Tutor
Amiga BASIC.info        Amiga BASIC
Disk.info               Basic Demos.info
Trashcan.info           Tfiles.info
```

Figure 14—Root directory of Extras disk

If you have created and saved any Amiga BASIC programs they will appear in the root directory. Since each program is saved with its own icon, files will appear in pairs. For example:

factorial	factorial.info
TV grid	TV grid.info

Figure 15—Each file you save has a ".info" counterpart

INFORMATION ABOUT FILES—The LIST command provides an extended list of all files held in the current directory or in the device specified. For example, if you have created a RAM disk or temporarily moved something into RAM:, you can obtain a list of the files there:

LIST RAM:

You can list by disk drive device, too:

LIST DF0:

With the "Extras" disk still in the drive type:

LIST EXTRAS:

Single drive owners will be prompted to replace Workbench and then replace the "Extras" disk. You will see the following:

```
Trashcan          Dir     rwed              21:57:33
.info             66      rwed    Today     21:25:32
AmigaBasic        93156   rwed    Today     18:28:51
  .               .                           .
  .               .                           .
  .               .                           .
Disk.info         306     rwed    30-Oct-85 18:31:26
Trashcan.info     442     rwed    30-Oct-85 21-55-36
Amiga Tutor       83208   rwed    30-Oct-85 18:37:54
  9 files - 4 - 383 blocks used - 18:38:21
```

Figure 16—Information about files on "Extras" disk

Directories have the word "Dir" in the second column instead of the file length. The letters "rwed" refer to the protection status of the file or directory. The letter "r" means that you can read it, "w" means that you can write to it, "e" means that you can execute it, and "d"

means that you can delete it. Currently only "d" is implemented. If the "d" doesn't show up in the "rwed" column for a file name, AmigaDOS won't delete that file when you execute the DELETE command.

OBTAINING INFORMATION ABOUT THE FILE SYSTEM— The INFO command is used to find out what disks (Volume name) are active and how many drives are attached to your system. It also tells you how much used and free space is on your disks, and whether the disks are read-only or read-write.

Dual disk drive owners need only type INFO to obtain the information, assuming Workbench is still in drive DF0.

INFO

Mounted Disks

Unit	Size	Used	Free	Full	Errs	Status	Name
DF01:	880K	1471	287	83%	0	Read/Write	Extras
DF0:	880K	1310	448	74%	0	Read/Write	Workbench
RAM:	83K	164	0	100%	0	Read/Write	

Volumes Available

Figure 17—System Info

Note that the status of the RAM disk is not quite accurate in the free and Full columns. Since memory for the RAM disk is dynamically allocated it will always appear full.

Single disk drive owners can obtain information about a disk other than Workbench by typing:

INFO ?

AmigaDOS responds:

none

AmigaDOS has loaded the INFO command from your CLI disk and shows you the template for the command. The response "none" means that you only have to press the Return key once you have replaced Workbench with the disk whose information you want to see.

COPYING A DISK—Disks can be easily duplicated using the DISKCOPY command with either single- or dual-drive systems. The advantage with dual-drive systems is that disks don't have to be swapped during the copy process. Single drive systems with 512K of memory require three passes, or sets of disk swaps, to copy a disk, while 256K single-drive computers require six passes. The reason is that the 880K bytes of disk data is approximately three or six times as much data as your copy buffer can accommodate.

For a single-drive system, type:

DISKCOPY FROM DF0: TO DF0:

For a dual-drive system, type:

DISKCOPY FROM DF0: TO DF1:

Follow the instructions as they appear. In single-drive systems, AmigaDOS instructs you to place the master (FROM) disk in the drive, then, as the copying proceeds, it prompts you to replace it with the copy (TO) disk. Keep swapping the master with copy until all of the disk has been duplicated. The cylinder (track) counter indicates your progress.

Copying is much easier with two drives. Place the master (FROM) disk in the internal drive and the copy (TO) disk in the external drive. The copying takes place automatically without prompts.

Never remove any disk while the drive light is on, doing so can ruin the disk. Even after all eighty tracks have been copied, you still have to wait several seconds because the system pauses for a three second timeout while it flushes its buffer and updates the bit map on the disk. If you remove the disk during this timeout, the disk will be marked as invalid.

DISKCOPY does not verify the track data that it writes to the copy disk against the data in its track buffer. You won't find out about physical flaws (bad sectors) on the destination disk until you either attempt to make a another copy of your duplicate or have trouble loading one of the files. You can avoid this by formatting the disk first.

FORMATTING A DISK—It is often necessary to have a blank prepared disk to hold data files for applications programs such as word processors, spreadsheets, databases, graphics, and music programs. The FORMAT command prepares such a disk by

segmenting it into tracks and sectors. It also places a "root block" in the center of the disk surface. Formatted disk are verified track by track. Although they are ready to accept data, they are not bootable. That means you can't start the system with one of these disks.

Disks can be formatted in either drive. You only have to give AmigaDOS the drive number and choose a Volume name for the disk.

FORMAT DRIVE DF0: NAME "MUSICFILES"

Then follow the prompts.

RELABELING A DISK—If you are not satisfied with a disk's Volume name after copying or formatting it, you can change it with the RELABEL command. For example, you could change the preceding disk name "MusicFiles" to "DataFiles."

RELABEL MUSICFILES: DATAFILES

You will be asked to insert the correct disk if it isn't already present.

MAKING A DISK BOOTABLE—A bootable disk is one that you can use to start up your Amiga following the Kickstart process. You can change a formatted disk into a CLI disk with the INSTALL command. This new CLI disk will contain a CLI interpreter and nothing else, not even a Trashcan as supplied when you format via the Workbench.

If you have a single drive system, type:

INSTALL ?

AmigaDOS responds:

Drive/A:

Remove your Workbench disk and insert a formatted disk. Then type:

DF0:

AmigaDOS copies boot sectors to the disk. *Remember, especially with single-drive systems, to use the question mark after the INSTALL command.* If you don't AmigaDOS will try to do an install on the disk currently in drive 0.

If you perform a system reset (CTRL-Amiga-Amiga), you will

enter the CLI mode rather than the Workbench when the system reboots. The CLI prompt appears, but the disk contains only the CLI interpreter, and none of its commands.

CREATING A NEW DIRECTORY—The MAKEDIR command allows you to create a new directory within the current directory. If you are in the root directory, you make a sub-directory, and if you are in a sub-directory, you make a sub-sub-directory within it.

Assuming you haven't reset the computer and you booted the system with the Workbench disk, you can create a new directory (drawer) named "Data" on your disk named "DataFiles."

CD DATAFILES
MAKEDIR DATA

If you issue the DIR command you will see an entry for:

DATA(dir)

COPYING FILES—With the COPY command you can copy files in any directory to another disk, or even move files from one sub-directory to another on the same disk. The best way to copy is from one drive to a different drive on a dual-drive system. The format is:

COPY FROM DF0:*sourcepath TO DF1: destinationpath*

or

COPY DF0:*sourcepath* DF1:*destinationpath*

You'll still have to do a disk swap if the file you are planning to copy is not on the System Disk, since the System disk must be present so AmigaDOS can retrieve the copy program from the command directory. Be sure to include the disk's Volume name in the pathname or AmigaDOS won't find the file. For example, if you wanted to copy "Amiga Tutor" from your "Extras" disk to your "DataFiles," disk you would enter:

COPY FROM "DF0:EXTRAS/AMIGA TUTOR" TO DF1:

This would copy "Amiga Tutor" to the new disk's root directory. The quote (") marks surround the file name because there is a space within it.

It is not practical to copy files from one disk to another in the same drive because it takes an innumerable number of disk swaps. It appears that the COPY program copies one sector per pass. Chances

are that you will mess up your disk either because you insert the wrong disk at the wrong time, or become impatient and pop a disk too soon. Hopefully, future releases will remedy this problem.

If you have only one disk drive the best way to copy files is to copy the file on the disk in drive DF0: to RAM, and then from RAM to a new disk replaced in drive DF0:. Owners of an external drive (DF1) may also wish to use this copy method since none of the AmigaDOS command library is in memory so it's constantly read from the systems disk in drive 0.

COPY DF0:" AMIGA TUTOR" TO RAM:

Change disks

COPY "RAM:AMIGA TUTOR" TO DF1:

Since copying files is much more complex on a single-drive system, you should create a RAM disk by copying several of the files from your system disk into memory. The procedure discussed earlier (page 28) sets up a RAM disk by copying the necessary files to it. You copy data files by copying them to the RAM disk, changing your directory to the RAM disk, then copying the files from the RAM disk to a destination disk.

With the Workbench or CLI disk in the internal drive, first create a RAM disk with the following commands:

COPY DF0:C/CD RAM:
COPY DF0:C/COPY RAM:
CD RAM:

Now insert the source data disk ("Extras") into the internal drive. Type:

COPY "DF0:AMIGA TUTOR" RAM:

Remove the source disk and insert the destination disk. Type:

CD RAM:
COPY "RAM:AMIGA TUTOR" DF0:

ASSIGN—TELLING AMIGADOS WHERE TO LOOK FOR THINGS—Now that you have a CLI disk with the entire executable command library, it would be best to assign the command library to the new disk. If you don't use the ASSIGN command you will be required to insert the Workbench disk to access commands.

CD DATAFILES:
ASSIGN C: DATAFILES:C

Now all AmigaDOS commands will be obtained from the "Command" (C) directory on this disk.

DELETING AND PROTECTING FILES—Let's create several disposable files on your CLI bootable disk named "DataFile". The easiest way to create a quick, disposable file is to copy a directory to a file. You can route the directory of any disk to a file called "Dirfile" with the redirect command symbols (> and <).

DIR > DIRFILE

The PROTECT command protects or unprotects a file from being deleted accidentally. You can protect "DIRFILE" from being deleted with the following command.

PROTECT DIRFILE

If you LIST "DIRFILE" you will see that all of the protected status letters are blank. Now if you try to delete it:

DELETE DIRFILE

AmigaDOS responds:

Not deleted - file is protected from deletion

To change the protection status type:

PROTECT DIRFILE D

or

PROTECT DIRFILE RWED

The DELETE command lets you erase unwanted files. The advantage to deleting a file is that it frees up disk space for re-use by AmigaDOS. *Always be sure that you really want to delete the file, because once you do so, it is impossible to retrieve it.* Let's delete the file "DIRFILE" which you should have just unprotected. If you haven't unprotected it, do so now. Now delete "DIRFILE."

DELETE DIRFILE

If you try to open "DIRFILE" now, AmigaDOS will respond:

Can't Open Dirfile

This indicates that the file is not on the disk, which should be obvious since you just deleted it.

TYPING A TEXTFILE TO THE SCREEN—You can use the TYPE command to display the contents of a text file on the screen. If it is a long file and you wish to stop the scrolling, press the space bar. To restart it press the Backspace key. Or use the space bar and then Return key. To end the TYPE command, use CTRL-C.

Type a file now. We'll use a "Dirfile" in our example. You can use any data file.

TYPE DIRFILE

Watch the text scroll by, or stop and start it using the keys mentioned above.

SETTING DATE AND TIME—The Amiga's internal clock is not battery powered, so you have to reset it each time you turn on the computer. To set the time and date, type:

DATE 4:30:00 23-Feb-86

Advance the date by one day by typing

DATE TOMORROW

Sometimes when you LIST a directory, you see dates that say "Future." This occurs when someone has transferred a file to your disk that has a later creation date than your disk's. This can happen when you or that other Amiga user don't always set the system time and date before working with files.

RUNNING A PROGRAM—You can run a program by just typing its name. But to run a program and still be able to enter CLI commands, you must use RUN followed by the program name. For example,

CLOCK

will show the clock. However, you will have to close the clock if you wish to continue using the CLI. But if you type

RUN CLOCK

you can keep track of time and still execute CLI commands.

SAY COMMAND—The SAY command uses AmigaDOS's speech capabilities. Type a phrase with no quotes after the SAY command and the computer will output fairly understandable speech. Try:

SAY HELLO I AM A FRIENDLY COMPUTER

Parameters can be set to change the voice from male to female, the rate of word flow from very slow to fast, and the pitch from a very deep voice to one that is high pitched. Parameters are set by preceding the phrase with a minus sign (-) followed by the parameter. For example, pitch can be varied with a value (65-320). To get a deep voice type:

SAY -P65 HELLO I AM A FRIENDLY COMPUTER

For a female voice try:

SAY -P130
SAY -F HELLO I AM A FRIENDLY COMPUTER

Other parameters are discussed in the reference section of this book.

You may notice that the disk drive light turns on as the computer accesses both the DOS SAY command and the text to speech translator library. If you plan to play with the speech generator extensively and want to do so without the disk access delay, move the necessary files on to a RAM disk. Type:

MAKEDIR RAM:LIBS
COPY LIBS:TRANSLATOR.LIBRARY TO RAM:LIBS
COPY C:SAY TO RAM:
CD RAM:
ASSIGN LIBS: RAM:LIBS

This sequence will take up approximately 24K of RAM.

Entering the SAY command by itself without parameters opens up two windows: an input window and a phoneme window. Text entered in the input window is converted to phonemes in the phoneme window before speech is outputted through the speakers.

CREATING A NEW CLI—Since AmigaDOS is multi-tasking, it can run one or more command line interpreters at the same time. Each of these CLI windows operates independently, each issuing its own commands. One might be requesting a disk directory, while another is downloading information from an on-line database through the serial port. Create a new CLI:

NEWCLI

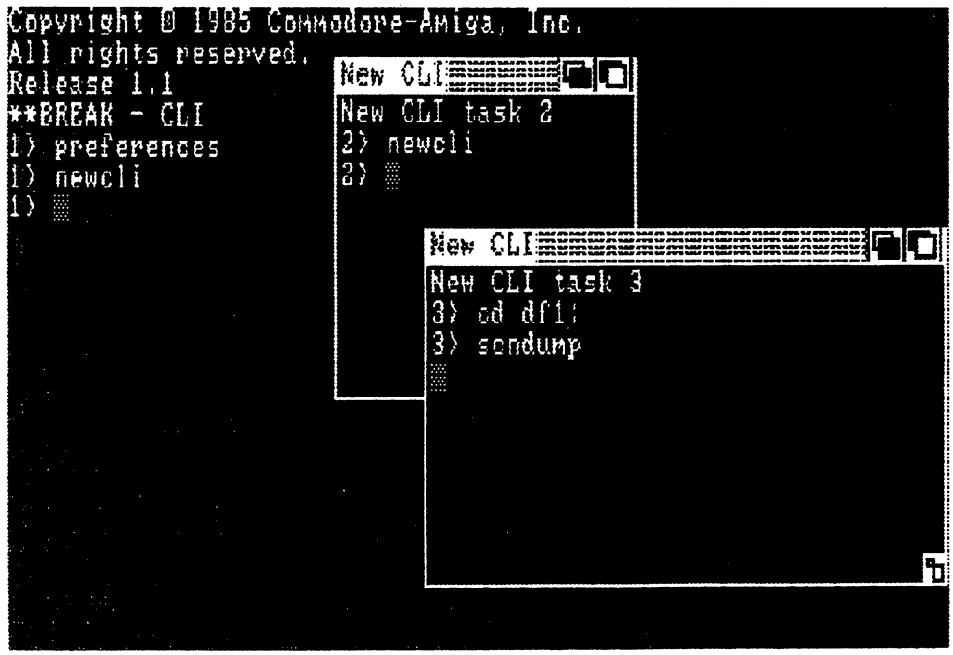


Figure 18—CLI Windows

This command opens a separate window with a prompt that identifies the current task or process. If the first window has a 1> prompt then the second will have a 2. prompt. Although both can produce output at the same time, only one can accept input at a time. The window that is active has a solid title bar. Let's demonstrate the possibilities. Click in window one with the mouse, then type:

DIR >PRT: DF0: OPT A

Then quickly click into window two and type:

INFO

CLOSING THE CLI—You can terminate a CLI task by typing:

ENDCLI

or

LOADWB

to get back to the icon driven Workbench. Use ENDCLI if you can return to the Workbench yourself. Use LOADWB if you modified the startup sequence and did not enter from the Workbench.

HARD DISKS—If you decide to buy a hard disk (device DH0:) you'll need to assign all of the system files to it as follows:

```
ASSIGN SYS: DH0:
ASSIGN C: DH0:C
ASSIGN L: DH0:L
ASSIGN S: DH0:S
ASSIGN LIBS: DH0:LIBS
ASSIGN DEVS: DH0:DEVS
ASSIGN FONTS: DH0:FONTS
```

Chapter 3

CLI COMMAND LIBRARY

ASSIGN

Assign is a shortcut, used to create a new device name to be associated with a directory. By calling on the "assigned" name, the user can generally save time accessing files in a sub-directory and not care about the current directory.

Example: **ASSIGN**

Lists current logical device names.

Example: **ASSIGN C: RAM:C**

Changes the Command directory "c" to the RAM disk. In this case, any CLI commands you intend to use would *first* have to be copied to the directory "c" in the RAM disk.

Example: **ASSIGN C: SYS:C**

Removes the assignment of RAM:C made in the previous example.

Example:

ASSIGN QUICK: DF1:DEMOS/ANIMATIONS

Creates a new device, "Quick:" and assigns it to the sub-directory "Animations." Accessing a file named "Boing!" in that directory is now is simple as typing "quick:boing!", which is the same as entering "df1:demos/animations/ Boing!".

Example: **ASSIGN QUICK:**

Cancels the assignment of "quick:" as used in the previous example.

BREAK

Break is used to stop a process in progress. Normally this is done by pressing CTRL-C on the keyboard. The BREAK command allows the user to change this to include any of the keys from CTRL-C to CTRL-F. To set the break flag for a process you must know its number. In a CLI window, this is generally the number that appears as the prompt. If you are unsure of the process number, try the STATUS command first. BREAK only gets the attention of the task that is running, it cannot force it to stop.

Example: **BREAK 2 C**

or

BREAK 2

Sets the attention/break flag of process 2 to CTRL-C.

Example: **BREAK 2 C E**

Sets process 2 break key to CTRL-C and CTRL-E.

Example: **BREAK 3 ALL**

Sets the attention flag of process 3 to CTRL-C through CTRL-F.

> **NOTE:** Some processes may not be aborted by using their break flag, such as aborting a printout when a printer isn't connected.

CD

Allows the user to set the current directory.

Example: **CD**

Displays the current directory.

Example: **CD DF1:**

Change the current directory to drive 1.

Example: **CD SYS:DEVS/PRINTERS**

Changes the current directory to "printers," a sub-directory of "devs" on the system disk. Remember that using "SYS:" or a specific disk's name such as "Workbench:" sets the current directory to that specific disk regardless of which disks are in the drives.

Example: **CD /**

Changes the current directory "up" one level.

Example: **CD //**

Changes the current directory "up" two levels.

Example: **CD :**

Changes the current directory to the root directory. (Goes all the way to the top of the directory tree.)

Example: **CD RAM:**

Changes the current directory to the RAM disk.

COPY

Makes a copy of a file or files, or an entire directory. The copy can be made from or to any logical device. Naturally, trying to copy from a device that doesn't send data, such as the printer won't work. The two options of this command are "all" and "quiet." "All" includes the contents of any sub-directories in the copy. "Quiet" tells the system not to print the names of the files on the screen as they are copied. You can also use wildcard patterns in specifying directories and filenames. For details on patterns, see Appendix A. The keyword "TO" is optional.

Example: **COPY DF0: TO DF1:**

Copies *only* the *files* in the root directory on drive 0 to drive 1.

Example: **COPY DF0: TO DF1: ALL**

Copies all files and all directories on drive 0 to drive 1. This is not quite the same as using DISKCOPY, since any files that existed on drive 1 would remain. Also, system disks such as Workbench copied with COPY will not boot unless you INSTALL it as well.

Example:
COPY DRAFT TO DF0:LETTERS/DRAFT
or
COPY DRAFT TO DF0:LETTERS

Makes a copy of the file "draft" from the current directory to the file "draft" in the directory "letters" on drive 1. COPY looked first for a directory called "draft" in the "letters" directory on drive 1. If a "draft" directory existed, the file "draft" would have been copied there. This example assumes that it did not exist and therefore AmigaDOS would use "draft" as the name of the destination file. If the destination filename is not specified, then the original name is used.

Example:
COPY DRAFT TO DF0:LETTERS/FINAL

Makes a copy of "draft" from the current directory to the directory "letters" on drive 0 with the new filename "final". As in the previous example, this assumes that "letters" does not contain a sub-directory named "final". If a file named "final" existed it would be overwritten with the contents of "draft". No warning would be given.

DELETE

Lets you delete files or directories you no longer want. DELETE accepts wildcard patterns (see Appendix A) and also the options "All" and "Quiet." Directories must be empty to be deleted unless "all" is used.

Example: **DELETE RAM: ALL**

Erases the contents of the RAM disk.

Example: **DELETE JUNK**

Deletes the file "Junk" from the current directory, or deletes the directory "Junk" if it is empty.

Example: **DELETE JUNK ALL**

Same as above, except if "Junk" is a directory, then it and its contents will be deleted.

*Example:***DELETE #?.BAS**

Erases all files ending with ".bas" from the current directory.

DIR

Displays an alphabetized list of the contents of a directory. Options available are "OPT A" to include a list of any sub-directories and their contents, "OPT D" lists only directories, "OPT I" lists the contents in interactive mode. The interactive mode allows you several options while the listing is in progress. The listing pauses after each item and waits for you to respond. Your options after the "?" prompt are:

Return Key—Do nothing, continue listing.
Del—Delete the item if it is a file or an empty directory. User must type "DEL".
T—Type the file to the screen. (Use CTRL-C to abort.)
E—Enter, if the item listed is a directory. Listing will continue with its contents.
B—The opposite of "E". Use it to get out of a sub-directory and go back up a level in the directory.
?—List these options.
Q—Quit the listing.

Example: **DIR :**

Lists contents of the root directory.

*Example:***DIR DF1:WORK/SOURCE**

Lists the contents of the directory "Source".

Example: **DIR >PRT: PROGRAMS: OPT A**

Lists to the printer the entire contents of a disk named "Programs," regardless of what drive it is in.

> **Advanced Users Note:** "Programs:" could also have referred to the ASSIGNed name of a directory rather than a disk name.

Example: **DIR DF0: OPT I**

Lists the disk in drive 0 in interactive mode.

Example: **DIR SYS: OPT D**

Lists the directories on the system disk.

Example: **DIR >PRT: DF0: OPT D**

lists only the directories on the disk in the internal drive to the printer.

DISKCOPY DISKCOPY makes a copy of one disk to another. Both disks must be identical (3½ to 3½ etc.). DISKCOPY will format the destination disk automatically. You will also be told when to insert the source and destination disks so you can make copies of disks other than the system disk.

Example: **DISKCOPY DF0: TO DF1:**

Copies a disk in drive 0 onto a disk in drive 1.

Example: **DISKCOPY DF0: TO DF1: NAME BACKUP**

Copies the disk in drive 0 on to a disk named "Backup:"

Example: **DISKCOPY DF0: TO DF0:**

Makes a backup copy of a disk using only one drive. AmigaDOS will tell you how many times you will have to swap disks to make the copy (depending on available memory) and prompt you each time you have to swap disks.

ECHO Prints a message on the screen (unless another device is specified). If there are spaces in the message then the entire message must be enclosed in quotes. Echo is generally useful as part of a sequence of commands as might be used with RUN or EXECUTE.

Example: **ECHO "HELLO"**

or

ECHO HELLO

Prints the word "HELLO" on the screen.

Example: **ECHO >PRT: "THIS IS A COMMENT"**

Prints the message in quotes on the printer.

ED The text editor. This is really a simple word processor, useful for editing documents, programs, execute files, etc. ED is very similar to "VI," which is available under most UNIX systems. An entire section of this book has been devoted to this command. Refer to it for details.

EDIT A sequential, line based editor similar to ED. We have found it to be more or less useless, however, you can refer to the section in this book that discusses it in detail.

ENDCLI Ends the current CLI window. You should only do this when you have some means of returning to the Workbench or another CLI window. Otherwise you will have closed the only interface between you and the computer.

Example: **ENDCLI**

EXECUTE Takes a text file that is composed of one or more CLI commands and executes the commands as if they were entered from the keyboard. If you find yourself typing a specific series of commands frequently, you could put them in a text file then use EXECUTE to have them executed as a "batch" or "macro." The text editor ED is very handy for creating EXECUTE files. An EXECUTE file is very much like a small program. Advanced users can even pass parameters such as filenames to their command sequences from the EXECUTE command. The break flag of an EXECUTE file is set to CTRL-D. The examples given go from basic to advanced techniques using this command. If you are an experienced AmigaDOS user, or familiar with "shellscript" on UNIX systems, the following information may be of use to you. Otherwise, we strongly suggest that you skip the following discussion and read only the first example.

AmigaDOS contains commands that only have meaning in an EXECUTE file. They exist to give the EXECUTE command greater flexibility and power. These commands are: FAILAT, IF, ELSE, ENDIF, LAB, SKIP, and QUIT. A description of each follows. Refer to the examples to see how they are actually used in an EXECUTE file.

FAILAT—Sets the fail limit of the command sequence. When a command fails, it returns a value greater than zero. The higher the value, the more serious the error encountered. *Do not confuse this number with the error type!* Generally, the value returned is 20, although 5, 10, and 20 are all very common. (See also QUIT below.) FAILAT is set to 10 unless you change it. If the FAILAT value is equaled or surpassed, the entire command sequence will stop.

IF...ELSE...ENDIF—IF lets you specify that some commands should be executed only if certain condition(s) are true. IF tests for a specific condition, and the command(s) following it are only executed if the condition specified is true. If the condition is not true, then the commands after IF are skipped and EXECUTE checks for an ELSE statement. The commands following ELSE are only executed if the commands after IF were not. ENDIF is used to mark the end of the entire sequence of conditional commands. *For each IF there must be a corresponding ENDIF.* ELSE is optional, and must be between the IF and its corresponding ENDIF. IF...ELSE...ENDIFs may contain (nest) more IF...ELSE...ENDIF statements. An IF statement can make six tests:

EQ—Compares two strings. It's useful for testing the user's input from the command line. The strings will be matched ignoring case.

Example: **IF <A> EQ "HELP"**

Tests to see if "HELP" was entered as a parameter by the user.

> **NOTE:** Version 1.0 cannot compare an empty argument. Thus you CANNOT use the statement:
>
> **IF < EQ""**
>
> to test if the user entered the parameter for <A>. See examples for a way around this.

EXISTS—Tests whether a filename exists.

Example: **IF EXISTS SYS:MYLIB**

Tests if the file "mylib" exists on the system disk.

NOT—Used to test if the opposite condition is true.

Example: **IF NOT EXISTS SYS:MYLIB**

The opposite of the previous example. The command(s) on the lines following would only be executed if "mylib" does not exist.

WARN—Tests for a minor command failed error (>=5).

ERROR—Tests for a command failed error value (>=10).

FAIL—Tests for a more serious command failed error (>=20).

> **NOTE:** For WARN, ERROR, and FAIL to work, the FAILAT value (see above) must be greater than their threshold values (5, 10, and 20 respectively). Otherwise the command sequence will terminate from reaching the FAILAT value and the IF statement will be meaningless.

LAB—Used to label a section of the EXECUTE file. It is used so that sections of the command file may be skipped over if necessary.

SKIP—Tells AmigaDOS to skip forward to a specified label.

Example:

```
SKIP HOWDY
ECHO "THESE TWO LINES"
ECHO "WILL NEVER BE PRINTED"
LAB HOWDY
ECHO "HI, WHAT KEPT YOU."
```

QUIT—Terminates the command sequence and returns with the error code specified.

Example: **QUIT 20**

stops the command sequence and returns an error code of 20.

EXECUTE Directives

In addition to the preceding commands, EXECUTE also allows certain directives to identify input passed to it from the command line. All directives begin with a period (.).

.key

or

.k

This is the most important directive. It is used to specify and identify parameters entered on the command line for the EXECUTE file. *It must be the first statement in the command file.* Only one ".k" statement is allowed. All parameters to be used must be specified in the first line.

Essentially, ".key" (don't forget the leading period) sets up a name for substitution text.
For example, the EXECUTE file named "typit" contains:

```
.k file
TYPE <FILE>
```

The user types:

```
Execute typit letter
```

AmigaDOS takes the parameter "letter" from the command line and substitutes it wherever "file" appears in the command file. Consequently, "letter" is printed to the screen.

Identifying additional parameters is as simple as:

```
.k file1,file2,a,b,c
```

There are three options for a parameter:

.k file—No option specified. This should be used if the parameter is not required.

.k file/a—The user must give the parameter. Failure to enter it will give an error and the command file will not be executed.

.k file/k—Not only is this parameter required, but it also must be preceded by its identifying name ("file" in this case), as in:

```
EXECUTE typit file letter
```

.k file/s—This parameter type looks for the identifier "file" on the command line just like "file/k," above but does not look for a parameter to be entered.

The last two probably seem very confusing, but if you are familiar with CLI commands you have probably used their type already. Consider the command to initialize a disk, FORMAT. It requires the form: FORMAT DRIVE DF0: NAME "Mydisk". The command requires you to use the words "DRIVE" and "NAME" as identifiers. AmigaDOS would certainly be smart enough to know "DF0:" is a drive and "Mydisk" is a name, but the form is required so that the user is less likely erase his disk accidentally. Consider the formats of the commands COPY and DELETE. After the filenames are specified you can enter the options "ALL" and "QUIET". The command checks for specific words to use as flags, or more accurately, as switches.

.def—Gives a parameter a default value rather than leaving it unspecified if not entered by the user.

"."—followed by at least one space is used to identify a comment line. Comment lines are not executed, but help make your program more readable. A period by itself is treated the same as a blank line. A semicolon (;) is also used to comment commands.

.dot—Is used if for some reason you want to redefine the period used in these directives.

.bra—Redefine "<" to some other character.

.ket—Redefine ">" to some other character.

.dol or .dollar—Redefine the "$" to another character.

Finally, there is the "$" option, which complements ".def". In some cases you may use a parameter that you do not want to have the same default value every time. Use the "$" option like this:

assem <file> <include$system.lib>

If the parameter for "include" was not specified, then the filename "system.lib" would be substituted for it.

EXECUTE Examples:

1.
The file "RC" contains:

```
makedir ram:c          ; create a sub-directory in the
                         RAM disk
cd sys:c               ; change current directory to
                         save some typing
assign r: ram:c        ; save having to type more long
                         names
copy cd to r:          ; move CD command to RAM
                         disk
copy dir to r:         ; move another command to the
                         RAM disk
copy diskcopy to r:    ; ditto
copy list to r:
copy copy to r:
copy execute to r:
copy assign to r:
copy info to r:
copy delete to r:
copy echo to r:
assign r:              ; R: no longer needed
assign c: ram:c        ; AmigaDOS will look in ram:c
                         for commands
echo "Mini command library now in ram."
```

The file "RC" is now executed from a CLI prompt with:

EXECUTE RC

This command file moves certain CLI commands into RAM and then tells AmigaDOS to look for commands in RAM instead of the system disk. This would limit you to only the commands copied to the RAM disk, but you would not have the delay of loading commands from the disk, or the inconvenience of being told to reinsert the system disk when you change disks (really helpful on single-drive systems).

2.
The file "kr" contains:

```
Assign c: sys:c      ; cancel RAM disk assignment
                       from "RC"
delete ram:c all     ; delete "c" and all its files
```

Execute from CLI with:

EXECUTE KR

Naturally the command file "kr" would have to be in the current directory since no other directory was specified. This example cancels out what was done in the last example and tells AmigaDOS to look for CLI commands back on the system disk.

3.
The file "printout" contains:

```
.k file/a                ; a file must be specified
echo >prt: "<file>"      ; a print filename to printer

date to prt:             ; print date
echo >prt: ""            ; line feed
echo >prt: ""            ; line feed
; This prints the file as background process.
run type >prt: <file>+
echo "<file> printing complete"
```

Execute this file with:

EXECUTE PRINTOUT "FILENAME"

"Filename" is the name of the file you want sent to the printer. This command file will print the filename and date on the printer and then dump the file to the printer as a background process so that you don't have to wait for the printout before moving on to other tasks.

4.
The file "mcli" contains:

```
.key name
.def name "con:0/10/640/190/OtherCLI"
failat 30
newcli <name>
if error
newcli "con:0/10/640/190/OtherCLI"
endif
```

Execute the command file with:

EXECUTE MCLI

or

EXECUTE MCLI CON:20/10/640/80/NEWCLI

This example creates a new CLI window. If the parameters and name of the new CLI window are not given, the default console window size and name will be used. If the user's entry parameters caused an error, the command file checks for it and creates a new CLI window anyway with the name "OTHERCLI".

5.
The file "ckfl" contains:

```
.k file
if exists <file>
list <file>
type <file>
else
echo "<file> does not exist"
endif
```

This example checks to see if the filename that the user has entered actually exists. This prevents a potential error message that might occur if you attempt to LIST or TYPE a non-existent file.

6.
The file "cc" contains:

```
.k file/a,link/s
failat 20
if not exists <file>
echo "<file> not in current directory"
quit 20
endif
lc-i Cl.0:include/<file>.c
echo " "
if <link> eq "link"  ;  switch on?
if exists <file>.o   ;  don't bother if not there
                     ;  to the next blank line is one
                        single line
alink Cl.0:lib/Lstartup.obj+<file>.o
lib=cl.0:lib/lc.lib+ :lib/amiga.lib to <file>
delete <file>.o
echo "<file>.c compiled and linked."
else
echo "no object to link"
endif
else
echo "Compile complete (no run file)."
endif
```

Execute the command file with:

EXECUTE CC HELLO

or

EXECUTE CC HELLO LINK

We warned you this command was for advanced users. This command file is used to compile a program written in "C" with the Lattice C Compiler. If the option "LINK" is specified in the command line, then it will also attempt to link the object file produced into an executable file. In order to call the program "Alink" to link the output of the compiler, two conditions must be met. First, the caller must want the object file linked. This is done by specifying the parameter "link" on the command line. Second, the compiler must have produced an object file for

the linker, "alink," to work on. If the program is linked then the intermediate object file is deleted. The command file assumes the standard "C" format: "file.c" for source code; "file.o" for the object file produced by the compiler; and "file" for the final runable file. The command file takes the filename passed to it and adds the appropriate ".c" or ".o" where needed.

FAULT Displays the error message for a given error code, saving you the trouble of digging out a manual and looking it up.

Example: **FAULT 223**

Displays the error message for Error 223, file write protected.

Example: **FAULT 223 226**

Displays the error messages for errors 223 and 226. Up to ten values may be entered at a time.

FILENOTE Attaches a remark to a file. The filenote will be displayed with the filename when the file is LISTed. The comment may be up to eighty characters.

Example:

FILENOTE PROG.C COMMENT "FIXED TODAY 12/18/85"

When a LIST is used to display the file "prog.c" the comment "fixed today 12/18/85" will appear in the line immediately following.

FORMAT Lets you format a new disk for use. The format process erases anything previously stored on a disk. AmigaDOS also sets up space for directories, the disk's name, and other information to make it recognizable in the future. Any disk you wish to store information on must be formatted. If you use the DISKCOPY command from either a CLI or the Workbench, the destination disk will be formatted automatically. The drive to use and the new disk's name must be specified.

Example:
FORMAT DRIVE DF0: NAME "OTHER DATA DISK"

This formats a disk in drive 0 and gives it the name "Another data disk." If you give two disks the same name, it is likely you will confuse only yourself. AmigaDOS can tell the difference since it uses the date and time of each disk's creation as part of its identification.

> **NOTE:** The keywords "drive" and "name" must be part of the command.

INFO

Use INFO to find out what disks and drives are attached to the system. Each disk installed is listed with its size, name, and blocks free.

> **NOTE:** The RAM disk is also listed. The RAM disk is always full since it is only allocated as much memory as it needs.

Example: **INFO**

INSTALL

Use INSTALL to make a new system disk. All files and directories required for the disk to be used as a system disk must already be present; INSTALL only makes it recognizable as a system disk. Use INSTALL after you have copied system files to the disk. You don't need to use INSTALL if you used DISKCOPY to create a copy of a system disk.

Example: **INSTALL FRED:**

Makes a disk named "fred" bootable.

JOIN

Copies two or more files into one new file. The original files remain. Up to fifteen files can be JOINed at a time.

Example:
JOIN CHAPTER1 CHAPTER2 CHAPTER3 AS BOOK1

This combines the contents of three files in the current directory (chapters 1-3) as a new file named "book1."

LIST

Like DIR, LIST is used to list the contents of a directory. The listing is not alphabetized. LIST also displays file size, protection status, time and date of creation, plus any comment assigned with the FILENOTE command. LIST also accepts wildcard patterns (see appendix) for specifying filenames and has several unique options. The options are:

DATES—Displays dates (default mode).

KEYS—Displays block header numbers.

NODATES—Omits dates and times from listing.

P—Searches for a specified pattern.

QUICK—Lists names only.

S—Searches for a specified string.

SINCE—Displays only files created on or after a specified date.

TO—Directs the listing to a specified device.

UPTO—Displays files created on or before a specified date.

Example: **LIST**

Gives the standard list

Example: **LIST QUICK**

Lists only directory and file names.

Example: **LIST SINCE 01-JAN-86**

Shows all files updated since New Year's Day.

Example: **LIST UPTO 01-JAN-86**

Shows all files updated before 1986.

Example: **LIST DF1: TO PRT:**

Lists the contents of drive 1 to the printer.

Example: **LIST S .BAS**

Lists only files and directories that contain ".bas" in their name.

Example: **LIST DF0: P .(C|O)**

Lists only files and directories in drive 0 that contain ".c" or ".o".

MAKEDIR Creates a new directory.

Example: **MAKEDIR T**

Creates the directory "t" in the current directory.

Example: **MAKEDIR "DF0:BASIC PROGRAMS"**

Creates the directory "Basic Programs" in the root directory on drive 0. Notice that quote marks are required since the name contains a space.

NEWCLI Creates a new CLI process. The new window can be customized to specify its position, size, and name ("CON:x/y/hsize/vsize/window name").

Example: **NEWCLI**

Creates a new CLI window using a default size and name. The prompt is the process number assigned by AmigaDOS to the new window.

Example: **NEWCLI CON:0/0/640/200/FRED**

Creates a new CLI window positioned at 0,0 (upper left) with a size of 640 pixels across by 200 scan lines (i.e., the entire screen). Even if you only wanted to give it a name, you must specify the new window's size and position.

PROMPT Changes the prompt in the CLI window. If you want the process number as part of the prompt use "%n" in specifying the new prompt.

Example: **PROMPT**

Sets the prompt to ">"

Example: **PROMPT "YOUR COMMAND? "**

Sets the prompt to the phrase "YOUR COMMAND?"

Example: **PROMPT "COMMAND %N:"**

Sets the prompt to "COMMAND" plus the process number followed by ":"

PROTECT Sets the protection status of a file. A file can have four protect options: Readable (r), Writable (w), Deletable (d), and Executable (e). So far with AmigaDOS 1.0 none of these options work except for (d). The (e) option is to identify a file as one that can execute or run (a program). (r) and (w) are for keeping unauthorized users from accessing or changing files. These last two options are generally only needed in a multi-user environment. Of course, if you FORMAT the disk or DISKCOPY over it, these options are ignored.

Example: **PROTECT DF0:LASTRESORT R**

Sets the protection on the file "lastresort" to read only.

Example: **PROTECT JUNK RWD**

Sets "junk"s status to readable, writable and deletable.

RELABEL Change a disk's name.

Example: **RELABEL DF0: "DIFFERENT"**

Changes the name of the disk in drive 0 to "different."

Example: **RELABEL WORK: "WORK BACKUP"**

Changes the disk named "WORK" to "WORK BACKUP."

RENAME Renames a file or directory, or associates it with a different directory. In a sense the file "moves" around on the same disk.

Example: **RENAME FRED AS ETHEL**

or

RENAME FRED TO ETHEL

or

RENAME FRED ETHEL

Renames "fred" as "ethel"

Example: **RENAME T/ED-BACKUP AS :LETTER**

Renames the file "ed-backup" in the directory "t."

The new name is "letter" and is now in the root directory.

> **NOTE:** You cannot rename from one disk to another. Use COPY.

RUN

Executes one or more commands in the background. When RUN is used a new CLI process is started but no window is created. The new process continues to run until it is completed, but it leaves your current CLI window active so you do not have to wait for the process to finish. This is useful for printing a file while continuing to work on something else.

Example: **RUN TYPE >PRT: FRED**

Prints the file "fred" to the printer.

Example: **RUN TYPE PRT: FRED+**
ECHO "DONE."

Prints the file "fred" to the printer and displays "DONE." in the CLI window.

> **NOTE:** The "+" at the end of a line tells AmigaDOS that the following line is also part of the RUN command. Each command must be on its own line.

SAY

Voice synthesizer. Uses the Translator Library to convert text to speech. Entering the command SAY without parameters opens two windows: an input window and a phoneme window. Text entered in the Input Window and followed by a "RETURN" will be converted to speech. Exit by typing "RETURN."

Options available are:

-m male voice
-f female voice
-s speed #(40-400)
-p pitch #(65-320)
-x "filename"

Example: **say**

Enter interactive voice mode.

Example: **SAY -X s/startup-sequence**

Reads the file "startup-sequence" out loud.

Example: **say I am a party animal.**

Says the sentence on the command line. Unlike other CLI commands no quotes are necessary.

SEARCH Searches all filenames for a specified string. Using the "ALL" option, files in any sub-directories can be included in the search. CTRL-C aborts. CTRL-D quits the search in any file. Wildcard patterns may be used to specify the filenames searched.

Example: **SEARCH DF0: "JOHN SMITH"**

Searches every file in the root directory on drive 0 for the phrase "JOHN SMITH."

Example: **SEARCH DF0: "JOHN SMITH" ALL**

Same as above, but searches the entire disk.

SORT Sorts standard text files. Each line in the file is sorted alphabetically. As an option, you can specify which column the sort is to begin with.

> **NOTE:** Each task is given a certain amount of "stack" to work with. SORT does not test whether it has used up this workspace as it should. It will continue to sort and when it runs out the Amiga will crash and you will have to reboot. Use the STACK command to give yourself more workspace if you intend to sort anything other than small files.

Example: **SORT NAMES TO PRT:**

Sorts the file "names" in the current directory and sends the sorted list to the printer.

Example:

SORT "MYFILE" TO "TEMP" COLSTART 5

Sorts "myfile" in the current directory to the file "temp." Skips the text in the first four columns of each line, and sorts on the fifth.

STACK Lets you set or see the current stack size. All programs must have a scratch workspace. Each program requests memory for stack space from AmigaDOS. If a program runs out of stack space it should stop its operation and give a warning. Programs that do not do this will crash the Amiga (see SORT). If you have five or more levels of directories on a disk you may need to increase stack space. Each CLI task is allocated 4000 bytes of stack space.

Example: **STACK**

Displays current stack size in bytes.

Example: **STACK 10000**

Set the current stack size to 10000 bytes.

STATUS Displays information on each current process. Options available are:

CLI—Displays the section name of the current command.

SEGS—Displays the names of sections of the segment list.

TCB—Shows stack size, process priority, and global vector.

FULL—All of the above options.

Example: **STATUS**

List each process and the command it is running.

Example: **STATUS 2**

Gives info on process 2.

Example: **STATUS FULL**

Lists all info on all current processes.

TYPE Prints a file to the screen or a specified device. There are two options: OPT N and OPT H. OPT N prints the file with line numbers. OPT H prints the file in hexadecimal numbers. This option is handy if you need to look at the contents of a file that isn't text.

Example: **TYPE README**

Prints the file "readme" to the screen.

Example: **TYPE README OPT N**

Prints the file "readme" to the screen with line numbers.

Example: **TYPE >PRT: README**

Prints "readme" to the printer.

Example: **TYPE README TEMP OPT N**

Prints a copy of the file "readme" to the file "temp" with line numbers.

WAIT Wait a specified amount of time or until a specified time. The process can do nothing else until the wait period is up.

Example: **WAIT**

Wait one second.

Example: **WAIT 5**

Wait five seconds.

Example: **WAIT 5 MINS**

Wait five minutes.

Example: **WAIT UNTIL 13:30**

Wait until 1:30 PM.

WHY Explains why the last command failed. This is pretty useless since in most cases it simply repeats the last error message given without much more elaboration.

Example: **WHY**

CLI
Command Library
Summary

CLI Command Summary

System and Disk Management

ASSIGN	Assigns a new logical device name to the disk directory.
DATE	Sets or displays the system date and time.
DISKCOPY	Copies the entire contents of one floppy disk to another.
FAULT	Displays error message codes.
FORMAT	Formats and initializes a new disk.
INFO	Gives information about the disks installed.
INSTALL	Makes a new system disk, one that is bootable.
RELABEL	Changes the volume name of the disk.

File Utilities

COPY	Makes a copy of a file, or copies all the files from one directory to another.
DELETE	Deletes files or directories no longer needed.
DIR	Displays an alphabetized lists of files in a directory.
ED	Enters a full screen editor for text files.
EDIT	Enters a line editor useful for editing binary or non ASCII files that have embedded control characters.
FILENOTE	Attaches a note with a maximum of 80 characters to a specified file.
JOIN	Copies and merges two or more files into a new file.
LIST	Examines and displays detailed information about a file or directory.
MAKEDIR	Creates a new directory.
PROTECT	Sets the protection status of a file.
RENAME	Renames a file or directory.
SEARCH	Searches all filenames in a directory for a specified string.
SORT	Sorts lines alphabetically in standard text files.
TYPE	Prints a file to the screen or a specified device.

CLI Control

BREAK	Enables the user to stop a process in progress.
CD	Sets a current directory and/or drive.
ENDCLI	Terminates the current CLI window.
NEWCLI	Creates a new interactive CLI window.
PROMPT	Changes the prompt in the current CLI window.
RUN	Executes one or more processes in the background.
STACK	Sets or displays the stack size (scratch workspace).
STATUS	Displays information on each current process.
WHY	Explains why the last command failed.

Command Sequence Control

ECHO	Prints a message.
ELSE	Tests specific actions within a command sequence (See the EXECUTE command).
ENDIF	Marks the end of the entire command sequence of conditional commands. (See the EXECUTE command).
EXECUTE	Takes a text file composed of one or more CLI commands and executes them as if they were entered from the keyboard.
EXISTS	Tests whether a filename exists (See the EXECUTE command).
FAILAT	Fails a command sequence if a program returns an error code greater than or equal to a specified number.
IF	Tests specific actions within a command sequence. (See the EXECUTE command).
LAB	Defines a label for SKIP to jump to during an execute file. (See the EXECUTE command)
QUIT	Exits from a command sequence with a given error code.
SAY	Uses the Translator Library to convert a text string to speech.
SKIP	Instructs AmigaDOS to skip forward to a specified LABel in a command sequence. (See the EXECUTE command)
WAIT	Waits a specified amount of time or until a specified time.

Chapter 4

MASTERING CLI

IN Chapter Three we presented working examples for each command rather than giving command syntax or providing a command template to follow. This is because many of the commands take optional parameters and keywords in what may appear to be a random and confusing fashion. The examples provided should cover the proper syntax you need for quick reference.

There is, however, a built in quick reference for nearly all of the CLI commands. Typing a question mark preceded by a space after most commands displays their command syntax and a prompt. This is extremely useful if you have only one disk drive to work with. Why? Because the command is loaded into memory and remains there until you respond to the prompt.

If you have a single-drive system and want the directory of a disk other than the startup disk, you would normally enter:

DIR MYDISK:

and wait for the requester to prompt you to insert "Mydisk." This can be a real problem if you can't remember the exact name of the disk you want to inspect. But if you enter:

DIR ?

AmigaDos responds with:

DIR,OPT/K:

Then insert the disk you want a directory of, and hit Return. The contents of the disk in the drive will be displayed. You could also enter optional parameters as described in Chapter Three.

This space-question mark syntax works with nearly all the AmigaDOS commands. The commands that do not work are:

EXECUTE	LOADWB
NEWCLI	RUN
SAY	

and commands such as IF, FAILAT, and QUIT which only function in EXECUTE files. Commands which take no arguments, such as INFO, return a prompt such as "NONE:".

Deciphering Command Prompts

AmigaDOS's prompts can appear very confusing until you become familiar with them. Let's look again at:

DIR ?

AmigaDos responded with:

DIR,OPT/K:

This tells you that the command can take up to two arguments or parameters. (For the record—and anyone who is confused—in the context of CLI commands, "argument" and "parameter" mean the same thing.) The comma sets off the description of each argument. "DIR" tells you that the first argument can be the directory name. "OPT/K" tells you that if an option is specified it must be preceded by the keyword "OPT". The "/K" tells you that the word "OPT" must be part of the command line if an option is desired. Unfortunately, the list of the options for DIR is not displayed. Refer to Chapter Three for a complete list.

There are two other descriptors for arguments. These are "/A" and "/S". "/A" tells you that an argument is always required and may not be omitted. "/S" signifies that a keyword is used as a switch and takes no argument. To illustrate these points try the command:

COPY ?

The AmigaDos response is:

FROM,TO/A,ALL/S,QUIET/S:

If you are familiar with this command you know that entering:

SYS:FONTS TO RAM: ALL QUIET

at this point will copy the entire contents of the "fonts" directory, including any sub-directories, to the RAM disk without displaying the filenames as they were copied. "SYS:FONTS" satisfies the argument "FROM", "RAM:" satisfies the argument "TO", and "ALL" and "QUIET" are optional keywords to "switch on" special aspects of the copy. If you are following along you may have noticed that the word "TO" is not a required keyword in the command line. Also "FROM/A" should have been used in the prompt for clarity.

"/A", "/K", and "/S" are used in the same manner in EXECUTE files. If you refer to the section of Chapter Three on EXECUTE you will notice that these descriptors allow you to specify how you want input from the command line passed to your EXECUTE file.

Special Keys

Several key combinations perform special functions in the CLI. Only a few are of any use on the command line and the rest might produce unwelcome results. They exist because the console device used to interface you and AmigaDOS supports them, and it is possible another application using this device can use them.

These keys are:

CTRL-C-F—Break/attention flags (see the BREAK command, Chapter Three).
CTRL-G—Bell. Actually flashes the screen.
CTRL-H—Same as the Backspace key.
CTRL-I—Same as the Tab key.
CTRL-J—New line.
CTRL-K—Cursor up one row. Sorry, you can't edit the previous line.
CTRL-L—Clear the CLI window.
CTRL-M—Same as the Return key. (Auto repeats, too!)
CTRL-N—Switch to alternate character set.
CTRL-O—Switch to normal character set.
CTRL-V—Verify/update the window.
CTRL-X—Rub-out the current line. (Faster than backspacing.)
CTRL-—End-of-file.

Timesavers

When you type a filename, AmigaDOS first looks in the current directory for it. If the filename is not there, AmigaDOS looks for it in the command directory. (This is the "C" directory on the system disk, unless you changed it with the ASSIGN command.) AmigaDos command files are really small programs. This means that if you COPY or RENAME a program to the command directory, AmigaDOS will always be able to find it regardless of the current directory. For the same reason you should put any EXECUTE files in the Sequence directory. This is the "S" directory on the system disk unless you change it. It is also where the "Startup-Sequence" can be found.

Using the RAM disk to store the most commonly used CLI commands will save you the time of loading them from disk. Although it cuts down on your available memory the RAM disk can be very useful if you want a disk other than the system disk in drive 0.

If you want to create a short EXECUTE file and don't want to go into ED, try using copy. For example:

COPY * TO S:DC

Creates the file "DC" in the Sequence directory and copies everything entered from the keyboard directly into it. To end the file type CTRL-\.

We have already shown you two ways to enter the CLI. The first was by clicking a bunch of icons from the Workbench. The second was by editing the startup file. The third is the simplest of all. Just press CTRL-D when the Workbench disk first boots up, while it is still printing messages on the screen.

Why not "RENAME EXECUTE as X"? It is a lot more convenient than typing out the word "EXECUTE" every time you want to use a batch file.

Tired of sliding graphics screens up and down? Type Amiga-N and Amiga-M to toggle back and forth.

More Power

The heart of the Amiga is a 68000 microprocessor. There are two other processors in the Motorola 68000 family that are compatible with the Amiga—the 68010 and 68020. The following briefly touches on their benefits in the Amiga and is probably of interest only to those readers who are programmers or hardware oriented.

The 68010 caches the last instruction. This means probably no more than a five percent speed increase.

The typical 68020 contains 256 bytes of 32-bit ram used to cache instructions. Kickstart 1.1 and later versions automatically enable this cache at boot time. Coupled with a math coprocessor, this chip should make the Amiga fly.

All existing and future software should run on either of these chips, but there is a possibility of incompatibility, since the MOVE from SR instruction is privileged on the 68010 and the 68020 uses all 32 bits for addressing. The Calculator program on the Workbench 1.1 disk is known not to function.

Disk Organization

Amiga's three and a half inch mini-disks are double-sided double density. Each disk contains 80 tracks or cylinders, forty per side. Each track in turn contains eleven sectors of 512 bytes each for a total usable disk capacity of 880K (kilobytes). In addition, each sector

contains sixteen bytes of sector label area for an additional 28K bytes of label area per floppy disk.

The tracks are arranged so that the even numbered ones (0, 2, 4...78) are on the top side, and the odd numbered ones (1,3,5...79) are on the bottom side. This interleaving of tracks shortens the load/save time as the read/write head is positioned less often during sequential read/writes of long files consisting of adjacent sectors. Actually, the computer reads an entire track into its buffer, all 5.5K of data, and extracts the necessary sectors from the track buffer. This method increases the storage capacity of the disk by 20% by allowing access to the disk with no interleaving of sectors. The disk driver, using the blitter to move the data to and from the track buffer, extracts the needed sectors or blocks. Since the blitter is much faster than the 68000 microprocessor in transferring data, the drive is considered fast compared to, say, Macintosh standards.

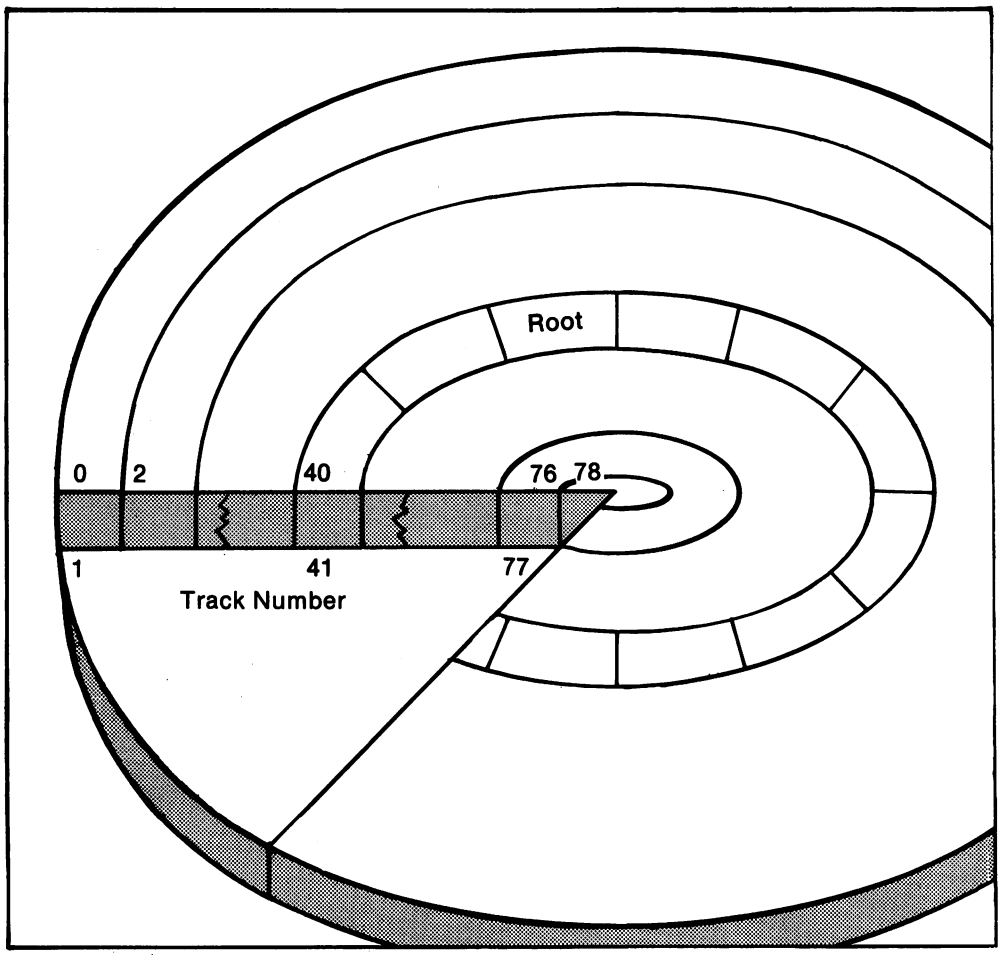


Figure 19—The disk

AmigaDOS, unlike disk operating systems of other personal computers, doesn't dedicate a track exclusively to the disk directory. Indeed, there is no directory in the usual sense with a table of easily accessible filenames. Instead, AmigaDOS uses a "root block" which is placed at the center of the disk surface (block #880—track 40 sector 0) rather than on track zero as customary. This "root block" points to other directories or files in tree fashion. While this method allows an indefinitely deep hierarchy of directories, particularly useful with large storage devices, it does take a considerable amount of time to search through this tree structure of sub-directories and files scattered on the disk surface to obtain the requested directory.

The "root block" contains the volume name of the disk and the date and time of creation to the nearest 50th of a second. This is the reason that you can't substitute a copy of a disk for the one AmigaDOS rejects. At least this applies to disks copied through Workbench or AmigaDOS commands. A hash table, where file and sub-directory names are routed to block numbers, follows. Each of the blocks that are pointed to can be either a directory or a file. It is possible that one or more file names 'hash' to the same block number. If that is the case, DOS reads through a chain of extension blocks until it finds the name that matches.

Sub-directories have the same structure as the "root block," while file headers contain a filename, date, and a table of data-block numbers that make up the file. If there are more blocks in the file that can be specified in the block list, then the list points to another block list containing the remainder. Thus there is no limit to the size of the file except by the constraints imposed by the physical storage capacity of the disk. All of the files are considered random-access with no distinction between binary and ASCII format.

All of the blocks that make up a file contain pointers to the next block in line (for efficient sequential access), and also a back pointer to their header block. This makes it possible to reconstruct a file by reading the pointers even if the file header is wrecked.

Data access has been further optimized by placing file headers and sub-directories inward from the "root block." Data blocks are allocated outward so that consecutive blocks can be kept close together.

While Metacomco, the creators of AmigaDOS, have done an admirable job to optimize AmigaDOS, they sacrified speed for a

more powerful and complex directory and data structure. Unfortunately, it gives the appearance that the disk drive is the culprit when it is not, for it is reading at least three tracks of 5.5K bytes of data per second. AmigaDOS spends most of its disk access time searching through a virtual chain of sector pointers on different tracks as it attempts to either construct a directory or find the pieces of a complete file. While it is unlikely that there will be any fundamental change to AmigaDOS that will alleviate this bottleneck in the near future, hopefully some commercial software will read and cache the directory in memory for faster disk access.

Chapter 5

ED—THE TEXT EDITOR

From a Command Line Interface (CLI) window you can access the built-in text editor, ED. ED is useful for editing programs, EXECUTE files, and simple documents. We say simple documents because there are no print formatting commands. What you see is what you get. Underlining, changing print styles, etc., are not possible.

As you would expect with any text editor, text is inserted at the cursor, and as each line is entered the cursor moves down to the next line. When the cursor reaches the last line on the screen, text is scrolled upward one line at a time so that you can continue. The horizontal width of the screen is sixty or eighty characters, depending on what you set in the Preferences program. The maximum length of a single line, however, is 255 characters. The screen scrolls horizontally, ten characters at a time, as necessary.

To use the editor, you must give it a filename to create or modify:

ED DF0:TEST

If the file "test" exists on the root directory on drive 0 then ED will load it in for modification. "Test" must be a pure text file and not contain any binary codes. If "test" does not exist, it will be created when you exit ED. ED checks to see if it is overwriting an old file when it saves a new one. Before overwriting an existing file, ED attempts to rename the old file to ":T/ED-BACKUP," so that you can go back and recover your old file if you made a mistake. For the backup to be made, however, the directory "t" must exist in the root directory of the drive you are using.

There are two ways to exit ED:

ESC-X—Exists and saves changes (press ESC, X then Return).

ESC-Q—Quit without saving changes (press ESC, Q, then Return).

If you have not experimented with ED, why not try editing a simple test file now to get a feel for it? Notice that the arrow keys work (unlike in a CLI window). The mouse, however, is still only used to manipulate the window gadgets. Try typing a long line and

see what happens. ED will try to force a new line rather than extend the current one. Use the arrow keys to position the cursor at the start of an already existing line. If you type a little, you will see that text is pushed to the right. If you type a lot more on the line, you will notice there is no indication that a line has gone off the edge of the screen. Enter too much and you will get the message "Line too long."

Getting the cursor to where you want it on the screen is vital. There are several alternatives in addition to the arrow keys:

CTRL-U—Scrolls text up.
CTRL-D—Scrolls text down.
CTRL-E—Moves the cursor to the top of the screen. Typing CTRL-E again moves the cursor to the bottom of the screen.
CTRL-]—Moves the cursor to the bottom of a line. Typing CTRL-] again moves the cursor to the start of a line.
CTRL-R—Moves the cursor to previous word.
CTRL-T—Moves the cursor to the next word.

These commands require one or two keystrokes and are called Immediate Commands. There are also Extended Commands. Hitting the ESC key will put you in extended command mode. An asterisk will appear at the bottom of the screen next to your cursor. You type in a command and press Return.

The extended command cursor controls are:

B—Cursor to bottom of file.
CE—Cursor to end of line.
CL—Cursor left. (Why not just use the left arrow key instead?)
CR—Cursor right.
CS—Cursor to start of line.
M#—Move to line #.
N—Move to start of next line.
P—Move to start of previous line.
T—Cursor to top of file.

The immediate commands to delete text are:

CTRL-B—Delete current line.
CTRL-H—Same as backspace.
CTRL-O—If the cursor is on a space, delete spaces up to the next character. If the cursor is on a character, delete characters up to the next space.
CTRL-Y—Delete remainder of line.

The Backspace key deletes text to the left of the cursor. The Del key deletes text under the cursor.

The extended commands to delete text are:

D—Delete line.
DB—Delete a block.
DC—Same as Del key.

A text block is a group of characters that have been identified in extended mode with:

BS—Marks the start of the block.
BE—Marks the end of the block.

The additional block commands are:

IB—Insert Block.
SB—Show block Scrolls screen until block appears.
WB"*filename*"—Write the block to specified filename.

Immediate insert text commands are:

CTRL-A—Insert line.

The extended insert commands are:

A—Insert line after current line.
A"*string*"—Insert string on next line. Enclose the string in quotes (") or slashes (/).
I—Insert line before current line.
I"*string*"—Insert string on line before current line.
IF"*name*"—Insert text file at cursor.

Command for search and replace are:

F"*string*"—Find string.
BF"*string*"—Backwards search for string.
E*"a"b"*—Exchange string "a" for string "b."
EQ*"a"b"*—Exchange string "a" for string "b" and verify.

Strings and filenames can be contained in quotes or slashes. There are also two ways to continue a search and replace:

CTRL-G—Repeat last command. Immediate mode.
RP—Repeat until error (end of file). Extended mode.

Example: **RP E / THIS / THAT /**

Changes every occurrence of "THIS" to "THAT" starting from the current cursor position.

Searches can be set to use or ignore lowercase.

LC—Distinguish between upper- and lowercases when searching.

UC—Ignores upper- and lowercase.

Hitting the Return key in the middle of a line will split the line. To rejoin the line, the extended command is:

J—Attaches the next line to the current one.

The remaining immediate commands are:

CTRL-F—Flip upper-/lowercase.
CTRL-M—Same as Return key.
CTRL-V—Verify/redraw the screen. Rarely needed.
CTRL-[—Same as ESC key.

The remaining extended commands are:

EX—Extend the right margin.
S—Split line. Same as Return key.
SA—Saves the file using the current file name.
SH—Shows current file information. (Size, etc.)
SL#—Set left margin to #.
SR#—Set right margin to #.
ST#—Set distance between tabs.

SR can be used to stop automatic word wrap when the right edge of the screen is reached.

Example: **SR 100**

ST 10 would set tab stops at the 10th, 20th, 30th, etc. screen positions.

There is only one more thing you need to know about ED. ED allocates 40000 bytes of RAM for your file. If you are going to edit a very large file, you must specify a larger workspace.

Example: **ED "THE GREAT AMERICAN NOVEL" SIZE 100000**

This allocates 100000 bytes of workspace for the file if it is available.

Summary of ED Commands

Summary of ED Commands

Immediate Commands

Cursor movement:

CTRL-D	Scroll down half a screen
CTRL-E	Move to top of screen, or to bottom if already at top.
CTRL-I	Move to next tab position.
CTRL-R	Move to end of previous word.
CTRL-T	Move to start of next word.
CTRL-U	Scroll up half a screen.
CTRL-]	Move to end of line, or to start if already at end.
TAB	Move cursor right to next tab position.

Text Insertion/Deletion:

BACKSPACE	Delete character to left of cursor.
CTRL-A	Insert line after current line.
CTRL-B	Delete current line.
CTRL-H	Delete character to left of cursor.
CTRL-O	Delete word or spaces.
CTRL-Y	Erase to end of line.

Miscellaneous:

CTRL-F	Flip case and move cursor right.
CTRL-G	Repeast last extended command.
CTRL-M	Carriage return (also RETURN key).
CTRL-V	Redraw screen.
ESC	Enter extended command mode.

Extended Commands

Cursor Movement:

ESC-B	Move to bottom of file.
ESC-CE	Move to end of line.
ESC-CL	Move left (nondestructive backspace)
ESC-CR	Move right.
ESC-CS	Move to start of line.
ESC-M #	Move to line number n
ESC-N	Move to start of next line.
ESC-P	Move to start of previous line.
ESC-T	Move to top of file.

Text Insertion/Delete:

ESC-A/*string*/	Insert string after current line.
ESC-D	Delete current line.
ESC-DC	Delete character under cursor.
ESC-I/*string*/	Insert string before current line.
ESC-U	Undo changes made to current line.

Block Manipulation:

ESC-BE	Mark block end.
ESC-BS	Mark block start.
ESC-DB	Delete block.
ESC-IB	Insert copy of block.
ESC-SB	Show block on screen.
ESC-WB/*filename*/	Write block to file.

Text Searches:

ESC-BF/*string*/	Find string, moving backwards toward top of file.
ESC-E/*str1/str2*/	Find first occurrence of str1 and change to str2.
ESC-EQ/*str1/str2*/	Find first occurrence of str1 and query whether to change.
ESC-F/*string*	Find string, moving forward toward end of file.
ESC-LC	Distinguish between upper and lower cases when searching.
ESC-UC	Ignore case when searching.

File manipulation:

ESC-IF/*filename*/	Insert file.
ESC-Q	Quit without saving text.
ESC-SA	Save text to file.
ESC-X	Save text to file and exit.

Margins:

ESC-EX	Extend right margin (margin release).
ESC-SL #	Set left margin.
ESC-SR #	Set right margin.
ESC-ST #	Set tab distance.

Miscellaneous:

ESC-J	Join current line with next line.
ESC-RP	Repeat command until error.
ESC-S	Split line at cursor.
ESC-SH	Show information on text.

Chapter 6

EDIT

EDIT is the other text editor. Before you try to use it on any file you wish to keep, we strongly suggest that you take a few hours to become familiar with it. In fact, we suggest you not use it unless you have a strong need for a particular editing feature that only it offers. There is virtually *no* similarity between ED and EDIT, except for the fact that they both work on text files. EDIT is not intuitive in its operation and it is very likely that a new user will have disastrous results with it. We have included its description for the sake of completeness.

You cannot create a file with EDIT, but you can modify and add to an existing one. Unlike ED, EDIT does not create a window for you to edit in, but uses the current CLI window.

EDIT is a sequential line editor. Most text editors are screen editors like ED—You move the cursor around the screen and enter or delete text. Line editors only allow you to modify the current line. EDIT only keeps a limited number of lines in memory.

Entering Edit

You must supply at least the name of a file to modify. You may also supply additional arguments on the command line to specify the output file, a command file, an error file and the available text buffer.

If no argument other than the file to be edited is given, then EDIT assumes it will take commands from the keyboard and send messages out to the current window. It sets a default buffer of 40 text lines with a maximum of 120 characters per line. You can set these options, for example, with:

EDIT DF1:MUGWUMP.C TO RAM:TEMP WITH EFILE VER PRT: OPT P300W100

This tells EDIT that:

1. The file to modify is "DF1:MUGWUMP.C"
2. The edited result will be "RAM:TEMP" ("MUGWUMP.C" will not be changed)

3. The file "efile" is a list of EDIT commands. Read from "efile" instead of the keyboard
4. Send any error messages etc. to the printer
5. Use a text buffer large enough to hold 300 previous lines of a maximum width of 100 characters each.

Trying it Out

To become familiar with EDIT, let's use a copy of a familiar file: "Startup-Sequence" from the Workbench disk. Since we don't want a permanent copy, first enter:

```
COPY S:STARTUP-SEQUENCE TO RAM:TEMP1
CD RAM:
MAKEDIR T
```

then:

```
EDIT TEMP1 TO TEMP2
```

EDIT responds with:

```
Editor
:
```

The ":" is the command prompt for EDIT. There are a couple of ways to scan through the file. Type an **N** followed by the Return key and EDIT displays the Next line in the edit buffer. Since the file was just loaded, EDIT responds with:

```
2.
ECHO " "
```

or whatever the second line in the file is. Enter**P** and EDIT displays:

```
1.
ECHO "WORKBENCH DISK. RELEASE 1.1"
```

or whatever the first line of the file is. Enter **4N** and EDIT skips ahead four lines and displays:

```
5.
LOADWB
```

Enter **T** and EDIT types from the current line (5) to the end:

```
LOADWB
ENDCLI> NIL:
7*
```

Enter T again and the response is:

7*

since EDIT has moved to the end of the file. There are three ways to go back to the top:

M1 or **M-**—M1 makes line 1 the current line (only if it is still in memory). M- makes the current line the highest line in memory. Since this file contains only six lines and the default buffer allows forty, either command will work in this case. M+ is used to make the last line the current line.

6P—changes the current line to the sixth previous line.

REWIND—makes line 1 the current line of the file, whether or not it was in memory. Please note that REWIND is very slow and may take up to thirty seconds, even if the files are on the RAM disk.

Trying to move too far forward or backward in the file will cause either:

INPUT EXHAUSTED

or

NO MORE PREVIOUS LINES

to be displayed.

Now let's try changing the message on line 3 to prompt for the date. Enter:

```
M-                  | your entry
1.
echo "Workbench disk. Release 1.1"
F/DATE/             | your entry, find "date"
3.
ECHO "USE PREFERENCES TOOL TO SET DATE"
D                   | your entry, deletes the current line
4.
ECHO " "
I                   | your entry, insert line command.
DATE ?              | your entry, new line
Z                   | your entry, terminate insert
:
```

The D and I commands are used to delete and insert lines. If no line number is specified, the current line is assumed. I inserts text before the current line. A range of lines can be deleted, such as:

D40 50

or

D. *

The first example deletes lines 40 through 50. The second example deletes from the current line to the end of the file. The period refers to the current line and the asterisk refers to the end of the file. When inserting text, you continue to insert lines until you exit insert mode by entering a "z" by itself on a line. If we had known which line we wanted to delete, we could have entered:

D3
I3

followed by the text to be inserted. This can be simplified one step further with:

R3

which stands for "replace." R deletes the range of lines specified and then begins inserting text.

The find command, F, searches forward through a file for a specified string. The current line is set to either the line where the string was found or the end of the file if the search failed. BF can be used to search backwards through a file. A string can be up to eighty characters in length and must be enclosed in delimiters. Delimiters are characters that EDIT will recognize so it knows where your string starts and stops. The available characters for delimiters are:

/ ? . , + - : *

It is possible to search and replace text using the exchange command, E, to replace one string with another.

> **NOTE:** You cannot use different delimiting characters on the same line. For example:
>
> **E/CRASH/TOAST**
>
> replaces the first occurrence of "crash" with "toast" and
>
> **E/BOGUS//**
>
> replaces the next occurrence of "bogus" with nothing, deleting it.

There are also special find and insert commands. For example:

A/EXPAND/ABLE/

searches for the next occurrence of "expand" and adds "able" immediately after it, changing it to "expandable," While:

B/PRE/DESTINED/

searches for "destined" and inserts "pre" immediately before it.

The Delete-Find command (DF) is an odd combination of search and delete functions. This command deletes lines *until* the specified string is found.

EDIT assigns a line number to each line of the source file. You can use the TL command to type the file displaying line numbers. Position the current line back to line 1 and try it now. Notice that the third line is not numbered as line 3 but "***". This is because the original line 3 was deleted, and EDIT does not associate new or inserted lines with line numbers. This can be fixed using the renumber command, "=". Enter the following:

M-
=1
TL

The new line is now line 3. You can use "=" to assign any number to a line, but in most cases it is best to go back to the first line and enter "=1." Try different values and use TL to see the results. To display the text buffer without line numbers use TP. This is the same as M- followed by T. T or TL followed by a number will display the specified number of lines.

There are several ways to exit EDIT:

W—Windup. Scans through the remainder of the source file to copy it to the output file.

STOP—If you have made errors, you do not want the new output file to be made.

Q—Quit. Stops the output file at the current line. Unless the current line is the end of the file your output file is going to be truncated. Q can also be used to terminate a command input file. Command input files end when EDIT encounters a Q command or the end of the file.

EDIT will save a backup of the file it overwrites in the "t" directory. EDIT also use the "t" directory for temporary files during EDIT sessions and deletes them when they are no longer needed. It is important to make sure that a "t" directory exists in the current root directory. The backup file EDIT saves is ":T/EDIT-BACKUP," similar to ED which saves its backup file as ":T/ED-BACKUP." Also, EDIT creates a temporary file and renames it at the end of the

session to your output filename. RENAME does not work across devices, therefore the "t" directory and output files must be on the same disk.

EDIT has many more commands, but to describe them all in the form of a practice session would probably be more confusing to most readers than beneficial. For this reason, the remainder of this chapter will simply describe these commands and their meanings.

String Searches

Similar to the exchange options of "b," "a," and "p" are special qualifiers for strings:

F B/HELLO/—Finds "Hello" only if it is at the beginning of the line.

G E/ALL./—Finds "all." only if it is at the end of a line.

G L/AGAIN/—Searches for "again" from left to right so that the last occurrence on the line is found first.

G P/ALL ALONE./—Finds "ALL ALONE." only if it exists on a line by itself.

FU/FRED/—Finds "fred" whether it exists in upper- or lowercase.

Splitting and Joining Lines

SA/*here,*/—Splits current line after "here,"

SB/*here*/—Splits current line before "here"

SB L/*here*/—Splits current line before the last occurrence of "here"

CL//—Joins the current line with the next one.

CL/ AND/—Appends "and" to the current line and joins it with the next.

CL—(Concatenate Line) joins two lines and a specified string. If you do not want a string, use "//" which is an empty string. Otherwise EDIT becomes confused and returns an error message.

Global Commands

Global commands work on source lines in the buffer when they are invoked and on new lines automatically as they enter the buffer. If a global command is suspended and re-enabled, later it will not automatically work on new lines that entered the buffer. Most likely this is because it does not keep track of lines it has modified and would end up re-doing the same procedure on them. Also, it is likely

that AmigaDOS's creators thought that a global command would be suspended to skip past sections where they were not wanted. GA, GB, and GE accept strings or qualified strings for input. REWIND cancels all globals.

GA/EQU /CUSTOM+/—Inserts "custom+" after "equ" whenever it appears in the source file.

GB/COP/;/—Inserts ";" before occurrences of "cop" whenever it appears in the command file

GE U/$DFF/CUSTOM+$/—Exchanges all occurrences of "$dff" or "$DFF", etc. with "CUSTOM+$" whenever they appear in the buffer.

SHD—Show data. Displays saved values on string searches and such.

SHG—Show globals. Displays all globals with their assigned number, a "D" if they are disabled, and the number of times they have been used. The above examples might produce:

1. 14 GA /EQU /CUSTOM+
2. 8 GB /COP/;
3. 0 GE U/$DFF/CUSTOM+$

CG—Cancel all globals.
CG 1—Cancel global #1.
DG—Disable all globals.
DG 3—Disable global #3.
EG—Enable globals.
EG 1—Enable global 1.

The Command Line

By now you may have noticed what appears to be an inconsistency in EDIT. The prompt ":" is not always present when EDIT is ready for command input. This is because the prompt only appears after EDIT verifies (displays) the line. "?" and "!" are commands to verify the current line. Because EDIT can work on files containing some of the nonprinting characters (binary), "!" is used to display non-alphanumeric characters in hexadecimal. Lines are automatically verified after a change. The commands:

V+
V-

are used to toggle automatic verification on and off. If EDIT commands are taken from a source other than the keyboard then automatic verification is turned off.

Line Windows and Single Character Commands

Another character that you may have noticed is ">". This character is used by EDIT to show its current position on a line. EDIT is able to work on single characters within the current line as well as blocks of strings.

When you enter ">", the pointer moves one character to the right for each ">" entered. "<" moves the character pointer to the left. Entering PR resets it to the start of the line.

PA stands for Point After. Entered with a string it sets the character pointer to the position immediately following the first occurrence of the specified string in the current line. Use "BP" followed by a string to position the Point Before the specified string. Don't forget you can use the string qualifiers "B," "E," "L," "P," or "U" with these commands.

Remember that the character pointer will move to the next appropriate character in the line. You may also enter more than one command at a time. The single character edit commands are:

#—Deletes one character.
2#—Deletes two characters, etc.
$—Change character to lowercase.
%—Change character to uppercase.
_—Change character to a space.

Now you have just enough EDIT commands to get you into trouble. There are several you'll probably need. We can only suggest that you experiment on your own with a test file before you use them on anything important, since EDIT is usually more than happy to destroy your text file for you when you exit.

These are the remaining EDIT commands:

DTA Q/*string*/—Deletes from the current line pointer to just after the specified string. You have the option of using qualifiers for the string.

DTB Q/*string*/—Deletes to just before the specified string.

DFA Q/*string*/—Deletes the remainder of the line after the specified string.

DFB Q/*string*/—Deletes the specified string and the remainder of the current line.

AP Q/*str1*/*str2*/—Finds the first string and inserts the second string immediately after it.

BP Q/*str1*/*str2*/—Same as "AP" but inserts the second string in front of the first one in the current line.

EP Q/*str1*/*str2*—Exchanges string two for string one.

> **NOTE:** "AP," "BP" and "EP" leave the character pointer positioned after the change. Unlike E (Exchange), which leaves the character pointer where it was. This allows you to do multiple exchanges on the current line: "2EP," "3EP," etc.

;—Like ED, you can use a semi-colon to separate commands (i.e., enter more than one at a time on a line).

( )—Parentheses can be used to group commands, but EDIT cannot accept more than one physical line of input.

B-*nn*—Most of the EDIT commands can be preceded by a number so that they will be repeated a specified number of times. Specifying "O" will cause the command to repeat until the end of the buffer or file, depending on the type of command. If you couple this with command groups in parentheses, you can construct a loop that repeats endlessly until you hit CTRL-C.

TR-—Trailing spaces will be removed from the end of lines. This is the default mode.

TR+—Trailing spaces will be left alone.

Z/*string*/—Changes the Insert mode terminator from "z" to a specified string.

FROM/*filename*/—EDIT switches to the specified file and reads it for text. Remember, you can use delimiters other than the slash.

FROM—EDIT resumes reading from the original file for text.

TO /*filename*/—Redirects EDIT's output to a specified file. You can switch at any time. Subsequent edited lines go to the new file.

TO—Redirects output back to the original output file.

CF /*filename*/—Close the file. Be sure to close any files you open with FROM and TO commands. Otherwise you may lose what you edited.

I/*filename*/—Inserts text from the specified file.

R /*filename*/—Replaces the current line with text from the specified file.

R 4 12/*filename*/—Replaces lines 4 through 12 with text from the specified file.

H *nnn*—Sets a line number as an EDIT limit. EDIT will not edit past the specified line number.

APPENDIX A

Wildcards and Patterns

Sometimes you want a command to affect more than one file. For example, you may want to use COPY to duplicate all the basic programs on one disk onto another. Wildcard characters allow you to do this. The concept is simple, and not unlike wildcards in poker. You specify a portion of a filename, and the computer matches any name that contains that partial name. The partial name, or group of characters, is referred to as a "pattern." Many other computers allow special characters such as "*," "?," or "=" as wildcards. AmigaDOS improves on this basic concept by allowing several special characters. They are:

? # | % ( ) '

You will probably find life much easier if you avoid using these special characters as part of your filenames. You should also avoid using other special characters used by AmigaDOS, such as:

< > ; + "

The question mark (?) matches any *one* character. Therefore,

LETTER?

will match filenames: "letter," "letter1," "letter2," "letter3," "letterz," and "letter." but not "letter10."

The pound symbol (#) matches the zero or more occurrences of the specified group of characters that follow it. Therefore,

#?

matches *everything*. Also:

#?#.BAS

matches "demo.bas," "account.BAS," and "temp..bas," but not "junk.b." "more.bs," "fish.bass," nor "sample.basic."

LE#TER#?

will match "letter1," "letter2," "letter3," "letter10," "letterz," "letter.doc," "leter," and "lettter" but not "letr" nor "lettteer". And

#?A#?

matches any file with the letter "a" in it. However,

#?#A#? matches everything.

Use a set of parentheses to group patterns. Therefore:

#(BAS)

matches "BAS" and "BASBAS,"while:

#BAS

matches "BAS" and "BBBAS."

The vertical bar (|) matches either of two patterns. Therefore:

#?(.C|.ASM)

matches all files ending with either ".C" or ".ASM".

The percent symbol (%) matches a null (empty, nonexistent) string. Therefore:

LET(T|%)ER#?

matches all files that begin with "leter" or "letter."

The single quote (') is used to specify that the following character is not a special character and is actually in the filename. Therefore:

WHATMEWORRY'?

Matches only the file "whatmeworry?" (question mark included).

The commands COPY, DELETE, and LIST allow you to use patterns in specifying filenames. LIST requires "P" as keyword before the pattern. LIST also uses "S" as a keyword if you want to match any files that contain a specified string.

APPENDIX B

AmigaDOS Error Codes

ERROR	MEANING
103	*Insufficient Memory* Either you need to add more memory to your Amiga or you have too many application programs running. It might help if you close unnecessary windows, or stop some of the other tasks. It is also possible that, if you have been previously running other programs, available memory is fragmented. In that case re-booting would help.
104	*Task Table Full* You have more than twenty CLI tasks running.
120	*Command Line Invalid* Either your command line format is incorrect or you have entered too many options.
121	*File is Not an Object Module* The requested file, possibly a picture or text document, is not a runable file. Only binary program files will run when you type the correct filename.
202	*Object in Use* Some other program is using the file your current program requested. For example, your program may be trying to read a file that another program is writing to.
203	*Object Already Exists* Usually, you tried to create a directory with a name that already exists. You must either delete or rename the first directory before you can use that name.
204	*Directory Not Found* Either the directory does not exist or you misspelled the name.
205	*Object Not Found* AmigaDOS can't find the device or filename. Usually you've made a typographical mistake, but you can get this error if you attempt to create a file in a directory that doesn't exist.

206 *Invalid Window*
Somehow you gave bogus parameters for your window size or device. Sometimes you have failed to define an entire window by forgetting the final slash.

209 *Unknown Command, Command Type Invalid*
You have asked a device handler to attempt an impossible task, like trying to DIR the console (CON:).

210 *Invalid Stream Name*
Check for illegal control characters in the name. Also, a file or directory can't be longer than thirty characters.

211 *Object Lock Invalid*
Usually caused by an error in an application program.

212 *Object Wrong Type*
The specified operation was a filename instead of a directory, or *vice versa*. Check the command usage before proceeding.

213 *Disk Not Validated*
Either you tried using a disk before AmigaDOS had validated it, or your disk has bad sectors. Then again, perhaps you forgot to format it. You can't save data to a disk that hasn't been formatted.

214 *Disk Write Protected*
AmigaDOS can't write to a disk that is write protected. You must *not* be able to see light through the write protect hole if you plan to write data to a disk. Move the switch.

215 *Rename Across Devices Attempted*
You tried to move a file between different disks using the RENAME command. Use the COPY command to move files from one disk to another.

216 *Directory Not Empty*
DELETE only works on a directory that is empty. Either you made a mistake or you need to enter "DELETE *dirname* ALL" to wipe out its contents first.

218 *Device Not Mounted*
If you haven't made a typographical error, then the disk isn't in the drive. Put the disk in the drive to use it.

219 *Seek Error*
Usually a programming error in which you called SEEK with invalid arguments. If you are not programming, you probably just have a bad disk.

220 *Comment Too Big*
Filenotes can't exceed eighty characters.

221 *Disk Full*
There is no more room on your disk. Either use another formatted disk or delete unneeded files to free up some space.

222 *File Protected from Deletion*
Use PROTECT to change the protection status. Use the LIST command to check the status.

223 *File is Write Protected*
Use PROTECT to change the protection status.

224 *File is Read Protected*
Use PROTECT to change the protection status. Read protection of files is currently not implemented (version 1.1).

225 *Not a DOS Disk*
The disk is probably not formatted or it is a bad disk. You can't save files on your Kickstart disk, either.

226 *Drive is Empty*
You forgot to put a disk in the drive.

232 *No More DIR Entries*
This is a programming error in which you scanned to the end of a directory and there are no more entries.

INDEX

About The Authors

Jeffrey Stanton received a BME (1967) and a MSME (1969) from Rennsselaer Polytechnic Institute. He worked as a control systems engineer and mechanical engineer for the aerospace industry in the early 1970's. His interest in computer game design sidetracked his career as a photographer and book illustrator in the late 1970's. In addition to writing several Apple and Atari 800 arcade games and doing some occasional consulting, he is the author of *Apple Graphics and Arcade Game Design,* co-author of *Atari Graphics and Arcade Game Design,* and one of the editor/reviewers for the books of *Apple, Commodore/Amiga,* and *Atari Software.* He is a certified Amiga developer. Jeff currently divides his time between writing and reviewing software in the mornings, and operating a postcard stand on the Venice Beach boardwalk in the afternoons. He lives in Venice, California.

Dan Pinal, typical of many of the early computer hobbyists, is self educated. Dan consulted, taught and did game programming for two software houses at the peak of the game market in 1983. He is co-author of the book *Atari Graphics and Arcade Game Design.* He is a certified Amiga developer and designs and translates entertainment software for a major software house. Dan currently lives in Los Angeles, California.

THE BOOK OF ADVENTURE GAMES

By Kim Schuette

Once upon a time a little book about fantasy lands made it very big in the real world. *The Book of Adventure Games* is one of the bestselling computer books ever.

"...both a must buy and a terrible temptation for all fans of adventure games."—*inCider* Magazine

Adventure games were one of the first types of computer software invented; the predecessors of currently popular games ran on the first mainframes ever installed. Adventure games and computers have grown up together. Like modern personal computers, today's adventure games are more sophisticated and complex. They can excite your imagination, and they can be among the most frustrating programs you'll ever own.

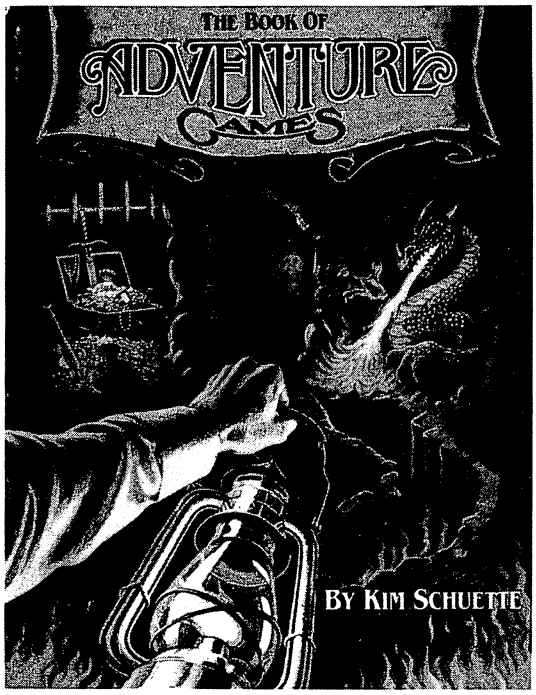
If all you need is just a hint or peek at a map to get you unstuck (and who ever needs more than that?), help lies between these covers.

The Book of Adventure Games includes descriptions, maps, illustrations, and clues for over seventy-five of the most popular games available on the Apple and many other computers. Maps and hints are presented in a way that offers help without giving away the whole game.

Paperbound; 8½" x 10⅞"; 350 pp.

ISBN 0-912003-08-1

$19.95



THE BOOK OF ADVENTURE GAMES II

By Kim Schuette

Kim has been playing more games, creeping down dungeon corridors and slogging through misty swamps with lantern raised to light the way for his faithful readers. (In fact, every few days for the past year he's emerged from the silicon depths of his Apple to cry out "Get me more Infocom! I *need* Adventure International!) And now he's amassed a superb collection of the *very latest* adventures running on the Apple, Macintosh, and most other computers.

The Book of Adventure Games II includes descriptions, maps, hints, and other helpful information for forty games, including Infocom's enormously popular *Hitchhiker's Guide to the Galaxy*, many of the new Tellarium interactive literature series, and *Gateway*, the first adventure game originally released only for the Macintosh.

Paperbound; 8½" by 11"; 250 pp.

ISBN 0-912003-41-3

$19.95

Dive Under the Workbench

Mastering AmigaDOS is the one essential book for anyone who wants to delve beyond their Amiga's icons and Workbench. As a complete user's guide to AmigaDOS, Amiga's disk operating system, it thoroughly explains disk and directory structure, multi-tasking, the Command Line Interface (CLI), and all of the available DOS commands with example applications.

Written in an easy to read, non-threatening style, **Mastering AmigaDOS** guides novices through field tested sample sessions, while advanced users are introduced to the subtleties of AmigaDOS's more advanced features. The most informative and easiest to use guide to AmigaDOS.

- Covers all available AmigaDOS commands.
- Explains the directory structure and logical devices, including the RAM disk.
- A complete CLI command library with hundreds of practical examples.
- Sample sessions with field tested examples.
- Additional information about the Workbench.
- Extensive tutorials cover both ED, the full screen editor, and EDIT, the line oriented editor.
- Complete guide to AmigaDOS and its multi-tasking operating system.

ISBN 0-912003-55-3

ARRAYS, INC.
6711 Valjean Avenue, Van Nuys, CA 91406

$16.95

Designed by Estela Montesinos