Attached is a set of errata pages for your AmigaDOS User's Manual.

Instructions are provided in the errata to indicate how the manual pages should be modified.

You may choose to make these changes or simply to file this errata package with the manual pages for future reference.

Commodore-Amiga, Inc.

Errata to the AmigaDOS User's Manual.
CBM Product Number 3272 64-1 rev 1.0 8.27.85

A brief note about terminology: In the world of Amigas there are two major user interfaces. The first, workbench, you should already be familiar with. The second, CLI, is what this manual is all about. CLI stands for "Command Line Interface", the user interface that involves typing commands in response to a prompt.

The CLI is also a command line interpreter, that is, its job is to read commands, interpret them, and run the necessary programs. Although both interpretations of CLI are correct, "Command Line Interface" is preferred. Please consider "Interpreter" to be changed to "Interface" throughout this manual.

The errata to the manual are enumerated below:

Page	Fix
1-1, Section 1.1	Fourth paragraph should read: AmigaDOS provides a process that you can use, called a Command Line Interface or CLI. There may be several CLI processes running simultaneously, numbered from 1 onwards. The CLI processes read commands and then execute them. All commands and user programs will run under any CLI. To make additional CLI processes you use the NEWCLI or RUN commands. To remove a CLI process use the ENDCLI command. (You can find a full description of these commands in Chapter 2 of this manual.)
1-2 Section 1.3	Insert the following paragraph immediately above sub-section 1.3.1: A file is the smallest named object used by AmigaDOS. The simplest identification of a file is by its filename, discussed below in section 1.3.1. However, it may be necessary to identify a file more fully. Such an identification may include the device or volume name, and/or directory name(s) as well as the filename. These will be discussed in following subsections.
1-4 Section 1.3.3	Fourth paragraph, last sentence (the sentence immediately preceding `CD :bill') should read: To refer to the files in :bill, first type

Page	Fix
1-5 Section 1.3.4	First paragraph, last line should read: Each individual disk is also associated with a unique name, known as a volume name (see below for more details.)
1-11 Section 1.4	First paragraph should be replaced with: An AmigaDOS command consists of the command-name and its arguments, if any. To execute an AmigaDOS command, you type the command-name and its arguments after the CLI prompt.
1-14 Section 1.4.5	Second paragraph above section 1.5, first sentence should read: Although this manual lists all the arguments expected by the commands, you can display the argument template by simply typing the name of the command, followed by a space and a question mark (?).
2-8	(the second page describing the COPY command) After the first example "COPY * TO CON:10/10/200/100/", replace the comment with: Click in the window that you typed the copy command into. This "reactivates" it so that console input is taken from there. Every time you type a line it will be displayed in the new window. Press CTRL- when you are done and the new window will close.
2-9	add the following sentence to the first paragraph: Time is displayed using a 24-hour clock.
2-9	comment after second example should read: sets the date to 6th September 1982. The time is not reset.
2-11	sentence starting with "If you specify" should read: If you specify ALL with a directory name, DELETE will delete that directory and all subdirectories and files within that directory and its subdirectories.
2-11	change "1/2" to "1|2" in the third example. It should read: DELETE t#?/#?(1|2)

Page	Fix
2-12	after the first paragraph, insert the following: If you want to know if a file exists type LIST filename. Typing DIR filename, where filename is a file which exists results in the Amiga responding with: "filename is not a directory".
2-12	references to DEL should indicate: typing the three letters D-E-L, not the DEL key.
2-15	last sentence in first paragraph should read: If the FROM filename does not exist, AmigaDOS creates a new file.
2-18	append the following sentence to the first paragraph: If the execution creates a new CLI window, the results may not be identical to typing at the keyboard.
2-18	It may not be clear from the table that .KEY and .K are synonyms as are .DOLLAR and .DOL. Note that the ")"s in the second column of the table and at the end of .DOLLAR are meaningless.
2-19	Notes about the EXECUTE command: A file can only use the dot commands if the first line has a dot command on it. The CLI looks at the first line. If it starts with a dot command, for example, a comment (.<space>txt) then the CLI will scan the file looking for parameter substitution and build a temporary file in the :T directory. If the file doesnt start with a dot command, then it is assumed that there are NO dot commands in the file, which also means no parameter substitution will be performed. For the no-dot case, the CLI starts executing the file directly without having to copy it to :T. Note that you can still embed comments in an execute-file by using the CLI's comment character, the semicolon (;). Thus: Expensive EXECUTE file Echo "Hello There" ; Cheap EXECUTE file Echo "Hello There" If you don't need parameter substitution and dot commands, don't use them. They save you extra accesses to the disk for the temporary file.

Page	Fix
2-19	EXECUTE file example should read: .k filename/a run copy <filename> to lp: echo "Printing of <filename> done" Second EXECUTE file example should read: .k name/a IF EXISTS <name>+ TYPE <name> OPT n . If the file given is on the current directory, type it . with line numbers IF NOT EXISTS <name>+ ECHO "<name>" is not on this directory
2-21	the first and second examples are dull and should read: FAULT 222 FAULT 221 103 121 218
2-26	the line in the example starting with RAM should be deleted.
2-30	the Format line should indicate that the <dir> parameter is optional by starting: LIST [[DIR] <dir>]...
2-31	The double quote shown in the sentence "Thus ´? matches ? and " matches ´." should be two single quotes.
2-31	The line "Thus:" should be changed to: The following pattern matches can be used if preceded by: "LIST PAT" or "LIST <dir> PAT", as PAT is always required by the LIST if these are used.

Page	Fix
2-32	the line which starts with LIST QUIET should start with LIST QUICK instead. Note that LIST s produces the response: Args no good for key.... because there is an "s" directory. LIST "s" will list this directory's contents.
2-37	in the first paragraph and the first example, replace the word STATUS with FLAGS.
2-37	Delete the last two sentences of the first paragraph. Replace then with "AmigaDOS only pays attention to the delete (d) flag in the current release. Users and user programs, however, can set and test these flags if they wish.
2-50	replace the example with: 1> type df0: can't open df0: 1> why Last command failed because object not of required type.
3-2	second paragraph from the bottom, second line, replace right hand with left hand.
3-8	section 3.3.4 append first paragraph to read: The string must be in quotes (or other delimiters "/", ".", ...). In order for a match to occur the strings must be of the same case, unless the UC command is used (see below).

Possible Misunderstandings:
==========

DOS:

When AmigaDOS requests a volume (diskette), it checks both the volume name and the creation date and time. Every volume should be unique. If all of your volume names are unique AND written on the diskette label, things can proceed more smoothly.

Note: the root directory name is the same as the volume name.

I/O redirection syntax is misleading:

It should read:

<command> [< <in file>] [> <out file>] [<args>...]

RENAME command will not rename if the only change is to change the "case" of one or more letters. For example,

RENAME fox to Fox does not work.

ED:

Cursor movement keys will allow you to cursor off the end of a line. For example, if you have a "short line", it is quite valid to cursor key, say, 10 character positions past the end of last word. If you do not type any characters at this position, the line will remain unchanged. However if you do type something, 10 blank spaces will be inserted between the end of the line and the first character you typed.

The <Tab> key is a cursor movement key. It does not insert tab-characters into your file.

Ctrl-N is no longer supported; DEL performs this task.

Errata for the AmigaDOS Users Manual, Rev 1.0 and 1.1:

This errata package contains additional tutorial material that is useful to all levels of users. This package consists of two distinct parts: a replacement for section 1.6 of the current AmigaDOS Users Manual, and a supplement for the examples section following the Execute command.

1.6 Commonly Used Commands, An Example Session

This manual describes the various AmigaDOS commands. The Command Line Interpreter (CLI) reads AmigaDOS commands typed into a CLI window and translates them into actions performed by the computer. In this sense the CLI is similar to more "traditional" computer interfaces: you type in commands and the interface displays text in return.

Because the Workbench interface is sufficient and friendly for most users, the Workbench diskettes are shipped with the CLI interface "disabled". To use the commands in this manual you must "enable" the CLI interface. This puts a new icon, labeled "cli" on your Workbench. When you have selected and opened this icon, a CLI window becomes available, and you can use it to issue text commands directly to AmigaDOS.

HOW TO ENABLE THE COMMAND LINE INTERFACE

Boot your computer using the Kickstart diskette and a writable copy of your Workbench diskette. Open the Workbench diskette icon. Open the "Preferences" tool. Near the left hand side of the screen, about two–thirds of the way down you will notice "CLI" with a button for "ON" and a button "OFF". Select the "ON" button. Select "Save" (lower right part of the Preferences screen) to leave Preferences.

HOW TO MAKE A NEW CLI WINDOW

To use the CLI commands, you open a CLI window. Open the "System" drawer. The CLI icon (a cube containing "1>") should now be visible. Open it.

USING THE CLI

To use the CLI interface, select the CLI window and type the desired CLI commands (described within this manual). The CLI window(s) may be sized and moved just like many others. To close the CLI window, type "ENDCLI".

WORKBENCH AND CLI, RELATIONSHIP AND DIFFERENCES

Type "DIR" to display a list of files (and directories) in the current disk directory. This is a list of files that makes up your workbench. You may notice that there are many more files in this directory than there are icons on the workbench. The reason for this is that Workbench will only display file "X" if it has an associated "X.info" file. In fact the ".info" (pronounced "dot info") file contains all of the icon display information.

For example, the diskcopy program has two files associated with it. "Diskcopy" the file "Diskcopy" contains the program and "Diskcopy.info" contains the Workbench information about it. In the case of painting data files like "mount.pic", the file "mount.pic.info" contains icon information and the name of the program (default) that should process it (GraphiCraft). In this case, when the user "opens" the data file (mount.pic.info), Workbench runs the program and passes the data file name (mount.pic) to it.

AmigaDOS sub–directories correspond to Workbench drawers. Random access block devices such as disks (DF0:) correspond to the diskette icons you have seen.

Not all programs or commands can be run under both Workbench and the CLI environment. None of the CLI commands described in chapter 2 of the AmigaDOS users manual can be run from Workbench. For example, there are two separate Diskcopy commands. The one in the :c/ directory works with AmigaDOS (CLI). The one in the system directory (drawer) works with Workbench.

AN INTRODUCTION TO SOME AMIGADOS COMMANDS

Although all of the commands that are available through the CLI are explained in detail in the reference part of the AmigaDOS Users Manual, we have found that most users will use very few of the advanced options. Therefore we have provided a summary here showing various commands in their most common form.

The commands summarized below (along with the actual AmigaDOS command name) ask AmigaDOS to do such commands as

- Copy a diskette (DISKCOPY)
- Format a new diskette (FORMAT)
- Make a formatted diskette bootable;
  Create a CLI disk (INSTALL)
- Relabel a diskette (RELABEL)
- Look at the directory of a diskette (DIR)
- Get information about files (LIST)
- Prevent a file from accidental deletion (PROTECT)
- Get Information about a file system (INFO)
- Change a current directory (CD)
- Set the date and time (DATE)
- Redirect the output of a command (>)
- Type a text file to the screen (TYPE)
- Rename a file (RENAME)

- Delete a file (DELETE)
- Create a new directory (MAKEDIR)
- Copy files on a dual–drive system (COPY)
- Copy files on a single–drive system (COPY)
- Find files on a diskette (DIR OPT A)
- Do something automatically at boot time (using Startup–Sequence)
- Tell AmigaDOS where to look for certain things (ASSIGN)
- Open a new CLI window (NEWCLI)
- Close an existing CLI window (ENDCLI)

All of the command sequences below assume that you have started your system with a CLI disk rather than a Workbench disk, or that you have turned on the CLI using the preferences tool and have entered the CLI by that path. The sequence for turning on the CLI is provided earlier in this document.

FOR A NEW USER

For a new user, we suggest that you read and try each of these items in sequence. Each command that is shown below leaves a test disk in a known state so that the command that immediately follows will work exactly as shown. Later, when you are more familiar with the system, the paragraph titles shown below will serve to refresh your memory.

HOW TO BEGIN

Before you begin this section, be sure you have two blank, double–sided diskettes, and either your Workbench disk or your CLI disk. Before you begin, write–protect your master diskette, and write–enable the blank diskettes Most of the commands given below assume that you have a single–drive system, however, for convenience of those with dual–drive systems, the dual–drive version of the command is occasionally given.

Commands that instruct AmigaDOS to execute are shown in the following sections, indented from the left margin. After typing each command, press the RETURN key to return control to AmigaDOS. Although the commands are all shown in capital letters, this is simply to distinguish them from the rest of the text. AmigaDOS will accept the commands in lower case as well as upper case.

In the sections that follow, the notations "df0:" and "drive 0" refer to the disk drive that is built into the Amiga. The notation "df1:" refers to the first external 3 1/2 inch disk drive.

You will occasionally see a semicolon on a command line that you are told to type. What follows the semicolon is treated as a comment by AmigaDOS. Since AmigaDOS ignores the rest of the line, you don't need to type the comment along with the command. It is for your information only.

For most commands, you can get a very limited form of help by typing the command name, followed by a question mark (?) and pressing Return. It shows you the "template" of a command, containing the sequence of parameters it expects and the keywords it recognizes.

COPYING A DISK

You can use this sequence to back up your system master disk or any other disk.

For a 1 disk system

DISKCOPY FROM df0: TO df0:

For a 2 disk system

DISKCOPY FROM df0: TO df1:

Follow the instructions as they appear. For a single drive system, you'll be instructed to insert the master (FROM) disk. Then, as the copying progresses, AmigaDOS asks you to insert the copy (TO) disk, swapping master and copy in and out until all of the diskette has been duplicated. For a two disk system, you'll be instructed to put the master diskette into drive df0: (the built–in drive) and the copy diskette onto which to copy into df1: (the first external drive).

Remove your master diskette (either Workbench or CLI disk) and put your master diskette in a safe place. Leave the copy write–enabled so that it you can store information on it. Insert the copy you have just made into the built–in drive and reboot your system from the copy. (See Introduction To Amiga for the reboot process).

After the reboot, re–enter the CLI mode again. If you boot with a CLI disk, the reboot enters the CLI automatically. If you are using a Workbench disk, you must open the CLI icon in the system drawer of the Workbench.

FORMATTING A DISK

To try this command, your Workbench or CLI diskette copy should be in drive 0, and you should have a blank diskette available.

Sometimes, rather than simply copy a disk, you'll want to prepare a data disk for your system. Then later you can copy selected files to this data disk. Format your second blank disk by using the FORMAT command:

FORMAT DRIVE df0: NAME "AnyName"

Follow the instructions. You can format diskettes in either drive 0 (df0:, built–in to your Amiga) or an external drive.

After the format is completed, wait for the disk activity light to go off and remove the freshly formatted diskette. Reinsert your Workbench or CLI diskette. The formatted diskette can now be used to hold data files. It is not bootable, however.

MAKING A DISK BOOTABLE

To try this command, your Workbench or CLI diskette copy should be in drive 0, and you should have your freshly formatted disk available.

There are several different ways to create a CLI diskette. Two of these ways are shown below.

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 by typing the command:

```
INSTALL ?          ;NOTE: to use this command on a single drive
                   ;      system, you MUST use the question mark!
                   ;      Otherwise AmigaDOS will try to do the
                   ;      install on the disk currently in drive 0.
```

AmigaDOS responds: **DRIVE/A**

Remove your Workbench diskette copy and insert the freshly formatted disk. Then type:

df0:

and press Return. AmigaDOS copies boot sectors to the diskette. Now, if you wait until the disk activity light goes out, you can then perform a full reset (CTRL–Amiga–Amiga). When the system reboots, you will go directly into the CLI rather than into the workbench.

Your formatted diskette now contains a CLI and nothing else. This means that although you see the interpreter, it can't perform any of the commands shown in this section. A CLI needs several files before its commands can be performed. All of the command files are located in the C directory of your master diskette.

The second way to produce a CLI disk gives you a more useful disk in that it leaves the CLI command directories intact. Here is a step–by–step process to change a writable copy of a Workbench diskette into a CLI diskette:

1. Copy your Workbench diskette.
2. Open the CLI as described above.
3. Click the selection button in the CLI window and type the command:

RENAME FROM s/startup–sequence TO s/NO–startup–sequence

Now if you wait for the disk activity light to go off and perform a full reset, your Workbench diskette copy will have become a CLI. To restore the Workbench, perform the rename again, but with the name sequence reversed. You see, if AmigaDOS can't find a file with the exact name "startup–sequence" in the "s" directory, it will enter command mode and wait for you to type a command.

RELABELING A DISK

Before you try this command, your Workbench or CLI diskette copy should be in drive 0.

If, after either copying or formatting a diskette, you are not satisfied with the volume name you have given it, you can change the name of the volume by using the RELABEL command:

relabel AnyName: DifferentName

In this example, we have referred to the diskette we just formatted by its volume name. You will be asked to insert volume AnyName into any disk drive so that RELABEL can relabel it.

After this command completes, remove the diskette and reinsert your Workbench or CLI diskette. The diskette you removed now has the new name.

LOOKING AT THE DIRECTORY

Before you try this command, your Workbench or CLI diskette copy should be in drive 0.

You look at the contents of a diskette with the command:

DIR or **DIR df0:**

This form lists the contents of your current directory. You can list the contents of a different directory by specifying the pathname for that directory. For example, the command:

DIR df0:c or **DIR c**

lists the contents of the c(dir) on drive df0. Directories are equivalent to the drawers you see when the Workbench screen is visible.

You can look at the directory of a different disk unit, if you have one, by specifying its name. For example:

DIR df1:

lists the contents of a diskette inserted in drive 1 (the first external drive if you have one attached).

You can even look at the directory of a diskettte that isn't currently in the drive by specifying its volume name. For example, the contents of that freshly formatted diskette whose name we changed can be displayed by the command:

DIR DifferentName:

AmigaDOS will ask you to insert diskette DifferentName into the drive so that DIR can read it and report the contents of the directory. Don't do it yet, however, because there are no files present for DIR to read. We'll add some files later.

USING THE LIST COMMAND

To try this command, your Workbench or CLI diskette copy should be in drive 0.

The DIR command tells you the names of files that are in your directory. The LIST command provides additional information about those files. Type the command:

LIST or **LIST df0:**

AmigaDOS provides information about all files in the current directory, including how large each file is, whether it may or may not be deleted, whether it is a file or a directory, and the date and time of its creation.

If you specify the name of a directory with LIST, it lists information about the files within that directory:

LIST c

The "rwed" are called protection flags, for read, write, execute and delete. When each flag is set, using the PROTECT command, a file is supposed to be readable, write–able, executable or delete–able. As of the current release, AmigaDOS only pays attention to the delete–flag. If the "d" doesn't show up in the "rwed" column for a file name, AmigaDOS won't delete that file during a DELETE command.

USING THE PROTECT COMMAND

To try this command, your Workbench or CLI diskette copy should be in drive 0.

This command protects (or unprotects) a file from being deleted accidently. Try the command:

```
DATE > myfile
PROTECT myfile
LIST myfile
```

You will see that all of the protect–flags have been set to "----". Now if you try:

DELETE myfile

AmigaDOS responds: "Not Deleted – file is protected from deletion"

To re–enable deletion of the file:

PROTECT myfile d or **PROTECT myfile rwed**

GETTING INFORMATION ABOUT THE FILE SYSTEM

Your Workbench or CLI diskette copy should still be in drive 0. Type the command:

INFO

It tells you how much space is used and how much is free on your diskettes, whether they are read–only or read–write, and the name of the volume. You can make more space on the diskette by deleting files. You can change the name of the volume by using the RELABEL command.

If you want to get information about a disk that isn't in your single–drive at the moment, issue the command as:

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:" says that you don't have to type anything other than a Return key to have it perform the command. Remove your CLI disk and insert the disk on which you want INFO to operate. Wait for the disk activity light to go on and off. Then press Return. AmigaDOS gives you INFO about this other disk. This works for DIR as well as INFO.

CHANGING YOUR CURRENT DIRECTORY

Until now, we have only stayed at the "root" or topmost hierarchical level of the diskette directory. You will find more information about the directory tree structure in section 1.3 of this manual. To see the level at which you are currently positioned in your directory tree, you use the command:

CD

To change to a different current directory, you tell the system which directory is to become the current one. For example, when you did a "dir" command on df0: the CLI diskette you saw an entry c(dir). If you want to make this directory the current one, you issue the command:

CD C or **CD df0:c**

Now when you issue the command DIR, it shows the contents of this level of the filing system. The command CD (alone) shows you the name of your current directory. You go up to the root directory (the top level) by specifying:

CD :

on the current volume (if you refer to your diskettes by volume name) or

CD df0:

on the built–in drive.

SETTING THE DATE AND TIME

You can set the AmigaDOS clock by using the DATE command:

DATE 12:00:00 12–oct–85

Now the system clock counts up from this date and time.

REDIRECTING THE OUTPUT OF A COMMAND

Before you try this command, your Workbench or CLI diskette should be in drive 0.

Normally the output of all commands goes to the monitor screen. You can change where the system puts the output by using the redirect command ">". The forward arrow symbol means send the output towards this output file name. Here's an example:

DATE > datefile

Execute the command so that you can use the datefile described below. This command creates (or overwrites) a file named 'datefile' in your current directory.

Or, just to have something on that formatted diskette named DifferentName, type the following:

DATE > DifferentName:datefile

AmigaDOS prompts you to insert the volume with that name. After the disk activity light goes out, remove DifferentName and reinsert your CLI or Workbench diskette. Now issue the command:

DIR DifferentName:

Again you are prompted to insert DifferentName into any drive.
AmigaDOS lists the directory of this diskette, which now contains
a file named datefile.

Replace your CLI or Workbench diskette in the drive.

TYPING A TEXTFILE TO THE SCREEN

You can see the contents of a textfile by using the TYPE command:

TYPE datefile

This command will display whatever you have in the specified file. If you wish to stop the output momentarily to read something on the screen, press the space bar. To restart it again, press the BACKSP–key. If you wish to end the TYPE command, hold down the CTRL–key, and press the C–key.

If you wish to verify that another diskette also has the datefile contents on it, you can perform the command

TYPE DifferentName:datefile

CHANGING THE NAME OF A FILE

Before you try this command, your Workbench or CLI diskette copy should be in drive 0.

You can change the name of a file by using the RENAME command:

RENAME FROM datefile TO newname
or **RENAME datefile newname**

Now use TYPE to verify that the new name refers to the same contents.

TYPE newname

Notice that the alternate form of the command doesn't require that you use the FROM and TO. Most of the AmigaDOS commands have an alternate form, abbreviated from that shown in this preface section. The longer form has been used primarily to introduce you to what the command does. Be sure to examine the summary pages to familiarize yourself with the alternate command forms that are available.

DELETING FILES

To try this command, your Workbench or CLI diskette should be in drive 0.

You may be working on several versions of a program or textfile, and eventually wish to delete versions of that file that you don't need any more. The DELETE command lets you erase files and releases the disk space to AmigaDOS for reuse.

NOTE: If you DELETE files, it is not possible to retrieve them. Be certain that you really do wish to delete them.

Here is a sample command sequence, that creates a file using the redirection command, types it to verify that it is really there, then deletes it.

DIR > directorystuff
TYPE directorystuff
DELETE directorystuff
TYPE directorystuff

To the final command in the above sequence, AmigaDOS responds

Can't Open directorystuff

indicating that the file can't be found, because you deleted it.

COPYING FILES

Before you enter this command, your Workbench or CLI diskette should be in drive 0.

On a dual–drive system, copying files is easy:

COPY FROM df0:sourcepath TO df1:destinationpath

or

COPY df0:sourcepath df1:destinationpath

On a single–drive system, copying files is a little more complex. You must copy certain system files from your system diskette into the system memory. This is also called using the RAM: device, often known as a ramdisk. Copy the file(s) to the ramdisk, change your directory to the ramdisk, then copy from the ramdisk onto the destination diskette. Here is a sample sequence.

Be sure your Workbench or CLI diskette is in the internal disk drive. Issue the commands

COPY df0:c/cd RAM:
COPY df0:c/copy RAM:
CD RAM:

Insert the source data diskette into the drive. (For this example, copy something from the Workbench or CLI diskette, which is already in the drive). Type:

COPY df0:c/execute ram:execute
or
COPY df0:c/execute execute
or
COPY df0:c/execute ram:

Remove the source diskette, and insert the destination diskette into the drive. Type:

COPY ram:execute df0:execute
or
COPY execute df0:execute ;if you did the CD RAM: this form works

Remove the destination diskette and insert your CLI or Workbench diskette again. Type:

CD df0:

and you are back where you started. The only other command you may want to perform is:

DELETE RAM:cd RAM:copy RAM:execute

which releases the ramdisk memory to the system for other uses.

CREATING A NEW DIRECTORY

You can create a new directory (new drawer) within the current directory by using the MAKEDIR command:

MAKEDIR newdrawer

Now if you issue the DIR command, you will see that there is an entry for:

newdrawer(dir)

You can also use the RENAME command to move a file from one directory (drawer) to another on the same diskette:

MAKEDIR newdrawer
RENAME FROM newname TO newdrawer/newname

moves the file from the current directory into the newdrawer you have created. To check that it has really been moved, issue the command:

DIR

Then type:

DIR newdrawer

AmigaDOS looks in the new drawer, and shows you that the file named "newname" is there.

IS MY FILE SOMEWHERE ON THIS DISK?

Before you enter this command, your Workbench or CLI diskette copy should be in drive 0.

Sometimes you wish to see everything on the diskette, instead of only one directory at a time. You can use the DIR command with one of its options:

DIR OPT A

which lists all directories and subdirectories on the diskette. Keep in mind the <space><BACKSP> combination to pause and restart the listing.

To get a closer look at the disk's contents, you might redirect the output to a file:

DIR > mydiskdir OPT A

Notice that the redirect–the–output command character and file name MUST come before the list of options for the DIR command.

Now, if you wish, you can TYPE the file mydiskdir and press the space bar to pause the listing. Use the BACKSP key to resume the listing. Or, you can use ED to view the file, as follows:

ED mydiskdir

Use the cursor keys to move up and down in the file. Use the key combination ESC then T <Return> to move to the top of the file. Such a combination can be referred to as "ESC–T", meaning ESC followed by T. Use the key combination ESC–B <Return> to move to the bottom of the file. Use the key combination ESC–M then a number <Return> to move to a specific line number within the file. Use the key combination ESC–Q <Return> to Quit without changing the file or Use ESC–X <Return> to write any changes to your file back into the original file name.

Chapter 3 of the AmigaDOS Users Manual has more detailed information on using ED.

DOING SOMETHING AUTOMATICALLY AT BOOT TIME

There is a file in the "s" subdirectory on your Workbench or CLI diskette called Startup–Sequence. This is an execute–file. It contains a sequence of CLI commands that AmigaDOS performs whenever you reboot the system. The last two commands in your Workbench diskette startup sequence are LoadWb (load the workbench program) and ENDCLI which basically leaves the workbench program in control. You can make up your own startup–sequence file using ED or EDIT to create a custom version of an execute command sequence. The EXECUTE command summary and tutorial section in the AmigaDOS Users Manual has details about various commands that you can have in this file. Note that startup–sequence can also be used to auto–run a program.

Take care to modify only a copy of your diskette,

—— never modify the master diskette ——

if you decide to change the startup–sequence.

ASSIGNING PLACES THAT AmigaDOS LOOKS FOR THINGS

Before you enter this command, your Workbench or CLI diskette copy should be in drive 0.

Occasionally, you might wish to change to a different diskette and then continue your work. For example, you may have booted the system using a Workbench diskette, then wish to change to a CLI diskette. If the CLI diskette has a directory on it that contains the

executable commands you want to perform, (for example, a c(dir)), you can change to that diskette by using the ASSIGN command.

If you don't use ASSIGN, you will have to swap diskettes to get commands done. Here is an example that doesn't use ASSIGN. The intent is to change diskettes and begin using "mydisk:" as the main diskette. Any unneeded files have already been deleted so as to provide workspace.

CD mydisk:

AmigaDOS responds "insert mydisk into any drive". Insert it, then type:

DIR

AmigaDOS prompts "insert Workbench [or whatever the boot diskette name was] in any drive". It knows, from boot time, that the DIR command is in the boot diskette, c directory. AmigaDOS reads the DIR command, then asks "insert mydisk in any drive". Any other AmigaDOS command also results in the need for a diskette swap. To avoid this, use the ASSIGN command as follows:

ASSIGN c: mydisk:c

AmigaDOS asks "insert mydisk into any drive". From now on, all commands to AmigaDOS will be sought from the command (c) directory of this other diskette and AmigaDOS won't ask for the original diskette back for simple commands.

Once you've done this, you'll probably want to type:

CD mydisk:

There are other things that AmigaDOS can assign. If you issue the command

ASSIGN LIST

you will see the other things as well. If you run a program that requires a serial device (modem, printer) or a parallel device (printer), AmigaDOS looks in the directory currently assigned to DEVS: to locate the device. If all of the system directories are on this new main diskette, you can avoid having AmigaDOS ask you to reinsert the original diskette by providing an execute–file on your diskettes that reassigns all devices to that diskette. The contents of this execute file for a diskette named "mydisk" are as follows:

```
ASSIGN SYS:   mydisk:
ASSIGN S:     mydisk:
ASSIGN DEVS:  mydisk:
ASSIGN L:     mydisk:
ASSIGN FONTS: mydisk:
ASSIGN LIBS:  mydisk:
```

To create this execute file, use the command:

COPY FROM * TO reassign

Then type the above ASSIGN lines. After you've typed the last line, enter the key combination CTRL–\ which ends the file. The "*" stands for the keyboard and current CLI window, so this method of creating a file is one possible alternative to using ED or EDIT.

CREATING A NEW CLI

AmigaDOS is a multi–tasking system. You can have multiple windows open at the same time, each with its own current directory and executing separate commands. You create a new CLI by using the command NEWCLI:

NEWCLI

This opens a separate window, with a prompt that identifies the current process. For example, if the first window has a prompt:

1>

then the new cli might have a prompt:

2>

You can move the new window around, make it bigger, make it smaller and so on. To issue commands to the new CLI, click within its window. Now anything you type goes into the window where you clicked the selection button most recently. Try the following:

Click in window 1, then type:

DIR df0:c

Quickly click in window 2, and type:

INFO

Both CLI's will work at the same time to fulfill your requests. This demonstrates the multi–tasking capabilities of the Amiga. Notice that you aren't limited to only two CLI's, you can, if there is memory available, open as many as 20 CLI's.

CLOSING A CLI

You finish with a CLI and close its window with the command ENDCLI. Click the selection button of the mouse in the window for the CLI you wish to close, and type

endcli

Thats all there is to it.

CLOSING COMMENTS

The above series of command descriptions introduces you to the kinds of things you can do with AmigaDOS commands from the CLI. There are several commands that haven't been covered in the above session at all. In addition, most of the commands described above have other "templates" (ways you can enter the commands) and options that haven't been demonstrated.

The AmigaDOS Users Manual contains a reference section that shows the templates for each of the commands in AmigaDOS. You can look at the description for each command in the reference section to find more information. Once you are familiar with the commands, and the forms in which you can use them, the quick reference listing in the back of the that manual will be useful to remind you of the commands that are available.

Additional Examples for the EXECUTE command:

==== EXAMPLE #1 ====

Parameter Substitution by Keyword Name and/or Position

The .KEY (or .K) statement supplies both keyword names and positions in command files. It tells EXECUTE how many parameters to expect and how to interpret them. In other words, .KEY serves as a "template" for the parameter values you specify. Only one .KEY statement is allowed per command file. If present, it should be the first command line in the file.

When you enter a command line, AmigaDOS resolves parameter substitutions for the keywords in two ways: by specification of the keyword in front of the parameter, and by the relative positions of the parameters in the line. Keyword name substitution takes precedence.

For example, the following .KEY statement:

.KEY flash,pan

tells AmigaDOS to expect two parameter substitutions, <flash> and <pan>. (The angle brackets indicate the keyword value to be substituted at execution time.)

Suppose you enter the following command line:

EXECUTE DEMO1 pan somename flash othername

The value "othername" is assigned to <flash>, and the value "somename" is assigned to <pan>.

You can omit the keyword names if the parameter substitutions are in the order given in the .KEY statement. For example, the following statement is equivalent to the preceding one:

EXECUTE DEMO1 othername somename

This is because the values correspond to the keyword order specified in the .KEY statement.

You can also mix the two methods of parameter substitution. Suppose you have a .KEY statement with several parameters, as follows:

.KEY word1,word2,word3,word4

The EXECUTE file processor removes parameter names from the input line to fill the meanings of any keyword values it finds. Then, with any remaining input, it fills the leftover keyword positions according to the position of the input value.

For example:

EXECUTE DEMO2 word3 ccc word1 aaa bbb ddd

The processor assigns ccc to <word3>, aaa to <word1>, and has two parameters left over. Scanning from left to right in the .KEY statement, it finds that <word2> is still unassigned. Thus, <word2> gets the next input word, bbb. Finally, <word4> hasn't been assigned either, so it gets the last input word, ddd.

You can indicate special conditions for parameter substitution, as follows:

.KEY name1/a, name2/a, name3, name4/k

The "/a" indicates that a value must be supplied to fill the parameters for name1 and name2. Values for name3 and name4 are optional, though the "/k" indicates that <name4> (if supplied) must be preceded by the explicit keyword "name4." For example:

EXECUTE DEMO3 fee fie foe name4 fum

If the user does not supply a required parameter (such as name1 or name2 in the preceding example), EXECUTE issues an error message.

As an example of the use of the /k option, suppose you have created an execute-file named COMPILE and it lets you optionally specify a file name to which a printout of the compilation is to be directed. Your .key statement might read:

.key compilewhat/a,printfile/k

If a user enters a line such as:

EXECUTE COMPILE myfile PRINTFILE myprint

the execute-file says the keyword PRINTFILE is optional and need not be supplied, but if used, there must be a value entered along with it. Thus the above line is correct, since myprint is specified as the target output file.

==== EXAMPLE # 2 ====

Assigning Default Parameters and Different Bracket Characters

Note: the following example can be executed as a batch file.

```
.KEY word1

; The  .DEF directive establishes a default value for a keyword ; if the user does not specify
a value on the command line. To ; detect an unsupplied parameter value, you can compare
it to "" ; (two double-quotes in a row). You must perform this comparison ; before
executing any .DEF statement in the execute file.

; You can assign defaults in either of two ways. The first way ; requires that you specify the
default every time you reference ; a parameter, using the "$" operator.

; For example, in the following statement:
```

ECHO "<word1$defword1> is the default for Word1."

; "defword1" is the default specified for word1 and is printed when the ; above statement executes. The second way is to define a default once. ; For example, with the following assignment:

.DEF word1 "defword1"

; you can execute the following statement:

ECHO "<word1> is the default for Word1."

; The output of both of the above ECHO statements will be:

; defword1 is the default for Word1.

; Note that a second use of .DEF for a given parameter has no effect:

.DEF word1 "--- New default ---"
ECHO "<word1> is STILL the default for Word1."

; (The first assignment, "defword1" will be substituted for word1 at ; execution time.)

; ASSIGNING DIFFERENT BRACKET CHARACTERS

; Wherever EXECUTE finds enclosing angle brackets, it looks within them ; to see if it can substitute a parameter. An unsupplied parameter ; with no default becomes a "null" string.

; Suppose you want to use a string that contains the angle bracket ; characters, < and > . You can use the directives .BRA and .KET ; to define substitutes for the bracket characters. For example,

ECHO "This line does NOT print <angle> brackets."
.BRA {
.KET }
ECHO "This line DOES print <angle> brackets." ECHO "The default for word1 is {word1}."

; The first ECHO statement causes the processor to look for the ; parameter substitution for "angle," since that's the current ; meaning of the angle bracket characters. Since "angle" wasn't ; included in the .KEY statement, the processor substitutes the ; null string for it. Then, after the .BRA and .KET directives ; redefine the bracket characters, the second ECHO statement prints ; the characters:

; This line DOES print <angle> brackets.

; The third ECHO statement illustrates that the braces ({ and }) ; now function to enclose keywords for the purpose of parameter substitution.

==== EXAMPLE # 3 ====

File Copy Simulation Showing Command File Structures

The IF statement lets you peform tests and cause different actions based on the results of those tests. Among the possible tests are testing strings for equality and testing to see if a file exists. You can use an ELSE statement with an IF to specify what should be done in case the IF condition is not true. The ELSE statement, if used, is considered a part of the IF statement block. An ENDIF terminates an IF statement block.

The example programs below also use a SKIP statement. The SKIP statement lets you skip FORWARD ONLY within your execute-file to a label defined by a LAB statement.

The IF...ENDIF structure is illustrated by the following short example. It is generally a good idea to test for keywords that might be omitted, or might be entered as null ("") in quotes, as shown below:

```
IF "<word1>" EQ "usage"
  SKIP USAGE
ENDIF
IF "<word2>" EQ ""
   SKIP USAGE
ENDIF
```

Enclosing your parameter substituition words in double-quotes within IF statements prevents EXECUTE from reporting an error if the keyword is omitted. If you omit the double quotes and the value is not supplied, the result can be a line that reads:

```
IF EQ "usage"
```

This produces an error, because the two operators IF and EQ are adjacent. Using double quotes around the keyword replacement indicators results in a line that reads:

```
IF "" EQ  "usage"
```

which is legal.

You can use NOT in an IF statement to reverse the meaning of the test you perform. For example:

```
IF NOT exists <from>
```

There can be nothing on the IF line other than the test condition. For example, the following is incorrect:

```
IF <something> EQ true SKIP DONE
```

The correct form of the above statement is as follows:

```
IF <something> EQ "true"
   SKIP DONE
ENDIF
```

As the example above shows, the string constant tested for need not be enclosed in double–quotes; in the preceding example, either "TRUE" or TRUE is acceptable.

As shown in the sample command file below, IF statements can be nested so that commands can be executed based on multiple true statements. Note that EXECUTE lets you indent to make the nesting of IF statements more readable.

The following sample command file simulates a file copying utility that illustrates certain useful structures in a command file: IF...[ELSE]...ENDIF, LAB, and SKIP.

```
.KEY from, to                    ; Assign parameter list
IF "<from>" eq ""                ; Check for a FROM file being supplied.
  SKIP usage                     ; No file, show user how to use.
ENDIF
IF "<to>"   eq ""              ; Check for a TO file being supplied.
  SKIP usage                   ; No file, show user how to use.
ENDIF

IF NOT exists <from>               ; Check if FROM file doesn't exist
  ECHO "The from file you selected (<from>) could not be found."
  ECHO "Please use the DIR or LIST command and try again."
  SKIP DONE                        ; Note: We can SKIP out of an IF.
ENDIF

IF exists <to>                   ; Check if TO file exists.
  IF "<o>" EQ "O"                  ; Did the user supply "O" on the line ?
    copy from <from> to <to>
    ECHO "Replaced file named <to> with a copy of file named <from>"
    ECHO "Request fulfilled."
  ELSE
    ECHO "Command will overwrite an existing file ONLY if"
    ECHO "the O parameter is specified."
   ECHO "Request Denied"
    SKIP usage                     ; Explain how to use this file
  ENDIF
ELSE
    ECHO "copy from <from> to <to>."
ENDIF
SKIP DONE

LAB usage
ECHO "cp:  usage...."
ECHO "The following copy forms are supported:"
ECHO "  x cp FROM sourcefile TO destinationfile"
ECHO "  x cp FROM sourcefile destinationfile"
ECHO "  x cp sourcefile TO destinationfile"
ECHO "  x cp sourcefile destinationfile"
ECHO "  x cp TO destinationfile FROM sourcefile"
ECHO "  x cp sourcefile destinationfile O"
ECHO "  x cp FROM sourcefile TO destinationfile O"
ECHO "  x cp O FROM sourcefile TO destinationfile"
```

```
ECHO "where: x is short for EXECUTE; cp is the name of"
ECHO "this command file, and "O" means 'overwrite existing file'."

LAB DONE
```

==== EXAMPLE # 4 ====

Sample Looping Batch File

The SKIP command allows only forward jumps. To create a loop structure within a command file, use EXECUTE iteratevely. That is, use the EXECUTE command within the file itself to send execution backwards to a label. The following executable example illustrates looping.

```
;  This file displays five messages:
;     "This message prints once at the beginning. (param1,param2)"
;     "Loop number I."
;     "Loop number II."
;     "Loop number III."
;     "This message prints once at the end. (parm1,parm2)"

.KEY  parm1,parm2,loopcnt,looplabel
FAILAT 20
IF NOT "<looplabel>" EQ ""                    ; Called with label?
   SKIP <looplabel>                           ;  Yes, then loop.
ENDIF

ECHO "This message prints once at the beginning. (<parm1>,<parm2>)"
;    ==== Start of loop ====
LAB 1st-loop
IF "<loopcnt>" EQ "III"                       ; Are we done looping ?
  SKIP loopend-<looplabel>                    ; Yes, unwind.
ENDIF
ECHO "Loop number <loopcnt>I."                ; Go 'backwards' in this file

EXECUTE. loop.sample "<parm1>" "<parm2>" <loopcnt>I 1st-loop

LAB loopend-<looplabel>
IF NOT "<loopcnt>" EQ ""
  SKIP EXIT
ENDIF                                         ; === end of loop ====
ECHO "This message prints once at the end. (<parm1>,<parm2>)"

LAB EXIT
```