MODEL 4
SUBDIRECTORY UTILITIES

By David Goben

Published and Distributed by

Computer News 80

PO Box 680

Casper, WY 82602

MODEL 4
SUBDIRECTORY UTILITIES

Published and distributed by

Computer News 80

PO Box 680 Casper WY 82602

TABLE OF CONTENTS

INTRODUCTION

Up until now, there was no practical way to place more than one directory onto a disk drive efficiently. Until now, the only way to include another directory on a drive was to also allocate large portions of the drive exclusively for the use of each individual directory. Allocatable disk space could not be shared between these different directories, as can be done on MS-DOS and Macintosh systems.

Granted, a hard disk can be (and most times due to large capacity drives, *must* be) divided into more than one partition, each of which can be used as a separate disk drive (most often called logical drives). The limiting problem with this process was that the separate directories are not interactive in that each directory is considered to be a master of its own little universe -- files from separate directories cannot share the same allocatable space without scrambling each others data. Thus each directory requires that it also have a large chunk of the disk's capacity available for its own exclusive, private use.

True, the need for logical hard disk partitions is essential in most cases for not only hard disk space economy, but for the fact that a single logical disk drive, as opposed to a total hard disk drive, has a maximum allocation capacity of 12,992 kilobytes (this would be a like using a partition that had 8 32-sector granules assigned to a logical cylinder, and 203 maximum logical cylinders -- 8 x 32 x 203 x 256 bytes = 13,303,808 bytes / 1024 = 12,992K). Thus even a small 15 megabyte hard disk would need to be divided into more than one partition in order to take full advantage of all its available room.

In pursuit of adding additional directories to drives without resorting to additional partitioning, one method introduced by Logical Systems Incorporated (LSI) was with the introduction of the DiskDISK utility (now owned by MISOSYS, Inc.; and also called SubDISKs on their hard disk support packages). A DiskDISK worked much like a partition in that a large portion of disk space was required to be preallocated to each DiskDISK directory for its own exclusive use. Again, a DiskDISK directory was a lone master in its individual universe, just as a hard disk partition was.

The major advantage of the DiskDISK-type partitioning idea is simple: A disk directory is capable of containing only a maximum of 256 files. And so as is most often the case, a person will normally run out of allowable file names *long* before they run out of actual free disk space. But by using only a single file and designating it as a DiskDISK partition file, it in turn can allocate up to 256 total files for its own reserved block of disk space. Additional DiskDISKs can also add up to 256 files each. This method allowed the user to therefore take back and be able to use the free disk space they might not otherwise be able to access.

The major disadvantage of the DiskDISK idea is the fact that so much disk space must be pre-allocated to each DiskDISK, since each block of space must be equal to the number of sectors assigned to an actual disk drive the DiskDISK is set up to emulate. For example, if a DiskDISK was created which simulated an 80 cylinder double-sided double-density disk drive, this would result in 2 sides x 18 sectors-per-side x 80 cylinders, or 2,880 sectors (720K) of disk space to be allocated to it. To save space, you should only create the DiskDISK to the size you need, which can be done when you select format, side counts, and cylinder counts. Unfortunately you cannot always anticipate exactly how much space you will require, especially if you do the logical thing and designate each DiskDISK to a specific application (most often you would create a DiskDISK partition to hold a related group of files), whose required size 6 months from now cannot always be accurately anticipated. If you later require a larger or even a smaller DiskDISK partition, you will have to create a new one with an appropriately allocated chunk of disk space. And maybe again in another 6 months.

SUBDIRECTORIES; THE IDEAL SOLUTION

For me, the ideal solution is to allow more than one directory to share allocatable disk space with other directories assigned to the same logical drive. Obviously they should never be able to over-write each other's files, but they should be able to share the same block of memory allocated for the drive. This means that they would be individual directories, but they would share one common disk space allocation table; thus when one directory needs to reserve a section of the disk for a file, the other directories will not later try to also use that same space.

I have chosen to call these additional directories *SubDirectories,* because they are actually branches off from the main disk directory, which can be referred to as the *root directory.*

The greatest advantage to this system is that there would no longer be a need to pre-allocate large chunks of disk space for a directory; rather the only space that would have to be allocated would be the space required for the separate directory itself. To emphasize this; the *maximum* disk space which would be required to be allocated to a directory file would be 34 sectors (directories *never* use more than 34 sectors, though due to granule sizes -- a granule is the smallest allocatable chunk of disk space, which normally consists of a number of disk sectors -- a directory may *seem* to use more than the maximum 8.5K it can possibly use).

Another advantage is that these SubDirectories could not only be accommodated on hard disks, but also on floppy disks as well (a feat not practical with the DiskDISK method except in very small blocks).

Still another advantage is that a SubDirectory can itself contain another SubDirectory (referred to as a *child directory* -- a SubDirectory that exists within another directory would call the directory above it its *parent directory).*

The uses for SubDirectories are many. For example, you could devote each SubDirectory to a particular grouping of files which are related to each other, such as is the case with document files, database files, spreadsheet files, etc. By placing related files into a SubDirectory, you would then have a common directory to access them from.

The most obvious advantage to the SubDirectory idea is the addition of many more filenames, which is especially important on hard disk systems that have immense data space available. This coupled with the total elimination of the need to preallocate huge chunks of the disk's surface to an individual directory whose allocation requirements may change drastically over time, and with the ability of the user to group related files in an individual centralized directory is the whole idea behind the MODEL 4 SUBDIRECTORY UTILITIES package, and why it was created.

PACKAGE CONTENTS

The disk accompanying this manual contains a number of important files which should be copied to a backup copy of the system disk you use every day, or onto your system hard disk partition. Each file is listed below, along with a brief explanation of its use:

CHDIR/CMD	*Used to activate a SubDirectory, assigning it to a selected logical drive.*
MKDIR/CMD	*Used to create a SubDirectory on a logical drive.*
REMDIR/CMD	*Used to remove a SubDirectory from a logical drive.*
SCAT/CMD	*A utility to display the contents of a SubDirectory in standard CAT format.*
SDIR/CMD	*A utility to display the contents of a SubDirectory in standard DIR format.*
UNREMOVE/CMD	*A file recovery utility to "unremove" files that have been removed via REMOVE and PURGE.*
SWAP/CMD	*A utility to swap logical drive numbers between drives.*
README/CMD	*A utility to display or print a README/TXT file.*
README/TXT	*A text file containing updated information that has been released after the documentation had been printed.*

COPYING THE FILES

The first thing you should do is copy the files on the distribution disk to your working system disk, or onto a partition on your hard disk. You can copy these files using the **BACKUP** command.

To copy the files from drive :1 to a working copy of your system disk in drive :0, place the **MODEL 4 SUBDIRECTORY UTILITIES** disk into drive :1 and enter the following command:

BACKUP -README:1 :0

This command will copy all the files except the README files to your working system disk in drive :0. Be sure that any write-protect tab is removed from your system disk. Also, *NEVER* use an original system disk as your working system disk (you wouldn't want people to think you were inbred, would you?).

The next step is to read the README file. So after the backup is completed, enter the command:

README

You will be asked if you wish to display or print the file. Type the "D" key to display it. If it has a lot of information, you may wish to turn on your printer and run the program again, and select "P" for print.

USING THE UTILITIES

The remainder of this manual explains how to use the utilities, in the order you will probably use them. Be sure to pay careful attention to all the rules and cautions. *Never* try using a utility before reading about it in this manual. You may just misuse it, and have only yourself to blame for any foul play.

Another note in regard to getting help using the utilities. Entering the command name followed by a space and the a question mark (?) will present you with a list of help and options you can use to take advantage of the program in question.

USING MKDIR

MKDIR allows you to create a SubDirectory file. Only through the use of this utility can SubDirectories be created. To create a SubDirectory, you must select a *path* you wish for it to exist within. A path is a term which would indicate the drive and directory name you wish to call the new directory. It can also include other directory names if it will exist within those directories.

For example, suppose you wanted to create a directory called MYFILES on a disk in drive :1. You would enter the command:

MKDIR MYFILES:1

This command will create a file called MYFILES/DIR on drive :1.

Since SubDirectories can exist within other SubDirectories, we must have a way of specifying the path to our target location. Suppose that the SubDirectory MYFILES exist on drive :1 and we want to create a directory within MYFILES called DATFILES. We would use the command:

MKDIR \MYFILES\DATFILES:1

Notice the use of the backslash "\" character. This is used to separate SubDirectory names from each other. Please note that the first backslash before the MYFILES designation is optional, but makes our intension more obvious.

Be aware that if the MYFILES SubDirectory had been active at the time, you could have also created the DATFILES SubDirectory within it by specifying the logical drive which had been assigned to the MYFILES directory. For example, suppose that the MYFILES directory was currently active as drive :6. You could have *alternatively* created the DATFILES directory using the command:

MKDIR \DATFILES:6

Notice that the optional backslash character had been used in this example.

Had you simply entered:

MKDIR

MKDIR would have asked you for a name of a SubDirectory to create. You would respond by entering a name or a path.

If you did no specify a drive number to create the SubDirectory on, MKDIR will prompt you for one.

Please note that /DIR is a default file extension, and you can specify your own if you wish, but for most purposes you should not tamper with it, as by allowing the default to be used, MKDIR and the other utilities will be much easier to use.

MKDIR offers *one* parameter: the question mark (?). By entering the command ***MKDIR ?,*** a line of help will be displayed, showing you the syntax required for using MKDIR.

MKDIR ERROR MESSAGES

MKDIR delivers the following error messages if things are not running smoothly:

passwords not allowed: You cannot specify a password when creating SubDirectories. All SubDirectories have a default password of ".DIRECTRY", and are listed in a directory listing as protected with the NO ACCESS level. Exercise caution when you have passwords disabled due to a possible system patch. Losing your directory will also lose all the files stored within it.

Invalid name specification: The user entered an invalid name or path.

Invalid syntax during JCL: Bad information was provided to MKDIR from a Job Control File (JCL). Since a JCL cannot be reprompted for correct data, MKDIR will abort to DOS and JCL processing will stop.

Too many file extents. Removing SubDirectory: When a SubDirectory was created, the allocation given it by DOS had more than 4 file extents. This would make for a very inefficient SubDirectory. The *ideal* number of extents is *one.* You may try using the CREATE command to create a file called TEMP/TMP on the drive and allocated several kilobytes worth of space to it, use MKDIR to try creating you SubDirectory again, and then using REMOVE to remove the TEMP/TMP file.

Close error. Removing SubDirectory: A problem occurred when MKDIR attempted to close the newly created SubDirectory. Since its data would be corrupted, it was removed. This may occur when a bad sector is on the target disk's directory. Back up the files from the disk onto a new disk, and try again with the new disk.

A Selected directory does not exist: A directory name included in a path does not yet exist. If you are attempting to create a SubDirectory within another SubDirectory, the SubDirectory you wish to create it within must already exist.

Invalid drive specification: The user specified an invalid drive to create the SubDirectory on.

Selected drive not active: The drive specified by the user to create the SubDirectory on is legal, but the drive is not currently active.

Selected drive not valid, or no disk present: The user specified a disk drive which does not have a disk which has a recognizable format, or the drive does not have a disk in it, or the drive door is not closed.

Selected directory already exists: The user attempted to create a SubDirectory which already exists in the specified path.

USING CHDIR

CHDIR allows you to make a SubDirectory accessable for file processing. When a SubDirectory is not active, it is considered just another file stored within its parent directory. Although when active it will still be considered just a file in its parent directory, it is seen as a fully functional logical disk to the drive number (slot) it is assigned to. To activate a SubDirectory, you must select the path for the Disk Operating System (DOS) to find it, and select a drive slot for it to be assigned to. The path can also include other directory names if it will exist within those directories.

For example, suppose you wanted to activate a directory called MYFILES on a disk in drive :1, and you wanted to assign it to logical drive :6. You would enter the command:

CHDIR :6 MYFILES:1

This command will activate a file called MYFILES/DIR on drive :1, and make this directory accessible *as* a directory through drive :6. Notice that a DIR or CAT command directed at the logical drive assigned to this subdirectory will display the name and date of the root directory, because the SubDirectory shares the allocation table of the root directory, where disk name, date, and various other items concerning formatting information are stored. However, the first thing you should note is that a directory listing of the drive will show only files which are assigned to the SubDirectory.

As had been said before, since SubDirectories can exist within other SubDirectories, we must have a way of specifying the path to our target location. Suppose that the SubDirectory MYFILES exist on drive :1 and we want to create a directory within MYFILES called DATFILES. We would use the command:

CHDIR :6 \MYFILES\DATFILES:1

Again, as had been said before, if the MYFILES SubDirectory had been active at the time, you could have also created the DATFILES SubDirectory within by specifying the logical drive which had been assigned to the MYFILES directory. For example, suppose that the MYFILES directory was currently active as drive :5. You could have *alternatively* created the DATFILES directory using the command:

CHDIR :6 \DATFILES:5

Had you instead chosen to activate the DATFILES directory on the *same* drive as the MYFILES directory was active on (:5), using the command:

CHDIR :5 \DATFILES:5

The DATFILES directory would have been made active as logical drive :5, and the MYFILES directory would have been rendered inaccessible, just as though you had entered the command: ***CHDIR :5 \MYFILES\DATFILES:1.***

Please take note of the fact that the drive number tagged *onto* the last (rightmost) subdirectory in a path indicates the logical drive to start the search from. Therefore, to reiterate for clarification, if DATFILES is a SubDirectory of the MYFILES directory, and the MYFILES directory is active, you can specify the DATFILES directory directly from the drive number assigned to the active MYFILES directory, or by specifying a path of \MYFILES\DATFILE through the drive number assigned to the root directory.

If you had entered the CHDIR command without specifying the logical drive number to assign a SubDirectory to, or you had not specified a SubDirectory (path), you would be prompted for them.

Please be aware that you *cannot* assign the logical drive number assigned to the SubDirectory's *root directory* as the logical drive to activate the SubDirectory on. The root directory must be accessible at all times, because actual reads

and writes to the SubDirectory are accomplished by making accesses to the root. If you attempt to use the root drive number, CHDIR will alert you to the problem and abort.

BACKING DOWN SUBDIRECTORIES

You can back down a branch of SubDirectories in one of two ways.

The first is to simply use CHDIR to activate the parent directory of the current directory in the chain up from the root directory.

The second is to use the 2-dot ".." operator. This is used as a shorthand notation for the parent directory of the current directory (some may note that this idea was borrowed from MS-DOS).

Therefore, if the DATFILES directory existed within the MYFILES directory, and the DATFILES directory was currently active as drive :5, you could back down the directory chain to DATFILES' parent directory (MYFILES) and make it active by issuing the command:

CHDIR :5 ..

Afterward the MYFILES directory will be active on logical drive :5. If you again issued the command, and the parent directory of MYFILES was the root directory, then the SubDirectory branch would be disconnected, and the SubDirectory memory modules assigned to logical drive :5 would be released to DOS. Also, if no other SubDirectories are active, then the main support driver (called $SDIR to the system) will also be removed, since it will no longer be needed.

CHDIR PARAMETERS

CHDIR offers you three parameters. The first is the question mark, and is used in the form: ***CHDIR ?.*** This will display a small screen of help, showing you the required syntax for using CHDIR.

The other two parameters must be enclosed within parentheses "()", and cannot be used together.

The first of these two is ***WHO.*** The WHO parameter allows you to display current drive and directory assignments. If we currently had the DATFILES directory, which exists within the MYFILES directory on drive :1, assigned to logical drive :5, the command ***CHDIR (WHO)*** would display:

Current drive(s) are SubDirectories:
—> Drive :6 assigned to \MYFILES\DATFILES:1

The second parameter is ***DISABLE.*** This will deactivate *all* currently active SubDirectories if no logical drive number is specified. In this case *NO* attempt will be made to recover the memory space used by the driver(s). Specifying a drive number assigned to a SubDirectory will cause CHDIR to attempt to return the memory space used by the driver to the system.

For example, the command:

CHDIR (DISABLE)

Will disable all active Subdirectories. The space occupied by any active SubDirectories will not be returned to the system. This is sort of a "panic button" command.

However, if a SubDirectory were assigned to drive :5, and you entered the command:

CHDIR :5 (DISABLE)

Then CHDIR will disable only that drive (even if other SubDirectories are active) and attempt to return its memory space to the system. If it is the only, or last SubDirectory drive to be deactivated, the memory space occupied by the main driver will also be returned to the system. If you have more than one SubDirectory active, and you wish to disable them and return the memory space they occupied to the system, be sure to disable them in the reverse order that logical drives were activated, otherwise memory space may be "blocked" by other memory modules, and although the drives will be deactivated, their memory space may not be recoverable.

Please note that you can abbreviate these parameters to their first characters "W" and "D".

CHDIR ERROR MESSAGES

CHDIR delivers the following error messages if things are not running smoothly:

passwords not allowed: You cannot specify a password when creating SubDirectories. All SubDirectories have a default password of ".DIRECTRY", and are listed in a directory listing as protected with the NO ACCESS level. Exercise caution when you have passwords disabled due to a possible system patch. Losing your directory will also lose all the files stored within it.

Invalid name specification: The user entered an invalid name or path.

Invalid syntax during JCL: Bad information was provided to MKDIR from a Job Control File (JCL). Since a JCL cannot be reprompted for correct data, MKDIR will abort to DOS and JCL processing will stop.

Parameter error: The user specified an invalid parameter.

Cannot specify more than one parameter: The user attempted to use both the WHO and DISABLES parameters together.

Cannot specify WHO with other specifications: The user specified the WHO parameter, but also include a drive number or a directory path. You can only use the WHO parameter in the form: ***CHDIR (WHO).***

Not a subdirectory drive: The user attempted to disable a drive using the DISABLE parameters, and specified a logical drive number which was not assigned to a SubDirectory.

SubDirectory cannot overwrite its source drive: The user attempted to assigned the logical drive number assigned to its root directory.

Invalid drive specification: The user specified an invalid drive to create the SubDirectory on.

Cannot reassign drive used by other SubDirectories: The user attempted to assign a logical drive number to a SubDirectory which was already assigned to another SubDirectory, but not within its path chain. You can only reassign a drive number when it is in a path stream that is common to the old SubDirectory and the new one. Otherwise you must first disable the drive before trying to use the logical drive with a new SubDirectory.

No SubDirectories active in system: The user specified the WHO parameter, but no SubDirectories were currently active.

Cannot recover drive's module space: A SubDirectory path was disabled, but another module blocked the module so that its memory space could not be released back to the system.

Cannot recover main driver module's space: The main driver for the SubDirectories was blocked by another memory module so that its space could not be returned to the system.

Cannot find drive's module. Must be hidden by *UNKNOWN* module: CHDIR could not find a drive module that its knows has to exist. It may be "hidden" by a memory module that does not have a proper memory header.

Selected drive is not a SubDirectory: The user appended a drive number on a directory path that does not contain a specified directory.

USING REMDIR

REMDIR allows you to safely remove a SubDirectory. You should ***never*** attempt to remove a SubDirectory using the REMOVE or PURGE commands! If the SubDirectories are not empty, and you use REMOVE or PURGE on it, then *all* the files stored within the SubDirectory, *including* any SubDirectories and their files stored within it, will be lost, but the disk space allocated to them will be retained; technically locking out the space from furture use.

REMDIR checks to ensure that the selected SubDirectory path is first empty. If it contains no files, then the directory will be removed. However, if it is not empty, then it will warn you that it contains files. You should always first remove all the files from the SubDirectory before attempting to remove it.

For example, to remove a Subdirectory on drive :5 called DATFILES, you would use the command:

REMDIR DATFILES:5

If DATFILES is stored within a directory called MYFILES, and MYFILES is a currently *active* SubDirectory assigned to drive :5, but contained on a disk in drive :1 (the root directory), you could remove it using the above command to extract it directly from the MYFILES directory, or you could use:

REMDIR \MYFILES\DATFILES:1

to remove it from the root directory level.

Again, if DATFILES contains other files (except BOOT/SYS and DIR/SYS; two files which will always be present on a directory), then REMDIR will not allow you to immediately remove the directory. This is for your own protection.

CLEARING A SUBDIRECTORY

If the selected SubDirectory indeed contains files, you will need to remove them. You can use the REMOVE command to remove files, using the format:

REMOVE filename

where *filename* will consist of the name of the file to remove, including any possible extension, plus the drive number assigned to the SubDirectory, if you choose. Also include passwords if they require them. If you have more than one file to remove, you can include them on the REMOVE command line by separating each filename with a space, in the format:

REMOVE filename1 filename2 filename3...

If there are a lot of files, you can use the PURGE command in the form:

PURGE :d (INV,SYS)

where ***:d*** represents the logical drive assigned to the selected *active* SubDirectory. For example, suppose that the root directory on drive :1 contains an active SubDirectory called MYFILES/DIR (referred to without the extension as MYFILES), which is assigned to logical drive :5. In turn, MYFILES contains three files called DBASE/BAS, MAILLIST/BAS, and DATFILES/DIR (DATFILES). Finally, the DATFILES directory contains three files called LIST1/DAT, LIST2/DAT, and LIST3/DAT.

Were we to attempt to use REMDIR to remove the DATFILES SubDirectory, we could do so using two possible ways. The first is through the root directory:

REMDIR \MYFILES\DATFILES:1

This is the usual method. The other method would be to go through the MYFILES directory which is assigned to logical drive :5:

REMDIR \DATFILES:5

In either case, since the DATFILES SubDirectory contains files, you will be altered with the message:

Selected SubDirectory contains data!!! Remove (Y/N)?

Under normal circumstances, you should *always* answer with "N" for NO. The only time you may wish to answer with "Y" for YES is when the only files stored within the SubDirectory contain bad sectors which you wish to be locked out from use by the system. By not removing the files and going ahead and removed the SubDirectory, the disk space allocated to the files stored within the SubDirectory will not be returned to the system for use, but will instead lock out the allocated space from future use, since the actual files the space is allocated to no longer exist *as* accessible files.

To clear the files from the DATFILES directory, you must first activate it. You would use CHDIR for this. You can activate it through drive :5 (which is currently active as MYFILES) by using the command:

CHDIR :5 \DATFILES:5

or by going through the root directory:

CHDIR :5 \MYFILES\DATFILES:1

You would then remove the three files stored there using the command:

REMOVE LIST1/DAT:5 LIST2/DAT:5 LIST3/DAT:5

Or you could use the command:

PURGE :5

We did not use the INV or SYS parameters because we can assume that the files we stored in that SubDirectory were normal data files, not system files, or attributed to be invisible. Technically, since we know that DATFILES contains no child directories, we *could* have used the command:

PURGE :5(Q=

to purge all files on the drive without being queries for each file. However, you should first be sure that SubDirectories *are not* stored there before doing so (if there were, you should *immediately* use the UNREMOVE command to restore the accidentally purged directories).

Once the files are removed, you can back down to the MYFILES SubDirectory by using the command:

CHDIR :5..

You can then remove the directory using the commands:

REMDIR \MYFILES\DATFILES:1 or ***REMDIR \DATFILES:5***

REMDIR ERROR MESSAGES

REMDIR delivers the following error messages if things are not running smoothly:

passwords not allowed: You cannot specify a password when creating SubDirectories. All SubDirectories have a default password of ".DIRECTRY", and are listed in a directory listing as protected with the NO ACCESS level. Exercise caution when you have passwords disabled due to a possible system patch. Losing your directory will also lose all the files stored within it.

Invalid name specification: The user entered an invalid name or path.

Invalid syntax during JCL: Bad information was provided to MKDIR from a Job Control File (JCL). Since a JCL cannot be reprompted for correct data, MKDIR will abort to DOS and JCL processing will stop.

A selected SubDirectory does not exist: The user specified a directory name in the path which does not exist in the directory chain.

Invalid drive specification: The user entered a logical drive number which was not valid.

USING SCAT AND SDIR

SCAT allows you to display a CAT-type listing of the contents of a SubDirectory which is not currently open. SDIR allows you to display a DIR-type listing of the contents of a SubDirectory which is not currently open.

Suppose you wanted to display a CAT-type listing of the files contained in a SubDirectory called DATFILES, which is in turn contained in a SubDirectory called MYFILES on the root directory in drive :1. You could display the contents of DATFILES using the command:

SCAT \MYFILES\DATFILES:1

If the MYFILES SubDirectory was currently active and associated with drive :5, you could also display the contents of DATFILES using the command:

SCAT DATFILES:5

or

SCAT \DATFILES:5

SDIR can be used in exactly the same way as SCAT.

You may notice that when a directory listing is displayed that the drive number displayed for the directory may be at odds with the directory's current drive path, or seems arbitrary to any "formula". SCAT and SDIR will temporarily utilize *any* drive number that is not drive :0 and is not the the drive associated with the selected SubDirectory's root directory. Thus if the root directory is associated with any drive from :1 through drive :6, then drive :7 will be associated with the display of the contents of the SubDirectory. If the root directory is on drive :7, then drive :6 will be used. Once the display is completed, then the drive used will be returned to its original driver.

Note that the directory name displayed will be the same as that for the root directory, as this information is stored within the allocation table, which is commonly used by all SubDirectories associated with a root directory.

Additionally, please make note of the fact that you can use any and all optional parameters associated with the standard DIR and CAT commands.

If you simply enter the SCAT or SDIR commands without specifying a directory path, then you will be prompted for one.

SCAT AND SDIR ERROR MESSAGES

SCAT and SDIR deliver the following error messages if things are not running smoothly:

passwords not allowed: You cannot specify a password when creating SubDirectories. All SubDirectories have a default password of ".DIRECTRY", and are listed in a directory listing as protected with the NO ACCESS level. Exercise caution when you have passwords disabled due to a possible system patch. Losing your directory will also lose all the files stored within it.

Invalid name specification: The user entered an invalid name or path.

Invalid syntax during JCL: Bad information was provided to MKDIR from a Job Control File (JCL). Since a JCL cannot be reprompted for correct data, MKDIR will abort to DOS and JCL processing will stop.

A selected SubDirectory does not exist: The user specified a directory name in the path which does not exist in the directory chain.

Invalid drive specification: The user entered a logical drive number which was not valid.

USING UNREMOVE

UNREMOVE allows you to recover files (such as accidentally deleted SubDirectories) that have previously been deleted using the REMOVE or PURGE functions.

When REMOVE or PURGE are used to eliminate a file, a single bit is reset in the file's File Directory Entry (its "alive" bit), its "hash" code is removed from the directory's Hash Index Table (a table used to speed finding a file in the directory), and the disk granules (a granule is the smallest block of space allocatable to a file) are released for re-use by the system. UNREMOVE can restore a file by turning the file's "alive" bit back one, restoring its "hash" code in the Hash Index Table, and re-allocating its previously released granules back to it. If the file had a password, it will be restored by UNREMOVE. Notice that you do not need to specify this password in the UNREMOVE command line, because it is already intact within the "dead" directory entry.

The success of UNREMOVE is only possible when 1) the file's directory entry has not been over-written by another new file (or also one of its possible extended directory entries in the case of an expanded file), and 2) if its granules have not already been re-allocated to another file. If either of these two thing are not met, then the file is considered unrecoverable.

UNREMOVE can also be used to list a directory of removed files on a disk.

The syntax for using UNREMOVE is:

UNREMOVE [:]d\|filespec1[filespec2...]

By specifying only a drive number (notice the the colon is optional), a directory of files that have been removed will be displayed. If any of the listed files are surrounded by square brackets "[ ]", this indicates that the file appears to be fully recoverable in that the granules that had been assigned to it are currently still unallocated. If a file is displayed that is not surrounded by square brackets, this indicates that even though the file's directory entry still exists in the directory (but in a "removed" state), some or all of the granules that had been previously assigned to it are now being used by another file. This renders a file unrecoverable.

By specifying at least one file specification, UNREMOVE will search the drive specified in the filespec (if no drive number is supplied, then UNREMOVE will default to searching drive :0) and locate its entry in the directory. It will then check the index in the entry for the location(s) of the granules that had been assigned to it. It will then check the granule allocation table and see if ALL of these granules are currently unassigned. If they are, then the granules will be re-allocated, the proper "hash" code for the file will be generated and placed in the proper location in the Hash Index Table, and the "alive" bit in the file entry will be set. In the case of exceptionally long or fragmented files which forces the directory to maintain more than one directory entry for the file (subsequent entries are called File Extended Directory Entries), UNREMOVE will process them and their corresponding entries in the Hash Index Table as required.

UNREMOVE should be used as soon as you realize that a desired file has been accidentally removed, purged or killed. This ensures that the granules that were assigned to the desired file have not been over-written with data and then later released back for re-allocation again. What this means is that if you remove a file, write data to another file that may by chance use a portion of the de-allocated space, thus over-writing the old data, and then later this space is again released either because the data file was removed or shortened, then even though using UNREMOVE may successfully recover the old file, the data that it had contained may be corrupted between the interval of initial removal and recovery. Thus the emphasis on recovery as soon as possible must be stressed. Though in general you will "luck out" and get it back OK, you can never be too careful. Remember one of Murphy's Laws: The more important the lost data file; the more impossible it is to recover it. Therefore never procrastinate about recovering a file and putting it off until later. In the interim you may forget and first over-write its data before you read your reminder list. With this in mind, you should never have any problems.

UNREMOVE EXAMPLES

UNREMOVE :1 will list a directory of all files on the disk in drive :1 that have been previously removed or purged. Those filenames displayed within brackets indicate files that can be recovered using UNREMOVE.

UNREMOVE MYFILES/DIR:0 will recover the file MYFILES/DIR on drive :0.

UNREMOVE THISFILE/DAT:1 THATFILE/OVL will recover THISFILE/DAT on drive :1 and THATFILE/DAT on drive :0 (notice that here UNREMOVE will default to drive :0 when no drive specification is supplied).

UNREMOVE ERROR MESSAGES

UNREMOVE delivers the following error messages if things are not running smoothly:

No removed files were found on drive :d: UNREMOVE could not find any files that had been previously removed or purged on the drive number specified.

Granules for FILESPEC are ALREADY allocated. Cannot recover it!: The specified FILESPEC cannot be restored because all or a portion of the disk space that had been assigned to it has been allocated to a different file or files.

FILESPEC was not found!: The specified FILESPEC was not found on the desired drive. This may be due to either the file having never been on that disk, or its directory entry has been over-written by another file entry.

FILESPEC is ALREADY active!: The specified FILESPEC was found on the disk, but it is currently marked as being already active, or "alive".

Any other error reports are DOS errors, and you must refer to your DOS manual to get a full explanation of what they represent.

When an error is encountered, the UNREMOVE function will abort. Therefore any subsequent files to be recovered, listed after the error-causing file on the command line, will be ignored.

USING SWAP

SWAP is a program that allows you to switch logical drive numbers. Unlike other SWAP utilities, this version has been written so that it will *also* properly swap logical drive numbers that one or the other may be assigned to a SubDirectory.

Using SWAP is easy. The syntax for using the utility is:

SWAP :d1 :d2

Where ***:d1*** represents one logical drive number, and ***:d2*** represents the other logical drive number. Note that the leading colons for each drive number is required.

For example, to swap logical drives :1 and :5, you would use the command:

SWAP :1 :5

Afterward the directory and disk associated with drive :1 will now be assigned to drive :5, and the directory and disk previously associated with drive :5 will now be assigned to drive :1. The advantage of using this program over the system command of **SYSTEM (DRIVE=d1,SWAP=d2)** is that this version of SWAP/CMD will search for active SubDirectories which may be associated with either of the selected drives, so that changes within the SubDirectory drives can *also* be swapped (otherwise the system may crash). Using SWAP is therefore safe, and is handy even if you do not have SubDirectories active, but simply want to swap a couple of drive numbers.

If SWAP detects that the user is swapping the SYSTEM drive with another drive, it will check the new drive which will become the SYSTEM drive for containing SYSTEM files. If it does not, then the user will be prompted to load a SYSTEM disk into the required drive. Once a SYSTEM disk is loading into the drive and the user presses the ENTER key, SWAP will activate the swap and return control to the user.

SWAP ERROR MESSAGES

SWAP delivers the following error messages if things are not running smoothly:

Source and destination drives are the same: The user attempted to swap drives using the same drive number. This is redundant, and is not allowed.

Can't prompt for a SYSTEM disk during JCL: An executing JCL file using the SWAP command swapped the system drive number with another drive, and the other drive was found to not have system files present. Prompting for a swap in the system drive is not allowed when a JCL file is executing.

Drive number specification error: The user failed to enter a valid drive number, or failed to enter a leading colon before a drive number, or did not enter two different drive numbers.