DataGeneral

Customer Documentation

Using the DG/RDOS Command Line Interpreter

Using the DG/RDOS Command Line Interpreter

093-000471-00

For the latest enhancements, cautions, documentation changes, and other information on this product, please see the Release Notice (085-series) supplied with the software.

Ordering No. 093-000471

Printed in the United States of America
Revision 00, October 1986

NOTICE

Using the DG/RDOS Command Line Interpreter
093-000471

Revision History:	Effective with:
Original Release - October 1986	DG/RDOS Rev. 2.00

Contents

Figures

Figure

Tables

Table

Preface

This manual introduces and describes the Command Line Interpreter for DG/RDOS revisions 2.00 and higher. The Command Line Interpreter, known as the CLI, serves as an interpreter between you and the operating system. With the CLI, you communicate with DG/RDOS to start and stop the system, manage files, control input and output, and develop programs. You also use CLI commands to invoke system utilities, such as the SPEED text editor or the IMOVE backup program.

Organization of the Manual

Chapter 1 provides a basic introduction to the syntax and uses of the DG/RDOS CLI.

Chapter 2 explains the DG/RDOS file system, including how to create and manage files and directories.

Chapter 3 describes and gives examples of all of the CLI commands. After you have become familiar with DG/RDOS, this chapter will be your primary reference.

Chapter 4 explains how to execute multiple CLI commands with the use of macros and indirect files.

Chapter 5 provides a brief description of the BURST and IMOVE utilities used to back up and move files.

Appendix A, CLI Command Summary, provides a brief summary of all commands.

Appendix B lists and explains CLI error messages.

The glossary defines pertinent terms. When you see an unfamiliar term, check the glossary.

Syntax Notation

Any mention of the *system console* refers to the operator's terminal--the one from which you bring up DG/RDOS. Other consoles on the system are called *user terminals.*

The following syntax notation is used throughout this manual in descriptions of command and statement syntax:

UPPERCASE	Uppercase indicates portions of statements or commands that must be entered exactly as shown.
lowercase	Lowercase indicates that you must enter some argument (such as the name of a file).

[ ]	Brackets indicate that the enclosed entry is optional.
{ }	Braces indicate that you have a choice among two or more entries. At least one of the entries enclosed in braces must be chosen unless the entries are also enclosed in square brackets.
< >	Angle brackets indicate that the enclosed key should be pressed.
...	Ellipses indicate that an entry may be repeated as many times as needed or desired.

All other punctuation, such as commas, colons, slash marks, and equal signs, must be entered exactly as shown.

The following special symbols are also used:

<NL>	The key that enters DG/RDOS commands. It is normally the key labeled NEW LINE. If your keyboard does not have a NEW LINE key, you enter DG/RDOS commands with the carriage return key, labeled CR or RETURN.
CTRL-character	A control character, which is used to overide CLI processing. You enter a control character by pressing and holding the key marked CTRL and then pressing the appropriate key, such as CTRL-S.

All numbers are decimal unless indicated otherwise.

We show commands in UPPERCASE, but you may type them in lowercase, UPPERCASE, or any combination.

In examples we use:

THIS TYPEFACE TO SHOW YOUR ENTRY
`This typeface, in lowercase or UPPERCASE, for system queries and responses.`

Changes for DG/RDOS Rev. 2.00

If you have been using previous revisions of DG/RDOS and the predecessor of this manual *RDOS, DOS, and DG/RDOS Command Line Interpreter,* 069-400015, please note the following changes:

- This manual is specific to DG/RDOS. All material relating to DOS or RDOS has been removed.
- CLI changes from DG/RDOS Rev. 1.09 through 1.30, that were in documentation update files on the release media, are included in this manual.
- Two new CLI commands, ABBREVIATE and XEQ, have been added.

- BURST is now a stand-among utility and is discussed in this manual.
- BURST, IMOVE, and LABEL (which was introduced in Rev. 1.30) have been enhanced.

Related Documents

The document set for DG/RDOS Rev. 2.00 and higher consists of this manual plus the manuals listed below. If you have used DG/RDOS previously, please note the changes in the set.

How to Generate and Run DG/RDOS, 093-000470, provides complete instructions on starting and stopping your system, handling an abnormal shutdown, backing up and maintaining files, and adding new hardware or software to your system.

Introduction to RDOS, 069-400011, describes the fundamentals and summarizes the features and utilities of the RDOS operating system. Since RDOS is the predecessor of DG/RDOS and the two are functionally similar, the manual is also useful as an introduction to DG/RDOS.

RDOS/DOS Superedit Text Editor, 069-400017, explains the editor which is also supplied with DG/RDOS.

RDOS/DOS Assembly Language and Program Utilities, 069-400019, describes the Extended Assembler, Macroassembler, Extended Relocatable Loader, and Library File Editor utilities that aid you in programming DG/RDOS.

RDOS/DOS Debugging Utilities, 069-400020, describes five utilities that assist you if you will be editing and debugging programs.

RDOS System Reference, 069-400027, describes and explains system features, calls, and user device driver implementation needed for assembly language or high-level programming with DG/RDOS.

RDOS/DOS User's Handbook, 093-000105, is a pocket-sized reference to CLI and utility program commands, and it also contains system error messages. Most of the information in this reference also applies to DG/RDOS.

–End of Preface–

Chapter 1
Introduction to the CLI

This chapter explains how to use the DG/RDOS Command Line Interpreter. It describes the syntax of command lines, including switches and the use of semicolons and angle brackets to repeat commands. It also explains CLI special characters and command abbreviations, and how to end a CLI session and shut DG/RDOS down.

Before you can use the CLI, the DG/RDOS operating system must be running on your computer. When you load DG/RDOS, the operating system starts the CLI. The CLI is available when the DG/RDOS CLI prompt, **R**, and the cursor appear at your terminal. The cursor marks your place on a line in the form of an underline (a blinking underline on many terminals) or a highlighted box. It is not visible on hard-copy terminals.

Command Line Syntax

A command line consists of a CLI command, optional switches and arguments, and a required command line terminator. The order and the manner in which you specify each component on a line is important to the interpretation of the line. The syntax of a command line is:

COMMAND[/global-switches] [arguments[/local-switches]] terminator

As the syntax shows, switches and arguments are both optional; a CLI command line may consist of simply a valid command name and a terminator. For example, the LIST command displays the names of files in the current directory. LIST offers many options and accepts arguments, but does not require an argument. To display the names of files in the current directory, you simply type the command:

LIST <NL>

When the CLI processes a command, it displays any information requested by the command (or an error message if it could not execute the command), and then displays the **R** prompt. You may then enter another command.

If the first word in a command line is not a valid CLI command, the CLI checks to see if the word is a valid abbreviation of a command. (See "Command Abbreviations" later in this chapter.) If it is not a valid abbreviation, the CLI looks for a file having the name you typed with a .MC extension (a macro file), then one with a .SV extension (an executable file). For example, if you want to list files and accidentally type

LSIT

the CLI first looks for a command named LSIT. When it does not find this command, it checks to see if LSIT is a valid abbreviation. Since it is not, the CLI next looks for a file

named LSIT.MC and then a file named LSIT.SV. If it cannot find any of these, you receive the message:

```
File does not exist:  LSIT.SV
```

The **R** prompt returns after the error message, and you can re-enter the command.

The Command Line Terminator

A command line terminator enters a command line for execution. The command line terminator for CLI commands is the NEW LINE key. This manual uses the symbol <NL> to represent the NEW LINE key.

If your keyboard does not have a NEW LINE key, the command line terminator is the key labeled RETURN or CR (carriage return). You can determine which key functions as the line terminator by pressing the key without typing anything else on the line. The key that functions as your command line terminator always returns the **R** prompt. The other key will not enter a CLI command (but could move the cursor down a line).

Arguments

An *argument* to a command names an object upon which the command is to take action. Usually, arguments are the names of files or devices.

Some commands require arguments. For example, in order to print a file you must supply the name of the file, which is the argument, in the command line:

PRINT FILE88

Some commands accept optional arguments. As shown in the earlier example, you can use the LIST command by itself to display a list of files. You can also direct LIST to display a particular file by specifying the name of the file as an argument:

LIST FILE1

```
FILE1.              68  D
```

LIST and many other commands accept multiple arguments. For example, you can display several files by specifically naming them:

LIST FILE1 PROGX FILE2

```
FILE1.              68  D
PROGX.             400  D
FILE2.              50  D
```

You can specify arguments in any order on the command line, unless the command line syntax for that command specifies otherwise.

Delimiters

You must precede each argument on a command line with a delimiter, so that the CLI can recognize it as an argument. The valid delimiters are:

- one or more spaces
- a comma
- a combination of a comma and spaces

Choose a simple delimiter that you prefer, and use it consistently. This manual uses a space to separate an argument from a command, and to separate one argument from another.

Switches

A *switch* is an option that modifies a command or its arguments. A switch is made up of a slash (/) and a single letter or number, such as /E. Many commands accept multiple switches. Note that switches have meaning only to the command under which they are listed. Many commands may have an /E switch, but the /E switch will cause a different, specific action for each command.

The CLI recognizes two types of switches—global switches and local switches—as discussed separately below.

Global Switches

A *global switch* is appended to a command name, and affects the action of the command. Many commands can take multiple switches. Note that no spaces may appear between a switch and the command, or between a switch and another switch.

For example, the LIST command has an "everything" switch, /E, that causes LIST to display all information about each file—the size of the file, the date and time the file was created and was last modified, and so forth.

LIST/E

```
FILE1.          68  D     01/06/86 16:27  01/06/86  [006564]   0
PROGX.         400  D     03/09/86 13:49  03/09/86  [006565]   0
FILE2.          50  D     02/13/86 10:03  02/13/86  [006566]   0
```

The following example appends more than one switch to the LIST command. The /S switch instructs LIST to sort and display the filenames in alphabetic order:

LIST/E/S

```
FILE1.          68  D     01/06/86 16:27  01/06/86  [006564]   0
FILE2.          50  D     02/13/86 10:03  02/13/86  [006566]   0
PROGX.         400  D     03/09/86 13:49  03/09/86  [006565]   0
```


You can add arguments to this command line. In the following example, the arguments are the two files on which you want information:

LIST/E/S PROGX FILE2

```
FILE2.              50  D      02/13/86 10:03  02/13/86  [006566]    0
PROGX.             400  D      03/09/86 13:49  03/09/86  [006565]    0
```

For the two arguments PROGX and FILE2, LIST displays all information about each file (/E) and lists the filenames in alphabetic order (/S).

Local Switches

A *local switch* attaches to an argument and changes the command's treatment of the argument.

Note that local switches may also affect other arguments on the same command line. To prevent this you must explicitly name the other arguments. An explicitly named argument is one that does not include a template. A *template* is a symbol that allows you to specify a number of files. Templates are discussed in detail in chapter 2, but for the purpose of the next few examples you need to know that the argument

FI-.-

means "all files that begin with the letters FI."

For example, in a LIST command line you can use a local /N switch to instruct LIST to exclude a particular argument from the display:

LIST FI-.- FILE2/N

```
FILE1.              68  D
FIDO.01            900  D
```

This command lists all files that begin with the letters FI except FILE2.

One of the most useful types of local switches allows you to specify files created before or after a given date. For example, in the LIST command, you can use the "after" switch, /A, to list only files created after a certain date. You specify the date as an argument, in the form mm-dd-yy, and append the switch to it. The date argument applies to every argument in the command line:

LIST/S/E FIL-.- PRO-.- 1-21-86/A

```
FILE2.              50  D      02/13/86 10:03  02/13/86  [006566]    0
PROGX.             400  D      03/09/86 13:49  03/09/86  [006565]    0
```

Since FILE1 and FIDO.01 were not created on or after January 21, 1986, the CLI does not include them in the display. The other files satisfy the date condition.

Note that the /A switch does not apply to any arguments that you enter explicitly, without templates. For example, the command

LIST/S/E FILE-.- SINGLEOUT 01-09-86/A

lists all files that match the template FILE-.- only if they have creation dates that are on or after 01-09-86, and it lists the file SINGLEOUT, regardless of its creation date, because it is named explicitly.

Correcting Typing Mistakes

Two keys enable you to correct typing mistakes on a line--a character delete key and a line delete key.

The key marked DEL or RUBOUT (depending on your keyboard) erases the character immediately preceding the cursor.

If you type LIDR instead of LIST, you can correct the line by pressing the character delete key, and then typing the T:

LIDR<DEL><DEL>T

results in LIST.

A backslash (\) erases the entire line and repositions the cursor at the beginning of the line so that you can re-enter the command. Note that \ resets the cursor but does not display an R prompt. For example, the sequence

QWERTY \
LIST

results in LIST.

If you enter a command line before you realize it has a typing mistake, and the CLI does not recognize the word as a command or abbreviation, a macro file, or an executable file, you receive the message:

```
File does not exist:  FILENAME.SV
```

You can then retype the command correctly.

Command Abbreviations

DG/RDOS provides an ABBREVIATE command that enables the CLI to recognize abbreviations of all CLI commands. When DG/RDOS starts, abbreviate mode is off, unless your system supplier or manager has modified your system. You check by typing

ABBREVIATE

The CLI responds with either `ON` or `OFF`. To turn abbreviate mode on, type the command

ABBREVIATE ON

When abbreviate mode is on, the CLI accepts the unique abbreviation of any command. For example, AB is a unique abbreviation for the ABBREVIATE command; it is the only command that starts with the letters AB. Thus if you type *AB,* the CLI responds `ON`.

Depending on the command, a unique abbreviation may be from one to four characters. For example, since the INIT command (which initializes a directory or device for input/output operations) is the only CLI command that starts with the letter I, you can initialize a directory by typing:

I directoryname

If the abbreviation is not unique, you receive an error message. For example, if you want to LIST files and type *LI,* you receive the error message

```
Abbreviation not unique:  LI
```

because another CLI command, LINK, begins with the letters LI. The unique abbreviation for LIST is LIS.

Special Characters

The CLI recognizes a number of special characters and symbols. Some of these, such as <NL> or the backslash (\) key have been used in the previous examples. Table 1-1 provides a complete list of these special characters and their functions.

Table 1-1. CLI Special Characters

Character	Keyboard Symbol	Function
NEW LINE[1]	NEW LINE or CTRL-J	Terminates command line. (Ignored on systems that recognize CR.)
Carriage Return[1]	CR or CTRL-M	Terminates command line. (Ignored on systems that recognize NEW LINE.)
Delete	DEL or RUBOUT	Deletes individual characters from right to left on a line.
Erase Page	ERASE PAGE or CTRL-L	Terminates command line and erases the display screen.
Tab	TAB or CTRL-I	Advance the cursor one tab stop.
Backslash	\	Deletes current line.
Space		Separates arguments in command line. Extra spaces have no effect.
Comma	,	Separates arguments in command line. Multiple commas indicate null arguments.
Slash	/	Sets off command line switch.
Semicolon	;	Delimits CLI commands, separating multiple commands entered on one line.
SHIFT-6	^ (caret)	SHIFT-6 followed by line terminator extends command to following line. Do not confuse with the uparrow cursor control key.
Period	.	Separates filename from extension. Also, adds or removes the time of day at the CLI **R** prompt.
Colon	:	Separates device names, directory names, and filenames in a pathname.

Table 1-1. CLI Special Characters (continued)

Character	Keyboard Symbol	Function
Asterisk	*	Template character that represents any one character, except a period, in a filename or extension.
Hyphen	-	Template character that represents any number of characters, except a period, in a filename or extension.
Parentheses	( )	Enclose multiple commands or arguments in one command line for the CLI to expand to multiple lines.
Angle brackets	< >	Enclose multiple arguments in a command line for the CLI to repeat.
Commercial at	@ @	Enclose a filename in command line to invoke the file as an indirect file.
Quotation marks	" "	Enclose a text string for literal interpretation.
Percent signs	% %	Enclose a CLI-designated variable.

[1] By default, the console driver interchanges the codes for Carriage Return and NEW LINE, unless the console is opened in a binary mode.

Control Characters

Control characters execute functions by overriding current CLI processing. You enter a control character by pressing and holding the key marked CTRL and then pressing the appropriate key. In this manual we indicate a control character in the form CTRL-character, such as CTRL-S.

If a control character is *echoed,* it appears on your console as a caret (^) followed by a printable character. For example, a CTRL-C appears as ^C. Table 1-2 lists DG/RDOS control characters.

Table 1-2. Control Characters

Control Character	Function
CTRL-C	First character of a two-character control sequence. Echoed.
CTRL-Q	Resume terminal output suspended with CTRL-S. Not echoed.
CTRL-S	Suspend terminal output. (Resume with CTRL-Q.) Not echoed.
CTRL-Z	Close a file being created from the terminal with the XFER/A command. End-of-file character. Echoed.

Note that CTRL-S and CTRL-Q do not display any messages. If your terminal seems "dead," you may have accidentally pressed CTRL-S. In this case, pressing CTRL-Q will restore normal terminal operation.

Control Sequences

You enter a *control sequence* by pressing and holding the CTRL key followed by pressing C and then the second key in the sequence. In this manual we indicate a control sequence in the form CTRL-C CTRL-A.

CTRL-C sequences are used to interrupt programs. They must be entered from either the foreground or the background console. When you enter a CTRL-C, the action taken by the console driver depends on the second character entered:

a) If the second character is a CTRL-A, -B, -C, -D, or -F, the actions described in Table 1-3 are taken.

b) If the second character is a CTRL-I, -J, -L, -M, -Q, or -S, or if the second character is not a CTRL character, the console driver handles the second character as if the CTRL-C were never entered.

c) If the second character is a control character that is not covered by rule a or b above, it is echoed but no action is taken.

Table 1-3 lists DG/RDOS control sequences.

Table 1-3. Control Sequences

CTRL-C followed by	Function
CTRL-A	Terminate execution and return to the next higher-level program (usually the CLI).
CTRL-B	Terminate execution, but first wait for task I/O to complete, and create a breakfile of the current memory image called BREAK.SV (background) or FBREAK.SV (foreground).
CTRL-C	Echoed, but no action taken.
CTRL-D	Special interrupt used by Interactive COBOL.
CTRL-F	Terminate a foreground program from the background console, releasing foreground devices and directories.

You can use CTRL-C CTRL-A to exit from any CLI command and from many programs. CTRL-C CTRL-A displays the message `INT`, meaning "interrupt," before it returns the **R** prompt. At that point, you can enter a CLI command.

CTRL-C CTRL-B displays the message `BREAK` before it returns the **R** prompt.

CLI Responses

The CLI responds to the command lines you enter in one of the following ways:

- It displays any output the command may have generated, and returns an R prompt.
- It displays an error message, and returns the **R** prompt.

Some commands do not generate any output, and you receive only the **R** prompt.

The CLI executes everything it can. For example, if your command has many filename arguments, and the CLI cannot locate one of the files, the command executes for all files except that one. You receive an error message that the CLI could not find the file.

Most CLI commands do not execute commands that threaten to overwrite an existing file. The message

```
File already exists:  FILENAME.SV
```

informs you that the CLI did not execute a command because it would be forced to overwrite an existing file (in this example, the file FILENAME.SV).

You receive an error message if a command cannot execute properly, or if the CLI cannot interpret the command line. The message explains why a command, or a part of a command, did not execute. Appendix B lists error messages, their meanings, and methods for correcting errors.

Typing Multiple Commands on a Single Line

You can include more than one command on a single line by using a semicolon (;) to separate commands. You then use a command line terminator at the end of the line to enter the commands to the CLI, all at once. For example:

LIST FILE1;TYPE FILE1 <NL>

```
FILE1.              68  D
This is FILE1,
which has only these two lines of information in it.
```

The CLI executes this line from left to right. First it executes the LIST command and then the TYPE command (which displays the contents of a file). When all of the requested output has been displayed, the **R** prompt returns.

Typing Commands on Multiple Lines

Your display screen or hard-copy terminal can accept some maximum number of characters across a single line; the maximum is usually 80 characters per line, but could be 132 characters. Many display screens automatically wrap a line that is over 80 characters.

The CLI accepts up to 132 characters per line. To enter a command line that is greater than 132 characters, press SHIFT-6 (which appears on the terminal as ^) after 131 characters, then press <NL>. This continues or *wraps* the command onto the next line. You then use the command line terminator when you have finished typing everything you want to include in the command line. For example:

LIST FILE1 FILE2 FILE3 PROGX NEWFILE^<NL>
OLDFILE ANYFILE<NL>

```
FILE1.               68  D
FILE2.               50  D
FILE3.             4386  D
PROGX.              400  D
NEWFILE.             28  D
OLDFILE.             74  D
ANYFILE.            137  D
```

Command Line Shorthand: Parentheses and Angle Brackets

DG/RDOS provides two shorthand methods for cutting down on repetitive typing in command lines—using parentheses and angle brackets to let the CLI perform the repetitions for you. This CLI operation is known as *expansion*.

Using Parentheses

Parentheses cause the CLI to expand the enclosed commands or arguments into multiple commands. For example,

```
(LIST,TYPE) FILE1 FILE2
FILE1.              68  D
FILE2.              50  D
This is FILE1,
which has only these two lines of information in it.
This is FILE2, which contains only this sentence.
```

The command (LIST,TYPE) expands to two commands, LIST and TYPE. Note that these commands execute for each argument specified on the command line: LIST lists both files, and TYPE displays the contents of both files.

Syntax Rules for Parentheses

The following rules define how to use parentheses in command lines.

- The CLI executes elements in parentheses from left to right:

  (LIST,PRINT) FILE1

  expands to

  LIST FILE1
  PRINT FILE1

- You must use commas to separate multiple commands enclosed within parentheses:

  (LIST,PRINT,DELETE) FILE1

- If you have multiple arguments to a command enclosed within parentheses, the action taken depends upon the type of delimiter. If you have spaces between arguments, the CLI expands to one command line. For example, the command

  LIST (FILE1 FILE2)

  expands to: *LIST FILE1 FILE2*

If you have commas between arguments, the CLI expands to multiple command lines. For example, the command

LIST (FILE1,FILE2)

expands to:

LIST FILE1
LIST FILE2

In a command such as LIST this difference has no practical effect, but it can be useful for some commands. For example, the command

MAC (A,B C,D)

expands to:

MAC A
MAC B C
MAC C

- You cannot nest parentheses within another set of parentheses.

- When the CLI encounters more than one set of parentheses in a command line, the action performed depends on the type of delimiter in the parentheses surrounding the arguments. For example, the command

  (LIST,PRINT) (FILE1,FILE2)

  expands to

  LIST FILE1
  PRINT FILE2

  However, the command

  (LIST,PRINT) (FILE1 FILE2)

  expands to

  LIST FILE1 FILE2
  PRINT

You can use the MESSAGE command to test your use of parentheses. Substitute MESSAGE for the command you intend and enter the arguments in parentheses. MESSAGE will repeat the arguments on your console (returning your "message"), allowing you to see how the CLI interpreted the arguments in parentheses. For example:

MESSAGE FILE(1,2,3)
```
FILE1
FILE2
FILE3
```


Using Angle Brackets

Angle brackets cause the CLI to expand the enclosed portions of an argument into a single command line. For example,

```
LIST FILE<1 2 3>
FILE1.              68  D
FILE2.              50  D
FILE3.            4386  D
```

The shorthand argument FILE<1 2 3> expands to the three arguments FILE1, FILE2, and FILE3

As with parentheses, you can use the MESSAGE command to test whether you're using angle brackets correctly.

Syntax Rules for Angle Brackets

The following rules define how to use angle brackets in command lines.

- The CLI expands arguments enclosed in angle brackets from left to right:

  LIST FILE<1,2>

  expands to

  LIST FILE1 FILE2

- Commands or arguments enclosed by angle brackets may be separated by either a comma or a space. You can use multiple commas or spaces to indicate a null argument, as follows:

  LIST FILE<,1,2>

  expands to

  LIST FILE FILE1 FILE2

  The null argument (,) causes the CLI to include the argument FILE without expanding it.

- You can nest angle brackets within other angle brackets, without limit.

  In the following examples, our file arguments include directory names in the form directory:filename. Directories contain files, and are discussed in chapter 2.

  LIST PROJ<83:FILE1,84:FILE<1<A,B>,2>>

  This line expands to

  LIST PROJ83:FILE1 PROJ84:FILE1A PROJ84:FILE1B PROJ84:FILE2

- When the CLI encounters nested angle brackets, it expands the innermost brackets first, and then proceeds to the outermost brackets:

  LIST PROJ<83:FILE1,84:FILE<1<A,B>,2>>

  The CLI begins expanding this line with <A,B>:

  LIST PROJ<83:FILE1,84:FILE<1A,1B>,2>

  then expands the <1A,1B>:

  LIST PROJ<83:FILE1,84:FILE1A,84:FILE1B,84:FILE2>

  and finally expands the outermost angle brackets:

  LIST PROJ83:FILE1 PROJ84:FILE1A PROJ84:FILE1B PROJ84:FILE2

Combining Parentheses and Angle Brackets

By combining the shorthand techniques of parentheses and angle brackets, it is possible to execute multiple commands with multiple arguments with a single command line. For example, the command

(LIST/E,PRINT) FILE<1,2,3>

is equivalent to:

LIST/E FILE1 FILE2 FILE3
PRINT FILE1 FILE2 FILE3

The CLI first lists full information on all three files and then prints all three files.

Ending a CLI Session

When you finish with a CLI session, you can do any of the following, according to your requirements:

- Leave the operating system and the computer running.
- Shut down the operating system, which leaves the computer in a halted state.
- Shut down the operating system and turn off the computer.

NOTE: Always shut down the operating system before turning your computer off.

Shutting Down DG/RDOS

If no one else will be using the system, you may want to shut down the operating system for the sake of security. To shut down DG/RDOS, first move to the master directory by typing the command:

DIR %MDIR%

DIR is a command that moves you to a different directory; %MDIR% is a *variable* that directs the CLI to supply the name of the system master directory.

Once located in the master directory, you can shut down the operating system by using the BYE.MC macro supplied by Data General on the DG/RDOS release medium. When you type

BYE<NL>

the following actions are taken:

1) The CLI displays the message `Starting system shutdown.`

2) The system closes all directories, subdirectories, and devices (such as diskette and tape drives), and updates all files. You receive the message `Master device released.`

3) You receive the prompt `Filename?` This indicates that the system has completed the shutdown procedure.

   NOTE: If any diskette or tape drives were initialized (opened for I/O; see the INIT command in chapter 3), do *not* remove the diskette or tape from the drive until the system has shut down. Removing a diskette or tape before DG/RDOS updates and closes the files on it may cause problems the next time you want to use the files.

You can bring DG/RDOS up again from this point by entering the name of the operating system, or you can turn off power to the equipment.

–End of Chapter–

Chapter 2
Using Files and Directories

This chapter explains how to create, access, and manage disk files and directories. These actions form the foundation for using your DG/RDOS system.

Storing Information on Disks and Diskettes

Disks and diskettes are devices on which you store and retrieve data. The processes of storing and retrieving data are often referred to as input/output, I/O, or reading and writing. In this manual, we generally use the term disk to refer to both disks and diskettes. We specify diskette when necessary.

Preparing a Disk for Use

Before you can store files on a disk, the disk must be hardware and software formatted to accept DG/RDOS file and directory structures. Disks supplied by Data General are hardware formatted. You software format a disk with the DKINIT utility followed by the CLI INIT/F command. See *How to Generate and Run DG/RDOS* for details.

During software formatting with INIT/F, DG/RDOS places MAP.DR and SYS.DR on the disk. These files contain locations and identifiers for the files you later store on the disk.

Files

A *file* is a collection of information that is given a name and stored on a disk. Files are the primary method of storing information on your system because it is easy to place and organize files on a disk, and to retrieve files from the disk in order to display, execute, or make changes to them. You use the CLI to create and work with files.

Before discussing how to create files, we need to explain the rules for naming files and the differences between the three DG/RDOS file types—random, contiguous, and sequential.

Naming Files

DG/RDOS filenames can have up to ten characters, and you can add a period and a one- or two-character extension to the name to further identify the file. MYFILENAME.XT is an example of a legal DG/RDOS filename.

You can use the following characters in a filename:

- alphanumeric characters

  a through z
  A through Z
  0 through 9

- the character $ (dollar sign)

Note that the period is not a filename character. DG/RDOS interprets a period as a filename delimiter; anything after the period is considered an extension. For example, if you entered the name BIG.FILE, DG/RDOS would recognize it as BIG.FI.

DG/RDOS truncates filenames having more than ten characters in the main body of the filename or more than two characters in the extension. For example, DG/RDOS recognizes the filename TREMENDOUSFILE.EXTENSION as TREMENDOUS.EX.

DG/RDOS stores only the uppercase version of a filename. For example, if you type the name *file1,* DG/RDOS displays the name as `FILE1.`

You can assign a file any name you choose, with the exception of the DG/RDOS reserved filenames discussed in the following section. Generally, you should assign a name that describes the contents of the file, such as ACCTSREC.01 for January receivables.

You do not have to use filename extensions, unless you use *macro* files, which must have an .MC extension. DG/RDOS adds predefined extensions to particular types of files (see Table 2-3). You can choose your own extensions for other types of files.

The following is a list of valid DG/RDOS filenames:

FILE1 (or FILE1.)
File1.SV
Longername.1
$$FILENAME
Z123abc.$$

Because the period is a filename delimiter, the names FILE1 and FILE1. are identical.

Reserved Filenames

Certain filenames are reserved for use by DG/RDOS. These reserved filenames include the names of DG/RDOS command and utility files and input/output devices. Do *not* use any of these reserved filenames for files you create. Table 2-1 lists DG/RDOS reserved filenames, and Table 2-2 lists device names.

Table 2-1. DG/RDOS Reserved Filenames

Filename	Function
CLI.ER	Error file used by the SPEED text editor.
CLI.SV	CLI system file.
(F)CLI.C*n*, (F)CLI.S*n*, (F)CLI.T*n*[1]	CLI virtual buffer files. *n* represents current system level.
(F)CLI.CM, (F)COM.CM	CLI command files
(F)LOG.CM	Log file created by LOG command.
(F)TML.TM	Temporary sort file of LIST/S command.
MAP.DR, SYS.DR	Operating system files that identify disk file locations.

[1] The letter F prefixes DG/RDOS files generated by a foreground process.

Table 2-2. DG/RDOS Device Names

Device Name	Description
DA*n*, DE*n*, DJ*n*[1]	Disk and diskette drives
$LPT(1)	Printer
MT*n*, UT*n*	Tape drive
$PLT	Plotter
QTY:*n*	Multiplexor line
$TTO	System (background) console
$TTI	System console keyboard
$TTO1	Foreground console
$TTI1	Foreground console keyboard

[1] The letter *n* indicates a number.

Filename Extensions

DG/RDOS assigns predefined filename extensions to files with special properties. Other extensions, not required but accepted as standards, are recommended for certain types of files. Consistency in filename extensions helps identify files.

Table 2-3 lists required filename extensions. For most of these file types, DG/RDOS or the creating utility automatically assigns the extensions. For a macro file (.MC), you must append the extension when you create the file. These file types are explained in more detail later, as we discuss their uses.

Table 2-3. Required Filename Extensions

Extension	File Type
.BU	Backup copy of a file, created by a text editor.
.CM	CLI command file, created by DG/RDOS.
.DR	Directory, created by the CPART or CDIR command.
.LB	Library file, created by the LFE utility.
.LS	Program assembly listing file, created by the assembler.
.MC	Macro command file, created by the user.
.OL	Program overlay file, created by the RLDR utility.
.RB	Relocatable binary file, created by a program compiler or assembler.
.SV	Executable program file, created by the RLDR utility.

Random, Contiguous, and Sequential Files

DG/RDOS supports three types of file constructions—random, contiguous, and sequential. Random files are used most because they combine rapid access time with flexibility in modifying file contents. For applications that require faster access time, you can use contiguous files. For sequential access, you can create sequential files.

File Storage and Retrieval

This section presents an overview of file storage and retrieval as a background for exploring the differences among random, contiguous, and sequential file constructions.

The storage space on a disk is divided into sections of 512 bytes each, called *blocks*. Each block has a physical address associated with its physical location on the disk.

DG/RDOS divides a file into blocks and stores the file in as many blocks as necessary to hold the entire file. DG/RDOS assigns a *logical address* to each part of the file. A logical address contains references to the corresponding physical address of a block. Every file has its own set of logical addresses.

For example, we can represent the logical addresses for a file that takes up three blocks as BLOCK1, BLOCK2, and BLOCK3. BLOCK1 contains the actual physical address of the first block used to store the file. BLOCK2 contains the physical address of the next block used to store the file, which may be located wherever DG/RDOS found a free block. BLOCK3 contains the physical address of the last block used to store the file.

Because the logical addresses BLOCK1, BLOCK2, and BLOCK3 contain the physical addresses of the blocks used to store the file, we do not have to concern ourselves with what those actual physical addresses are.

Random Files

DG/RDOS stores a random file in blocks as the blocks become available on a disk. It assigns a logical address to each physical block location and, in addition, processes each logical block through an index. Using this index, DG/RDOS can jump from any block in a file to any other block, without having to access any of the blocks that may logically lie between them. For example, for DG/RDOS to move from the first block to the last block of a random file, it uses the index to locate the address of the last block, then accesses only that block.

Random files provide great flexibility for the operating system and for the user. Use of disk space is efficient, because random files use free blocks as they occur. Access time is fast, because random files allow flexible movement from one block to any other block in the file. In addition, a random file expands as you add data to it; it can be as large as the partition in which it is located (up to a maximum size of 33 MB or 65,535 blocks).

Contiguous Files

Contiguous files are stored in disk blocks that are physically located next to one another on a disk. When you create a contiguous file, you must specify the number of blocks, up to a maximum of 65,535, and DG/RDOS reserves the amount of space you specify.

You can include only as much information in a contiguous file as its size can hold; a contiguous file, unlike a random file, cannot expand. If you do not use all of the space reserved for the file, the space is wasted. However, access to contiguous files is faster than access to random or sequential files because all of the file exists on adjoining physical blocks.

Sequential Files

DG/RDOS stores sequential files in physical blocks as the blocks become available on a disk. DG/RDOS assigns a logical address to each physical block used, and it also keeps track of the address of the last block used to store the file and the next block that will be used to store the file. From any given block in a sequential file, DG/RDOS can move only to the addresses kept for that block, which are the addresses of the next or the previous block used to store the file.

For example, for DG/RDOS to move from the first block to the last block of a sequential file, it finds the address of the second block of the file, accesses that block, finds the address of the third block of the file, accesses that block, and so on until it reaches the end of the file.

Access to sequential files is thus not as rapid as access to random or contiguous files, because to move from one block to any other block, DG/RDOS must access every block of the file that lies between the two. For this reason, you normally use sequential organization only for small files.

Creating Files With CLI Commands

Three separate CLI commands control the creation of random, contiguous, and sequential files. After you have created the files, you can insert data into them with the CLI XFER command, or by using a text editor. You can also create and modify files using a text editor.

Creating Random Files

The CRAND command creates a random file and assigns it the name you supply. The following command creates the random file MYFILE:

CRAND MYFILE

If we then type *LIST MYFILE,* the following display appears:

```
MYFILE.              0  D
```

The display shows that MYFILE has a length of zero bytes (indicating that it is empty), and the characteristic D (which indicates a random file). The length of the file will expand as we add data to it.

File characteristics are discussed in more detail at the end of the chapter under "File Characteristics and Attributes."

Creating Contiguous Files

To create a contiguous file, use the CCONT command, supplying a filename as the first argument, and the size of the file, in blocks, as the second argument.

CCONT CTGFILE 144

This command creates a contiguous file called CTGFILE and assigns it a length of 144 blocks.

LIST CTGFILE
```
CTGFILE.            73728  C
```

The LIST command shows that CTGFILE has a length of 73728 bytes (144 blocks), and the characteristic C (which indicates a contiguous file). The 144 blocks represent the space allocated to the file, and not the actual length of the information it contains.

Creating Sequential Files

The CLI command for creating sequential files is CREATE.

CREATE SEQFILE
LIST SEQFILE
```
SEQFILE.            0
```

After the sequential file SEQFILE is created, the LIST command shows that it has a length of zero. Sequential files have no file characteristic.

Inserting Data into Files With the XFER Command

The CLI command XFER lets you add data to the files you create. The format is:

XFER/A/B $TTI filename

/A	specifies a transfer of ASCII text.
/B	appends the information to the file you have created.
$TTI	tells XFER you'll be typing in text from the keyboard (device name $TTI).
filename	tells XFER to add the information to this file.

When you enter the XFER command, everything you type at your keyboard is transferred into the specified file. To stop the transfer, enter CTRL-Z.

The XFER command has many additional capabilities. You can create files and transfer files to and from devices such as a line printer or a tape. XFER is described fully in chapter 3.

Creating Files With a Text Editor

You can also create files with a text editor. Your system comes with the text editor SPEED. The advantages of using a text editor are:

- You can enter information into the file when you create it.
- You can move anywhere in the file to examine or make changes to the file.
- You can change, add, and delete information in the file.

Text editors by default create random files, but you can also edit other types of files with a text editor.

Text editors come with their own commands for manipulating files. For this reason, a separate manual covers each editor. SPEED is described in the *RDOS/DOS Superedit Text Editor* manual.

Displaying Files

This section introduces two commands that display the contents of files. The TYPE command displays the contents of a file at your terminal, and the PRINT command prints the contents of a file on the line printer.

To use the TYPE command, supply the filenames of one or more files with the command. The file is then displayed on your screen, as in the following example:

TYPE FILE1
```
This is my first DG/RDOS file.
XFER accepts input from my keyboard.
That's enough for now.
```

NOTE: Use the TYPE or PRINT commands *only on text files;* do not use them on binary files (such as .OL and .SV files).

Because the TYPE command displays the entire contents of a file, a long file may scroll off your display screen too fast for you to read it. Type a CTRL-S to pause the display, and when you're ready to see more, use CTRL-Q to resume it.

The PRINT command displays the contents of a file on the line printer (device name $LPT):

PRINT FILE<1,2,3>

This command prints the files FILE1, FILE2, and FILE3 on the line printer, as soon as the printer becomes available (i.e., when it completes any print requests submitted before yours). The PRINT command displays no confirmation message.

Specifying Groups of Files With Templates

Templates allow you to group files that have similar elements in their filenames. When you use template symbols to represent any part of a filename argument, DG/RDOS replaces the argument with all filenames that apply. DG/RDOS template symbols are:

- The hyphen, which represents any group (or string) of legal filename characters. For example, the command

  LIST C-

  lists all files that begin with C and do not have filename extensions. The command

  LIST E-.-

  lists all files that begin with E, whether or not they have filename extensions.

* The asterisk, which represents any single character. For example, the command

  *LIST FILE**

  lists any files with 5-character filenames, where the first 4 characters are FILE.

You can use templates anywhere within a filename. The command

LIST -.SV

lists all executable program files (i.e., any files that have a .SV extension).

You can mix template symbols in filename arguments. The command

*LIST -.***

lists all files that have a two-character filename extension.

You can use templates with the following CLI commands:

BUILD
DELETE
DUMP
LIST
LOAD
MOVE
PRINT
UNLINK

The ability to use templates can influence the way you name your files. For example, if you keep progress files for each month of the year, you could organize their filenames so that you can access them in a number of ways. By giving each progress file the same first two letters (say, PR for progress), followed by three characters for the month, and an extension that indicates the year, you could access these files:

- by specifying all progress files with the argument PR-.-
- by specifying all progress files for a particular month with the argument -JAN.- or -APR.-, for instance.
- by specifying all progress files for a particular year with the argument -.86 or -.87.

Templates can also be helpful when you are not sure of the spelling of a filename.

Directories

A *directory* is a special type of disk file that holds other disk files. Directories allow you to arrange the many files stored on a disk into functional groups. Note, however, that *you cannot have two files with the same name within the same directory*. Even when files exist in different directories, you should avoid using duplicate names.

You use CLI commands to create and work with directories. You can work within one directory, or easily move to work in another directory. You can also move files between directories. Note, however, that DG/RDOS does not recognize a directory until you open it for access with the DIR command or initialize it with the INIT command. See the section "Accessing Directories" for details.

DG/RDOS has three types of directories:

- primary partition
- secondary partition
- subdirectory

When we use the term *directory,* we are referring to any one of these three types.

Two directories have a special status: the directory that acts as your current directory, and the directory designated as your master directory.

The Current Directory

The directory you are presently working in is known as your *current directory*. DG/RDOS assumes that the files you refer to reside in your current directory, unless you explicitly refer to another directory.

There are three ways to gain access to the files in any directory:

- You can change your current directory to another directory with the DIR command.
- From the current directory, you can refer explicitly in a command line to a file in another directory. This explicit reference is known as a *pathname* or *directory specifier.*

  Note that you cannot use filename templates in pathnames to refer to files in another directory.
- With the LINK command, you can create in your current directory a file that *links* to a file in another directory.

The Master Directory

The disk with which you bootstrap DG/RDOS, called the master device, contains a *master directory*. The master directory is the directory that contains the DG/RDOS system files necessary to run the operating system and the CLI. You use the master directory to start up and shut down DG/RDOS. When you start DG/RDOS, the master directory is your current directory.

Setting Up a Directory Structure

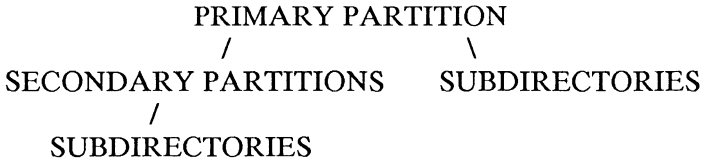
As mentioned previously, DG/RDOS recognizes three kinds of directories—the primary partition, secondary partitions, and subdirectories.

The *primary partition* is the top level in the directory structure, comprising the entire disk or diskette.

Secondary partitions are a subset of the primary partition, forming a second level in the structure.

Subdirectories can be a subset of a primary partition, forming a second level in the directory structure, or a subset of a secondary partition, forming a third level.

You can use these directories to create a directory structure that is three levels deep, as shown in the following illustration:



Simple data files (as compared to directory files) can reside in any of the three directories.

The Primary Partition

Before you create any directories on a disk, that disk has one directory, the primary partition. The name of the primary partition is the device name of the disk itself, such as DE0 or DJ0. Often these terms are used interchangeably. For example, disk DE0 is the same as primary partition DE0; diskette DJ0 is the same as primary partition DJ0.

A primary partition forms the first level in your directory structure for a given disk or diskette. A primary partition can contain files, secondary partitions, and subdirectories.

Secondary Partitions

A secondary partition is a directory that contains a fixed amount of contiguous disk space for file storage. DG/RDOS forms a secondary partition by assigning it space from the primary partition. You specify the amount of disk space, in blocks, when you create the secondary partition.

Secondary partitions are constructed as contiguous files. The space DG/RDOS allocates to a secondary partition is made up of disk blocks located next to each other, and the files in the partition are stored in this space. You cannot change the size that you specified when you created the partition.

Creating Secondary Partitions

To create a secondary partition, use the CPART command, and supply a name and a *block count* for the partition. For example, the command

CPART PART1 256

creates a secondary partition called PART1 that has 256 contiguous blocks.

The block count allocates a fixed amount of disk storage space to the partition. The information contained in a secondary partition cannot exceed the amount of space allocated. (The section "Managing Files and Directories" discusses managing your storage space.)

Since directories are actually special types of files, we can list them with the LIST command:

LIST
```
PART1.DR            131072 CTY
```

The LIST command tells us that the command *CPART PART1 256* created a file called PART1.DR, with 131072 bytes of storage space (equivalent to 256 blocks), and characteristics CTY (which indicate a secondary partition).

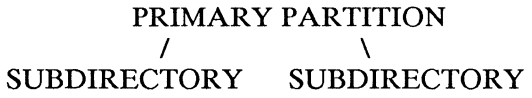
Characteristics of a file are explained further at the end of the chapter under "File Characteristics and Attributes."

The CPART command automatically assigns the directory a .DR filename extension. You do not need to specify the .DR as part of the directory name, unless you use it as a filename argument.

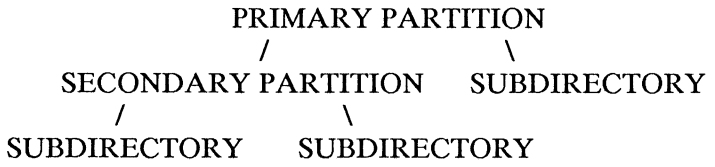
Subdirectories

A subdirectory is a variable length directory that can be a subset of the primary partition or a subset of a secondary partition.

When a subdirectory is a subset of the primary partition, it forms the second level of directory structure, as in the following illustration:



When a subdirectory is created within a secondary partition, it forms the third level of directory structure:



Subdirectories are constructed as random files containing filenames and location information about each file. Unlike secondary partitions, subdirectories are of variable length. When you create a subdirectory, DG/RDOS sets aside an initial amount of disk space, but the subdirectory expands as you add files or data.

The partition containing the subdirectory is known as the *parent directory* of the subdirectory. A subdirectory can occupy as many disk blocks as the subdirectory's parent directory can accommodate.

Creating Subdirectories

To create a subdirectory, use the CDIR command and supply a name for the directory:

CDIR MYDIR

This command creates subdirectory MYDIR as a subset of the current partition. The current partition is the parent directory of subdirectory MYDIR.

When you create a subdirectory from a primary partition, it is a subset of the primary partition. When you create a subdirectory from a secondary partition, it is a subset of that secondary partition.

We can list this subdirectory with the LIST command:

```
LIST
MYDIR.DR              512   DY
```

LIST shows that the command *CDIR MYDIR* created a file called MYDIR.DR with an initial length of 512 bytes and characteristics DY (which indicate a subdirectory). The size of MYDIR will change as you add files or data.

The CDIR command automatically assigns the directory a .DR filename extension. You do not need to specify the .DR as part of the directory name, unless you use it as a filename argument.

Examples of a Directory Structure

Figures 2-1 and 2-2 illustrate DG/RDOS directory structures. Figure 2-1 does this by means of a tree structure, so called because it resembles an inverted tree. Figure 2-2 shows the division of a primary partition into various directories and files.

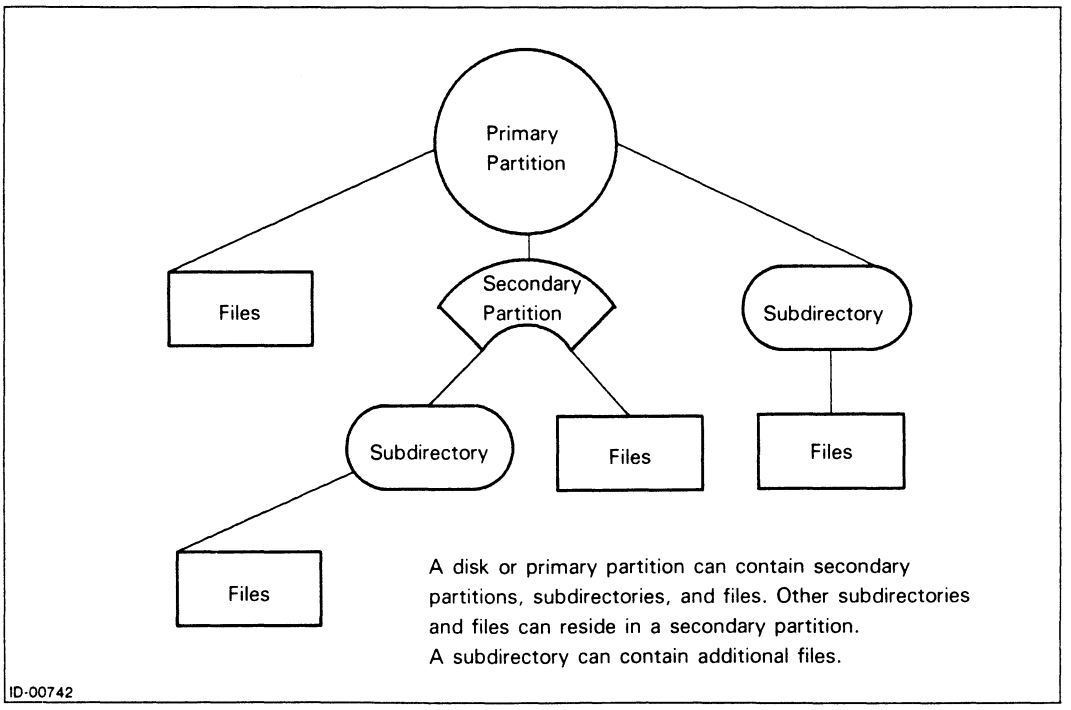


Figure 2-1. DG/RDOS Directory Tree Structure

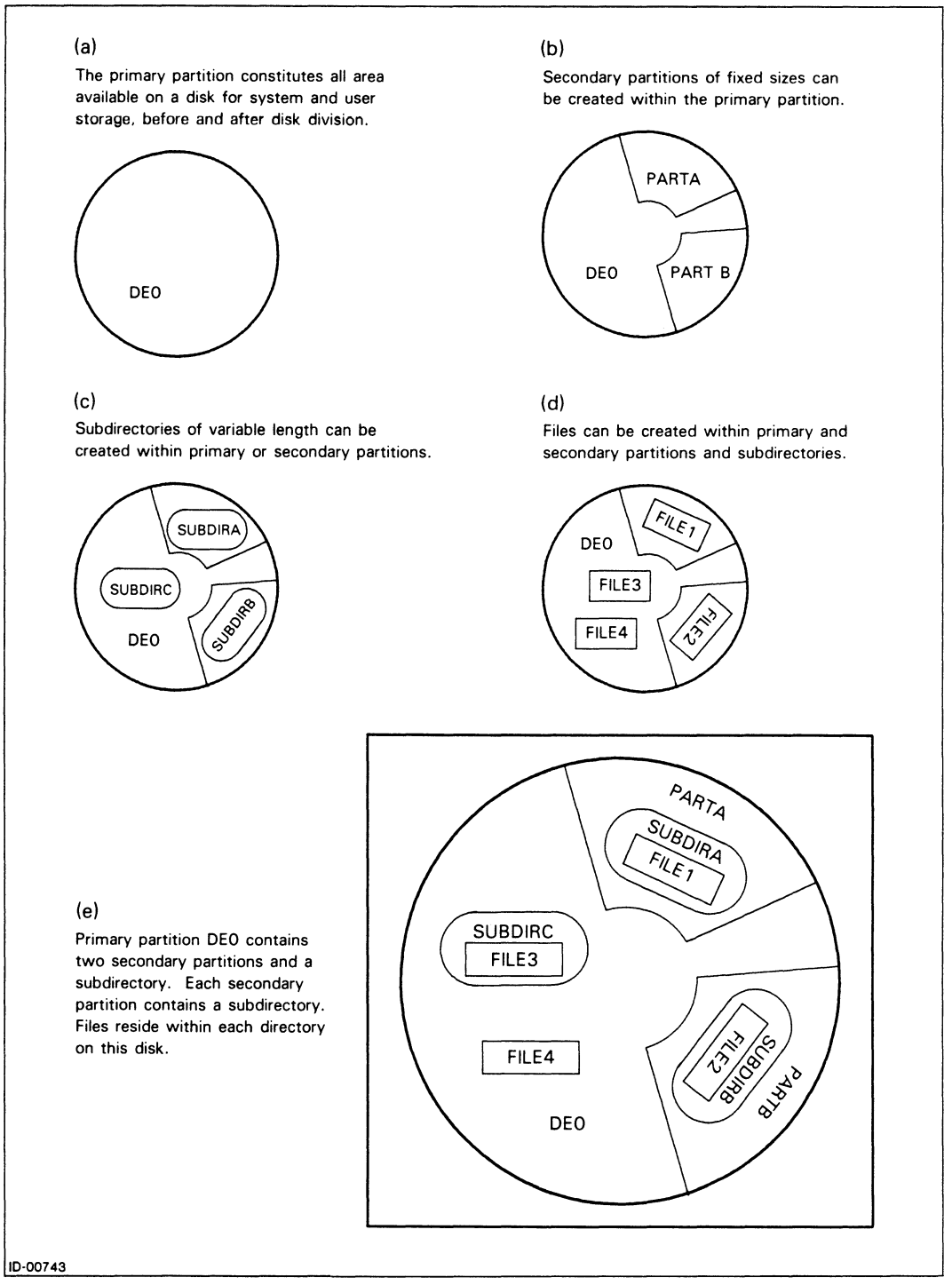


Figure 2-2. Formation of a DG/RDOS Directory Structure

Summary of Directory Types

The following lists summarize the most important characteristics that differentiate directories.

Primary Partition

- fixed length (the size of the disk or diskette)
- constitutes the first level of a directory structure
- can contain secondary partitions, subdirectories, and files

Secondary Partition

- fixed length (size set by user at creation)
- constitutes the second level in a directory structure, being a subset of the primary partition
- can contain subdirectories and files

Subdirectory

- variable length
- constitutes either the second or third level of a directory structure, depending on whether it's a subset of the primary partition or a secondary partition
- can contain files

Accessing Directories

When you *access* a directory, the directory and its files must be available to you and to the operating system. DG/RDOS does not recognize a directory until you *initialize* it, which opens it for access.

You can also designate any directory as your *current directory,* that is, the directory from which you are presently working. You have direct access to any files in your current directory.

Releasing a directory reverses the initialization, and designates a directory as unavailable for access.

You initialize and release directories, whether they are partitions or subdirectories, with the same commands. The only exception is the master directory. When you start up DG/RDOS, the master directory becomes your current directory automatically. Releasing the master directory causes DG/RDOS to release all subpartitions and subdirectories in it.

Releasing a partition releases all directories and subdirectories in the partition.

Initializing Directories

Before you can access a file within a directory, the directory must be initialized. The term "accessing a file" refers to any file operation, including displaying a list of files, displaying the contents of files, modifying files, or executing program files.

There are two ways to initialize a directory:

- by using the INIT command
- by changing the current directory with the DIR command

The INIT command initializes the directory you specify in the command line. For example, the following command initializes the directory MYDIR:

INIT MYDIR

INIT opens a directory for access, without affecting the current directory. You can refer to files in an initialized directory that is not the current directory by naming the path to that directory in filename arguments. This is explained in the section "Using Pathnames."

You can have many directories initialized at one time. You select the number you want during the CONFIG program. See *How to Generate and Run DG/RDOS* for details.

Changing the Current Directory

When you first start up your DG/RDOS system, your current directory is the master directory. You can change your current directory with the DIR command. The command

DIR MYDIR

places you in subdirectory (or secondary partition) MYDIR, and also initializes MYDIR. All file commands, such as LIST and TYPE, will now apply to the files in MYDIR (unless you specify otherwise in a pathname). Your previous current directory also remains initialized.

Releasing Directories

Releasing a directory closes it to input and output, and prohibits access to the files it contains. To release directory MYDIR, use the command:

RELEASE MYDIR

A RELEASE command cancels both the INIT and DIR commands. If you release your current directory, the master directory (which may or may not be the parent directory) becomes the new current directory.

You can use DIR or INIT to reinitialize a directory that has been released.

Releasing Diskettes

Always release a diskette before physically removing it from its drive. Otherwise, the files on the diskette are not updated, leaving the diskette in an inconsistent format.

To release a diskette, give its device name as the argument to the RELEASE command:

RELEASE DJ0

Displaying Information About a Directory Structure

The commands MDIR, GDIR, and LDIR display information about the current state of your directory structure.

The MDIR command displays the name of the master directory:

MDIR
```
DA0
```

In this example, the master directory is DA0.

The GDIR command displays the name of your current directory:

GDIR
```
DA0
```

DA0 is the current directory.

If we change directories with the DIR command, GDIR displays the new current directory.

DIR SOLUTIONS
GDIR
```
SOLUTIONS
```

The current directory is SOLUTIONS.

The LDIR command displays the name of the previous current directory:

LDIR
```
DA0
```

The previous current directory was DA0.

Using Pathnames

A *pathname* or *directory specifier* allows you to access a directory or file that is not in your current directory. A pathname may consist of just the directory name or the directory name and a filename separated by a colon. In order for a pathname to work, all of the directories between the current directory and the destination directory or file must be initialized.

Example of Using Pathnames

Figure 2-3 depicts a directory structure with which we can experiment with pathnames.

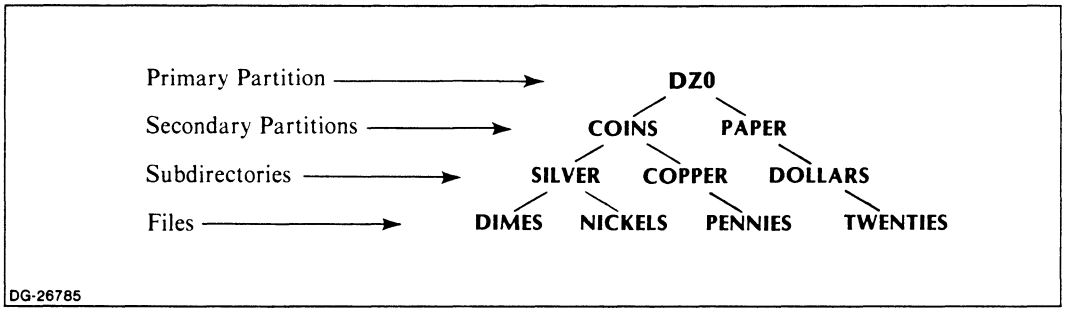


Figure 2-3. Directory Structure

Master directory DE0 is our current directory; no other directories are initialized. The following commands initialize all of the directories in this directory structure; note that, by being named in the path, PAPER and COINS are initialized.

INIT PAPER:DOLLARS
INIT COINS:SILVER
INIT COINS:COPPER

Once the directories are initialized, we can list a file without changing the current directory (DE0) by specifying the directory and the filename. To list the file NICKELS in directory SILVER, use the command:

LIST SILVER:NICKELS

To move from directory DE0 to directory SILVER, use the command:

DIR SILVER

From SILVER we can list the file PENNIES with the command:

LIST COPPER:PENNIES

We can move from directory SILVER to directory DOLLARS with the command:

DIR DOLLARS

To edit the file TWENTIES, using the SPEED editor located in DE0, we use the command:

DE0:SPEED TWENTIES

Managing Files and Directories

The remainder of this chapter explains how to:

- Display the amount of space available on a disk.
- Rename, delete, and move files between directories.
- Link to a file in another directory.
- Set access privileges to files using file attributes.

Determining the Amount of Free Disk Space

The DISK command displays the allocation of space for the current partition.

For a primary partition, DISK displays the total amount of space on the disk. Blocks allocated to files, secondary partitions, and subdirectories are listed as USED; the number of USED and LEFT (unused) blocks is the total number of blocks in the partition.

For a secondary partition, DISK displays the amount of space allocated to that partition and the amount of space used by files and subdirectories within the partition.

For a subdirectory, DISK calculates the amount of space for its parent partition.

Figure 2-4 illustrates how the DISK command displays file allocation.

`Left: xxxx`	`Used: yyyy`	`Max. contiguous: zzzz`
xxxx is the number of blocks free for storage	yyyy is the number of blocks in use	zzzz is the number of blocks forming the largest contiguous free space

Figure 2-4. DISK Command Interpretation

If you add the number of blocks LEFT to the number of blocks USED, the result is the total amount of space allocated to the partition. If you want to create a secondary partition or other contiguous file, check the number given for Max. contiguous to see how much space is available:

DISK
```
Left: 1499   Used: 23161   Max. contiguous: 910
```

In this example, the current partition has 1499 free blocks available for use, 23161 blocks currently in use, and 910 blocks is the largest amount of contiguous space available for use.

In order to avoid exhausting your file storage space on a disk, you need to decide which files you normally use, and which files (or perhaps entire directories) are either outdated or needed only rarely. You can then delete outdated files and store the old information on another device, such as a tape, thereby freeing disk space.

Another way to gain more contiguous space is to back up all of your files with IMOVE, delete all of the files on the disk, and then restore the files. The IMOVE restore operation compacts the files toward the beginning of the disk, giving you more room at the end.

DG/RDOS provides CLI command utilities for moving files and directories to a variety of devices, and for transferring those files back to disk when they're needed. Chapter 3 describes the BURST and IMOVE utilities, and chapter 5 explains briefly when and how they should be used. For a full discussion of backup and move utilities, see *How to Generate and Run DG/RDOS.*

Renaming and Deleting Files and Directories

The RENAME command lets you change the names of files and directories. You specify the old name as the first argument, and give the new name as the second argument:

RENAME PART1.DR GRAPHICS.DR

This command assigns the new name GRAPHICS.DR to secondary partition PART1.DR. Renaming this directory has no effect on the files in the directory.

The DELETE command erases a file or directory. When you use the /C switch, DELETE requests confirmation from you before it deletes the file.

NOTE: We *recommend strongly* the use of the /C switch with the DELETE command, *especially* when you use templates.

Consider the following example:

DELETE/C FILEX.-
```
FILEX? N
FILEX.1? <NL> *
FILEX.2? <NL> *
```


The CLI displays each filename that matches the template FILEX.- and pauses for confirmation after each name. If you type a command line terminator, the CLI displays an asterisk (*) and deletes the file. If you type any other key, the CLI does not delete the file. In this example, we confirmed the deletion of FILEX.1 and FILEX.2, but not FILEX.

CAUTION: If the filename you supply in the DELETE command line is a directory, the directory and all of the files and subdirectories contained in that directory will be deleted.

Note also that you should *never* use the DELETE command to erase a link file. DELETE will erase the resolution file, not the link file. Use the UNLINK command instead. Link files are discussed at the end of this chapter under "Linking Files."

When you delete a file, you cannot retrieve it unless you have made a backup copy of the file on another medium. Creating backup copies of your files regularly can save time, effort, and frustration if you inadvertently delete a needed file. (See chapter 5.)

Another way of protecting a file from inadvertent deletion is to assign it attribute P (permanent). See the section "File Characteristics and Attributes" for details.

Moving Files

The MOVE command allows you to move files from the current directory to another directory (the destination directory).

You supply the name of the destination directory as the first argument, and then specify the file or files you want to move. Including a /V (verify) switch with the MOVE command causes the CLI to list each file moved:

MOVE/V GRAPHICS XCHART.SV, CHART-.-
```
XCHART.SV
CHART1
CHART1A
CHARTFIN
```

This command moves the file XCHART.SV and all of the files matching the template CHART-.- from the current directory into the directory GRAPHICS.

Note that MOVE does not recognize permanent files (i.e., files with the attribute P), unless you include the /A switch. For example, the command

MOVE/A/V GRAPHICS C-.-

moves all files in the current directory that begin with a C to directory GRAPHICS, including permanent files.

Often, you create a new directory when you see that a group of files you have been working on deserve their own directory. The following example shows how to do this by using a

combination of the commands discussed so far.

The example shown in Figure 2-5 creates and initializes a directory called GRAPHICS, and moves four files from the secondary partition, USR$APPL, to the subdirectory GRAPHICS. Once the files exist in GRAPHICS (which we verify by listing the files), we no longer need them in USR$APPL, so we delete them.

```
DIR USR$APPL                                  Move to partition USR$APPL

CDIR GRAPHICS                                 Create directory GRAPHICS

INIT GRAPHICS                                 Initialize GRAPHICS

MOVE/V GRAPHICS XCHART.SV CHART-.-            Move files to GRAPHICS
XCHART.SV
CHART1
CHART1A
CHARTFIN

DIR GRAPHICS                                  Move to GRAPHICS

LIST                                          List its files
XCHART.SV     54376  D
CHART1          945  D
CHART1A        1085  D
CHARTFIN      26386  D

DIR USR$APPL                                  Move to USR$APPL

DELETE/V XCHART.SV CHART-.-                   Delete the files that were moved
DELETED XCHART.SV
DELETED CHART1
DELETED CHART1A
DELETED CHARTFIN
```

Figure 2-5. Moving Files into a New Directory

Linking Files

Creating links to files in another directory allows you access to files without using pathnames, changing your current directory, or moving copies of the files into your current directory.

Links are often used to access utilities residing in the master directory. Let's assume the master directory is the primary partition DA0. We want to use the SPEED editor from our current directory (subdirectory GRAPHICS) and have DG/RDOS access the file SPEED.SV that resides in the master directory.

The LINK command takes two arguments. The first argument assigns the link a name, and the second argument specifies the *resolution file,* the name of the file you want to reach. You enter the resolution file argument in the form of a pathname. For example, the command

LINK SPEED.SV DA0:SPEED.SV

creates a link in GRAPHICS to the SPEED program file in DA0.

Generally you should give the link file the same name as the file you want to access, so that you do not have to remember two different names.

When you list a link file, the LIST command displays the name of the link and its resolution:

LIST SPEED.SV
```
SPEED.SV          DA0:SPEED.SV
```

The link file is SPEED.SV and its resolution file is SPEED.SV in directory DA0.

To remove a link, use the UNLINK command with the link filename:

UNLINK SPEED.SV

This removes the link file SPEED.SV from your current directory.

CAUTION: Be sure to use the UNLINK command to remove link files. Do *not* use the DELETE command; it erases the resolution file.

File Characteristics and Attributes

DG/RDOS keeps information about all files on a disk in a system file directory called SYS.DR. The information kept about your files is called a *user file definition* (UFD). Part of the UFD for a file contains the file's characteristics and attributes.

File Characteristics

DG/RDOS uses *file characteristics* to identify particular properties of files, such as whether a file has random organization or is a link or directory file. DG/RDOS assigns the characteristics when you create a file. You cannot change a file's characteristics. Table 2-4 describes DG/RDOS file characteristics.

Table 2-4. File Characteristics

Characteristic	Meaning
C	Contiguous file.
D	Random file.
L	Link file.
T	Secondary partition. (Partitions also have characteristics C and Y.)
Y	Directory file. (Partitions, subdirectories.)

The LIST command displays the characteristics for each file listed.

```
LIST
FILE1.              68  D
FILE2.              50  D
GRAPHICS.DR        512  DY
```

In this example, FILE1 and FILE2 have the characteristic D, indicating a random file, and file GRAPHICS.DR has characteristics D, indicating a random file, and Y, indicating a directory.

File Attributes

File *attributes* are codes assigned by you or by DG/RDOS that restrict or permit file manipulation or access. The CHATR (CHange ATtRibute) command allows you to assign and remove a file's attributes. Table 2-5 describes DG/RDOS file attributes.

Table 2-5. File Attributes

Attribute	Type of File	Comment
A	Attribute-protected file	You cannot change the file's characteristics or attributes, including attribute A. The A attribute can be set only with the .CHATR system call.
N	Resolution-protected file	You cannot access the file by way of a link.
P	Permanent file	You cannot delete or rename a file with P set. Note, however, that the parent directory can be deleted unless *it* is permanent. Note also that BUILD, DUMP, LIST, LOAD, or MOVE commands do not work on files with the P attribute unless you include the /A switch in the command line.
R	Read-protected file	You cannot read the file (with the TYPE or PRINT commands), edit it with a text editor, or copy it with the DUMP, IMOVE, MOVE, or XFER commands when R is set.
S	Save file.	An .SV file needs this attribute to execute. The RLDR utility assigns S when creating a file.
W	Write-protected file	You cannot modify the file. Note, however, that W does not protect a file from deletion.

To display file characteristics and attributes, use the LIST command:

LIST FILE10

```
FILE10.          0  DN
```

The letter D is the file characteristic for a random file, and the letter N is the file attribute that prevents links to the file FILE10.

The CHATR command takes two or more arguments. The first is a filename; the second and subsequent arguments specify the desired attributes. You precede each attribute with a plus sign (+) to assign the attribute, or with a minus sign (-) to remove the attribute. For example, the command

CHATR FILE1 +W

assigns the write-protect attribute to FILE1.

The command

```
LIST FILE1
FILE1.          214  DW
```

shows that FILE1 has the characteristic D and attribute W. The command

CHATR FILE1 -W

removes the write-protect attribute from FILE1.

The CHLAT (CHange Link ATtribute) command changes a file's link access attributes. Link access attributes are the same attributes described in Table 2-5 , but they take effect only when a file is the target, or resolution file, of a link. Link access attributes have no effect on direct access of a file (i.e., access from within the file's own directory or through use of a pathname).

You use CHLAT the same way you use CHATR. The LIST command displays link access attributes by appending them to existing attributes and characteristics with a slash (/).

```
CHLAT DISPLAY.SV +PW
LIST DISP-.-
DISPLAY.SV      634  SD/PW
```

This example assigns the attributes P and W as link access attributes to file DISPLAY.SV. The LIST command shows that DISPLAY.SV has file characteristics S (executable) and D (random) and link access attributes P (permanent) and W (write-protect). DISPLAY.SV cannot be deleted or modified through a link file, but you can modify or delete it from within its own directory, or by using a pathname.

–End of Chapter–

Chapter 3
CLI Commands and System Utilities

This chapter describes all CLI commands and system utilities in alphabetical order.

Before proceeding, you may want to review the notational conventions described in the preface; these will help you understand the command format. Each entry begins with the command name, a brief description, and the command syntax. The entries include a discussion of the command and its uses, lists of global and local switches, and one or more examples of the command.

Generally, when a CLI command takes a filename argument, that argument can include a pathname, as in the command *PRINT SUBDIR:FILENAME*. We note exceptions in the individual entries.

Table 3-1 through Table 3-4 briefly list the commands according to function. Note that some commands appear in more than one functional table. The categories are:

- File management commands
- Directory management commands
- System control commands
- System utilities

Note that the commands described in Table 3-4 invoke system utilities. These are program files (with an .SV extension) that you start from the CLI, but they are not part of the CLI, and the CLI is not active when they run. When a utility terminates (either normally or after a fatal error), control returns to the CLI.

Table 3-1. File Management Commands

Command	Function
APPEND	Combine two or more files.
BUILD	Build a file of filenames.
CCONT	Create a contiguous file of specified size.
CHATR	Change a file's attributes.
CHLAT	Change a file's link access attributes
CLEAR	Set file or device use count to zero.
CRAND	Create a random file.
CREATE	Create a sequential file
DEB	Debug a program.
DELETE	Delete a file.
DUMP	Dump a file in CLI DUMP format.
ENDLOG	Close the (F)LOG.CM file.
FILCOM	Compare the contents of two files.
FPRINT	Print a file in octal, decimal, hexadecimal, or byte format.
LINK	Create a link entry to a resolution file.
LIST	List file information.
LOAD	Load dumped (DUMP) files.
LOG	Start recording in the log file (F)LOG.CM.
MOVE	Copy files to a directory.
PRINT	Print a file on the line printer.
RENAME	Rename a file.
REPLACE	Replace overlays in an overlay file.
REV	Display the revision level of a program.
TYPE	Type a file on the terminal.
UNLINK	Remove a link entry.
XEQ	Execute a program (.SV) file.
XFER	Copy the contents of one file to another file.

Table 3-2. Directory Management Commands

Command	Function
CDIR	Create a subdirectory.
CPART	Create a secondary partition.
DELETE	Delete a directory.
DIR	Change the current directory.
DUMP	Dump a file in CLI DUMP format.
EQUIV	Temporarily rename a disk or magnetic tape.
GDIR	Display the name of the current directory.
INIT	Initialize a directory or device for system recognition.
LDIR	Display the name of the last current directory.
LOAD	Load dumped (DUMP) files.
LIST	List file information.
MDIR	Display the name of the master directory.
MOVE	Copy files to a directory.
RELEASE	Release a directory or device from initialization.

Table 3-3. System Control Commands

Command	Function
ABBREVIATE	Turn abbreviate mode on or off.
BOOT	Bootstrap a system from disk.
CHAIN	Overwrite the CLI with an executable program.
CLEAR	Set file or device use count to zero.
DISK	Display the number of disk blocks used, free, and free contiguous.
ENDLOG	Stop recording the log file (F)LOG.CM.
EXFG	Execute a program in the foreground.
FGND	Describe the foregound program status.
GMEM	Display background/foreground memory usage.
GSYS	Display the current system name.
GTOD	Display the current system time.
INIT	Initialize a disk directory or device for system recognition.
LOG	Start recording in log file (F)LOG.CM.
MESSAGE	Display text message on terminal display.
POP	Return to the program on the next higher level.
RELEASE	Release a directory or device from initialization.
SDAY	Set system calendar.
SMEM	Set the background/foreground memory areas.
SPDIS	Disable spooling to device.
SPEBL	Enable spooling to device.
SPKILL	Delete data spooled to device.
STOD	Set system clock.
XEQ	Execute a program (.SV) file.

Table 3-4. System Utilities

Command	Function
BURST	Dump and load disk images between disk and tape.
CONFIG	Examine or change system chacteristics.
DO	Execute a command file, replacing dummy arguments with specified arguments.
ENPAT	Insert patches in filename; also see PATCH.
FCOPY	Duplicate a diskette or copy a file.
IMOVE	Dump and load files between disk and tape.
LABEL	Write and examine diskette and tape labels.
LFE	Create or edit .RB library files.
MAC	Assemble a source file into a relocatable binary (RB) file with the Macroassembler.
OVLDR	Create an overlay replacement file.
PATCH	Install patches created by ENPAT.
RLDR	Process relocatable binary files to form an executable program.
SEDIT	Analyze and edit disk files with the Symbolic Editor.
SPEED	Edit text with the ECLIPSE Supereditor.
SYSGEN	Generate a new operating system.

ABBREVIATE

Command

Turn abbreviate mode on or off.

ABBREVIATE [ON / OFF]

The ABBREVIATE command enables the CLI to recognize unique abbreviations of all CLI commands. When DG/RDOS starts, abbreviate mode is off, unless your system supplier or manager has modified your system. You check by typing:

ABBREVIATE

The CLI responds either `ON` or `OFF`. To turn abbreviate mode on, type the command:

ABBREVIATE ON

When abbreviate mode is on, the CLI accepts the unique abbreviation of any command. For example, AB is a unique abbreviation for the ABBREVIATE command; it is the only command that starts with the letters AB.

Depending on the command, a unique abbreviation may be from one to four characters. For example, since the INIT command (which initializes a directory or device for input/output operations) is the only CLI command that starts with the letter I, you can initialize a device simply by typing *I devicename.*

If the abbreviation is not unique, you receive an error message. For example, if you type *LI,* the CLI displays `Abbreviation not unique: LI` because two commands, LINK and LIST, begin with the letters LI. You must enter LIN to link files or LIS to list files.

Type *AB OFF* to turn abbreviate mode off.

Note: Macro and executable files with the same name as a command abbreviation (such as LIS.MC or LIS.SV) will not work when abbreviate mode is on, because the CLI executes commands (or abbreviations) before macro or executable files. If your system has macro (.MC) or executable (.SV) files with the same names as CLI abbreviations, you must do one of the following:

- Change the names of the files.
- Turn abbreviate mode off.
- Enter the name of the macro or executable file explicitly (LIS.MC or LIS.SV).
- Use the XEQ command to execute files. Enter the filename with or without the .SV extension (*XEQ LIS* or *XEQ LIS.SV*).

APPEND

Command

Copy one or more files to a new file.

APPEND outputfilename inputfilename [...inputfilename]

APPEND creates *outputfilename,* and copies the contents of the *inputfilenames* into it. The original input files do not change.

You can use a pathname argument for any of the filename arguments.

If an *inputfilename* is read-protected (attribute R), APPEND copies all of the files up to the read-protected one and then terminates with the error message `File read protected.` Note that the error message does not list the name of the read-protected file. Use the LIST command with the list of *inputfilenames* to find out which one is read-protected; you can then remove the R attribute with the CHATR command.

Examples

APPEND QTRFILE JANFILE FEBFILE MARFILE

Creates QTRFILE and copies into it the contents of JANFILE, FEBFILE, and MARFILE.

APPEND TAPEFILES MT0:1 MT0:2

Creates TAPEFILES and copies into it the contents of tape files 1 and 2 from the tape on drive MT0.

BOOT

Command

Load and run an operating system or a stand-alone program.

BOOT { disk | [directory:]sysname }

BOOT releases the current system and bootstraps a new system or a program into execution.

Sysname can be an operating system program file or a stand-alone program file (see "Stand-Alone Programs," below). It can also name a link entry to an operating system file, if both the system's save and overlay files are in the same directory. If *sysname* is a link, all intermediate directories must be initialized. In every case, when the bootstrap succeeds, the partition containing sysname becomes the master directory.

The program xBOOT.SV must reside in the primary partition of the disk that holds sysname. The program name is either ABOOT.SV or MBOOT.SV, depending on your system configuration.

When you use the *BOOT disk* format, BOOT displays a `Filename?` prompt. Enter the system filename or the name of a stand-alone program. You can respond to `Filename?` with <NL> to bootstrap a system with the name SYS.SV, the default system name.

All directories involved in the BOOT command must be initialized on the system. A disk named as an argument must be a valid DG/RDOS disk.

Stand-Alone Programs

A stand-alone program is a program (such as the DKINIT disk formatter program) that runs without an operating system. You can execute a stand-alone program from the CLI with the BOOT command if the program conforms to the following rules:

1) The save (.SV) file must be randomly organized. (.SV files created with RLDR are always random, but using XFER to transfer a file can change the file's organization.)

2) The stand-alone program must begin at location 0. Location 0 must contain either 0 or a byte-pointer to a text string (the program name). If it contains the latter, the text string will be displayed on the console.

3) Location 2 must contain either the starting address of the .SV file, or -1. If it contains the starting address, the program will self-start. If location 2 contains -1, the computer will halt after loading; press the indicated key to continue.

4) Location 5 must contain 0.

See the RLDR global /Z switch for loading information.

All DG/RDOS stand-alone programs conform to the rules. If you need to bootstrap a program that does not follow them, use the BOOT command and name the directory that contains the program. When BOOT asks `Filename?`, type the program name and append the /A switch.

Restart Feature

When you start BOOT, it tries to bootstrap the system specified. If you omit *sysname,* BOOT tries to bootstrap the default system name, SYS.SV. When BOOT finds an operating system, the operating system then tries to run RESTART.

Local Switches

/A Appending /A to your response to the `Filename?` query informs BOOT that you are loading a stand-alone program that does not conform to default BOOT conventions.

Examples

BOOT DE0

```
Master device released
Filename?  MYSYS
```

Loads BOOT.SV from DE0, releases the master directory, and asks for the system name. When you supply the system name MYSYS, BOOT bootstraps MYSYS from disk directory DE0. MYSYS requests the date and time, then activates the CLI.

BOOT DE0:SYS64K

Bootstraps the system named SYS64K from directory DE0.

BOOT DJ0

```
Master device released
Filename?  FOO.SV/A
```

Invoke BOOT and execute the program FOO.SV. FOO.SV does not conform to BOOT conventions, thus the /A switch.

BUILD

Command

Build a file consisting of filenames.

BUILD outputfilename inputfilename [...]

BUILD creates *outputfilename* and copies all qualifying *inputfilenames* into it. If *outputfilename* already exists, BUILD deletes and recreates it. You can use template characters and date switches to select the filenames. BUILD looks in the current directory to satisfy the templates and date switches you set. You can edit the resulting output file as needed.

BUILD is useful for creating a file containing filename arguments. You can use the output file as an argument to a command by invoking it as an indirect file. The command then uses each filename in the output file as a filename argument. This is particularly helpful with commands or utilities that do not allow the use of templates, or for using a given group of filenames repeatedly.

In *outputfilename,* the filenames are arranged in lines of 80 characters or less, and each line is terminated with the line continuation character (^).

Global Switches

/A Include all filenames, whether or not they have permanent attributes. (By default, BUILD does not recognize files that have the P attribute.)

/K Exclude all link entry filenames.

/N Write filenames to the output file without including their filename extensions.

Local Switches

mm-dd-yy/A Include only filenames whose creation dates are this date or after.

mm-dd-yy/B Include only filenames whose creation dates are before this date.

name/N Exclude from the output file any filenames that match the specified name. *Name* can include template characters.

Arguments *mm* (month) and *dd* (day) can be one or two digits.

Examples

BUILD ABC -.SR TEST-.

Creates file ABC and writes two categories of filenames into it: names that have a .SR extension, and names that begin with TEST and have no extension.

BUILD RBFILES -.RB ABC.RB/N

Creates RBFILES and includes all relocatable binary files (i.e., all filenames that have a .RB extension) except for file ABC.RB.

BUILD RECENT -.SR 8-2-84/A

Builds file RECENT from all filenames with .SR extensions created on or after August 2, 1984.

BUILD/A TAPEFILES -.SR TEST-.SR/N
LIST/S/A/E @TAPEFILES@

```
FOO.SR            266  D  07/06/84  16:31  07/06/84  [006564]  0
```

The BUILD command creates TAPEFILES, which contains all filenames that have a .SR extension, excluding .SR files whose filenames begin with the letters TEST. Because of the /A switch, TAPEFILES contains the names of permanent (attribute P) files.

The LIST/S/A/E @TAPEFILES@ command line invokes TAPEFILES as an indirect file (by enclosing the filename with @ symbols), as an argument to the LIST command. The result is a display, in alphabetical order, of the size, date created, and characteristics of each filename contained in file TAPEFILES.

BURST

Utility

Dump and load disk images between disk, diskette, or magnetic tape.

BURST [buffersize/S] [command] [source] [destination]

The main function of the BURST utility is to dump and restore disk images. However, BURST also enables you to duplicate the contents of one disk on another of the same type and to trace the ownership of disk blocks.

By default BURST uses 8 KB buffers for dump and load operations. The optional *buffersize/S* argument can be used to select a larger buffersize in order to improve the performance of the tape unit on certain systems. Refer to *How to Generate and Run DG/RDOS* and the DG/RDOS software release notice to determine if you can use the argument on your system.

When you enter the simple command *BURST,* a screen similar to the following appears:

```
        Disk backup and block owner utility
For help, enter ? as the response to any question
 or HELP as the response to the Command? question

              Command?
```

If you enter *?* in response to the `Command?` prompt, BURST displays a summary of commands. If you enter *HELP,* you receive a more detailed display. The BURST commands are:

DUMP[/V]	Copy the contents of a disk to diskettes or tapes. If you use the /V switch, the program also compares the data on your disk to the data it writes on your backup media.
DUPLICATE[/V]	Copy the contents of a disk to another disk of the same type. If you use the /V switch, the utility, after performing the copy, compares the information on your source and destination disks.
HELP	Display information about the program.
LOAD[/V]	Restore the information on your BURST backup media to your disk. If you use the /V switch, BURST compares the data on your backup diskettes or tapes to the data it writes to disk.
OWNER	Determine which file, if any, owns a given disk block.
STOP	Terminate the program.

The *source* is the disk you are dumping files from; the *destination* is the disk or tape drive you are dumping the files to. When you enter the BURST command and specify the desired

command and the source and destination, you do not see the screen shown above; the BURST operation starts immediately.

You can put a complete command line in a macro file for easy execution. For example, you could put the following command in a file named BACKUP.MC:

BURST DUMP/V DE0 DJ0

You can then type *BACKUP* and BURST will start a dump from disk DE0 to diskette drive DJ0 without displaying the initial screen.

For a complete discussion of the BURST utility, see *How to Generate and Run DG/RDOS.*

CCONT

Command

Create a contiguous file.

CCONT filename blockcount [...filename blockcount]

CCONT creates a contiguously organized file in the current directory or in the specified directory. The filename you assign can be in a pathname. The file will have the fixed length you specify as the blockcount argument; blockcount is a decimal number (one block equals 512 bytes). The file will have the characteristic C, which indicates a contiguous file.

To insert information into the file, you can use a text editor or you can transfer information into it with the XFER command. The data in the file cannot exceed the *blockcount* length you specify.

Global Switches

/N Do not zero the data words in the file. Without /N, CCONT fills the file with null values; since this process can take an extremely long time, you should normally use this switch.

Examples

```
CCONT NEWFILE 16
LIST NEWFILE
NEWFILE                 8192  C
```

Creates the contiguous file NEWFILE in the current directory and allocates it a fixed length of 16 contiguous disk blocks. Note that LIST returns the size in bytes: 512 (bytes per block) multiplied by 16 (number of blocks) equals 8192 bytes.

```
CCONT FILE4 24 DE0:PART2:ACCESS 48
LIST FILE4 DE0:PART2:ACCESS
FILE4                  12288   C
DE0:PART2:ACCESS       24576   C
```

Creates FILE4 in the current directory, allocating a fixed length of 24 disk blocks to the file; and creates the file ACCESS in secondary partition PART2 on disk DE0, allocating 48 disk blocks to the file.

CDIR

Command

Create a subdirectory.

CDIR directoryname[.DR]

CDIR creates subdirectory *directoryname,* a variable-length directory in which to store files. Subdirectories can reside in a primary or secondary partition of a disk. The subdirectory is the lowest level of the three-tiered DG/RDOS directory structure.

The *directoryname* can be a pathname. DG/RDOS appends the .DR extension (the filename extension for directories) if you do not supply it.

DG/RDOS assigns the subdirectory an initial length of 512 bytes (one block), and assigns the file characteristics DY, where:

D denotes a random file
Y denotes a directory

You should assign each of your subdirectories a unique name, as you can initialize only one subdirectory of a given name at any one time.

You can create subdirectories from within a primary partition or a secondary partition. If you attempt to use CDIR to create a subdirectory within another subdirectory, CDIR does not execute and you receive the message `Directory depth exceeded.`

Examples

DIR SECPART
CDIR WHATEVER

Creates WHATEVER.DR as a subdirectory to secondary partition SECPART.

CDIR DA1:DRONER

Creates subdirectory DRONER.DR on disk DA1 (a primary partition), using a directory specifier to indicate that the subdirectory is to be created in DA1. Equivalent commands would be:

DIR DA1
CDIR DRONER

Note that when you use the name of a subdirectory as a directory name argument (with the DIR command, for example), you do not need to include the .DR extension:

DIR WHATEVER

However, when you use the name of a subdirectory as a filename argument (with the LIST command, for example), you must specify the extension:

LIST WHATEVER.DR

```
WHATEVER.DR              512  DY
```

CHAIN

Command

Load and run a program in place of the CLI.

CHAIN programname

CHAIN overwrites the CLI by chaining *programname* (an .SV file) into execution on the current level. The program should chain control back to the CLI with the .EXEC system call.

CHAIN is useful for programs that use five swap levels. (When you execute a program from the level 0 CLI, it runs on level 1; hence it has only levels 2, 3, and 4 available.)

Be careful when you use CHAIN. If the program you chain issues a .RTN instruction from level 0 in the background, the system tries to chain to CLI. You may get the message `Insert system disk; strike any key to continue.` In this case, press <NL> to return to the CLI. Also, the CTRL-C CTRL-A or CTRL-C CTRL-B sequences may terminate DG/RDOS.

Note that you cannot CHAIN when the log file is open in the current ground; close this file with the ENDLOG command.

Global Switches

/D Pass control to the debugger.

Examples

CHAIN BEHEMOTH

Program BEHEMOTH.SV executes on the level normally used by the CLI. The program uses the .EXEC system call to return control to the CLI.

CHATR

Command

Change a file's attributes.

CHATR filename [sign] attributes [...filename [sign] attributes]

Assigns or removes file access attributes for a file, thus controlling the types of access permitted to the file.

The *sign* can be + (plus), to add an attribute, or - (minus), to remove an attribute. When you specify an attribute without a sign, the new attribute replaces all existing attributes.

To remove all attributes (except attribute S), enter 0 (zero) as an argument. A file having no attributes allows full access to the file, except for execution access. In order to execute a file, the file *must* have the S attribute. The RLDR utility assigns this attribute when it creates an executable program file.

The LIST command displays file access attributes for a file as part of its output, as follows:

```
FILENAME      sss   xxxx
```

where *sss* represents the size of the file (in bytes) and *xxxx* represents the letters designating file characteristics and attributes. File attributes and characteristics are listed below.

The file access attributes for a file apply to all methods of accessing the file, including:

- Direct access from within the file's directory.
- Direct access through use of a directory specifier.
- Indirect access through a link file. (See the CHLAT command for a discussion of link access attributes, which control indirect access to a file but do not affect direct access.)

File Access Attributes

The file access attributes that you can set with the CHATR command are:

N Prohibit links to the file. (You can create a link to the file from another directory, but the link will not work.)

P Make the file permanent. The file cannot be deleted or renamed when it has the P attribute. Note, however, that the parent directory can be deleted unless *it* is permanent. Note also that the BUILD, DUMP, LIST, LOAD, or MOVE commands do not work on files with the P attribute unless you include the /A switch in the command line.

R Read-protect the file. You cannot read the file (with the TYPE or PRINT commands), edit it with a text editor, or copy it with the DUMP, IMOVE, MOVE, or XFER commands when R is set. However, the file can be executed.

S Permit execution of the file. RLDR assigns this attribute when it creates an executable program file. If you remove this attribute, the file cannot be executed.

W Write-protect the file. You cannot modify the file. Note that W does not protect a file from deletion.

0 Remove all removable attributes, except attribute S.

* Retain all existing attributes.

? User-defined attribute.

& User-defined attribute.

Examples

CHATR OLDFILE 0 NEWFILE R

Removes all attributes of OLDFILE and replaces existing attributes for NEWFILE with the read-protect attribute R.

CHATR MYFILE -R +W

Removes the read-protect attribute R from MYFILE and assigns the write-protect attribute W. All existing attributes remain the same.

CHATR PASSWORDS 0

Removes all existing attributes except S from file PASSWORDS.

CHATR PASSWORDS R

Assigns the read-protect attribute R to file PASSWORDS. Commands that display the contents of files are not effective for this file unless you first remove the R attribute.

CHLAT

Command

Change a file's link access attributes.

CHLAT filename [sign] attributes [...filename [sign] attributes]

Use CHLAT to assign or remove link access attributes for *filename*. A file's link access attributes control a link user's access rights to that file. *Sign* can be + (plus) to add an attribute, or - (minus) to remove an attribute. When you specify an attribute without a sign, the new attribute replaces all existing attributes.

To remove all link access attributes, enter 0 (zero) as an argument.

See the CHATR command for information on setting file access attributes. The difference between file access attributes and link access attributes is that file access attributes apply to all methods of access, including link access. Link access attributes apply only to link access; they have no effect when a file is accessed from within its own directory or through use of a directory specifier.

The LIST command displays link access attributes for a file, by appending them, preceded by a slash (/), to the display of any existing file access attributes.

See the LINK command for information on creating and using link files.

Link Access Attributes

N Prohibit links to the file. (You can create a link to the file from another directory, but the link will not work.)

P Make the file permanent. Users linking to this file cannot delete or rename it.

R Read-protect the file. Users linking to this file cannot display the file but can execute it.

S Permit execution of the file.

W Write-protect the file. Users linking to this file cannot modify the file. (Note that users can delete the file if they use DELETE on their link file. See the UNLINK command for information about removing links.)

0 Remove all removable attributes, except attribute S.

* Retain all existing attributes.

? User-defined attribute.

& User-defined attribute.

Examples

LIST ROSETTA.SV
```
ROSETTA.SV      331   SD
```

The LIST command shows that file ROSETTA.SV has no link access attributes, but has file attribute S (execute) and file characteristic D (random file).

CHLAT STONE.SV P
LIST STONE.SV
```
STONE.SV        331   SD/P
```

The CHLAT command assigns the permanent (P) link access attribute, ensuring that users who link to the file STONE cannot delete it.

CLEAR

Command

Set file use count to zero.

CLEAR [filename ...]

Clear the file use count in one or more SYS.DR entries. Each file's use count is 1 or more when the file is open, and 0 when the file is closed normally. If a system fails when a file is open, its use count remains nonzero when the system is rebootstrapped, and you cannot delete or rename the file.

Use the LIST/U or LIST/E command to determine a file's use count.

If the operating system shuts down abnormally, when you next bring up the system you receive the message:

```
Partition in use - type C to continue
```

Type C, log on, then type *CLEAR/A/V/D* from the master directory and from all other directories that were initialized at the abnormal shutdown. After clearing all directories, re-boot DG/RDOS.

You may also receive the message `Partition in use` if someone shut off power to the computer before releasing the master directory. Again, clear all directories that were initialized when power was turned off and re-boot.

CLEAR works only from the background CLI, at level 0, with no foreground program running.

Global Switches

/A Clear use count in all files in the current directory except the current CLI.OL, CLI.ER, sysname.OL, and LOG.CM. To clear these files, enter their names as arguments to CLEAR. (Arguments are ignored when you use this switch.)

/D Clear device entries.

/V Display on the terminal the names of the files cleared.

Examples

CLEAR/A/V/D
```
Cleared SYS.DR
```

Clears use count of all files and devices in the current directory except CLI files; verify filenames cleared.

CLEAR/V DE0:DEMPSEY
```
Cleared DE0:DEMPSEY
```

Clears and verifies use count of file DEMPSEY, in directory DE0.

CONFIG

Utility

Examine or change system configuration parameters.

CONFIG [systemname] [dialogfile.CF/V]

The CONFIG utility allows you to configure your system and hardware. When you install a DG/RDOS update or revision, or add or change a piece of hardware, you normally use SYSGEN to specify the system hardware, followed by CONFIG to define more precisely the parameters for memory usage, devices, and communications lines.

Systemname is the name of the system that you are configuring.

If you include the *dialogfile.CF/V* argument, DG/RDOS writes to this file a record of the parameters you specify during the CONFIG dialog. If you want to review your system configuration at a later date, you can type or print this file rather than running CONFIG. Note that if you use this argument, you *must* include both the .CF extension and the /V switch..

If you do not specify *systemname* on the command line, the first CONFIG screen prompts you for the name. The default value for *systemname* is DGRDOS.SV.

If you do specify *systemname,* CONFIG displays the first screen of system parameters. You then make the appropriate choices from the menu at the bottom of the screen. CONFIG displays additional screens as necessary.

If you are uncertain how to answer a question on any screen, you can type *?* or press SHIFT-F1 to display a help message.

After you have finished running CONFIG and have installed changes in your system, type the command:

BOOT systemname

This command shuts down your system and then restarts it with the new configuration parameters.

For a complete discussion of the CONFIG utility, see *How to Generate and Run DG/RDOS.*

Global Switches

/H Display help information about the CONFIG utility.

Examples

CONFIG DGRDOS JUL2286.CF/V <NL>

```
(first CONFIG screen)
.
.
(exit from utility)
Writing dialog to JUL2286.CF
```

Starts the CONFIG utility and displays the first screen of parameters. After you exit from CONFIG and install changes in the system, CONFIG writes the dialog to the file *JUL2286.CF*.

CPART

Command

Create a secondary partition.

CPART partitionname[.DR] blockcount

CPART creates the secondary partition *partitionname.* A secondary partition is a directory containing a fixed amount of contiguous disk space for storing files and subdirectories.

You supply a name and the number of contiguous blocks to be reserved for the partition on the CPART command line. If the number you specify is not an integer multiple of 16, the *blockcount* defaults to the next lower multiple. You cannot specify fewer than 48 blocks.

DG/RDOS appends the .DR extension (the filename extension for directories) unless you supply it, and also assigns the secondary partition the file characteristics CTY, where:

C denotes a contiguous file
T denotes a secondary partition
Y denotes a directory

Secondary partitions must be subsets of a primary partition. If you attempt to create a secondary partition from a directory other than a primary partition, the CPART command does not execute, and you receive the error message `Directory depth exceeded.`

Examples

DIR DJ1
CPART SECPART 144

The DIR command makes the primary partition DJ1 (the device name for a diskette) the current directory. The CPART command creates a secondary partition called SECPART.DR on DJ1, and allocates a fixed length of 144 contiguous disk blocks to it.

When you use SECPART.DR as a directory argument, you do not have to specify its .DR extension. The command

DIR SECPART

makes SECPART the current directory. However, when you use SECPART.DR as a filename argument, you must include its extension, as in the command:

LIST SECPART.DR

```
SECPART.DR          73728  CTY
```

Note that the LIST command displays the 144 blocks as 73728 bytes. (One disk block equals 512 bytes; 144 disk blocks equal 73728 bytes.)

CRAND

Command

Create a random file.

CRAND filename [...filename]

CRAND creates filename, a randomly organized file in the current directory, or in the directory specified in a pathname. The created file will have an initial length of 0, no file attributes, and the file characteristic D, which indicates a random file.

To insert information into the file, use a text editor or the CLI command XFER. The length of the file expands as you add data to it.

Note that an executable program (.SV) file must be a random file.

Examples

CRAND NEWFILE
LIST
```
NEWFILE          0  D
```

Creates the random file NEWFILE in the current directory.

CRAND FILE1 FILE2 DE0:CORP:MEETINGS
LIST DE0:CORP:MEETINGS
```
DE0:CORP:MEETINGS       0  D
```

Creates FILE1 and FILE2 in the current directory and creates the file MEETINGS in directory CORP on disk DE0.

CREATE

Command

Create a sequential file.

CREATE filename [...filename]

CREATE creates a sequentially organized file in the current directory or in the specified directory. The file will have an initial length of 0 and is not assigned any file characteristics.

To add information to the file, use a text editor or the CLI command XFER. The length of the file expands as you add data to it.

Examples

```
CREATE SEQFILE
LIST
SEQFILE                 0
```

Creates the sequential file SEQFILE in the current directory.

```
CREATE FILE3 DA1:PART1:DATAFILE
LIST FILE3 DA1:PART1:DATAFILE
FILE3                   0
DA1:PART1:DATAFILE      0
```

Creates FILE3 in the current directory and DATAFILE in secondary partition PART1 on disk DA1.

DEB

Command

Load a program into memory and start executing at the debugger address.

DEB programname

DEB lets you debug a program while executing it. A symbolic debugger must have been loaded as part of the program .SV file; you do this with the /D switch in RLDR. DEB transfers control to the debugger within the .SV file; the debugger then performs a line feed and awaits debugging commands.

While debugging, you can examine memory, set break points, and run the program. After making any necessary changes, you can return to the CLI by running the program to its normal termination. You can also return to the CLI by issuing the ESC-V debug command or the DG/RDOS CTRL-C CTRL-A sequence.

After successfully debugging your program, you will probably want to correct the source program, and then reassemble and reload it.

For more information, see *RDOS/DOS Debugging Utilities.*

Examples

DEB PROG.SV

START + 15/006751
START + 15$B
$R
.
.
7BSTART + 15
0... 1... 2... 3...$P
.
.
$V
BREAK

The DEB command line invokes the debugger for file PROG.SV. The debug commands examine a location, set a breakpoint at START + 15, and start the program executing. (The ESC key in debug commands echoes as a $.) Debugging continues. ESC-V terminates the debugging session and returns to the CLI.

DELETE

Command

Delete a file or directory.

DELETE filename [...filename]

Deletes the files named. You can use pathnames to name files in directories if the directories have been initialized.

To delete a subdirectory or partition, you must first release it (with the RELEASE command), then type *DELETE directoryname.DR*. When you delete a directory, the CLI erases the directory and all of the files it contains.

CAUTION: Do *not* use the DELETE command on a link file; it erases the resolution file.

The DELETE command does not delete files that have the attribute P (permanent). See the CHATR command for more information.

Global Switches

/C Request confirmation of each deletion. The system repeats each filename, then waits for you to confirm the deletion by pressing the NEW LINE key. When you press <NL>, an asterisk indicates that the file has been deleted. To prevent the deletion, press any key other than the command line terminator. The key you press will not be echoed; the CLI immediately displays the name of the next file.

We recommend strongly that you *always* request confirmation of deletion when you use templates with the DELETE command in order to avoid accidentally deleting the wrong file. As an added precaution, use the LIST command with the same template before using DELETE.

/L Print a list of the filenames of deleted files on the line printer ($LPT). This overrides the /V switch.

/V Verify deletion by displaying filenames at the terminal.

Local Switches

mm-dd-yy/A Delete only files created on this date or after.

mm-dd-yy/B Delete only files created before this date.

name/N Do not delete any files that match this name.

Arguments *mm* (month) and *dd* (day) can be one or two digits.

Any command line arguments that include templates are affected by local switches, though the switches might be attached to other arguments.

Examples

DELETE/C PROGX.-
PROGX.: *<NL>* *
PROGX.LS: *<NL>* *
PROGX.SR: *[N]*
PROGX.RB: *<NL>* *
PROGX.SV: *[N]*

The command line instructs the CLI to delete all files named PROGX with any extension, and the /C switch allows us to confirm the deletions.

The CLI displays each filename, one at a time, that matches the template PROGX.-. Press the command line terminator in response to each file you want to delete; the CLI deletes the file and displays an asterisk. Type *N* or any other letter in response to each file you want to keep. The CLI does not echo the N, and continues with the command.

DELETE/V/C -.LS
A.LS: *<NL>* * Deleted A.LS
COM.LS: *<NL>* * Deleted COM.LS
MAP.LS: *<NL>* * Deleted MAP.LS

Deletes, after confirmation, all files having an .LS extension, verifying the deletion by listing the filenames.

DELETE/V/C -CHART-.- 1-6-84/B
NEWCHART.LS: *<NL>* * Deleted NEWCHART.LS
NEWCHART.IN: *<NL>* * Deleted NEWCHART.IN
XCHART1.OU: *<NL>* * Deleted XCHART1.OU
XCHART4.OU: *<NL>* * Deleted XCHART4.OU

Deletes with confirmation and verification all files that have the characters CHART in the middle of their names and have creation dates earlier than January 6, 1984.

DELETE GRAPHICS:FOOD.SR
R

Deletes the file FOOD.SR in directory GRAPHICS. Note that without global switches, the only indication of the deletion is the return of the **R** prompt.

DIR

Command

Change the current directory.

DIR directoryname

DIR makes *directoryname* your new current directory. Directoryname can be a subdirectory, secondary partition, or primary partition. Directoryname can be in a pathname.

The current directory is the directory from which you work and whose files you can access without pathnames. When you specify filename arguments without including directory specifiers, DG/RDOS assumes you are referring to files in the current directory.

If the directory you specify is not already initialized (opened for access), DIR initializes it. (See the INIT command description for more information on directory initialization.) When you change your current directory, the previous current directory remains initialized.

To access a directory with the DIR command, its parent directories, if any, must be initialized. If you use a pathname in an argument to the command, (as in the command *DIR PART1:SECPART:SUBDIR*), DIR initializes all directories listed in the path. When a directory is already initialized, you can supply its name to the DIR command without using a pathname, from anywhere within the directory structure.

When you start DG/RDOS, the master directory is your current directory.

You close directories that have been initialized through the DIR command with the RELEASE command. If you release the current directory, DG/RDOS places you in the master directory. (See the RELEASE command description for more information.)

Other commands that are helpful with the DIR command are GDIR, which displays the name of the current directory, MDIR, which displays the name of the master directory, and LDIR, which displays the name of the previous current directory.

Examples

DIR PARSETZ
GDIR
PARSETZ

The command DIR PARSETZ makes directory PARSETZ the new current directory. The GDIR command displays the name of the current directory—in this example, PARSETZ.

DIR DJ0:APPLICAT:DBMS
GDIR
DBMS

This example makes directory DBMS the new current directory. Directory DBMS resides on diskette DJ0 in secondary partition APPLICAT. DIR initializes all directories listed in its directory specifier argument.

DIR PARSETZ
GDIR
PARSETZ

We know that directory PARSETZ is initialized because it was a previous current directory. We can therefore use it as an argument to the DIR command without specifying all the directories in the path from directory DBMS to directory PARSETZ.

DISK

Command

Display the allocation of disk storage space for the current partition.

DISK

Displays the current allocation of disk space, in decimal blocks, for the current partition. One disk block equals 512 (decimal) bytes.

The display reports the allocation of disk blocks for the current partition as follows:

```
Left:  fffff    Used:  uuuuu    Max. contiguous:  ccccc
```

where:

`fffff` is the number of blocks free for use.

`uuuuu` is the number of blocks in use.

`ccccc` is the number of blocks forming the largest contiguous unallocated space.

If you add the number of unused blocks to the number of blocks in use, the result is the total amount of space allocated to the partition.

If the current directory is a subdirectory, DISK calculates the space usage of its parent partition. If the current directory is a primary partition, DISK surveys the entire disk for space usage.

When determining space requirements, note that the bootstrap program and system files require at least 16 blocks for the master directory. Other directories require some space for their system and map directories (SYS.DR and MAP.DR).

Examples

DIR DJ1
DISK
```
Left: 520     Used: 88     Max. contiguous: 414
```

This response, from diskette DJ0, indicates that 520 out of 608 blocks are still available for use and that the largest amount of free contiguous blocks is 414.

DIR DE0
DISK
```
Left: 1499     Used: 23161   Max. contiguous: 910
```

Disk DE0 has 1499 free blocks and 23161 blocks in use, out of 37660 blocks. The largest free contiguous space is 910 blocks.

DISK
```
Left: 1478     Used: 8298    Max. contiguous: 544
```

This response indicates that 1478 blocks from the original 9776 of the disk are still available for use, with 544 contiguous blocks available.

DO

Utility

Execute command files, replacing dummy arguments with the specified arguments.

DO filename [argument ...]

The DO utility executes command files—groups of CLI commands and their actual or dummy arguments—replacing dummy arguments within the files with the arguments you specify on the command line.

Placing a global /H switch in the command line outputs a help message.

The utility requires that all lines in the command file and all lines output to (F)CLI.CM conform to standard .RDL/.WRL formats: lines cannot exceed 132 characters and must contain a terminator.

Dummy Arguments

A dummy argument has the form %n%, where n is a number other than 0. The DO utility can handle up to 256 dummy arguments in a single command file.

You use a dummy argument in a command line in this way:

PRINT %1%

If you supplied a filename argument such as THISFILE, the command line would expand to read:

PRINT THISFILE

Supplying the Arguments on the DO Command Line

DO uses the first argument you specify on the command line to expand dummy argument %1%; the second argument on the command line corresponds to dummy argument %2%, and so on. If a particular dummy argument appears more than once in the command file, DO replaces the dummy argument with its corresponding actual argument each time it appears throughout the command file.

If you specify fewer arguments on the command line than exist in the command file, DO replaces the leftover dummy arguments in the command file with nulls.

You can specify a null on the command line by using the number sign, #, as an argument. For example, the following line exists in the command file PRINTIT.DO:

PRINT %1% %2% %3%

The command line DO PRINTIT.DO FILE1 FILE2 FILE3 causes the line to expand to:

PRINT FILE1 FILE2 FILE3

The command line DO PRINTIT.DO # FILE2 causes the line to expand to:

PRINT FILE2

Examples

TYPE CLOSEDIR.DO

```
MESSAGE CLOSING DIRECTORY %1% ...
DIR %1%
DELETE/V -.BU
MESSAGE PRINTING DIRECTORY %1% LISTING ON THE PRINTER
LIST/S/A/E/L
RELEASE %1%
```

Displays the contents of command file CLOSEDIR.DO, showing that it contains a dummy argument, %1%, allowing CLOSEDIR.DO to work for any directory you specify.

DO CLOSEDIR.DO PROGS

```
CLOSING DIRECTORY PROGS ...
DELETED TEST1.BU
DELETED FOO.BU
PRINTING DIRECTORY PROGS LISTING ON THE PRINTER
```

The DO command line executes the command file CLOSEDIR.DO, supplying one argument, PROGS, to replace any %1% dummy arguments that exist in the command file.

DUMP

Command

Store one or more disk files from the current directory in a dump format file on another directory or device.

DUMP [destination:]dumpfilename [filename...]

Destination specifies the name of the directory or device (disk, diskette, or magnetic tape) to which you are transferring the files. This directory or device must be initialized. If you omit this part of the argument, DG/RDOS creates the dump file in your current directory, which is probably not what you intend.

Dumpfilename is the name of the file that will contain the dumped files. You need to use this name when you load the files back (with the LOAD command). When dumping to disk, DUMP creates this file. When dumping to tape, give a tape file number, such as MT0:0, for this argument.

Filename specifies the files to be dumped. If you do not specify any filenames, DUMP dumps all nonpermanent files (files that do not have the attribute P) and directories subordinate to the current directory. If the current directory is a primary partition, all nonpermanent files are dumped; if the current directory is a secondary partition, the secondary partition and all its subdirectories and files are dumped.

If you specify filename arguments, only those files are dumped. To dump directories subordinate to the current directory (and their contents), specify their names with the .DR extension. Note that you cannot use a primary partition name such as DE0 as a valid filename argument.

You can use templates when referring to files in the current directory.

DUMP creates at the named destination the dump file *dumpfilename* to contain the files dumped from the current directory.

The dump file contains file information and file contents for each file. When you load the dump file back to disk (with the LOAD command), the information is used to reconstruct the original files and directory structures.

Use DUMP to create backup copies of files and directories for storage rather than for interactive use. If you want to copy files and directories to another disk or directory and use them just as you would in your current directory, use the MOVE command.

NOTE: For system backup you should use IMOVE or BURST rather than DUMP. See chapter 5 for details.

When you use the /V or /L switches, DUMP displays the names of dumped files, each on a separate line, as follows: files from the current directory are indented two spaces, directories are preceded by an asterisk, and files from a subordinate directory are indented four spaces, to help you determine exactly which files were dumped.

Global Switches

/A Include permanent (attribute P) files in the dump.

/K Excludes link files.

/L List on the line printer (device name $LPT) the names of the dumped files (This switch overrides the /V switch.)

/V Display on the terminal the filenames of each file dumped.

/W Include the resolution files for any link files dumped, provided that all directories in the path to the resolution file are currently initialized.

Local Switches

mm-dd-yy/A	Dump only files created on this date or after.
mm-dd-yy/B	Dump only files created before this date.
filename/N	Exclude files that match filename from the transfer; filename may include templates.
oldname/S newname	Assign *newname* to file *oldname* when it is transferred, but the file retains its original name (oldname) in the current directory.

Arguments *mm* (month) and *dd* (day) can be one or two digits. Local switches /A, /B, and /N do not affect filename arguments that are specified explicitly (i.e., named without the use of templates) on the command line.

Examples

```
DIR MYDIR
INIT MT0
DUMP/A/L MT0:0 7-12-85/A
RELEASE MT0
```

Dumps all directories and files, including permanent (attribute P) files (/A), from current directory MYDIR that were created on or after July 12, 1985, to file 0 of the tape on drive MT0. This tape now provides a backup for these files. The line printer produces a listing of the dumped filenames (/L).

DUMP/V DA1:SAVEFILES -.SV

```
RED.SV
BLUE.SV
.
.
CHARTREUSE.SV
```

Dumps to file SAVEFILES on disk DA1 all files in the current directory with an .SV extension, and lists at the terminal (/V) the filename of each file dumped.

DUMP/A MT0:2 APRIL/S APRIL.BU

Dumps permanent file APRIL to file 2 of the tape on MT0. Names the dumped copy APRIL.BU (which allows you to load it into the same directory as APRIL, if necessary).

DUMP/A/L DE0:85JUN22.BU -.RB/N 6-22-85/A

Creates dumpfile 85JUN22.BU (named for a date) on disk DE0, then dumps all files (except .RB files) created on or after June 22, 1985 to file 85JUN22.BU on disk DE0. /L lists, on the line printer, the filenames of each file and directory dumped.

ENDLOG

Command

Close the log file opened by LOG.

ENDLOG [password]

ENDLOG orders the system to stop output to the log file previously opened with a LOG command. You must close a log file before you can examine or delete it.

To print or delete the log file after closing it, use its full name as an argument. The LOG command names the background log file LOG.CM, and the foreground log file FLOG.CM.

If you included a password argument with the LOG command to open the log file, you must include the same password argument with ENDLOG to close the log file. This command, ENDLOG [password], appears as a last entry in the log file.

Examples

LOG/H GSTONE

```
[terminal dialogue after you opened the log file]
.
```

ENDLOG GSTONE

The password GSTONE is required with ENDLOG since it was used when the log file was opened with LOG/H.

ENPAT

Utility

Create a patch file.

ENPAT [patchfile]

The ENPAT utility lets you create patches for .SV and .OL files; a patch is a change to a program or overlay file. Patches are often used to update operating system files. Once you create a patchfile with ENPAT, you install the patches into the program or overlay file with the PATCH utility.

ENPAT creates *patchfile,* or opens it for appending if it already exists. ENPAT then asks five questions about each patch, accepts valid answers, and places them in the patchfile. Later you install the patchfile with the PATCH utility.

A patch for a program (.SV) file can contain symbols (provided that a load map is available to the PATCH program), octal numbers, or expressions including a symbol, an operator, and an octal number. The most common operators are:

+ (addition)
- (subtraction)
@ (indirection)

The PATCH program cannot resolve symbols unless you have a load map of the save file on disk. You can instruct SYSGEN to save such a map with the SYSGEN local switch /L.

ENPAT is explained in greater detail in *RDOS/DOS Debugging Utilities.*

ENPAT asks the following questions for each one-word patch. If you give an invalid response, it returns an error message and repeats the question.

```
SAVE FILE (0) OR OVERLAY FILE(1)?
```

Answer 0 if the patches in this file will be installed in an .SV file; answer 1 for a .OL file. Next, it asks:

```
PATCH LOCATION?
```

Enter the location to be patched—number, symbol or expression. ENPAT now asks about the contents of this location:

```
CURRENT CONTENTS?
```
Type in the current contents of the location.

Now, ENPAT asks:

```
NEW CONTENTS?
```

Respond with the new contents. Next, ENPAT asks about conditions for patch installation:

```
CONDITIONAL?
```

If you want to install the patch unconditionally, simply press <NL>. To install the patch only if a symbol is defined in the load map, enter the symbol name. (You might do this if the patch relates to a given module, and you don't know if the module is part of your system.)

For example, assume that a patch relates only to a device driver named ALPHA. You want this patch installed only if ALPHA is defined in your system load map, so you type *ALPHA*. (To install the patch only if the symbol is *not* defined in the load map, type a minus sign (-), then the symbol name; e.g., -ALPHA).

If you want this patch installed conditionally, and the condition is the same as you entered for the previous patch, press SHIFT-6 and <NL> in response to this question. This allows you to enter multiple patches that concern one condition.

Next, ENPAT wants to know if you have more patches to enter:

```
EXIT (0=NO, 1=YES)
```

Answer 0 to return to question 1, and enter another patch; answer 1 to create the patch file and return to the CLI.

Examples

You have the patches listed in the following table.

Location	**Old Contents**	**New Contents**
IPBQ-1	100000+IPSTK	401@0
(401@0)+15	0	100000+IPSTK
401	401@0	401@0+16

The following illustrates the insertion of the patches in patch file IPB.PF, for later installation in save file SYS.SV:

```
ENPAT IPB.PF
CREATING NEW PATCHFILE
SAVE FILE (0) OR OVERLAY FILE (1)? 0
PATCH LOCATION: IBPQ-1
CURRENT CONTENTS: 100000+IPSTK
NEW CONTENTS: 401@0
CONDITIONAL: <NL>
EXIT (0=NO, 1=YES)? 0
SAVE FILE (0) OR OVERLAY FILE (1)? 0
PATCH LOCATION: (401@0)+15
CURRENT CONTENTS: 0
NEW CONTENTS: 100000+IPSTK
CONDITIONAL: <NL>
EXIT (0=NO, 1=YES)? 0
SAVE FILE (0) OR OVERLAY FILE (1)? 0
PATCH LOCATION: 401
CURRENT CONTENTS: 401@0
NEW CONTENTS: 401@0+16
CONDITIONAL: <NL>
EXIT (0=NO, 1=YES) 1
```

The PATCH utility example in this chapter continues this example to show installation of the patches in patchfile IPB.PF.

EQUIV

Command

Assign a temporary name to a disk or tape drive.

EQUIV newname oldname

EQUIV assigns the temporary name *newname* to a disk or magnetic tape unit. *Newname* replaces *oldname* until the device is released (with RELEASE).

Properly applied, EQUIV enables you to write device-independent programs. You can write a generic device specifier in your programs, and use EQUIV at runtime to match the generic specifier to the actual device.

You must rename a device *before* you initialize it. The new name then exists only until you release the device. After release, all devices revert to their original specifiers. You cannot use EQUIV on a secondary partition, subdirectory, or the master device.

Global Switches

/P Display a message to mount a tape or disk, and pause for user intervention.

Examples

Your program refers to all magnetic tape files as TAPEDECK. This gives your program device independence at runtime. Before running the program, you issue the command:

EQUIV/P TAPEDECK MT0
```
MOUNT TAPEDECK ON UNIT MT0, STRIKE ANY KEY
```

This command temporarily changes the name of tape drive MT0 to TAPEDECK. The /P switch produces the request to mount the tape. After you issue the command, DG/RDOS no longer recognizes tape drive MT0. For example, if you type *INIT MT0,* you receive the message `File does not exist: MT0.` You must use the new name for all tape operations until you release the device.

INIT TAPEDECK
LOAD/V TAPEDECK:1
```
MASM2      05/17/82
MASM4.LS   06/01/82
```

The INIT command initializes the tape on drive MT0 (TAPEDECK). The LOAD command transfers the files from the second file (file 1) on tape MT0 (TAPEDECK) to disk.

EQUIV MT0 TAPEDECK
`Directory in use`

You cannot use EQUIV again for the same device until you release the device.

RELEASE TAPEDECK

The RELEASE command releases tape drive TAPEDECK. When you release the device, it reverts back to the original name MT0.

EXFG

Command

Execute a program in foreground memory.

EXFG programname

When there is no memory separation or when you want to use the background, you execute programs by typing in the name of the program. Once you have allocated memory for a foreground with the SMEM command, you execute a foreground program with the EXFG command. You terminate a foreground program by typing CTRL-C CTRL-F from the background console.

For an explanation of memory separation and the sharing of resources between the two grounds, see the SMEM command. Refer also to the GMEM command, which displays the amounts of memory allocated to the two grounds.

You can execute any program in the foreground if you have allocated enough memory with the SMEM command. You can execute noninteractive programs (such as assemblies or compilations) in the foreground, or you can execute interactive programs like the CLI, and access them through the foreground console.

A foreground CLI has different names for system files; LOG.CM, COM.CM, CLI.CM, and BREAK.SV are named FLOG.CM, FCOM.CM, FCLI.CM, and FBREAK.SV.

To execute an assembly, compilation, or any program in the foreground, type the command line exactly as you would for the background, but precede it with EXFG. For example, to execute the program SPROING.SV in the foreground, use the following command line:

EXFG SPROING

You can also execute utilities in the foreground by preceding the utility name and any needed arguments with the EXFG command.

Global Switches

/D Pass program control to the debugger.

/E Assign equal priority to foreground and background. (Normally, the foreground program has higher priority.)

Examples

EXFG DA1:CRESS

Executes program CRESS, on primary partition DA1, in foreground memory. The foreground as set by SMEM must accommodate CRESS; if CRESS won't fit, the system returns the error message:

```
Insufficient memory to execute program
```

If CRESS executes, the CLI remains active in the background; you can then try to execute a different program in the background. Again, if the background does not have enough space to execute the program, DG/RDOS displays the `Insufficient memory` message.

EXFG MAC/L/U PARU/S USR/S SOURCE<1,2,3,4> BACKUP PROG<1,2,3,4,5>

Assembles a complex program in the foreground.

FCOPY

Utility

Copy a file or duplicate a diskette.

FCOPY [source destination]

Source specifies the device name of the diskette or the name of the file that you want to duplicate. This argument can include a pathname to specify a file outside the current directory, but cannot include templates.

Destination specifies the device name of the diskette or the filename on which FCOPY creates the duplicate. The diskette must have a Data General hardware format, but it does not need to be software formatted with DKINIT and INIT/F.

If you omit arguments on the command line, FCOPY presents a series of menus from which you make choices. If you supply the arguments, you are asked only to confirm that the pathnames on the command line are correct.

If your system has only one diskette drive, you must use FCOPY to copy information from one diskette to another. FCOPY is also useful if you want to duplicate a diskette on systems with either one or two drives, since it does not require that the destination diskette be software formatted.

If your system has two diskette drives, or if you want to copy multiple files, MOVE is preferable to FCOPY. Although MOVE does require that the destination diskette be software formatted, it allows you to name multiple files on one command line or to use filename templates.

FCOPY stops with an error message if the destination diskette has a bad block. If you are duplicating a diskette, you must use another one that has no bad blocks.

Global Switches

/C Copy a single file.

/D Duplicate a diskette.

/V Check the copy to ensure that it is readable. We recommend that you always use this switch.

See *How to Generate and Run DG/RDOS* for a detailed description of FCOPY.

Examples

FCOPY <NL>

```
DG/RDOS Diskette Transfer Utility     REV n.nn
Command:             STRIKE FUNCTION-KEY-1 FOR OPTIONS
-------------------------------------------------------
    1.   Copy a file
    2.   Duplicate a Diskette
    3.   Exit from program

Select desired command: 2
Enter source diskette name: DJ0
Enter destination diskette name: DJ0

Do you want verification of the duplicate?  (Y/N) Y

Writing time will be longer due to the verification option.

Select CONTINUE if the pathnames above are correct.

(1) Continue (2) Restart (3) Return to Main Menu
    Select desired operation: 1

Insert SOURCE diskette:  DJ0

Elapsed time  min:sec
Reading from source file ...

Insert DESTINATION diskette: DJ0
Elapsed time  min:sec
Writing to destination file ....
.
.
Transfer and verification completed

(1) Continue (2) Restart (3) Return to Main Menu
```

When you select 2 on the main menu to duplicate a diskette, FCOPY asks for the name of the source diskette and reads as much information into memory as it can. FCOPY then asks for the name of the destination diskette and writes from memory all data from the source. This operation continues until the duplication is complete.

FGND

Command

Determine if a foreground program is running.

FGND

FGND allows you to check the status of the foreground. It returns one of two messages, depending on whether or not a program is currently running in the foreground.

Examples

FGND
```
No foreground program running
```
EXFG MYPROG
FGND
```
Foreground program running
```

FGND reveals that no program is running in the foreground. After the EXFG command executes a program in the foreground, FGND reveals that a program is running.

FILCOM

Command

Compare the contents of two files.

FILCOM filename1 filename2

FILCOM compares two files, word by word, and displays dissimilar word pairs, in octal, at the terminal. File organizations of the two files can differ; that is, you can compare a random file with a contiguous file.

Local Switches

outputfile/L Send output to *outputfile,* which can be a file or device name.

Examples

FILCOM YIN YANG $LPT/L

Compares files YIN and YANG word-by-word. Prints any dissimilar word pairs in octal on the line printer, along with their respective word displacements in the files, as follows:

```
025/            044516          042530
141/            000014          020044
142/            000000          046120
143/            000000          052057
144/            ---             046015
145/            ---             000014
```

YANG is two words longer that YIN (note the dashes for these locations in YIN). If either YIN or YANG is a null file, FILCOM prints dashes for the null file and the entire contents of the other file.

FPRINT

Command

Print a disk file in the specified format.

FPRINT filename

Displays a disk file at your terminal in either byte, decimal, hexadecimal, or octal format, with printable ASCII characters on the right side.

Any nonprinting characters are reported as periods. Without switches, locations are printed in octal. Use local switches to specify a first and last location. Locations are offset by 16 (octal) unless you include the /Z switch.

Global Switches

/B Print in byte format.

/D Print in decimal.

/H Print in hexadecimal.

/L Send output to the line printer.

/O Print in octal (default).

/Z Print locations starting at zero.

Local Switches

n/F Start at location n (octal).

filename/L Send output to file named (overrides global /L).

n/T Stop at location n (octal).

Examples

FPRINT/L TE1

Prints file TE1 on the line printer. The mode is octal by default.

FPRINT/B/L MYFILE 2000/F 3500/T

Prints MYFILE in byte format on the line printer, from location 2000 to 3500.

GDIR

Command

Display the name of the current directory.

GDIR

GDIR displays the name of the current directory. The value that the CLI displays for the GDIR command is the same as that of the %GDIR% CLI variable.

See the DIR command description for information on changing the current directory.

Examples

GDIR
```
MANHATTAN
```

MANHATTAN is the current directory.

GDIR
```
SKYSCRAPER
```
RELEASE SKYSCRAPER

In this example, we determine the name of the current directory with GDIR so that we can release it with the RELEASE command. An equivalent is:

RELEASE %GDIR%

where %GDIR% is the CLI variable that contains the same value the GDIR command displays.

GMEM

Command

Display background and foreground memory allocations.

GMEM

Displays the current memory allocation to background and foreground memory areas. The size is given in memory pages, where one page equals 1024 words (or 2048 bytes).

See the EXFG command for instructions on executing foreground programs.

Examples

GMEM

```
BG: 80     FG: 104
```

80 pages of memory are available to the background memory area (BG:); 104 pages of memory are available to the foreground (FG:).

GMEM

```
BG: 184  FG: 0
```

No memory is available to the foreground memory area; all available user memory is currently allocated to the background. This means that no foreground currently exists. Before you can run a program in the foreground, you must allocate memory to the foreground with the SMEM command.

GSYS

Command

Display the name of the current operating system.

GSYS

Displays on the terminal the filename of the operating system that is currently running.

Examples

GSYS
```
SYS
```

The current operating system file is named SYS. The system does not display its save file extension, .SV.

XFER/A $TTI OPSYS.MC
MESSAGE THE CURRENT OPERATING SYSTEM IS %GSYS%
^Z
OPSYS
```
THE CURRENT OPERATING SYSTEM IS SYS64K
```

This example creates a macro that contains the variable %GSYS%. When we use the macro, we see that the name of the currently running operating system is SYS64K.

GTOD

Command

Display the time and date.

GTOD

The GTOD command displays the current system date and time.

Examples

GTOD

```
06/17/86  15:24:20
```

The message indicates that the time is 3:24 p.m. and the date is June 17, 1986. Note that the date is displayed in numeric form even if you entered it as Jun 17 86 when you started DG/RDOS, and the time is displayed on a 24-hour basis even if you entered it with an AM or PM.

IMOVE

Utility

Dump and load files between disk, diskette, or magnetic tape.

IMOVE [argument/switch...] device [filename...]

IMOVE allows you to dump your whole disk or to dump only selected files or directories to unlabeled or labeled media and to create multi-diskette or multi-tape backup sets. Also, the utility writes in a dump format that other Data General operating systems can read and is able to read backup sets written under those same systems.

The following discussion of IMOVE gives basic instructions for its use in relatively simple backup and restore operations. For a complete discussion of all IMOVE features and capabilities, see *How to Generate and Run DG/RDOS.*

Argument/switches are options that provide greater control during IMOVE processing. Note that they must *precede* the device name.

Device is the name of the magnetic tape or diskette to which or from which you are moving files. The device name is the only argument that is required in the IMOVE command line.

If you use diskettes for backup, they must be hardware formatted, but they do not have to be software formatted. When you use diskettes, do *not* initialize the drive. If *device* is an initialized diskette drive, IMOVE aborts with the message `Directory is in use.`

If *device* is a tape drive, it may or may not be initialized. However, it is more convenient if it is not initialized, because IMOVE will initialize the drive, dump or load data, and release the device; the tape then rewinds automatically.

Filename specifies one or more files to be dumped or loaded. You can name multiple files, and you can also name files outside the current directory by placing a directory specifier in front of the filename. You cannot use filename templates to specify groups of files; however, you can dump the files named in an indirect file (created with the BUILD command). To invoke an indirect file as the IMOVE filename argument, use the command format:

IMOVE device @indirectfile@

If you do not specify filename arguments during a dump, IMOVE dumps all files from your current directory, including subordinate directories and their files. If you do not specify filename arguments during a load, IMOVE loads all dumped files, including subordinate directories and their files, into your current directory.

IMOVE dumps to unlabeled diskettes, unlabeled tape, labeled diskettes, or labeled tape. Dumps to both unlabeled and labeled diskettes can span several volumes. However, if your backup medium is unlabeled tape, your IMOVE dump file is restricted to the length of one tape. To overcome this restriction and create a multi-tape backup set, use IMOVE's global /A switch which writes ANSI-format labels on each tape.

If you use unlabeled magnetic tape, you must include a tape file number in the device name, as in MT0:0. You do not include such a specifier for labeled tape operations.

For multiple diskette and tape backup sets, IMOVE prompts you to insert the next diskette or tape needed. Be sure to write the number and date of the backup on each diskette or tape.

IMOVE writes its dumps in a format that AOS and AOS/VS can read. Use the AOS(VS) LOAD command to read an IMOVE dump file written to labeled diskettes, unlabeled tape, or labeled tape. Use the AOS MMOVE utility to load an IMOVE dump file recorded on unlabeled diskettes. You can also transfer files in the opposite direction by using the AOS(VS) DUMP command or the AOS MMOVE utility to dump files from your AOS(VS) system and IMOVE to load them onto your DG/RDOS system.

Global Switches

/A Specifies ANSI labeled media. For labeled tape operations, use the global /T switch in conjunction with /A.

/C During a dump operation, convert carriage return to NEW LINE in all of the files dumped; during a load operation, do the opposite. You need to make this conversion when you are transferring text files between DG/RDOS and AOS(VS).

NOTE: Do not use the global /C switch if any of the files you are dumping or loading are binary files.

/D Write data to diskettes or tape in dump format or read data that is in dump format. Note that with DG/RDOS Rev. 2.00 and higher, this switch is optional.

/F Load files in dump format from diskettes or tape. The dump file may have been created with IMOVE or with the AOS(VS) DUMP command or AOS MMOVE utility.

/H Display help text about the IMOVE command line and switches. /H prevents any file transfer regardless of other switches on the command line.

/L List on the printer the names of all files moved.

/N Do not perform any operation on the command line; list on the screen the files that would have been moved if the /N switch had not been present. Use /N to test the effect of your command.

/O Override the standard practice of not restoring a file from a backup diskette or tape if a file of the same name resides in the current directory. If you use the /O switch, the file you are trying to restore overwrites the existing file of the same name. Do not use /O and /R in the same command line.

/R During a load, restore a file that shares a name with a file in the current directory only if the file you are trying to load is of more recent date than the existing one. Do not use /R and /O in the same command line.

/T Dump files to or load files from tape. If you do not use /T, the utility assumes that you are using diskettes.

/V Display on the terminal the names of files as they are moved.

Argument/switches

buffersize/S By default IMOVE uses 8 KB buffers for dump and load operations. This argument/switch selects a larger buffersize in order to improve the performance of the tape unit on certain systems. Refer to *How to Generate and Run DG/RDOS* and the DG/RDOS software release notice to determine if you can use this argument on your system.

date/A/M Dump or load only files modified on or after the specified date. This date must be in the form dd-mmm-yy[:hh:mm:ss], as 12-JUN-86:12:00:00/A/M. Any parts of the time you do not enter default to zero.

date/A/R Dump or load only files last opened on or after the specified date.

date/B/M Dump or load only files modified before the specified date.

date/B/R Dump or load only files last opened before the specified date.

days/X Specifies the number of days (from the current date) for which data on a labeled diskette or tape should be protected from overwriting. Note that this protection exists *only* if you are using utilities in labeled mode on on labeled media. The default value for this protection is 90 days. Use this switch only in conjunction with the global /A switch.

There is no protection if you are using a utility in non-labeled mode.

name/F Specifies the name (1-17 characters) that is to be included in the HDR1 record of the diskette or tape you are dumping to. This switch also lets you specify the name that must be in the HDR1 label before a load operation takes place. Use this switch only in conjunction with the global /A switch.

filename/L Write to *filename* the names of files dumped or loaded. This argument overrides the global /L and /V switches.

Local Switches

/C During a dump operation, convert carriage return to NEW LINE in *filename;* during a load operation, do the opposite. Use this switch when you are dumping or loading both binary and text files; in that situation you cannot use the global /C switch.

Examples

From directory MONTHLYRPT.DR, the following command dumps all of the files in the directory to diskette:

DE0:IMOVE/V DJ0

```
Please insert disk 1, then press NEW LINE to continue the DUMP. <NL>
2-JUL-86 16:49:45
  COM.CM
  JAN3186
  FEB2886
  .
  .
  JUN3086
```

The /V switch displays on your screen the names of files as they are moved.

If you should accidentally delete all of the files in MONTHLYRPT.DR, you can restore them from your backup diskette by making MONTHLYRPT.DR your current directory, inserting the backup diskette in DJ0, and typing the following command.

DE0:IMOVE/F/V DJ0

```
Please insert disk 1, then press NEW LINE to continue the LOAD. <NL>
27-JUN-86 16:49:45
Attempt to create an existent file: COM.CM
  JAN3186
  FEB2886
  .
  .
  JUN3086
```

Note the message relating to COM.CM. If a dumped file has the same name as a file in your current directory, IMOVE does not restore the backup file. To restore a file with the same name, use the global /O switch, which directs IMOVE to overwrite the existing file with the backup file unconditionally. To overwrite the existing file only in the instance that the backup file is of more recent date, use the /R switch.

From directory MONTHLYRPT.DR, the following command backs up to file 0 of an unlabeled tape all files in the directory that were last modified on or after March 31, 1986:

DE0:IMOVE/T/V 31-MAR-86/A/M MT0:0

```
04-APR-86 16:02:09
  COM.CM
  APR3086
  MAY3186
  JUN3086
```

From directory WEEKLYRPT.DR, the following command backs up to labeled tape all of the files in the directory:

IMOVE/A/T/V 86WKLYJANJUN/F 200/X MT0

The /A switch indicates that you want to write ANSI-format labels on your backup tape, the /T switch indicates that you are backing up to tape, and the /V switch displays on your screen the names of files as they are moved. The /F switch is used to enter the filename *86WKLYJANJUNE* in your tape's HDR1 label; if you do not specify such a filename on your command line, IMOVE prompts you for one later. The /X switch is used to enter 200 days (from the current date) as the length of time to protect data from being overwritten. With a labeled tape you cannot specify a tape file number (such as MT0:0); you simply enter the device name MT0.

After you have entered the command line, IMOVE asks you to mount a tape and to press NEW LINE. Then, if the tape you are using does not already have a VOL1 label, the program prompts you for a volume name.

```
VOL1 does not exist.
Enter Volume ID for volume label VOL1 (1-6 char):
```

If your tape has a VOL1 label, IMOVE skips this request and begins to perform the requested operation. If the files do not fit on a single tape, IMOVE prompts you to insert the next tape in the set.

```
Please mount labeled tape 2, then press NEW LINE to continue the DUMP.
```

When IMOVE completes its operation, it rewinds your tape and returns you to the CLI.

From directory WEEKLYRPT.DR the following command line restores to the directory all of the files from the labeled tape, overwriting existing files with the same name:

IMOVE/A/T/F/V/O 86WKLYJANJUN/F MT0

86WKLYJANJUN specifies the name that appears in your tape's HDR1 label. If you do not include this argument, IMOVE simply ensures that all tapes in a multi-volume set have the same name in their HDR1 labels.

With a multi-tape set, you are prompted to insert the next tape in the sequence. When the restore operation is complete, IMOVE rewinds the tape and returns you to the CLI.

INIT

Command

Initialize a directory or a device for I/O.

INIT { directory / device }

INIT introduces a directory or device to the operating system for I/O. All files on an initialized tape unit, directory, or diskette are available until you release the device or directory with the RELEASE command. Before the introduction, all files on the device or directory are effectively closed.

To initialize a directory, the directory's parent directory must also be initialized. If you use a pathname as an argument to INIT (e.g., INIT DE1:SECPART:SUBDIR), all directories listed in the specifier are initialized.

Your operating system has a limit on the number of directories that can be initialized at any one time; this value is set at system generation with CONFIG. The default is 10 directories, with a maximum of 64. If you attempt to initialize a directory when at the limit, you receive the message `No more DCBs.` You can either use the RELEASE command to remove directories from system recognition to avoid running into the limit, or you can run the CONFIG program and increase the number of directories that can be initialized. See *How to Generate and Run DG/RDOS* for details.

Bootstrapping automatically initializes the directory that holds the current system (i.e., the master directory), and makes it the current directory.

The DIR command performs all INIT functions, and selects the current directory; this saves a step if you want to change the current directory. You cannot use DIR on a tape drive.

Global Switches

/F Fully initialize a tape, disk, or diskette by erasing all existing files and information. The /F switch is ignored when issued for a secondary partition or subdirectory.

After processing a disk with DKINIT, you initialize it fully with INIT/F; this is the last step in software formatting. INIT/F prepares a disk or diskette to accept DG/RDOS files by writing a new system file directory (SYS.DR) and map directory (MAP.DR). INIT/F destroys any existing data on the disk.

On a tape, INIT/F rewinds the tape and writes two end-of-file (EOF) marks at the beginning, effectively erasing the tape by allowing the system to overwrite existing information.

Because INIT/F erases data, the CLI issues a `Confirm?` inquiry before executing the command. Type *Y* to execute the command; the CLI displays `Yes` at the console and fully initializes the device. The CLI interprets any other character you type as NO; it displays `No`

at the console and cancels the command. Note that the CLI displays the inquiry and waits for a response when it encounters an INIT/F command in a macro.

Examples

INIT DJ1

Initializes the primary partition of the diskette in drive DJ1. Directories subordinate to primary partition DJ1 must now be initialized separately.

INIT SECPART:NEWDIR

Initializes both secondary partition SECPART and its subdirectory NEWDIR. All files in directories SECPART and NEWDIR are now available for use.

LIST ABC:FOLLY
```
No such directory:  ABC:FOLLY
```
INIT ABC
LIST ABC:FOLLY
```
ABC:FOLLY           88  D
```

In this example, we try to LIST a file in directory ABC. On receiving the message `No such directory,` we initialize ABC, and the command works when we try it again.

INIT/F MT0
```
Confirm? Yes
```

When we respond *Y* to the confirmation prompt, the CLI writes out `Yes` and begins operation; a <NL> is not needed. The command fully initializes the tape on tape drive MT0 by rewinding the tape and writing two end-of-file (EOF) marks at the beginning of the tape.

INIT/F DJ0
```
Confirm? No
```
R

In this example, when the system requests confirmation for a full initialization of diskette DJ0, we press any key except Y. The CLI writes out `No,` aborts the command, and returns the **R** prompt.

LABEL

Utility

Write and examine diskette and tape labels.

LABEL device/D [argument/switch]

Device is the name of the diskette or tape drive. The device/D argument and switch are required on the LABEL command line unless you use the /H global switch to request help.

The LABEL utility serves two primary purposes. First, it allows you to examine existing diskette or tape labels, and, second, it enables you to create new labels in preparation for an IMOVE (or AOS(VS) DUMP) backup session. If you plan to use BURST to back up your disk, do not prelabel your diskettes or tapes; BURST overwrites any existing labels.

If you plan to use IMOVE, it is not necessary to use LABEL to prepare your backup media. However, LABEL allows you to write up to nine user volume labels and give each tape in a multi-volume backup set a unique ID. This feature is important when you want to move files from a DG/RDOS to an AOS/VS system, because the latter system requires that each tape in a backup set have a different volume ID.

For a more extended discussion of the LABEL utility, see *How to Generate and Run DG/RDOS.*

Global Switches

/B Read diskette labels written by BURST. Do not use this switch with /T.

/F Examine existing labels.

/H Display a help message.

/I Examine or write labels in IBM format.

/L Send output to the printer.

/P Partially initialize a tape. This switch has the same effect as /S.

/S Same as /P.

/T Read from or write to tape. The absence of this switch indicates that you are using diskettes.

/V Send output to the console.

Argument/switches

volid/V	Supply a volume ID of from 1 to 6 characters. To write new labels or partially initialize a tape, you must use this switch.
owner/O	Specify an owner name in a diskette or tape's VOL1 label. If you are writing an ANSI-format label, this name can be up to 14 characters long; if you are using IBM format, it can be up to 10 characters long.
access/A	Specify a 1-character access code in a diskette or tape's VOL1 label. NOTE: Do not use an access code if you plan to move the files you are dumping to an AOS(VS) system.
filename/L	Send output to *filename*. This switch overrides the global /L switch.
["]uvltext["]/U	Create a user volume label. *Uvltext* is any text string with a length of 1 to 76 characters. Enclose the text string with double quotation marks it it has any spaces or punctuation. You can request a total of 9 user volume labels.

Examples

LABEL/V DJ0/D JUN86/V

Labels the diskette in drive DJ0 with an ANSI format, giving it volume ID *JUN86,* and sends the output of the LABEL operation to the console.

LABEL/F DJ0/D

Reads diskette labels created earlier by IMOVE.

LABEL/I/L/T MT0/D JUN86/V

Labels the tape in drive MT0 with an IBM-format and prints the output of the label operation.

LABEL/T/V MT0/D JUN86/V JOHN/O "BACK UP FILES"/U "OR ELSE"/U

Labels a tape with the volume ID *JUN86* and the name John, and supplies the text *BACK UP FILES* and *OR ELSE* for two user volume labels.

LDIR

Command

Display the name of the previous current directory.

LDIR

LDIR displays the name of the previously (or last) current directory. The name returned by LDIR is the value stored in the CLI variable %LDIR%.

See the DIR command description for information on changing the current directory.

Examples

GDIR
DRONER
DIR DJ0
GDIR
DJ0
LDIR
DRONER

The first GDIR command displays the name of the current directory, DRONER. DIR changes the current directory to directory DJ0. GDIR displays the name of the new current directory, DJ0. LDIR displays the name of the previous current directory, DRONER.

LDIR
DRONER
DIR %LDIR%

This example uses the CLI variable %LDIR% to change the current directory back to directory DRONER.

LFE

Utility

Create, edit, and analyze library files.

```
LFE { A inputmaster [...arg(n)] [listfile/L]
    { A/M inputmaster(1) [...inputmaster(n)] [lisfile/L]
    { D inputmaster [outputmaster/O] arg(1) [...arg(n)]
    { I inputmaster [outputmaster/O] file(1) [...file(n)]
    { M outputmaster/O inputmaster(1) [...inputmaster(n)]
    { N outputmaster/O file(1) [...file(n)]
    { R inputmaster [outputmaster/O] arg(1) file(1) [...arg(n) file(n)]
    { T inputmaster [listfile/L] [...inputmaster(n)]
    { X inputmaster arg(1) [...arg(n)]
```

LFE edits and analyzes library files, which are sets of relocatable binary files having special starting and ending blocks, and which are usually designated by the extension .LB.

A, D, I, M, N, R, T, and *X* are keys designating LFE functions; *inputmaster* and *outputmaster* represent library files; *arg* represents logical records on the library files, and *files* are update files.

Action taken by the LFE depends upon the function given in the command. The function keys are as follows:

A Analyze global declarations of *inputmaster* or a series of *inputmasters,* or of logical records specified from one *inputmaster.* Output is a listing with symbols, symbol type, and flags; no new output library file is created. Default output is to $LPT.

D Delete logical records, specified by *args* from *inputmaster,* producing *outputmaster.* Default output is to diskfile D.L1.

I Insert relocatable binary files, merging with logical records of inputmaster in the manner described under "Switches." Default output is to disk file I.L1.

M Merge library file (*inputmasters*) into a single library file named outputmaster. Default output is to disk file M.L1.

N Create new library file, *outputmaster,* from one or more relocatable binary files given by files. Default output is disk file N.L1.

R Replace logical records in *inputmaster* by relocatable binary files, producing *outputmaster.* Arguments are paired, with the first being the logical record and the

second the relocatable binary file that replaces the logical record. Default output is to disk file R.L1.

T Output to the listing device ($LPT by default) the titles of logical records on *inputmaster.*

X Extract from library file, *inputmaster,* one or more relocatable binary files given by *args.* Output is one or more relocatable binary files named args.RB.

Key Switches

/M Multiple input library files. The switch modifies the A function (not the filename LFE) and causes all library file names following, except the listing file, to be analyzed as one library.

Local Switches

/A Insert after. The switch modifies a logical record in an I function command line. Arguments following the switches are inserted after the logical record whose name precedes the switch. When neither a /A nor /B switch is given, inserts are made at the beginning of the new library file.

/B Insert before. The switch modifies a logical record in an I function command line. Arguments following the switch are inserted before the logical record whose name precedes the switch. When neither a /A nor /B switch is given, inserts are made at the beginning of the new library file.

/E Error listing directed to given filename.

/F Output analysis of each relocatable binary on a separate page (used only with /L).

/L Listing file. The switch modifies the name of a file to be used as listing output in the command line using the A or T functions. ($LPT is used by default.)

/O Output library file. The switch always modifies outputmaster in D, I, M, N, and R functions.

Extensions

If the .LB extension for inputmaster or the .RB extension for an update file are not given in the command, LFE searches for inputmaster.LB or arg.RB, respectively. If not found, LFE searches for inputmaster or arg, respectively.

Examples

LFE N ZED.LB/O ZED.RB ZEPHYR.RB

Creates a new library file called ZED.LB from binary files ZED.RB and ZEPHYR.RB.

LFE A/M ZUT1.LB ZUT0.RB

Analyzes as one library ZUT1.LB and the binary file ZUT0.RB. Analysis goes to the line printer.

LFE I MAL.LB MAL1.LB/O RSK/A MOD1

In library MAL.LB, inserts binary MOD1.RB after binary RSK. Names the new library MAL1.LB.

LFE A MYLIB(1,2,3).<LB,AN/L>

Analyzes libraries MYLIB1.LB, MYLIB2.LB, and MYLIB3.LB; stores analysis in disk files MYLIB1.AN, MYLIB2.AN, and MYLIB3.AN.

LINK

Command

Create a link to a file in another directory.

LINK { resolutionfile/2 | linkfile resolutionfile }

The LINK command creates the file *linkfile* that points to the *resolutionfile* file in the specified directory. *Linkfile* is created in the current directory unless you specify a different directory. Both *linkfile* and *resolutionfile* can be a pathname.

When you enter the name of a link file from the current directory, the CLI accesses its resolution file. In order for a link to work, all directories involved in the resolution chain must be initialized.

The *resolutionfile/2* argument creates a link file with the same name as the resolution file, when the resolution file is in the parent directory of the current directory. The parent directory could be a disk or diskette (primary partition) or a secondary partition. Since utility programs usually reside in the master directory (along with the operating system), you could link to each utility file using this format—if the master directory is the parent directory.

The use of links is the preferred way to execute programs and CLI utilities from directories other than those in which they reside. If a utility file (e.g. utility.SV) comes with an overlay file or error file (e.g. utility.OL or utility.ER), you need to create links to these files as well.

To create a link entry to another disk or diskette, use a pathname in *resolutionfile.* Be sure that the path contains only one directory specifier.

Normally you will assign a link file the name of its resolution file so that you do not have to keep track of different names for a file that performs the same function.

You may, of course, give your link file a name that differs from the name of the resolution file; in this case, the link file name is called an "alias" name. Note that when the resolution file to an alias name is an executable program file (.SV) or an overlay file (.OL), your link name must include the .SV or .OL extension.

If you create a link file to another link file, the CLI will link to its resolution file, and so on until it reaches the destination.

You can create a link to a file whose attributes prohibit linking (see the CHATR and CHLAT commands), but the link will not work.

CAUTION: To remove a link, use the UNLINK command. The DELETE command will delete the resolution file and not the link file. To prevent accidental deletion of a resolution file, assign attribute P to the file with the CHATR and CHLAT commands.

Examples

DIR BRANCH
LINK EDIT.SV/2

Creates the link file EDIT.SV to the editor EDIT.SV in DE0 (BRANCH's parent directory). Anyone in directory BRANCH can now edit a file without using a directory specifier. The link file appears in LIST command output as follows:

```
EDIT.SV          @:EDIT.SV
```

LIST displays the link filename and its resolution. The at sign (@) indicates that the link is to a file in the parent directory.

DIR DE0:LEAVES:GRASS
LINK TXT.SV DE0:EDIT.SV

Creates an alias link entry, TXT.SV, to EDIT.SV in DE0. From subdirectory GRASS, the command TXT will work exactly as the command EDIT works.

DIR DJ1
LINK EDIT.SV DE0:EDIT.SV

Creates a link file on diskette DJ1 to the resolution file EDIT.SV on disk DE0.

DIR DJ1
LINK EDIT.SV/2
EDIT MYFILE
```
Link depth exceeded: EDIT.SV
```

In this example, the LINK command does not report an error, but when we try to use the link we find a problem. In this case, the current directory is diskette DJ1. We use the form of the link command that searches for the resolution file in the parent directory. DJ1 does not have a parent directory, and the link was made to itself.

To correct the problem, remove the link and recreate it, including a directory specifier naming the directory in which the resolution file resides, as follows:

UNLINK EDIT.SV
LINK EDIT.SV DE0:EDIT.SV
EDIT MYFILE
*

The editor prompt (*) shows that the link works.

LIST

Command

Display information about the files in a directory.

LIST [[directory:]file ...]

Without arguments and switches, LIST displays all nonpermanent files in the current directory, in the form:

```
FILENAME        x  y
```

where `x` gives the size of the file in bytes, and `y` gives the file's characteristics and attributes.

When you include global switches with the command, LIST displays a variety of information about each file. The information depends on the switches used, and can include:

- file size, in bytes
- file characteristics (see list below)
- file access attributes and link access attributes (see list below)
- file creation date and time, in the form mm/dd/yy hh:mm
- the date the file was last opened for I/O, in the form mm/dd/yy
- the logical octal address of the first block in the file, in the form [nnnnnnnn]
- the use count of the file (in decimal) (the number of users who currently have the file open for I/O—see the CLEAR command for more information)

You can use local switches to specify files that were created before or after a specified date, or to exclude specified files from the display.

For a link file, LIST displays only the filename of the link and its resolution filename. An @ symbol indicates that the resolution file resides in the current directory's parent directory. The LINK command describes how to create and use link files.

File Characteristics

DG/RDOS assigns file characteristics to a file when it is created. The file characteristics are:

D Random file.

C Contiguous file.

L Link file.

T Partition. (Partitions also have characteristics C and Y.)

Y Directory. (Partitions, subdirectories.)

File and Link Access Attributes

The CHATR command changes a file's access attributes; the CHLAT command changes a file's link access attributes. The file and link access attributes are:

A Attribute-protected file. You cannot change the files's attributes, including attribute A, from the CLI. The A attribute can be set only with the .CHATR system call.

N Resolution-protected file. You cannot access the file through a link.

P Permanent file. You cannot delete or rename files with P set. Note, however, that the parent directory can be deleted unless *it* is permanent. Note also that the BUILD, DUMP, LIST, LOAD, and MOVE commands do not work on files with the P attribute unless you include the /A switch in the command line.

R Read-protected file. You cannot read the file (with the TYPE or PRINT commands), edit it with a text editor, or copy it with the DUMP, IMOVE, MOVE, or XFER commands when R is set.

S Save file. An .SV file needs this attribute to execute. The RLDR utility assigns S when creating a file.

W Write-protected file. You cannot modify the file. To prevent a file from being edited, use R. Note that W does not protect a file from deletion.

? User-definable attribute. You can use this to mark files for your own record-keeping.

& User-definable attribute. You can use this to mark files for your own record-keeping.

Global Switches

/A List permanent files. If you do not use the /A switch, LIST excludes permanent files (those with the P attribute) from the display.

/B Display filenames only.

/C Include file creation date and time in the display.

/E Display all categories of file information (overrides switches /B, /C, /F, /O, and /U).

/F Include the logical address of the first block in the file in the form nnnnnnn; displayed as 0 if unassigned.

/K Exclude link files from the list.

/L Send output to the line printer.

/N List only link files.

/O Display the date the file was last accessed.

/S Display the list of files sorted alphabetically by filename.

/U Display the file use count.

Local Switches

mm-dd-yy/A List files with creation dates on or after the specified date. Arguments *mm* (month) and *dd* (day) can be one or two digits.

mm-dd-yy/B List files with creation dates before the the specified date. Arguments *mm* (month) and *dd* (day) can be one or two digits.

filename/N Exclude files that match this filename.

Local switches have effect only when the other arguments in the command line are null arguments (when no files are specified) or template arguments.

Examples

LIST

```
FILE1.          68  D
PROGX.         479  D
FILE2.          50  D
```

Lists all nonpermanent files in the current directory, and displays the size of each in bytes, and any attributes and characteristics. In this example, all files have the characteristic D (indicating a random file), and no attributes.

LIST/A/S/E

```
FILE1.            68  D       01/06/86 16:37  02/16/86   [006564]  0
FILE2.            50  D       02/16/86 13:24  02/16/86   [006565]  0
MAP.DR          3262  APWC    12/23/85 09.38  07/15/86   [000017]  0
PROGX.           479  D       02/23/86  9:43  02/23/86   [006566]  0
PROJLIST         145  PD      01/04/86 10:56  02/24/85   [006567]  0
SYS.DR         27136  APWDY   12/23/85 09:38  07/16/86   [000006]  1
```

Lists all categories of information (/E) about the files in the current directory, including permanent ones (/A), in alphabetical order (/S), and displays the following, from left to right:

- the size of the file
- file characteristics and attributes
- file creation date and time
- the date the file was last opened for I/O
- the logical address of the first block in the file
- the file use count

LIST/K/S -.SV 1-25-86/A

```
INVEN.SV          1305  D
PROG4.SV           342  D
PROG5.SV           654  D
```

Lists all executable program files (-.SV), in alphabetical order (/S), that were created on or after January 25, 1986 (1-25-86/A). The output does not include link files (/K).

LIST/A SUBDIR:EDIT.SV

```
SUBDIR:EDIT.SV       @:EDIT.SV
```

Lists file EDIT.SV in directory SUBDIR. EDIT.SV is a link file whose resolution is a file of the same name residing in the parent directory of SUBDIR. The /A switch ensures that LIST will locate the file EDIT.SV, even if EDIT.SV is a permanent file.

LOAD

Command

Restore DUMP files to the current directory.

LOAD [source:]dumpfilename [filename ...]

Source: specifies the disk, diskette, directory or magnetic tape that contains the files you are loading into your current directory.

Dumpfilename specifies the filename you assigned to the dump file when you created it with the DUMP command.

Filename specifies the file or files to be transferred into your current directory from the specified source directory or device. This argument can include templates.

LOAD restores from a directory or device to the current directory the files and directories contained in a dump file (created with the DUMP command).

If you omit filenames and switches, all nonpermanent files from the dump file are loaded. With global switches, you can select filenames for loading. The /N (no load) global switch interprets the command line without loading any files, to allow you to check the files that will be loaded.

Files to be loaded must have different filenames from files in the current directory (unless you specify the /N, /O, or /R switches). Neither dumping nor loading changes a file's attributes, creation date, or directory characteristics.

Be certain that your current directory can accommodate the appropriate levels of subordinate directories to be loaded.

The LOAD command loads only files that that were previously dumped with DUMP.

Global Switches

/A Load all files, including permanent (attribute P) ones.

/B When used with the /V or /N switches, suppress the listing of file creation dates.

/K Exclude link files from the transfer.

/L Print the names of the files loaded. Overrides /V switch. When used with global /N switch, produces line printer listing without loading any files.

/N Do not load files; display on the terminal a list of filenames. File creation dates are displayed unless you also use the /B switch.

/O Delete files in the current directory that have the same filenames as the files being loaded, replacing the old file with the file loaded.

/R Load the most recent version of a file with the same name as a file in the current directory. If the existing file is older, it is replaced; if newer, it is retained.

/V Display on the terminal the name of each file loaded. File creation dates are also displayed unless you use the /B switch. Directory names are preceded by an *. Filenames in a directory are indented 2 spaces; note that they are listed *before* the directory name.

Local Switches

mm-dd-yy/A Load only files created on or after this date. Arguments *mm* (month) and *dd* (day) can be one or two digits.

mm-dd-yy/B Load only files created before this date. Arguments *mm* (month) and *dd* (day) can be one or two digits.

filename/N Exclude files that match *filename* from the load. *Filename* may contain templates.

Local arguments /A, /B, and /N do not apply to filenames that are specified explicitly (i.e., without the use of templates) on the command line.

Examples

```
INIT MT0
LOAD/V MT0:1
  RACES.01              01/31/86
  TIMES.TX              06/30/86
    SUZUKI.             03/22/86
    KAWASAKI.           04/03/86
    HARLEY.             02/01/86
* CYCLES.DR             01/02/86
RELEASE MT0
```

Loads into the current directory all nonpermanent files previously dumped to file 1 of the tape on tape drive MT0, and displays the filenames and their creation dates as they are loaded (/V). Indentations and asterisks reflect the directory structure of the files loaded.

LOAD/L/A MT0:3 -.SV 12-15-85/A TEST-.-/N

Loads from file 3 of the tape on MT0 all files with a .SV extension except those files whose names begin with the characters TEST and any files created before December 15, 1985; /L lists the filenames on the line printer.

DIR DE1
LOAD/V/R DE0:JUN86.BU -.SR

Loads (into current directory DE1) all .SR files from dumpfile JUN86.BU, on disk DE0. (See DUMP command example.) The /R switch specifies that if any file to be loaded has the same name as a file in DE1, the file should be loaded only if it is newer.

LOG

Command

Record the current CLI session in the log file (F)LOG.CM.

LOG [password] [directory]

LOG records CLI dialog that appears at your terminal in a file named LOG.CM (or FLOG.CM for a foreground CLI). The CLI creates the log file, or appends information to the existing log file, then records each line of CLI dialog in it. Only one current log file may exist at a time in any ground. You cannot examine, print, or delete the log file while it is open. Use the ENDLOG command to close the log file.

The *password* is an optional argument of up to 10 alphanumeric characters that you can use to prevent the log file from being closed inadvertently. If you specify a password, you must use the same password in the ENDLOG command to close LOG.CM.

Directory indicates a destination for the log file other than the current directory. You must initialize the directory before using its name in the LOG command.

Global Switches

/H Place a header at the beginning of LOG.CM. This header contains the title LOG FILE, and directory and date information.

/T Trace the execution of CLI commands. This tells the CLI to write each command in its final form to the log file before executing it. All trace lines will be preceded by the symbols ==> when LOG.CM is printed.

Local Switches

/O Write LOG.CM in the specified directory, overriding the default of using the current directory.

Examples

LOG/H GSTONE

Records all CLI dialog to file LOG.CM in the current directory. The password is GSTONE, and on output, the header of this file will look like:

```
***LOGFILE***  GATE  [DE0:SYS]  10/17/84  3:0:0
```

GATE is the name of the current directory.

Assume that you have built a macro file named LAST.MC, which contains the following commands:

DIR %LDIR%
MESSAGE Now in directory %GDIR%

Assume also that the value of %LDIR% is MYDIR. If you open the log file with global /T (trace) switch and type

LAST

the system writes the following lines into the log file:

```
LAST
==>LAST
==>DIR MYDIR
==>MESSAGE Now in directory MYDIR
Now in directory MYDIR
```

If you open the log file without /T, these lines appear in the log file without the trace, as follows:

```
LAST
Now in directory MYDIR
```


MAC

Utility

Assemble source file(s) with the Macroassembler to produce a relocatable binary (.RB) file.

MAC filename [...filename]

The Macroassembler, MAC, creates a relocatable binary (.RB) file from assembly language source files. Once you produce the .RB file and handle any errors MAC detects, you can use the .RB file to create an executable .SV file with the RLDR utility.

MAC will use macro definitions that you specify in your source file with macro calls and incorporate them as part of the .RB file. See the *RDOS/DOS Assembly Language and Program Utilities* manual for instructions.

If you omit switches, MAC produces a binary file named filename.RB and no listing file. You can include switches to produce an assembly listing, an error listing, a symbol table file, and a symbol cross-reference listing.

To use MAC, you need to have access to the following files:

MAC.SV

MACXR.SV

MAC.PS

For each source file you specify on the command line, the CLI searches for *filename* with the .SR extension first, and then searches for *filename*.

Global Switches

/A Cross-reference all user and semipermanent symbols (symbols defined by pseudo-ops).

/E Suppress error messages at the terminal. If you do not specify the creation of a listing file (global or local /L), error messages cannot be suppressed with /E.

/F Generate or suppress a form feed on printed listings, as necessary to produce an even number of listing pages. By default, MAC generates one form feed after listing each filename.

/K Keep the MAC.ST symbol table after the assembly. By default, MAC.ST is deleted at assembly completion.

/L Append an assembly listing to disk file *filename.LS*; /L creates the file if none exists, using the filename of the first file on the command line.

/M Flag multiple-defined symbols on pass one.

/N Suppress creation of the .RB file (useful for checking assembly errors).

/O Override all listing suppression controls (as specified with .NOLOC, .NOMAC, etc.).

/S Skip pass two and copies MAC.ST (symbol table and macro definitions) to file MAC.PS. Deletes the old MAC.PS (if any) first.

/T Recognize and store eight-character symbols from the source file(s). The resulting binary file is in extended .RB format. The cross-reference will show five-character symbols. By default, MAC recognizes and stores only the first five characters of symbol names. For restrictions on using this switch, see *RDOS/DOS Assembly Language and Program Utilities.*

/U Include user symbols in the relocatable binary output.

Local Switches

name/B Assign *name* to the .RB file (overrides global /N).

name/E Write error messages to *name.*

name/L Write the listing and the cross-reference to *name* (overrides global/L).

name/S Skip *name* on pass two of assembly. Use this switch only if name contains no storage words. Macro definition files can be skipped on pass two.

name/T Use *name* as a permanent symbol file for this assembly. If you omit this switch, the assembler uses file MAC.PS.

Error Codes

If a line of source code contains an error, the assembler places a letter at the left margin of the offending line in the listing. It can insert no more than three codes per line.

A Address error
B Bad character
C Macro error
D Radix error
E Equivalence error
F Format error
G Global symbol error
I Parity error on input
K Conditional or repetitive assembly error

L Location counter error
M Multiply-defined symbol error
N Number error
O Field overflow or stack error
P Phase error
Q Questionable line error
R Relocation error
U Undefined symbol error
V Variable label error
W Text error

Examples

MAC/L ZORRO

Assembles source file ZORRO.SR or ZORRO, producing relocatable binary file ZORRO.RB. The /L global switch sends the listing to file ZORRO.LS.

MAC LIB/S A B C $LPT/L

Assembles files A.SR or A, B.SR or B, and C.SR or C, producing binary file A.RB. The local /S switch scans file LIB (or LIB.SR) on pass one for macro definitions and values for externals in A, B, and C. The listing is sent to the line printer ($LPT/L).

MAC/L/U EX<1 2 3 4>

Assembles EX1, EX2, EX3, and EX4 (with or without .SR extensions), producing EX1.RB. The global /U switch includes user symbols in file EX1.RB; the /L switch sends the listing to file EX1.LS. (See chapter 2 for an explanation of the use of angle brackets.)

MDIR

Command

Display the name of the master directory.

MDIR

MDIR displays the name of the master directory, the directory in which DG/RDOS system files are kept. The value that DG/RDOS returns for MDIR is the same as that returned for the %MDIR% CLI variable.

Examples

MDIR

```
DE0
```

Primary partition DE0 is the master directory.

DIR DE0

This command, which makes master directory DE0 the current directory, is equivalent to the command

DIR %MDIR%

MESSAGE

Command

Display a text string.

MESSAGE ["]textstring["] ...

MESSAGE displays *textstring* as a message on the console. This command is useful for indirect and macro command files, which are explained in chapter 4.

The /P global switch allows you to delay further execution until someone strikes a key on the keyboard. This is useful if your macro or indirect file directs a user to mount a device, such as a disk or tape, and you need to halt processing until the device is ready.

The quotation marks are optional. If you use quotation marks, the MESSAGE command returns all characters literally, except for the quotation marks (which are text string delimiters), semicolons, carriage returns, and form feeds (which are command delimiters). A single quotation mark within a quoted string is illegal. The text string cannot have more than 72 characters.

If you omit quotation marks, the CLI interprets special characters such as % for variables and @ for indirect files. Angle brackets (<>), parentheses, commas, and slashes (/) are interpreted as they are in a CLI command line. (To have these characters interpreted literally, place them inside quotation marks.) Unquoted text strings are delimited as they are in a CLI command—with <NL> or ; (semicolon). An unquoted text string can have up to 132 characters.

Global Switches

/P Directs a pause; after displaying *textstring*, display the message `Strike any key to continue`, and wait for someone to strike a key.

Examples

The macro file LOGON.MC contains the following commands:

```
DIR USERSDISK
CHATR  SYS.<SV,OL>  +WP
CHATR  CLI.<SV,OL,ER> +WP
MESSAGE  Welcome aboard.
MESSAGE  User directories available are:
LIST  -.DR
```

We then execute the file:

LOGON

```
Welcome aboard.
User directories available are:
SECONDPART.DR      507904 CTY
SUBDIR.DR          512    DY
SUBDIRA.DR         512    DY
.
.
```

The macro file DUMPTAPE, below, uses the /P switch to delay execution until someone takes a particular action.

TYPE DUMPTAPE.MC

```
CHATR  SYS.<SV,OL> -WP  CLI.<SV,OL> -WP
MESSAGE  The current directory is %GDIR%
MESSAGE  "Mount dump tape on MT0, and ready the tape drive"
MESSAGE/P  Enjoy!
INIT MT0
DUMP/V MT0:0
```

MOVE

Command

Copy files from the current directory to the specified directory.

MOVE destination [filename ...]

Destination specifies the name of the directory to which the files are moved.

Filename specifies the files in the current directory that are to be moved. If you do not specify any filename argument, all nonpermanent files in the current directory are moved. Filename arguments permit the use of templates.

MOVE transfers copies of one or more files from the current directory to the specified directory. The directory does not have to be initialized if:

- It is a subdirectory of the current directory.
- You supply a pathname to it.

For each file it transfers, MOVE preserves file statistics such as filename, length, attributes, time of creation, and time of last access.

Link files retain their original resolutions, so you may want to use the /K switch, and create new link files from scratch so their resolutions are valid for the new directory.

Note that you cannot move a directory with MOVE.

Global Switches

/A Move all files, including permanent files.

/D Delete the original files from the current directory after copying them.

/K Exclude link files from the transfer.

/L List moved filenames on the line printer (overrides /V switch).

/R Retain most recent version of the file. When a file in the current directory has the same name as a file in the destination directory, the system checks both files' creation dates. If the file in the current directory is newer, it replaces the file in the destination directory. If the file in the current directory is older, no transfer takes place.

/V List on the terminal the names of each file moved.

Local Switches

mm-dd-yy/A Move only files created this date or after. Arguments *mm* (month) and *dd* (day) can be one or two digits.

mm-dd-yy/B Move files created before this date. Arguments *mm* (month) and *dd* (day) can be one or two digits.

name/N Exclude any files that match name from the transfer.

oldname/S newname
Assign newname to the file specified as the oldname argument, but retains its old name in the current directory.

/A, /B, and /N local switches apply to any filename arguments specified as templates, or to all files in the current directory when no filename arguments are specified. /A, /B, and /N ignore any files that are specified explicitly (that is, whose full filenames are given).

Examples

MOVE/V DJ1 TEXT-.-
```
TEXTA.OB
TEXTA.SR
TEXTA.SV
TEXTOBJ
TEXTVS
```

Copies to directory DJ1 all nonpermanent files in the current directory that begin with the letters TEXT. /V causes MOVE to list at the terminal the names of the files copied.

MOVE/D/K/V SOURCE -.SR TEST-.-/N
```
CHART1.SR
CHART4.SR
GOCHART.SR
SIMPLOT.SR
XCHART.SR
```

Copies to directory SOURCE all nonpermanent files with an .SR extension in the current directory (except link entries (/K) and files beginning with TEST (/N)). /V causes MOVE to list at the terminal the names of the files moved. /D causes MOVE to delete the original files from the current directory after the transfer takes place.

DIR ACCTSDUE
MOVE/A/V/R DE0:LEGALNOTES -.AD 2-24-86/B
```
ADR1ANCO.AD
JOHNSMART.AD
MAXSMART.AD
```

Copies to directory LEGALNOTES on unit DE0 all files having the extension .AD that were created before February 24, 1986. The recent (/R) switch prevents the replacement of any files in directory LEGALNOTES that are more recent than files of the same name in the current directory.

CDIR JONAS
MOVE/V JONAS NEWPROG.-
```
NEWPROG.LS
NEWPROG.OB
NEWPROG.SR
NEWPROG.SV
```

From directory DE0, creates subdirectory JONAS and moves files beginning with NEWPROG from the current directory to JONAS. Since JONAS is a subdirectory of the current directory (DE0), JONAS does not have to be initialized for the MOVE to work.

OVLDR

Utility

Create an overlay replacement file.

OVLDR programname old_overlay/N new_overlay(s) [old_overlay/N new_overlay(s)]

Creates and loads an overlay replacement (.OR) file for the overlay (.OL) file of a program previously created with RLDR. OVLDR can replace up to 127 overlays. The new overlays do not replace the old ones until you use the CLI command REPLACE to do so.

The argument *old_overlay/N* identifies the overlay that *new_overlay* is to replace. *Old_overlay* can be either the octal representation of the overlay segment and overlay number within the segment, or it can be the symbolic overlay name if the .ENTO pseudo-op defined the name in the root program.

See the *RDOS System Reference* for more on overlays and *RDOS/DOS Assembly Language and Program Utilities* for a description of OVLDR.

OVLDR requires a symbol table in the .SV program. You can do this when you create the .SV file with RLDR by specifying RLDR's global /D switch, or by including an .EXTN .SYM statement in a program module.

Global Switches

/A Produce an additional symbol table listing with symbols ordered alphabetically. (Use with the local /L switch.)

/E Display error messages at the terminal when a listing file has been specified (local /L). By default, when you specify a listing file, error messages to the console are suppressed.

/H Display all numeric output in hexadecimal (radix 16). By default, output is in octal.

Local Switches

name/E Send error messages to file name.

name/L Write the symbol table to *name*. The table lists symbols in numeric order.

Examples

OVLDR/A/E ROOT OLD1/N NEW1 NEW2 ROOT.OM/L

Creates overlay replacement file ROOT.OR. When ROOT.OR replaces the original overlay file ROOT.OL, overlays NEW1 and NEW2 replace old overlay OLD1 in ROOT's overlay file. The .ENTO pseudo-op was used in each overlay; thus we could reference overlays by name instead of by overlay number and node number. OVLDR's error messages and memory map of new symbols are written to disk file ROOT.OM. Error messages appear at the terminal.

The RLDR line that previously loaded ROOT might have looked something like this:

RLDR/D ROOT [OLD1,OLD2] ROOT1 [OLD3, OLD4]

PATCH

Utility

Install a patchfile in a program (.SV) or overlay (.OL) file.

PATCH [programname/S] [patchfile/P] [loadmap/L]

PATCH installs the patches you inserted into a patchfile with the ENPAT utility. To apply Data General-supplied patches to an operating system, you must have instructed SYSGEN to save the load map file. If no load map exists for a program, you can patch it by appending the global /N switch to PATCH. If you omit arguments, PATCH asks for the information.

When you use the global /N switch and the patchfile contains symbolic references, PATCH displays the symbol name in quotation marks and asks for the address of the symbol. You must respond with an octal number of less than six digits. Negative numbers have the same form preceded with a minus sign.

As it modifies the save or the overlay file with *PATCH.PF,*, the utility builds a file named *programname.PD*. If a file with this name exists, the dialog is appended to the file. The .PD file contains a line with the date and time of the patching, a list of all locations patched, and a string indicating whether the patch was installed. Patches applied during the last run of PATCH are marked with an asterisk. The .PD file is a text file that you can type or print.

Global Switches

/I Do not display comments from the patchfile on the terminal.

/N There is no load map available.

Local Switches

name.ex/L Load map name. You must include the extension to this name.

name/P Patchfile name, including extension; patchfile is created by ENPAT.

name/S Object program name; you can omit the .SV extension.

Examples

PATCH SYS/S SYS.LM/L IPB.PF/P

```
3 applicable patch(es)
3 patch(es) needed to be installed
```

Install the patches entered in the ENPAT example. Note that PATCH describes the number of applicable patches, and the number it installs.

PATCH/N MYPROG/S MYPROG.PF/P

```
2 applicable patch(es)
2 patch(es) needed to be installed
```

Here, the patches inserted in patchfile MYPROG.PF are applied to user program MYPROG. There is no load map, hence the global /N switch. (Eventually, for permanence, MYPROG'S author should make corrections to the source version, then reassemble and reload it.)

POP

Command

Return to the next higher level program.

POP

POP forces a return to the next higher level program after a user program has swapped in the CLI on level one or lower. Programs can use the system call .EXEC to swap in the CLI.

The CLI is initially on level zero. When you execute another program via the CLI, the CLI is swapped out of memory and the new program is brought in and executed on level one. The new program terminates itself (and swaps the CLI back in) with .RTN. In effect, POP issues a .RTN from the CLI.

You cannot POP from level zero.

Examples

```
SPEED MYFILE
.
.
!XCLI.SV$$

R
. .
POP
!
```

During a SPEED editing session, starts a new CLI process in order to perform some operation. After the operation is complete, the POP command terminates the second CLI and returns to SPEED.

PRINT

Command

Print a file on the line printer.

PRINT filename [..filename]

PRINT outputs the contents of ASCII (text) files at the line printer (device name $LPT). The files may reside on any device. PRINT is the equivalent of a series of *XFER/A filename $LPT* commands.

To print more than one copy of a file, use the local numeric switch to indicate the number of copies you want (up to 9). For example, the command *PRINT RIDDLER/5* prints 5 copies of the file RIDDLER. To print more than 9 copies, use multiple PRINT commands.

To print a binary file, see the FPRINT command description.

Local Switches

filename/n n specifies the number of copies of filename to print.

Examples

PRINT FOO.SR DJ1:COM.SR BLIXEN.SR/2

Prints one copy each of the files FOO.SR and DJ1:COM.SR, and prints two copies of BLIXEN.SR.

PRINT ARREARS:<JAN,FEB,MAR>.BL

Prints files JAN.BL, FEB.BL, and MAR.BL in directory ARREARS.

PRINT DA1:MYFILE/3

Prints three copies of DA1:MYFILE.

PRINT MT0:2

Prints the contents of file 2 of the tape on drive MT0.

RELEASE

Command

Release a directory or device from system initialization.

RELEASE { directory / device }

RELEASE closes a directory or device. For a directory, RELEASE updates and closes all files, including all subordinate directories. For a tape drive, RELEASE rewinds the tape.

After you release a directory, DG/RDOS no longer recognizes it. You can reopen it for access with the INIT or DIR commands.

The INIT command reopens a tape drive for access.

CAUTION: Always release a diskette before physically removing it from its drive; otherwise its files may not be updated properly and the diskette will be left in an inconsistent state.

The CLI cannot release the master directory while a foreground program is running. You must first type CTRL-C CTRL-F on the background console to end the foreground program.

If your current directory is not the master directory and you release the current directory, the master directory becomes your new current directory.

A normal system shutdown (with the BYE.MC macro or equivalent) releases all directories and devices.

Refer to the INIT and DIR command descriptions for information on initializing directories and devices.

Examples

RELEASE DJ0

Releases drive DJ0 so that the diskette can be removed from the drive.

RELEASE MT0

Rewinds the tape on drive MT0, and releases tape drive MT0.

RENAME

Command

Change the name of a file or directory.

RENAME oldfile newfile [...oldfile newfile]

Oldfile specifies the current filename of the file or directory. You must include the .DR extension if *oldfile* is a directory.

Newfile assigns a new filename to the file or directory. You must include the .DR extension if *newfile* is a directory.

RENAME changes a file's or directory's name but does not affect the file's contents, characteristics, or attributes.

You cannot rename any file that is protected with the P (permanent) attribute. You must first change the P attribute (CHATR command) before renaming such files.

You can rename directories without affecting the contents of the directory. However, if you have macros with pathnames that use the old name, or link files in other directories that recognize the old name, they will no longer work.

Examples

DELETE Q.SV
RENAME QTEST.SV Q.SV

Deletes the old version of program Q.SV, and replaces it with the most recent version of the program by changing the name of QTEST.SV to Q.SV. If we had not deleted Q.SV prior to the RENAME command, RENAME would have returned the error message `File already exists.`

RENAME DE0:A1.DR DE0:A.DR B1 B

Renames directory A1 to A on DE0 and file B1 to B in the current directory.

XFER/A $TTI FN
LONGFILE<CTRL-Z>
RENAME T @FN@.01 T1 @FN@.02 T2 @FN@.03

Opens file FN to accept input from the terminal. The text string LONGFILE is placed in FN and the file is closed. The RENAME command then assigns the string LONGFILE to existing files. The @ signs specify the contents of file FN, and the old names T, T1, and T2 become LONGFILE.01, LONGFILE.02, and LONGFILE.03.

REPLACE

Command

Replaces overlays in an overlay file.

REPLACE programname

REPLACE replaces overlays in overlay file *programname.OL* with the replacement overlays created in file *programname.OR* with the OVLDR utility.

REPLACE is the active sequel to OVLDR. Actual replacement occurs as soon as there are no outstanding overlay load requests.

Examples

REPLACE ROOT

Replaces the specified overlays in file ROOT.OL with new overlays in file ROOT.OR. The OVLDR utility created the overlay replacement file ROOT.OR.

REV

Command

Display the revision level of an .SV file.

REV program[.SV]

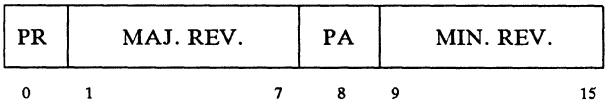
REV displays the revision level of a program (.SV file). The program file must have the S attribute. The system returns the major revision number followed by a period and a minor revision number.

Both major and minor revision levels can be in the range 0 through 99. Revision levels greater than 99 are displayed as 99.

Use the .REV assembler pseudo-op to assign a major and minor revision level number to a source file. If you omit this pseudo-op from all source programs, then the revision level of your .SV file is displayed as 00.00.

If bit 0 of the revision level word is set to 1, the word `Pre-release` is displayed after the revision level number to indicate that the save (.SV) file is a pre-release version. If bit 8 is set to 1, the word `Patched` is appended to the revision level number display to indicate that the .SV file is a patched version.

The format of the revision level word is:



Examples

REV CLI

```
CLI.SV  02.00
```

The major revision level of this CLI is 2, and the minor revision level is 0.

RLDR

Utility

Loads relocatable binary (.RB) files to produce an executable program (.SV) file.

RLDR binary_file... [overlay_binary...] library_file...

NOTE: Square brackets ([ ]) in the RLDR command format are part of the command line; they are not notation conventions to set off options, as in other command formats.

RLDR invokes the Relocatable Loader utility. RLDR takes assembled or compiled relocatable binary (.RB) files, assigns nonrelocatable addresses, and produces a program (.SV file) ready for execution. You can then execute programname.SV by entering programname to the CLI as a CLI command.

By default, RLDR uses the filename of the first binary file you specify in the command line to name the files it produces, including the program (.SV) file, overlay (.OL) file, and symbol table (.ST) file. The CLI searches for each binary file first with an .RB extension, then with no extension. You must include an extension, if any, of the library files you specify.

To create an overlay file for your program, specify the overlay binary files in square brackets on the command line. RLDR builds root binary files into the .SV file, and overlay binary files into the .OL file. During execution, the root binary files reside in memory all of the time, while the overlay binary files are called from the .OL file into their assigned memory nodes, according to the instructions in your program.

Each pair of square brackets ([ ]) in the RLDR command line defines a node in the .SV file and a segment in the .OL file. RLDR uses the node of the .SV file to reserve an area of memory to receive an overlay from the node's corresponding .OL file segment.

The individual overlay files reside independently in the .OL file, contiguous to one another in the order specified in the RLDR command line. Overlay files assigned to the same node (placed within the same pair of brackets) overwrite the node as the executing program loads them one by one (via the .OVLOD system call). For example, the command line

RLDR ABLE [A,B]

loads binary file ABLE as a root program, ABLE.SV. ABLE has one node for binary files A and B. The node in the disk file maintains itself in memory when ABLE executes; ABLE can load either A or B into this node without affecting the rest of ABLE.

If you want to debug the program at a later time, use the global /D switch to include a debugger and symbol table in your .SV file. To include local user symbols, add the local /U switch. (You must have previously specified global /U to the assembler or compiler.)

RLDR can produce a load map (memory map), which shows where program modules will reside when the program executes. This load map can help you patch the .SV file on disk. You can print it with the global /L switch, or save it in a file with the local /L switch.

If a program runs more than one task, you must specify multiple tasks, and the number of I/O channels for the program, to RLDR. RLDR will then include the multitask scheduler in the program. To specify tasks and channels, use either a .COMM TASK statement in your program, or RLDR local switches /K and /C. If you're working in a high-level language (as opposed to assembly language), your compiler manual will tell you how to specify multiple channels and tasks.

To use RLDR, you need access to the following files:

RLDR.SV	The RLDR program file.
RLDR.OL	The RLDR program's overlay file.
SYS.LB	The system library.

For more information about RLDR, see *RDOS/DOS Assembly Language and Program Utilities* and *RDOS System Reference.*

Global Switches

/A Produce an additional load map, with the symbols ordered alphabetically. (Use with the local /L switch.)

/D Include the symbolic debugger and program symbol table with the program. RLDR does not write the program symbol table to the .SV file unless you include the /D switch. Use global /S with /D to store the symbol table in high memory (instead of directly above the program). To load the interrupt-disable debugger, instead of the default symbolic debugger, include its name (AIDEB.RB) as an argument on the RLDR command line.

/E Display error messages at the terminal when a listing file has been specified (local /L). By default, when you specify a listing file, error messages go to it, not to the terminal.

/G When overlays refer to named common area, print a warning message at each occurrence. By default, RLDR prints a warning message at the first occurrence only.

/H Print numeric output in hexadecimal (radix 16). By default, numeric output is in octal radix.

/I Do not create UST, TCB, or other system tables; start NREL code at octal location 445 and ZREL code at octal location 50. The .SV file cannot execute under any Data General operating system.

/K Store RLDR's internal symbol table for this program in filename.ST. RLDR does not save this file unless you include /K.

/M Override the display of the load map and error messages at your terminal.

/N Override RLDR's search of the system library (SYS.LB). By default, RLDR searches the system library at the end of the command line to try to resolve undefined symbols.

/O Override the inclusion of the program symbol table as specified by the /D switch, but retains the debugger. (Use with global /D.)

/P Print the starting NREL value of each .RB file as it loads.

/S Leave the symbol table in high memory (you must include the global /D switch to produce the symbol table).

/Z Load the .SV file to start execution at location 0. Use /Z only to load stand-alone programs that use page zero locations 0-15 (octal). Note that the resulting .SV file cannot execute under DG/RDOS.

Local Switches

n/C Allot *n* I/O channels to the program. This octal value overrides any value specified in a .COMM TASK statement. If you omit both /C and .COMM TASK, RLDR allots eight channels to the program.

name/E Send error messages to file name.

n/F Start program's NREL address space at octal address n. (See the local /Z switch.)

n/K Allot n (octal) tasks to the program. This overrides any value in a .COMM TASK statement. By default, RLDR allots one task.

name/L Send the load map to name, where name may be a file or a device such as the line printer ($LPT). This map lists symbols in numeric order. Without this switch, the load map will not be saved.

n/N Move the NREL pointer, which indicates the location for loading the next file, to octal address n. Address n must exceed the current NREL pointer value. (The pointer value is originally 400(octal) plus space for a User Status Table, TCBs, etc.; RLDR moves it upward as it loads each binary file.)

name/S Assign name to the .SV file and the .OL file, overriding the RLDR default of using the name of the first binary file specified on the command line.

name/U Take binary file name and includes its user symbols in the symbol table. Local symbols are those used exclusively within this binary file. This does not work unless you previously specified a global /U switch in the MAC command line.

[binaries...]/V Load binary files (delimited by brackets) as virtual overlay files. You must specify all virtual overlay files before any other overlay files in the RLDR command line (e.g., RLDR ROOT [A,B]/V ROOT1 [C,D]).

n/Z Start program's ZREL address space at octal address n. By default, ZREL code starts at location 50(8). (See the local /F switch.)

Examples

RLDR A B C DA1:D

Loads files A, B, and C from the current directory, and file D from directory DA1; produces an executable file called A.SV in the current directory. All messages go to the console. Neither the symbol table nor the load map is saved.

RLDR/D A B C

Loads files A, B, and C with the symbolic debugger and a symbol table, to produce file A.SV.

RLDR/E MYFILE MYFILE.LM/L

Loads MYFILE to produce MYFILE.SV; creates listing file MYFILE.LM and sends the load map and all messages to it. Error messages will appear at your terminal, as well (/E).

RLDR MYPROG PROG1 MYLIB.LB $LPT/L 10/K 20/C

Loads MYPROG, PROG1, and library file MYLIB.LB to produce MYPROG.SV. All RLDR messages (plus the load map) print at the line printer ($LPT/L). MYPROG.SV is a multitask program; switch 10/K enables it to run up to 10 (octal) tasks, and switch 20/C enables it to use up to 20 (octal) channels.

RLDR ROOT1 [A,B,C,D] ROOT2 ROOT3 [E,F,G,H] ROOT1.LM/L

Loads binary files ROOT1, ROOT2, and ROOT3 into ROOT1.SV, and creates overlay file ROOT1.OL with two segments. ROOT1.SV can load overlays A, B, C, or D into the first node (node 0); it can load overlays E, F and G, or H into the second node (node 1). Switch ROOT1.LM/L sends the load map and console messages to file ROOT1.LM.

SDAY

Command

Set the system calendar date.

SDAY mm dd year

SDAY sets the system calendar to the date you specify. You can enter the year as either two or four digits (e.g., 86 or 1986). Use spaces or commas to separate each date argument.

Examples

SDAY 6 17 1986

Sets the system calendar date to June 17, 1986.

SEDIT

Utility

Analyze and edit a disk file.

SEDIT filename

Invokes the symbolic editor to examine, analyze, or modify the contents of a disk file. *filename* can be any nonsequential disk file.

If you omit an extension, SEDIT searches for *filename.SV;* if SEDIT does not find that file, it searches for *filename*. To specify an overlay file, include the .OL extension.

For more information on SEDIT, see the *RDOS/DOS Debugging Utilities* manual.

Global Switches

/N Do not search for the symbol table. Use this switch if there is no symbol table, or to edit a text or nonprogram file.

/Z Start file at location 0.

Examples

```
SEDIT MYPROG.SV
SEDIT REV x.x
.MYPROG%
.START + 10/ 106413
.$;
.START + 10/ SUB# 1 SNC
.$Z
DONE
R
```

SEDIT finds and opens MYPROG.SV, proceeds with editing, examines a location, and returns to the CLI as directed by ESC-Z (the ESC key echoes as a $).

You can find the current NMAX and ZMAX requirements for any program by typing the following SEDIT commands:

```
404/ nnnnnn
401/ zzzzzz
```

SEDIT returns values `nnnnnn` and `zzzzzz`, the User Status Table values for NMAX and ZMAX.

SMEM

Command

Allocate background and foreground memory.

SMEM pagesize

When DG/RDOS starts, it runs in what is called the *background*. However, you may also establish a second memory area known as the *foreground*. Because each ground can run a separate set of programs, more efficient use of system resources can result from running programs in two grounds.

To establish the foreground, you first use the SMEM command to specify a portion of total available user memory to the background; DG/RDOS allocates the remainder to the foreground.

You issue SMEM from the background CLI, while no foreground program is running. *Pagesize* specifies the amount of memory, in pages, that SMEM allocates to the background. (One page equals 1024 words, or 2048 bytes.)

Foreground and background programs run independently of one another. Each ground has its own user memory area, and has its own task scheduler for controlling program tasks.

The two grounds share system resources such as CPU time and I/O devices, so that while one ground is idle, the other can use the system. The grounds can have equal priority, or you can assign higher priority to the foreground. This allows the foreground to run programs requiring fast system response, while the background runs less urgent programs, making use of extra system time that would otherwise go unused.

Setting Up the Foreground and Background

The GMEM command displays the current memory allocations for each ground. As noted previously, however, when DG/RDOS starts it runs in the background. Therefore, you see a display similar to the following when you use the GMEM command:

```
BG: 341     FG: 0
```

You now use the SMEM command to allocate the amount of memory you want for the background, and DG/RDOS assigns the remainder of available user memory pages to the foreground. The command

SMEM 121

allocates 121 of the 341 available pages to the background, and the remaining 220 pages to the foreground. The GMEM command now displays:

```
BG: 121     FG: 220
```

Once you allocate the desired amount of memory to the background with the SMEM command, you can execute programs in either background or foreground.

Before you execute any program in the foreground, however, you may want to check its memory requirement with the SEDIT utility.

Communication Between Foreground and Background Programs

Foreground and background programs run in separate portions of memory, but they share system resources. DG/RDOS allows communication and resource sharing between foreground and background as described below.

When running two grounds, the system console ($TTI and $TTO) interacts with the background, and another terminal (usually $TTI1 and $TTO1) interacts with the foreground program.

To find out if a program is currently executing in the foreground, type *FGND* from the background. You receive one of two responses: `No foreground program running` or `Foreground already running.`

Programs running in foreground and background may pass information to each other through system calls.

You can use system calls to *checkpoint* a background program—to stop the background program, run something else, then recall the background program. For instance, if your background and foreground programs are functionally dependent upon each other, you can issue a call from the foreground that causes the background to process information that the foreground program needs. See the *RDOS System Reference* for more information.

Foreground and background programs can initialize tapes but cannot simultaneously read or write data to them. The first program to read or write to a tape will be successful.

Both can open, read, and write to the same file. Issuing the .EOPEN system call from one ground prevents the other ground from opening a file.

Both can initialize and use disks and directories at the same time, and either can release a directory without affecting the other's use of that directory. If one ground releases a directory that the other ground has initialized, the message

```
Directory shared: directoryname
```

appears at the ground's console if the CLI is running. This is an informational message only, not an indication of error. This directory remains initialized for one ground, and becomes released for the other.

If you attempt to release the master directory from the background while a foreground program is running, you receive the message:

```
Foreground already running: directoryname
```

To release the master directory from the background, you must first terminate any foreground program still running.

If the foreground program releases the master directory, DG/RDOS releases it for the foreground, and the master directory remains initialized for the background.

Terminating the Foreground Program

You terminate the foreground program by issuing a CTRL-C CTRL-F from the background console. After terminating the foreground program, use the SMEM command to restore all memory to the background.

You must terminate the foreground before shutting down DG/RDOS.

Examples

```
GMEM
BG: 341    FG: 0
SMEM 121
GMEM
BG: 121    FG: 220
```

The first GMEM command displays the total amount of available memory prior to setting up memory for two grounds. The SMEM command allocates 121 pages of memory to the background, and allocates the remainder of available memory to the foreground. The final GMEM command confirms that the background is set up with 121 pages of memory, and the foreground is set up with the remaining 220 pages.

```
GMEM
BG: 121    FG: 220
SMEM 341
GMEM
BG: 341    FG: 0
```

After checking the memory allocation for foreground and background with GMEM, the SMEM command allocates all available user memory pages (121 plus 220) to the background.

SPDIS

Command

Disable device spooling.

SPDIS device [...device]

DG/RDOS automatically spools data sent to any device defined as spoolable or enabled with the SPEBL command. During a spool operation, DG/RDOS stores in disk buffers data that is being sent to output devices. This frees the CPU for processing while the devices receive the data from the disk buffers.

Use the SPDIS command to stop spooling data to the specified device. Use SPEBL to re-enable spooling to the device.

If a spool operation requires more disk space than is available, the system itself will issue the equivalent of an SPDIS command, and will restart spooling when appropriate. This sequence is invisible to the user.

See also SPKILL for deleting a spool queue.

Examples

SPDIS $LPT

Prevents data output to the line printer from being spooled. To reinstitute spooling, you must issue the command SPEBL $LPT. If output is currently being spooled to the line printer, spooling will stop after the current spool is completed.

SPEBL

Command

Enable device spooling.

SPEBL device [...device]

SPEBL re-enables spooling on a device for which spooling has been disabled by SPDIS. The device can be any device defined as spoolable.

Note that you cannot spool to a multiplexor line (QTY).

See the SPKILL command for deleting a spool queue.

Examples

SPEBL $LPT

Re-enables spooling to the line printer.

SPEED

Utility

Invoke the Superedit text editor.

SPEED [filename]

The SPEED utility, also known as Superedit, allows you to create and modify random text files. *Filename* specifies the new or existing file to be edited.

If *filename* does not already exist, SPEED creates it. You can invoke SPEED without a filename argument and specify the file from within SPEED.

When you invoke SPEED, the utility generates an exclamation point (!) prompt; you type in SPEED commands in response to this prompt. SPEED commands can add delete, or change information in a text file, and store the modified file on disk. In addition, SPEED contains features that allow use of multiple buffers, multiple I/O files, programming of macros, and use of numeric variables.

To end an editing session and return to the CLI, type H and press the ESC key twice.

SPEED is documented in the *RDOS/DOS Superedit Text Editor* manual.

SPKILL

Command

Stop spooling and delete the current spool queue.

SPKILL device [...device]

Cancels a spool operation by deleting the spool files queued to the device. The device may be any device defined as spoolable.

After you kill spooling on a device, data on the output spool is lost. Spooling will resume to the device at the next command that sends data to the device (e.g., PRINT MYFILE).

Examples

SPKILL $LPT

Stops the spooling of data to the first line printer, and deletes the spool files.

STOD

Command

Set the system clock.

STOD hh mm ss

STOD sets the system clock to the time of day you specify. The system clock is a 24-hour clock (i.e., you would specify 2 p.m. as 14 00 00). Use spaces or commas to separate the time arguments.

Examples

STOD 21 24 0

Sets the system clock to 9:24 PM.

SYSGEN

Utility

Generate a new operating system.

SYSGEN [systemname]

Use SYSGEN to generate a brand new DG/RDOS system or a different version of an existing system after you have installed an update or new revision of DG/RDOS or have added new hardware to your system. SYSGEN asks a series of questions; after you have answered them, SYSGEN builds a system that meets your specifications.

NOTE: Do not use the name of the current system as *systemname,* and, unless you want to delete your DG/RDOS starter system, do not name the new system DGRDOS.

You can use the global /H switch in the command line to request help, or you can get help during the SYSGEN dialogue by entering a *?* as the response to any question.

As it builds the new system, SYSGEN produces *systemname.SV,* containing your tailored DG/RDOS system, and *systemname.OL,* containing the system overlays.

If you include on your command line the argument/local-switch combination *systemname.SG/V,* SYSGEN records its questions and your answers in this dialog file.

If you include on your command line the argument/local-switch combination *systemname.LM/L,* SYSGEN writes to this file the names and values of all the entry points in your system and overlay files. You *must* create this load map file if you plan to use the PATCH utility.

After you answer the last SYSGEN question, the SYSGEN program analyzes your answers to its questions, and then places the names of required modules and libraries for this system in CLI file CLI.CM (FCLI.CM for a foreground CLI). Then it processes CLI.CM with RLDR. If you use the global /N switch, SYSGEN skips the RLDR step, and leaves the RLDR command line in (F)CLI.CM. You can invoke the RLDR phase once, before executing another SYSGEN, with the command @CLI.CM@.

When the SYSGEN program is finished, the **R** prompt reappears, and you can boot your newly created system with the BOOT command (unless you want to run the CONFIG program).

See *How to Generate and Run DG/RDOS* for a complete description of the SYSGEN utility.

Global Switches

/H Display a help message that describes SYSGEN's global and local switches and shows some sample command lines.

/N Do not build a new system after completion of the dialog.

/R Build a system without an overlay file.

Local Switches

systemname.SG/A Build a system according to the specifications recorded in a dialog file (*systemname.SG*) from a previous SYSGEN session. If you use this argument/local-switch combination, the utility does not run interactively.

systemname.SG/E Use the values recorded in a dialog file (*systemname.SG*) from a previous SYSGEN session as the default answers to the questions it will pose during the current session.

NOTE: Do not use the /A and /E switches on the same command line.

systemname.LM/L List in a load map file the names and values of all the entry points in your system and overlay files.

systemname.SV/S Name the new operating system *systemname.SV*. If you do not supply this argument, SYSGEN uses the name SYS000.

systemname.SG/V Record the dialog with the utility in a file named *systemname.SG*. During later SYSGEN sessions, you can use the values recorded in this dialog file as input to the program (see the descriptions of the /A and /E switches above).

Examples

SYSGEN SYS1.<SV/S,LM/L,SG/V>

Activates the system generation program, allowing you to generate a new operating system by answering a series of questions. The new system is named SYS1.SV with overlay file SYS1.OL. The command also produces load map SYS1.LM and dialog file SYS1.SG.

TYPE

Command

Display the contents of a file on the console.

TYPE filename [...filename]

TYPE displays on the terminal the contents of *filename*.

TYPE works properly only for text (ASCII) files, such as those with an .MC extension. If you try to TYPE an .SV file, you will get strange results and might lock the terminal in graphics mode. If this should happen, try typing CTRL-C CTRL-A. If that does not work, press CMD and ERASE PAGE and then try CTRL-C CTRL-A again.

You can suspend the display from a TYPE command by typing CTRL-S, and resume it by typing CTRL-Q. This aids in reading long files.

To display the contents of a binary file, use the FPRINT command.

Examples

TYPE DJ0:$$READ.ME

```
This diskette contains updated information and testing
procedures for project goldfinger.  Feel free to use it.
                                -JB-
```

Displays the contents of disk file $$READ.ME on diskette DJ0.

TYPE MT0:0

```
DOCUMENT ARCHIVE   DG/RDOS CLI MANUAL  REV 093-000471-00
-------------------------------------------------------
LOAD TAPE FILES 1, 2, and 3.
```

Displays the contents of the file on tape MT0, file 0. In this example, the file contains information about the contents of the tape.

UNLINK

Command

Delete a link file.

UNLINK linkfile [...linkfile]

Unlinks files by deleting one or more link files. This command does not affect the resolution file.

CAUTION: Always use the UNLINK command to unlink files; *do not* use the DELETE command. DELETE will delete the resolution file (unless the resolution file has been assigned the attribute P; see the CHATR and CHLAT commands) and the link file will remain intact. If you delete the resolution file, you can restore it if you have kept a backup copy.

See the LINK command for information about creating and using link files.

Global Switches

/C Request confirmation of each deletion. The system repeats each specified link filename, and waits for you to confirm the deletion by typing a command line terminator. To prevent the removal of a link, press any key other than a command line terminator.

/L List names of the deleted link files on the line printer ($LPT). This switch overrides the /V switch.

/V Display on the console the name of each link file deleted.

Examples

UNLINK/C TEST.-
`TEST.SR` <*NL*> * `TEST.RB` <*NL*> * `TEST.SV` <*NL*> *

Removes any link files whose names begin with TEST. The /C switch causes UNLINK to request your confirmation before deleting each link entry from the current directory. When you confirm a deletion with a command line terminator, the system echoes an asterisk (*); when you prevent a deletion by typing any key except a command line terminator, the system echoes nothing.

XEQ

Command

Execute a program.

XEQ programname

Normally with DG/RDOS you have to enter only the name of an .SV file in order to execute the program. However, if ABBREVIATE mode is on and you have an .SV file with the same name as an acceptable CLI command abbreviation or a macro (.MC) file, the file will not execute if you simply enter its name.

You can solve this problem by turning ABBREVIATE mode off, but you then lose the advantage of being able to use abbreviated CLI commands. The XEQ command provides an alternative.

XEQ allows you to circumvent all CLI command and macro processing. For example, the command *XEQ LIS* will execute the program LIS.SV, even though LIS is the abbreviation of the LIST command, or even if a LIS.MC exists.

However, note the following:

- If you have a macro named XEQ.MC, you can execute it only by explicitly supplying the .MC extension in your command line, that is, by typing *XEQ.MC*.
- If you have a program named XEQ.SV, you can execute it in either of two ways:
  1) By explicitly supplying the .SV extension, that is, by typing *XEQ.SV*.
  2) By using the XEQ command, with or without the .SV extension, that is, by typing *XEQ XEQ* or *XEQ XEQ.SV*.

Examples

XEQ REP

Executes the program REP.SV (REP is the unique abbreviation of the REPLACE command.)

XFER

Command

Transfer the contents of a file to another file; transfer input from a terminal to a disk file.

XFER sourcefile destinationfile

Sourcefile specifies the filename or device name (but not a directory) from which you transfer information.

Destinationfile specifies the filename or device name (but not a directory) to which you transfer information. Both of these arguments can be in a pathname.

XFER transfers information from a file on any device to another file on any device. Note that with DG/RDOS, devices are logically constructed as files, and are referred to by their device names. (See Table 2-2 in chapter 2 for a list of DG/RDOS device names.)

If you omit local switches and the destination file does not exist, XFER creates the destination file as a sequential file.

When the destination file is an executable program (.SV) file, be sure to include the /R local switch. Also, a new destination .SV file will not be assigned the S attribute. Be sure to use a *CHATR filename +S* command on the destination file.

When you transfer information into an existing destination file, the file retains its original characteristics.

During a transfer, the system will detect parity errors in all ASCII source files and in binary tape files. On a parity error, DG/RDOS display the message `Parity error.` For magnetic tape files, DG/RDOS aborts the command when it detects a parity error.

You can use XFER to copy text directly into a file from the background terminal in the form:

XFER/A $TTI filename

where $TTI is the device name for the foreground keyboard.

To copy text into a file from the foreground terminal, use the command:

XFER/A $TTI1 filename

where $TTI1 is the device name for the background keyboard.

Alternately, from either ground you can use the command:

XFER/A %GCIN% filename

where %GCIN% is the CLI variable for the device name of your keyboard.

Include the /B switch if *filename* is an existing file. All of the information you subsequently type at your keyboard becomes part of the destination file. To terminate the transfer and return to the CLI, press CTRL-Z.

If you use XFER to put information on magnetic tape, then you must use XFER to retrieve the information. In other words, you cannot use LOAD or another utility to read magnetic tape files written with XFER.

Global Switches

/A An ASCII (text) transfer. XFER transfers ASCII characters, line by line, taking appropriate read/write action, such as inserting line feeds and carriage returns, if that applies to the destination file.

/B Append the source file to the end of the existing destination file. Use this switch when transferring data to a disk file or tape file that already exists; otherwise, the XFER command will abort and you will receive the error message `File already exists.`

Local Switches

destinationfile/C Organize the destination file contiguously. (Both the source and destination files must be disk files.)

destinationfile/R Organize the destination file randomly. (The destination file must be a disk file.) Be sure to use this switch to transfer information to an .SV file.

Examples

XFER/A/B $TTI OLDFILE

```
We're appending this stuff to oldfile.
DG/RDOS transfers everything we type in until we press
the CTRL key and a Z simultaneously.
<CTRL-Z>
```

Appends information typed at the keyboard to file OLDFILE. OLDFILE is an existing file, and we included the /B switch to append the information to the file. We terminated the transfer with a CTRL-Z. OLDFILE retains its original characteristics and attributes.

XFER/A DA1:OLDNOTES NEWFILE/R

Creates a random file, NEWFILE, in the current directory, and transfers the information from the file OLDNOTES in directory DA1 to the new file.

XFER MYPROG.SV MT0:0

Transfers the file MYPROG.SV to tape file 0 on tape MT0.

XFER MT0:1 NEWPROG.SV/R
CHATR NEWPROG.SV +S

Creates a random file, NEWPROG.SV, from tape file 1 on tape MT0. The CHATR command assigns the new file an S (executable) attribute.

XFER/A/B QTY:7 DA1:MUXNOTES

Appends the text input from QTY line 7 to disk file MUXNOTES in directory DA1. The user on QTY:7 can terminate the transfer with CTRL-Z.

–End of Chapter–

Chapter 4
Grouping CLI Commands: Macro and Indirect Files

Macro and indirect files enable you to use a single command to execute many CLI commands. By storing a set of CLI commands in a macro or indirect file, you can run involved or often-used procedures quickly and simply. You can modify the commands in these files, as needed, with a text editor.

Macro files are the simpler of the two types of files. A macro file can contain only valid command lines. To execute a macro file, you type in the name of the macro file, as you would a CLI command.

Indirect files have further capabilities. An indirect file can contain complete command lines, or just the command arguments. You specify an indirect file in a command line by enclosing its filename in @ symbols. You can invoke an indirect file as a command or as an argument to a command.

When you invoke a macro or indirect file, the CLI reads in and executes each line in the file sequentially. The CLI responds to the commands in the file exactly as if you had typed each command to the CLI interactively.

See also the DO utility in chapter 3 for executing a macro command file with variable arguments that you specify in the DO command line.

See the section "Improving Macro and Indirect Files" for a discussion of the use of macro and indirect files to call other macro or indirect files.

Creating a Macro File

You can assign a macro file a name of your choice, but you must include the filename extension .MC. The .MC enables the CLI to recognize the file as a macro file when you invoke it.

Remember, however, the implications of abbreviate mode when you are creating macro files. See the discussion of abbreviate mode in chapter 1 for information.

NOTE: All CLI commands and variables in macro files *must* be UPPERCASE. Message text in macro files may be upper- or lowercase.

We will be using the XFER command to create files for our examples; a text editor such as SPEED may be more appropriate for you.

The following example creates a macro file called ALLDONE.MC to execute a number of commands that you might normally use when you have finished using the CLI for the day.

XFER/A $TTI ALLDONE.MC <NL>
DELETE/V -.BU <NL>
LIST/S <NL>
DISK <NL>
DIR %MDIR% <NL>
^Z

The command *XFER/A $TTI ALLDONE.MC* creates file ALLDONE.MC and puts in it the text we type at the keyboard (device $TTI). CTRL-Z closes the file.

The commands in macro file ALLDONE.MC:

- Delete any backup files you have created.
- Provide a sorted list of the files in your current directory.
- Display information about the amount of available file storage space.
- Make the master directory the current directory.

Invoking a Macro File

To *invoke* (execute) a macro file, simply enter its macro name as you would a CLI command. A *macro name* is the name of a macro file, minus the .MC extension.

The example below invokes the ALLDONE.MC by typing in ALLDONE as a command.

ALLDONE
```
DELETED FILE3.BU
FILE1.              68  D  01/06/84 16:37  01/06/84  [006564]  0
FILE2.              50  D  01/06/84 13:24  01/09/84  [006565]  0
PROGX.               0  D  01/13/84 10:42  01/13/84  [006566]  0
Left: 21230  Used: 4866   Max. contiguous: 19961
```

As the example shows, you do not need to specify the .MC extension when you invoke a macro file (as long as the macro name is not a CLI command or abbreviation). However, if you had not included the .MC extention when you created the macro, the name you enter is meaningless to the CLI, and you receive the message:

```
File does not exist:  ALLDONE.SV
```

The "Command Interpretation Order" section in this chapter explains CLI execution hierarchy.

Creating an Indirect File

An indirect file may have any filename of your choosing. It does not require an extension, because the CLI recognizes an indirect file by the way in which you invoke it.

The following example creates an indirect file called PROJFILES.

XFER/A $TTI PROJFILES <NL>
PROJ83:1QTR83 PROJ84:<1QTR,JAN,FEB,MAR>84 <NL>
^Z

The file PROJFILES contains two filename arguments. The first argument refers to the file 1QTR83, in the directory PROJ83. The second argument refers to four files in the directory PROJ84 (files 1QTR84, JAN84, FEB84, and MAR84). (Chapter 2 explains the use of angle brackets as a shorthand method of specifying arguments.)

This file does not contain complete command lines; we will invoke it as an argument to another command.

Invoking an Indirect File

You invoke an indirect file by enclosing its filename with two @ symbols. Since our indirect file, PROJFILES, contains only arguments, we can invoke the file only as an argument to a command:

LIST @PROJFILES@
```
PROJ83:1QTR83   142   D
PROJ84:1QTR84   161   D
PROJ84:JAN84    234   D
PROJ84:FEB84    306   D
PROJ84:MAR84    271   D
```

In this example, the @ symbols cause the CLI to search for the file PROJFILES, and use its contents as arguments to the LIST command. We could use @PROJFILES@ as an argument to a number of commands, such as PRINT, MOVE, and DELETE.

Note that the CLI recognizes *all* characters in macro and indirect files, including the command line terminators we used to enter each line into the file.

We can also invoke ALLDONE.MC as an indirect file, since it has all the components of an indirect file. All we need to do is enclose ALLDONE.MC with @ symbols.

@ALLDONE.MC@

This command has the exact effect of typing in ALLDONE as a macro file. However, we needed to include the .MC extension, because the CLI searches for a filename when it encounters an @ symbol. In this example, the @ symbols caused the CLI to search for and execute the contents of the file ALLDONE.MC.

Improving Macro and Indirect Files

You can improve your macro and indirect files by making use of the following features:

- CLI variables
- the MESSAGE command
- calling macro and indirect files from within other macro and indirect files

CLI Variables

The CLI stores some information about your system in a set of eight predefined variables, described in Table 4-1 . When you include a variable in a command line, the CLI replaces the variable with the current value of the item that the variable represents.

Table 4-1. CLI Variables

Variable	Value
%DATE%	Today's date, in the form mm-dd-yy.
%FGND%	The character *F* if the CLI is running in the foreground; a null value if it is not.
%GCIN%	The device name for the keyboard (terminal input), $TTI or $TTI1.
%GCOUT% $TTO1.	The device name for the display screen (terminal output), $TTO or
%GDIR%	The name of the current directory.
%LDIR%	The name of the previous current directory.
%MDIR%	The name of the master directory.
%TIME%	The time of day, in the form hh:mm:ss.

The following command line uses the variable %DATE% to list all files in the current directory that were modified today:

LIST/S/E %DATE%/A

This command line is general enough to include in a macro or indirect file. It will list all files modified on the current date, even as the actual current date changes.

Displaying Messages in Command Files

The CLI MESSAGE command can be very useful as a way of displaying text during the execution of a macro or indirect file. The MESSAGE command accepts text strings of up to 72 characters as arguments. For example, if you have the following command in a macro file:

MESSAGE Message displays whatever you type.

when you execute the file, the CLI responds:

```
Message displays whatever you type.
```

When you enclose your text argument in quotation marks ("), the CLI displays every character you type literally, except for the quotation marks. When you do not use quotation marks, the CLI interprets and executes any special command line symbols, including angle brackets, commas, and control characters (see tables 1-1 and 1-2 in chapter 1). This can work to your advantage. For example, the following line in a macro file reports the value of a CLI variable:

MESSAGE The device name for your keyboard is %GCIN%
```
The device name for your keyboard is $TTI
```

The CLI replaces the variable %GCIN% with its real value.

Note that both the CLI command MESSAGE and the variable GCIN are in uppercase. If you had entered them in your macro file as *Message* and *gcin,* you receive the following error messages when you to execut the file:

```
File does not exist:  Message.sv
Illegal variable:  %gcin%
```

If you enclose a variable such as %GCIN% in quotation marks, the CLI treats it as a character string, as it does any other string in quotation marks. For example:

MESSAGE "The value of %GCIN% is" %GCIN%
```
The value of %GCIN% IS $TTI
```

The command MESSAGE/P halts execution and displays the following message:

```
Strike any key to continue
```

This can be useful if you write a macro that requires the user to perform some procedure, such as inserting a diskette or mounting a tape, before the macro continues.

For example, we have the following lines in a macro file:

MESSAGE Transfer complete.
MESSAGE Insert next diskette.
MESSAGE/P

When we invoke the file, the terminal displays:

```
Transfer complete.
Insert next diskette.

Strike any key to continue
```

Calling Files From Within a File

You can call other macro and indirect files from within a macro or indirect file. This allows you to write command or argument files in small, flexible modules and then put them together as needed. In addition, you can make changes to one file without modifying any of the files that use that one file.

To call another macro or indirect file, you call it in the file exactly as you call it interactively. For example, the macro file MASTERFILE.MC contains the following text:

PRINT @PROJFILES@
PRINT @MOREFILES@
ALLDONE

When you type *MASTERFILE,* the CLI prints the files contained in the indirect files PROJFILES and MOREFILES and then executes the ALLDONE macro.

Command Interpretation Order

The CLI can recognize four types of items as the first word in a command line:

- indirect files (enclosed in @ symbols)
- CLI commands
- macro files (with a .MC filename extension)
- program files and CLI utilities (with an .SV filename extension)

The CLI interprets the first word in a command line in in the following order:

1) When the CLI encounters the indirect file symbol, @, it searches for the filename enclosed by the @ symbols, and executes that file as an indirect file. If no files in the current directory match the specified filename, the CLI returns a `File does not exist` message.

2) If there are no @ symbols, the CLI attempts to match the name with one of its own commands or, if abbreviate mode is on, with the unique abbreviation of one of its own commands.

3) If the name is not a CLI command or abbreviation, the CLI attempts to match the name with *name.MC* from the current directory. (That is, the CLI tries to execute *name* as a macro file.)

4) If the CLI cannot find *name.MC,* it searches for *name.SV* in the current directory, and tries to execute *name* as a program file. Program (.SV) files include user programs and CLI utilities.

As a result, if a macro file and a program file in the same directory both have the same name, the CLI will always find the macro file first and execute it. You can avoid this problem by renaming one of the files, or you can direct the CLI to search specifically for the program file by including an .SV extension when you type in the program name.

If the CLI cannot match the first word of a command line with any of the items listed above, you receive an error message:

FIRSTWORD
```
File does not exist:  FIRSTWORD.SV
```

The CLI checked for a command, an abbreviation, then for a macro file (.MC), and finally for a program file (.SV), and could not match FIRSTWORD with any of these.

Command Line Execution Order

The CLI executes the components of a command line in the following order:

1) Text strings in quotation marks (MESSAGE command).

2) Indirect files and variables (as they appear from left to right on the line).

3) Angle brackets.

4) Parentheses.

5) Numeric switches.

6) Text strings outside of quotation marks (including the first word of the command line, arguments, and global and local alphabetic switches, in the order they appear on the command line).

Numbers 1 through 5 are all items the CLI will need to *expand* the command line. Once the CLI takes care of the expansions, it can begin executing the entire command line from left to right.

–End of Chapter–

Chapter 5
Backing Up Files

This chapter briefly describes how to plan and schedule backup procedures on DG/RDOS systems with a hard disk. See *How to Generate and Run DG/RDOS* for full discussions of the DG/RDOS backup utilities.

Why Do It?

Just a few years ago, all of our information was written on paper. In those days, it was standard practice to make carbon copies or photocopies of valuable documents so that, if the originals were lost or damaged, the information could be recovered.

Today much of our information is stored on computer disks, and we make copies by backing up the disks to tapes or diskettes. Because the method for making copies is different and somewhat time consuming, some computer users feel that regular backups are too much trouble.

We must remember, however, that the reason for having copies of valuable documents has not changed. The amount of time and effort spent to back up your computer files will be far less than the time and effort needed to recreate all of the information if it is destroyed by accidental (or malicious) deletions or mechanical failure.

One of the most important things you can do, therefore, is to plan and implement a regular backup program.

Planning a Backup Program

A backup program should include both *incremental backups,* which copy only part of the files, and *full backups,* which copy all files.

How often you perform backups depends on the rate that new information accumulates, and how important that new information is. Simply ask yourself the question: How much work can I afford to lose? In answering the question, remember that in a worst case situation you may lose all of the information you have stored on the disk since your last backup. If this happens, the amount of time lost will be doubled; that is, if you back up your files only once a week and a serious problem occurs just before your scheduled backup, it may take a week to recreate the lost information.

For that reason, we strongly recommend that you perform daily backups. This is not as difficult and time-consuming as you may think, because you can incrementally back up each day only those files that have changed since your last full backup.

Another approach is to organize the disk so that important new files are written to one or two directories. You can then do daily backups (incremental or full) only of these directories.

In addition to your daily backups, we recommend a full weekly backup. We realize, of course, that a full backup takes time. However, in the event of a disk problem, you will appreciate the greater ability to recover crucial lost information, and you will also find that the restoration task is simpler and faster.

Finally, we recommend that for backups you have at least three generations of diskettes or tapes. For example, if your daily backup requires two tapes, you should have at least six tapes. Use the first two tapes on Monday, the second two on Tuesday, and so forth until you have used all the tapes, then restart the cycle with the first two tapes. Using several generations of tapes helps ensure that a media problem or a new disk problem does not leave you without a good backup.

Selecting a Backup Program

BURST and IMOVE are the two major backup utilities provided by DG/RDOS. Although they are similar, there are differences that may make one more suitable for your situation. The following list provides general guidelines for their use; for a detailed description of the two, see *How to Generate and Run DG/RDOS.*

- If you are backing up to diskettes, IMOVE is recommended. IMOVE does not require diskettes to be software formatted (with the DKINIT utility.) In addition, because IMOVE uses diskette space more efficiently, it is much faster if you use diskettes.

- To perform incremental backups to either diskettes or tape, you *must* use IMOVE.

- If you are performing a full backup to tape, and if you do not need to transfer files between DG/RDOS and Data General's AOS or AOS/VS operating systems, BURST is recommended. It is much faster and it also allows you to trace the ownership of disk blocks.

  NOTE: If you decide to use BURST for your full backups, we strongly recommend that you use IMOVE occasionally for the following reasons:

  - In many cases a problem affects only a portion of the disk, and it is not necessary to restore every file on the disk. IMOVE allows you to restore individual files or directories.

  - Because BURST dumps exact block-by-block copies of the disk, any fragmentation on the disk is preserved when you back up and will exist on the disk if you have to restore files after a disk problem. As a result, disk access times will be greater than necessary and there may be unnecessary restrictions on the number of large contiguous files you can create.

    If such fragmentation becomes a problem, back up the disk with IMOVE, use *INIT/F* to create a new file system on the disk, and restore the files. This process will compact files toward the beginning of the disk, resulting in better access time.

- If you will be transferring files between DG/RDOS and other Data General systems, you *must* use IMOVE.

- If you want to duplicate the entire contents of one disk onto another disk, you *must* use BURST.

  Note, however, that the original and target disks must be the same type, the remap areas must start at the same place and be the same size, and that the disks' coresident diagnostic areas (if any) must also be the same size and have the same starting address.

Backup Pointers

The following are some additional important points about backup:

- Avoid backing up material you do not need to back up.

- Start each backup set from the same directory. Ideally, all backups should start from the master directory.

- Do backups when the system is idle. If any file is open when the backup program tries to copy it, the file will not be copied. You can avoid this by having the foreground program (if any) shut down, and doing the backup from the background CLI.

- Be sure to label tapes and diskettes. For example, for daily backups you might have four tapes labeled Mon, Tue, Wed, and Thu. Be sure to record the date of the backup on the label.

  For weekly full backups you might have three tapes labeled Week1, Week2, and Week3. Be sure to record the date of the backup on the label.

- When the backup takes more than one diskette or tape, make sure that the tapes or diskettes are labeled in sequence. When you restore files, you need to load the tapes or diskettes in the proper order.

- Follow any backup instructions supplied with applications software. Some products have specific backup and restoration procedures, and it is essential to follow these procedures.

- Use the online HELP provided by BURST and IMOVE whenever you are uncertain about how to use them. With BURST, either use the /H switch in the command line or enter a ? as the response to any question. With IMOVE, use the /H switch.

- *Always verify your backups.* With BURST, use the DUPLICATE/V command, which verifies as the backup is taking place. With IMOVE, use the IMOVE command with the /N and /V switches for each diskette or tape after files have been backed up to it.

–End of Chapter–

Appendix A
Summary of CLI Commands and Utilities

ABBREVIATE [**ON** | **OFF**]
Turn abbreviate mode on or off.

APPEND outputfilename inputfilename . . .
Combine two or more files.

BOOT {**disk** | **[directory:]sysname**}
Bootstrap a system from disk.

BUILD outputfilename inputfilename
Build a file consisting of filenames.

BURST
Dump and load disk images between disk, diskette, or magnetic tape.

CCONT filename blockcount
Create a contiguous file.

CDIR directoryname
Create a subdirectory.

CHAIN programname
Load and run a program in place of the CLI.

CHATR filename [sign] attributes
Change a file's attributes.

CHLAT filename [sign] attributes
Change a file's link access attributes.

CLEAR [filename]
Set file use count to zero.

CONFIG [systemname] [dialogfile.CF/V]
Examine or change system configuration parameters.

CPART partitionname blockcount
Create a secondary partition.

CRAND filename
Create a random file.

CREATE filename
Create a sequential file.

DEB programname
Debug a program.

DELETE filename
Delete a file or directory.

DIR directoryname
Change the current directory.

DISK
Display the allocation of disk storage space for the current partition.

DO filename [argument...]
Execute command files, replacing dummy arguments with the specified arguments.

DUMP [destination:]dumpfilename [filename]
Store one or more disk files from the current directory in a dump format file on another directory or device.

ENDLOG [password]
Close the log file opened by LOG.

ENPAT [patchfile]
Create a patch file.

EQUIV newname oldname
Assign a temporary name to a disk or tape drive.

EXFG programname
Execute a program in foreground memory.

FCOPY [source destination]
Copy a file or duplicate a diskette.

FGND
Determine if a foregound program is running.

FILCOM filename1 filename2
Compare the contents of two files.

FPRINT filename
Print a file in the specified format.

GDIR
Display the name of the current directory.

GMEM
Display background and foreground memory allocations.

GSYS
Display the name of the current operating system.

GTOD
Display the time and date.

IMOVE [argument/switch] devname [filename...]
Dump and load files between disk, diskette, or magnetic tape.

INIT { **directory** | **device** }
Initialize a directory or device for I/O.

LABEL devicename/D
Write and examine diskette and tape labels.

LDIR
Display the name of the previous current directory.

LFE {
A inputmaster [...arg(n)] [listfile/L]
A/M inputmaster(1) [...inputmaster(n)] [listfile/L]
D inputmaster [outputmaster/O] arg(1) [...arg(n)]
I inputmaster [outputmaster/O] file(1)[...file(n)]
M outputmaster/O inputmaster(1) [...inputmaster(n)]
N outputmaster/O] file(1) [...file(n)]
R inputmaster [outputmaster/O] arg(1) file(1) [...arg(n) file(n)]
T inputmaster [listfile/L][...inputmaster(n)]
X inputmaster arg(1) [...arg(n)]
}
Create, edit, and analyze library files.

LINK { **resolutionfile/2** | **linkfile resolutionfile** }
Create a link to a file in another directory.

LIST [[directory:]file]
Display information about the files in a directory.

LOAD [source:]dumpfilename [filename ...]
Restore DUMP files to the current directory.

LOG [password] [directory/O]
Record the current CLI session in the log file (F)LOG.CM.

MAC filename...
Assemble source file(s) with the Macroassembler to produce a relocatable binary (.RB) file.

MDIR
Display the name of the master directory.

MESSAGE ["]textstring["]
Display a text string.

MOVE destination [filename ...]
Copy files from the current directory to the specified directory.

OVLDR prog_name old-overlay/N new-overlay(s)
Create an overlay replacement file.

PATCH [programname/S] [patchfile/P] [loadmap/L]
Install a patchfile in a program (.SV) or overlay (.OL) file.

POP
Return to the next higher level program.

PRINT filename
Print a file on the line printer.

RELEASE { **directory** | **device** }
Release a directory or device from system initialization.

RENAME oldfile newfile
Change the name of a file or directory.

REPLACE programname
Replace overlays in an overlay file.

REV program[.SV]
Display the revision level of an .SV file.

RLDR binary_file [overlay_binary] library_file
Loads relocatable binary (.RB) files to produce an executable program (.SV) file.

SDAY mm dd yy
Set the system calendar date.

SEDIT filename
Analyze and edit a disk file.

SMEM pagesize
Allocate background and foreground memory.

SPDIS device
Disable device spooling.

SPEBL device
Enable device spooling.

SPEED [filename]
Invoke the Superedit text editor.

SPKILL device
Stop spooling and delete the current spool queue.

STOD hh mm ss
Set the system clock.

SYSGEN [systemname]
Generate a new operating system.

TYPE filename
Display on the console the contents of a file.

UNLINK linkfile
Delete a link file.

XEQ programname
Execute a program.

XFER sourcefile destinationfile
Transfer the contents of a file to another file; transfer input from a terminal to a disk file.

–End of Appendix–

Appendix B
Error Messages

This appendix presents CLI and DG/RDOS operating system error messages in alphabetical order, and gives possible causes and actions that should correct the error. For greater legibility, the messages are shown here in uppercase; they appear on the terminal in upper- and lowercase.

A ZERO .XMT OR .IXMT MESSAGE
Returned from SYS.LB when you attempt to transmit a message with an .XMT or .IXMT task call and the message you are trying to send (ac1) is zero.

ALM LINE NOT READY
DG/RDOS thinks a QTY line is not ready because the modem signals are in a bad state. Terminate the modem connection, then re-dial.

ABBREVIATION NOT UNIQUE: abbreviation
Abbreviate mode is on, but the CLI cannot tell which command you want. Enter enough characters for a unique abbreviation.

ADDRESS ERROR IN .SYST ARGUMENT
A program issued a system call with an address that is invalid (it probably exceeded the NMAX allocated for the program).

ATTEMPT TO CREATE A ZERO LENGTH CONTIGUOUS FILE
You have not specified a blockcount in the CCONT command.

ATTEMPT TO READ INTO SYSTEM SPACE
A program issued a system call with an address that is invalid—probably exceeding the NMAX allocated for the program.

ATTEMPT TO RELEASE AN OPEN DEVICE
You've attempted to release a magnetic tape unit while a file on the tape is open.

ATTEMPT TO RESTORE A NON-EXISTENT IMAGE
You issued the POP command from a level 0 CLI. The system has no program to restore.

ATTEMPT TO WRITE AN EXISTENT FILE
The program tried to write to a file that already exists.

BLANK TAPE
Probably the tape is new and has not been initialized with INIT/F.

BREAK
The program that is running encountered a severe error and aborted. You also receive this message if you terminate a command or process with CTRL-C CTRL-B.

CANNOT CHECKPOINT CURRENT BG
Checkpointing is the practice of suspending one background program (the checkpointed program) temporarily so that you can execute a new program in the background. You have apparently attempted to checkpoint a background program that performs multiplexor I/O or uses a .DELAY, .RDOP, .IDEF/.IRMV, or .DUCLK/.RUCLK system call.

CHANNEL ALREADY IN USE
You have attempted to open a channel that has already been opened.

CHANNEL CLOSED BY ANOTHER TASK
While one task was attempting to access a channel, another task closed it.

CHECKSUM ERROR
The drive hardware could not read the diskette or tape. Retry. If the error recurs, try another diskette or tape or a different drive. For a tape drive, cleaning the tape or cleaning the unit's read-write heads may help.

COMMON SIZE ERROR
The communications area of your program exceeds the program size or would overwrite the system, or the size of a message requested from another program exceeds the size of the communications area.

COMMON USAGE ERROR
You have attempted to write a message into another program's communications area, but no communications area is defined in the other program.

CONSOLE INTERRUPT RECEIVED
As you were reading from a QTY line, an interrupt character was entered. Your read terminates with this error message.

DEVICE ALREADY IN SYSTEM
You tried to initialize (INIT) a directory that is already initialized.

DEVICE NOT IN SYSTEM
You tried to access a tape or disk unit without initializing it. For tape, type *INIT device*; for a disk-based directory, you can use either INIT DIR.

DEVICE PREVIOUSLY OPENED
You have tried to open a magnetic tape unit that is already open.

DEVICE TIMEOUT: device
The system has tried for 20 seconds to access the device but cannot do so. For a diskette, this probably means that the diskette is not inserted in the unit properly or not inserted at all. Check the diskette in the drive.

DIRECT I/O ACCESS ONLY
A program tried to perform nondirect-block I/O on a file that requires direct-block I/O.

DIRECTORY DEPTH EXCEEDED
You (or a LOAD command) tried to create a subdirectory within a subdirectory, or a secondary partition within a secondary partition. Change directories or commands as appropriate, and re-enter the command.

DIRECTORY IN USE
The CLI thinks that the directory you tried to delete or rename is in use. First, try releasing the directory with the RELEASE command. If the message recurs, the foreground program may be using the directory. If so, you might not want to delete or rename the directory. If you do want to delete or rename it, have the foreground release it, then re-enter the command.

If you still get the message, there may be a file open in the directory. For example, you cannot release the directory from which you started logging (LOG command) until you type ENDLOG. Check for open files with the LIST/U/S command. If any file (except SYS.DR) has a use count other than zero, type the following commands:

DIR directoryname
CLEAR/A/V/D; CLEAR LOG.CM
RELEASE directoryname

Try again to delete or rename the directory. If the foreground is running, the system does not let you use the CLEAR command. Wait until the foreground program terminates (or terminate it yourself) and then clear the use counts.

DIRECTORY NOT INITIALIZED: name
You have tried to access an uninitialized directory. Initialize with DIR or INIT and re-enter the command.

DIRECTORY SHARED
After you release a directory (RELEASE), this message reminds you that the other ground has the directory initialized. Take no action; the message is informational only.

DIRECTORY SIZE INSUFFICIENT
You (or a program) tried to create a secondary partition smaller than 48 blocks in size. Repeat the command, specifying a blockcount of 48 or more.

DISK FORMAT ERROR
The disk is not in DG/RDOS directory format. It has been hardware formatted but not software formatted. You can use IMOVE with it, but if you want to use INIT or DIR, you must software format it with the DKINIT FULL and INIT/F commands.

DUPLICATE READ OR WRITE
You tried to issue a system call to read from or write to a QTY line, but some other task in your program is already doing that.

END OF FILE
The system hit an end of file when it tried to execute your command. With tape, this means you specified a nonexistent file number. Try a lower number, for example, MT0:0.

EXECUTE ERROR ON CHAIN
You have attempted to execute a program with the CHAIN command, but there is an error in the program (.SV) file.

EXPANDED ERROR TEXT NOT AVAILABLE
From SPEED text editor. SPEED needs two files for its error message text— CLI.ER and SPEED.ER. Either execute SPEED from the directory in which these files reside or create links to the files. If you're running SPEED from a nonmaster directory, and CLI.ER and SPEED.ER reside in the master directory, create needed links by typing:

LINK CLI.ER %MDIR%:CLI.ER
LINK SPEED.ER %MDIR%:SPEED.ER

FATAL SYSTEM UTILITY ERROR
You may have a program which issues an .ERTN system call with EREXQ in ac2 but (F)CLI.CM doesn't exist. Several DG/RDOS utilities will issue the .ERTN call with the message `Fatal utility error` in ac2 if a serious error occurs and the utility either has already issued an error message or does not know what else to do.

FG TERM
From the background CLI. The foreground program has terminated. Either you did it from the system console with CTRL-C CTRL-F, or the program terminated normally, or the program encountered a fatal error condition.

FILE ALREADY EXISTS
You tried to create a file whose name already exists in the specified directory, or you tried to use the MOVE, IMOVE, LOAD, or XFER commands on such a file without appropriate switches.

FILE ATTRIBUTE PROTECTED
You tried to change the attributes of a file whose attributes were fixed with the .CHATR system call. The attributes can't be changed. However, if the file does not have the P attribute, you can rename and delete it.

FILE DATA ERROR: `filename`
There is an inconsistency in the file you're reading or writing to. If `filename` is a diskette name, this message normally means that the diskette you tried to initialize (INIT or DIR) has not been hardware formatted. (If the diskette has been hardware formatted but not software formatted, the message is `Disk format error.`)

If `filename` is SYS.DR or MAP.DR, this means a system directory has inconsistent information in it. Probably, an INIT/F command is needed on the disk to fix things. You can try to back up existing files before doing the INIT/F, but the backup may be inconsistent.

FILE DOES NOT EXIST: `filename`
The system can't find the file in the current or specified directory. The problem may be one of the following:

- You made a typing error.
- The file exists, but you forgot a needed extension.

- The file exists, but in another directory, and there is no link to the file in the current directory.
- If name is DJ0 or another disk name, this means that the disk has not been fully initialized with INIT/F. Verify that this is the right disk. Then, if you still want to access it as a directory, type *INIT/F* and confirm with Y.
- The file really doesn't exist.

`FILE IN USE: filename`
You tried to read, rename, or delete a file that the system thinks is in use. If a file is open at abnormal shutdown, its use count remains nonzero when the operating system comes up again; DG/RDOS thus assumes that someone is still using the file. Check a file's use count by typing *LIST/U filename*. To set the use count to zero, type *CLEAR/A/V/D*. If this doesn't help, type *CLEAR/V filename*. Note that CLEAR does not fix any data inconsistency that may have developed because the file was left open.

`FILE NOT FOUND: filename.SV`
From the bootstrap program, at startup. The file you specified was not found. Programs you can start from the `Filename?` question include the operating system (DGRDOS.SV), or a stand alone utility such as DKINIT or MBOOT.

`FILE NOT OPEN`
You tried to issue a system call taking a channel number as an argument, but no file is open on that channel.

`FILE POSITION ERROR`
You tried to set the current file pointer (.SPOS system call) to a position that is not allowed.

`FILE READ PROTECTED`
You cannot read the file because it has the R (read-protect) attribute. If you need to read it, type *CHATR filename -R* and re-enter the command.

`FILE SPACE EXHAUSTED`
This message means that your command cannot be completed because the current disk or secondary partition is full, or, during a backup, all of a tape has been used. You can gain disk space by deleting some files. If you have some that you will not ever need, simply delete them. If you are not sure, back up the files and then delete them. (On a multiuser system, ask users which files they don't need before deletion.) With a hard disk, you need to have a minimum of 500-1000 free blocks.

If this message occurs during a backup operation, your backup is incomplete; load a tape that has more available space and re-enter the command.

`FILE WRITE PROTECTED`
From a text editor or the CLI. If it is a disk file, it has the W (write-protect) attribute. If you need to change it, type *CHATR filename -W*. If it is a tape file, insert a write ring. If it is a diskette file, make sure the diskette is write enabled.

FILES MUST EXIST IN THE SAME DIRECTORY
Your command (probably RENAME) requires both files to be in the same directory.

FOREGROUND ALREADY RUNNING
While a foreground program is running, you can't use SMEM to change memory allotments, CLEAR file use counts to 0, execute a program in the foreground, or shut down (BYE). Terminate the foreground program with CTRL-C CTRL-F, then re-enter the command.

ILLEGAL ARGUMENT
You specified an illegal argument. Check the command's syntax.

ILLEGAL ATTRIBUTE
You specified an illegal attribute (with CHATR). See the CHATR command for more detail.

ILLEGAL BLOCK TYPE
You tried to use the LOAD command with a file that was copied with XFER or to use XFER on a file that was copied with DUMP. Use the appropriate command.

ILLEGAL CHANNEL NUMBER
You tried to issue a system call taking a channel number as an argument, but that channel number is too large. Reduce the number, or use CONFIG to specify more channels.

ILLEGAL CHARACTER
From system, at startup, when you're trying to type the date or time. Re-enter the date or time in the correct format. Also try the numeric keypad to type numbers.

ILLEGAL DIRECTORY NAME
The directory name you specified was illegal. Legal filename characters are a-z, A-Z, 0-9, and $.

ILLEGAL FILE NAME
The filename you specified was illegal. Legal filename characters are a-z, A-Z, 0-9, and $.

ILLEGAL INDIRECT FILENAME
The indirect filename you specified was illegal. Retype the command, using the form @filename@ (legal filename characters are a-z, A-Z, 0-9, and $).

ILLEGAL NUMERIC ARGUMENT
There may be a nonnumeric character in a numeric argument, the numeric argument may be too large, or the radix may be wrong.

ILLEGAL PARTITION VALUE
In your SMEM command, you tried to allocate more memory to the background than is available to both grounds, or allocate too little memory (less than 20 pages) for the background CLI. Re-enter the command with an appropriate value.

ILLEGAL SYSTEM COMMAND
You have tried to issue a system call that does not exist. It may be an RDOS rather than DG/RDOS call.

ILLEGAL TEXT ARGUMENT
You omitted a matching quotation mark (") in a MESSAGE command.

ILLEGAL VARIABLE
The variable you typed doesn't exist or is illegal. Retype the command, using the form *%variable%*.

INSUFFICIENT CONTIGUOUS BLOCKS
There is not enough contiguous space to hold the contiguous file you want to place on the disk. The disk is probably nearly full; take action described under message `File space exhausted.`

INSUFFICIENT MEMORY TO EXECUTE PROGRAM
There is not enough memory available in the foreground to execute the program. Use GMEM to check memory allotment. If the foreground (FG) does not have at least 16 pages, use the SMEM command to reduce the amount of memory allocated to the background, thereby increasing foreground memory, and re-enter the EXFG command. In necessary, keep reducing the backgound memory until the EXFG command works or you get an error message. Another option is to check the product Release Notice for the amount of memory required to execute the program.

INSUFFICIENT ROOM IN DATA CHANNEL MAP
You tried to use the .IDEF system call for a data channel but there are not enough slots available in the data channel map, or you have tried to use the .MTDRW system call but the requested record size is larger than the system can handle through the data channel map slots allocated for the tape drive.

INSUFFICIENT SPACE IN BAD BLOCK POOL (CORE)
You tried to initialize a disk with the .INIT system call, but the number of bad blocks on it and all the other currently initialized disks exceeds your response to the SYSGEN question. This error may also occur when you use the .DIR system call to a non-initialized disk or during a BURST, FCOPY, or IMOVE operation.

INT
You interrupted a command or program with CTRL-C CTRL-A.

INVALID BAD BLOCK TABLE
The disk cannot be initialized in its current state. Run a DKINIT PARTIAL (if this fails, DKINIT FULL) command on the disk.

INVALID DATE OR TIME
You tried to specify an illegal time or date, or used an illegal character. Type *GTOD* for an example of the correct format (you can use spaces as separators).

INVALID WINDOW BLOCK NUMBER (.REMAP)
You issued the .REMAP task call with an invalid window block number as an argument.

LINE TOO LONG
The command line and system limit for line-oriented text is 132 characters, including the line terminator or the ^ (SHIFT-6) line continuation character. Place one of these characters in the applicable line.

LINK ACCESS NOT ALLOWED
The file's link attributes (CHLAT command) prevent access by a link entry.

LINK DEPTH EXCEEDED
This probably means the file was linked to itself. Remove the link (UNLINK linkname) and recreate it specifying the correct resolution file pathname.

MAP.DR ERROR
The space allocation file has inconsistent information. You probably should perform an INIT/F. Try to back up existing files before doing the INIT/F, but the backup may be inconsistent.

NO DEBUG ADDRESS
You tried to debug (DEB command) a program that does not include a debugger (RLDR/D switch).

NO DEFAULT DEVICE
No directory is current. Set a new current directory with the DIR command.

NO DIRECT I/O
You tried an .RDB/.WRB/.ERDB/.EWRB system call with a sequential file, or you tried to use an overlay file that is not contiguous.

NO FILES MATCH SPECIFIER
There are no files that match your filename template (characters - or *).

NO MORE DCBs
You (or another user or program) tried to initialize a directory (via INIT or DIR), but the number of directories in use has reached the maximum allowed (as set in CONFIG). For the INIT or DIR command to work, an initialized directory must be released (RELEASE command). If you get this error message often, you may want run CONFIG to increase the number of directories accessible at one time, or you may want to reduce the number of directories in your system.

NO ROOM FOR UFTS
There are not enough channels configured for the program ground. The CLI requires 16 channels and other applications may require more. Run CONFIG to specify more channels. This message can also mean that a program you built doesn't have enough channels reserved (with the RLDR /C switch). It can also happen when you try to execute rather than boot a stand-alone program from the CLI.

NO SOURCE FILE SPECIFIED
From a utility like the MAC assembler. You omitted a source filename from the command line.

NO STARTING ADDRESS
You tried to execute a program which passes several of the system's tests (it is an .SV file, or it is a random file with the S attribute, etc.), but the file does not have a valid starting address specified in its User Status Table.

NO SUCH DIRECTORY: file
The operating system cannot find the directory. This may mean that the directory, or its parent directory, is not initialized. You can check directory names within the current directory by typing *LIST -.DR*. Use INIT or DIR to initialize the directory, then try again.

NOT A COMMAND
This message indicates that the CLI is not operating properly. Shut DG/RDOS down and restart. If the error happens again, reload CLI files from your release media or from a backup copy.

NOT A LINK ENTRY
You tried to UNLINK a nonlink file. To remove nonlink files, use DELETE.

NOT A SAVE FILE
You cannot execute any program that is not a save (.SV) file. It must also be a randomly organized file and have the S attribute. Check with LIST and give the S attribute (CHATR filename + S) if needed. If the file is not randomly organized (R), copy it to a random file by typing *XFER filename newname.*

NOT ENOUGH ARGUMENTS
More arguments are needed; check the command's syntax.

OUT OF TCB's
Your user program needs to have more tasks defined. Run RLDR on it again and specify more tasks.

PARITY ERROR
For action, see message `Checksum error.`

PERMANENT FILE: filename
You tried to delete a permanent file (P attribute). This can happen after a DELETE command or after a MOVE, IMOVE or LOAD command that specifies deletion (the /R or /O switch). If you really want to delete the file, remove its P characteristic (CHATR name -P). (Some system files are attribute protected, which means you can't remove the P attribute.)

PROGRAM NOT SWAPPABLE
A program that uses a .DELAY, .RDOP, .IDEF/.IRMV, or .DUCLK/.RUCLK system call has tried to swap to another program.

PUSH DEPTH EXCEEDED
You have attempted to swap too many levels.

QTY BUFFER OVERFLOW
This means a multiplexor line error—probably a simultaneous read or write to the same line.

RDOS ERROR: n
A system utility could not not interpret the error code returned. This can happen if the program lacks disk space to create a needed file. It can also indicate incompatible revisions of the utility and the operating system.

READ FRAMING ERROR
Input from QTY lines consists of data and parity bits surrounded by start bits and stop bits. The QTY hardware signalled that something came in without the proper start/stop bits. Use CONFIG to check the number of stop bits on the line; the standard device has one start/stop bit. If the line is configured correctly, you have a problem with the hardware.

READ OVERRUN ERROR
Input from QTY lines is bufferred on the QTY board one character at a time; the operating system must pull each character in before the next one arrives. The QTY hardware signalled that the operating system did not react quickly enough. Either you are trying to use too high a baud rate (run CONFIG to check), or the operating system or a program has interrupts disabled for too long a period of time (INTDS instruction, INT.DS task call). If neither of these is the case, you have a problem with the QTY line or the hardware.

SIGNAL TO BUSY ADDRESS
This message is returned from SYS.LB when you attempt to transmit a message (with a .XMT or .IXMT task call) and the "mailbox" (address) you are trying to send the message to already contains a nonzero value. Someone has previously transmitted a message but it has not been received; when the message is received (with a .REC system call) the contents of the mailbox are reset to zero.

SPOOL FILES ACTIVE
You have tried to shut down or boot a system, and there is material in a spool queue, such as the queue for the line printer. Either wait for the queue to empty or kill the spool queue by typing *SPKILL devicename,* then proceed.

SYSTEM STACK OVERFLOW
There is not enough memory available to execute your command. For some commands, the CLI needs a minimum of 21K words. Check with GMEM, give the CLI at least 21K words with SMEM, and try again. (If the foreground is already running, you must wait until it ends to change memory with SMEM.)

SYNTAX ERROR INSIDE [ ]
Correct the line and re-enter.

SYNTAX ERROR: < WITHOUT > OR > WITHOUT <
Correct the line and re-enter.

SYNTAX ERROR: ILLEGAL NESTING OF ( ) AND [ ]
Correct the command line and re-enter.

SYNTAX ERROR: ILLEGAL NESTING OF < > AND ( )
Correct the line and re-enter.

SYNTAX ERROR: UNMATCHED OR NESTED ( )
You have left out a parenthesis or you have attempted to nest parentheses within another set of parentheses. Correct the line and re-enter.

SYNTAX ERROR: UNMATCHED OR NESTED [ ]
Correct the command line and re-enter.

SYS.DR ERROR: filename
There ss an inconsistency in the system file directory. If `filename` is a disk name, you may get this message if the disk has been hardware formatted but not software formatted, or if it has been written to by a program like IMOVE or a different operating system.

SYSTEM DEADLOCK
You have attempted to allocate disk space in the master directory and there is no space left. This error may occur if output to a spoolable device is active and the spooler is also trying to allocate disk space in the master directory. Check your files; you probably have old ones that you rarely use. Make a backup copy and then delete the files from the disk.

TASK ID ERROR
Several task calls in SYS.LB can return this error. Check the *RDOS System Reference.*

TASK NOT FOUND FOR ABORT
Several task calls in SYS.LB can return this error. Check the *RDOS System Reference.*

TEXT ARGUMENT TOO LONG
You typed more than 72 characters between quotation marks. Retype with a shorter text string.

TOO MANY ARGUMENTS
You typed too many arguments. Check to see if you accidentally put an extra space or comma in the command line. Also check the command syntax in chapter 3.

UNIT IMPROPERLY SELECTED
You tried to access a magnetic tape drive, but the drive is not turned on or not on line, or there is no tape in the drive.

YOU CAN'T DO THAT
You attempted something not allowed. If you were using the ENDLOG command, you did not give the password specified in LOG. If you were attempting to shut down DG/RDOS, you may be on a CLI level other than zero or logging may be active. Try the POP command; if you are at CLI level zero, use the ENDLOG command. You also get this message if you attempt to chain while logging is active; here again you need to use the ENDLOG command.

-End of Appendix-

Glossary

This glossary describes computer-related terms that may be new to you—either as words or in relation to Data General products.

abort
The termination of a program prior to completion because of a serious error or an interrupt sequence; for example, you can abort an executing program by typing CTRL-C CTRL-A.

ANSI (American National Standards Institute)
A committee that publishes standards for a large range of things, including computer languages and tapes.

AOS,AOS/VS (Advanced Operating System, Advanced Operating System/Virtual Storage)
Two of the operating systems used by Data General computers.

argument
Something that is acted upon by a command, statement, or instruction. For example, in the statement PRINT MYFILE, MYFILE is an argument to the PRINT command.

ASCII (American Standard Code for Information Interchange)
This code establishes standard numeric values for each character used in text; the numbers range from 000 for the null character to 177 (octal) for the DEL character. An international character set extends the ASCII set with numbers from 200 (octal) to 377 (octal); these numbers indicate non-US, language-specific characters (for example, the UK currency symbol).

asynchronous line
A communications line that uses an asynchronous protocol to transmit characters. In such a protocol, each character has its own "framing" information: traditionally one start bit (before the character) and one stop bit (after the character). Asynchronous lines are generally used for terminals and—sometimes—for intersystem communication.

background program
The program that is running on the system console. At startup DG/RDOS runs the CLI on the system console. From this background CLI, you can execute a different program in the background. You can also allocate memory to a second ground, called the foreground, and run a foreground program.

backup
Files copied for safekeeping, usually onto magnetic diskettes or tape.

bad block
On the magnetic surface of a disk or diskette, a bad block is a flawed area that won't hold information. The DKINIT software formatter notes such areas so the operating system will avoid them. If DG/RDOS encounters a new bad block, it usually displays a `FILE DATA ERROR` message.

BASIC (Beginner's All-purpose Symbolic Instruction Code) language
An easy, interpreted language, originally developed at Dartmouth College. BASICs available with DG/RDOS include Business BASIC, with special business and ISAM capabilities; and Extended BASIC, a traditional BASIC.

baud
The rate at which a line or modem can transfer data. Normally, each character requires 10 bits, so characters are transferred at 1/10 the baud rate. The standard (and default) baud rate for terminals is 9600. For modems it is 1200.

bit
A Binary digIT. A bit can assume one of two values: 0 or 1. A computer word of 16 bits can indicate 65,536 different numbers; two words of 32 bits can indicate over 4 billion numbers.

boot
To start up an operating system or stand-alone program. Usually, a small program on the beginning of the device reads a larger bootstrap program. The bootstrap program in turn asks which file you want to boot; after receiving a valid answer, it loads that program into memory and yields control to it.

break character
A break or interrupt character is one that lets you stop program execution. DG/RDOS break characters are the control sequences CTRL-C CTRL-A and CTRL-C CTRL-B.

buffer
A part of the computer's memory used to receive and temporarily store disk-based information. The information may or may not be changed while in the buffer. If changed, the new information is usually written back to disk—perhaps to the file from which it came. The DG/RDOS system uses buffers.

Business BASIC
An enhanced, business-oriented version of the BASIC language.

byte
Eight bits, capable of storing one ASCII character (for example, A) or any number from 0 to 255.

channel, I/O
A data structure used by DG/RDOS to identify and communicate with each open file.

CLI (Command Line Interpreter)
The DG/RDOS command language. CLI commands allow people to communicate with DG/RDOS. They provide control, help maintain files, execute other programs (like SPEED), and do many other things. When you start DG/RDOS it runs the CLI on the system console.

CMD key
A key on the terminal keyboard that by itself does nothing, but with other characters can do things like take user terminals off and on line.

COBOL (the COmmon Business Oriented Language)
A very popular programming language for business. It features English language construction, with paragraphs, sentences, and clauses. The COBOL available with DG/RDOS is Interactive COBOL (ICOBOL).

cold start
System startup that begins with computer power off, as compared to a warm start.

COM.CM
A command file, built by the CLI, for use by DG/RDOS and some utility programs. It contains the name of the last non-CLI command typed.

command
In this book, a keyword that tells the CLI, or any DG product, what to do.

compiler
A program that translates statements in a high-level programming language (such as COBOL) into binary instructions for the computer. The RLDR (Relocatable Loader) program then turns the binary instructions into an executable program file, called a save (.SV) file.

CONFIG
A program supplied with DG/RDOS that tailors a DG/RDOS system for certain hardware (USAM, printer) and software.

console
Another name for terminal. The operator's terminal is called the system console in this book. Other terminals are called user terminals.

control key
See *CTRL key.*

CPU (central processing unit)
One of the three sections of a computer system. The others are Input/Output and Main Memory. The CPU decodes and executes program instructions, performs arithmetic and logical operations, and holds critical data.

CR (Carriage Return)
On some keyboards the CR key, rather than NEW LINE, is the CLI command line terminator.

crash
See *panic.*

CRT (Cathode Ray Tube)
A television-like video screen, also called a terminal or screen, that displays characters.

CTRL key
A key, like the SHIFT or CMD key, that does nothing by itself but can do a lot with other keys. CTRL sequences are used to suspend and restore screen display and interrupt commands and programs.

current directory
The directory where you currently are; the name displayed by the GDIR command.

cursor
On a terminal screen, the cursor indicates the current position on a line. The cursor is analagous to the position of a pencil point on paper. It is an underline (a blinking underline on many terminals) or a highlighted box. The cursor is not visible on hard-copy terminals.

data tablet
A large tablet with an electric grid beneath its surface that uses either a cursor puck or a stylus to generate input signals.

database
An information structure (usually kept in one or more files) that a program requires for proper operation.

data-sensitive record
A type of record delimited by a special, agreed-upon character. A standard delimiter is the New Line key. Data-sensitive records are also known as *line-sequential* records.

deadlock
A condition in which a system is frozen; it does not act or respond. Usually you must break a deadlock manually, perhaps by turning power off and on.

default, by default
A value or parameter that a program uses if you do nothing about it. For example, the CLI command LIST displays by default an unsorted list of filenames. But the LIST/S command displays the filenames sorted.

delimiter
A special character that ends each record in a data-sensitive (line-sequential) file. The system treats all characters up to the next special character as a record. A common delimiter is the space between a command and an argument on a CLI command line.

device code
The number by which a computer recognizes an attached device such as a disk controller.

DIP (dual in-line package) switch
A very small switch, often in groups of eight, that enables or disables a hardware function; for example, 8-bit operation on a console. An alternative to DIP switches is hardware (wire) jumpers, but DIP switches are far more convenient. You can change the setting of a DIP switch with a pencil point.

directory
A file whose sole function is to contain other files. Directories can help you organize and keep track of your files; the system itself uses them for this purpose.

directory specifier
See *pathname.*

disk
A fast mass storage device, with metal platters that rotate rapidly. The platters have a magnetic coating that is written to and read from. Disks are frequently referred to as hard disks, to distinguish them from diskettes or floppy disks. Disks may be fixed (Winchester disks) or in a removable disk pack.

diskette
A flexible disk, with the magnetic coating on plastic, ranging in size from 3 to 8 inches.

DKINIT
A DG/RDOS utility program that formats diskettes and disks and checks their surfaces for flaws (bad blocks).

DOS (Disk Operating System)
An early version of DG/RDOS. DOS was replaced by RDOS (Real-time Disk Operating System), and DG/RDOS was developed from RDOS.

dump
In data processing, dump means "to copy." Often the copy is for safekeeping, as for backup.

echo
To confirm a character by displaying it. For example, when you type a character on the keyboard, DG/RDOS reads it and echoes it on the terminal screen.

emulator
A program that enables a terminal or computer system to act like another type of terminal.

fatal error
See *panic.*

file
A collection of information stored under a *filename.*

filename
Names given by you or DG/RDOS to identify files. Filenames can be from 1 to 10 characters long, plus a two-character extension, and can include letters, numbers, and $. You can give files any name you choose with the exception of certain names that are *reserved* by DG/RDOS for system files and devices (such as disk drives).

fixed record (also *fixed-length* or *fixed-sequential record*)
A fixed-length group of characters defined to be a data record. This contrasts with data-sensitive or line-sequential. With fixed records, the system does not look for a delimiter; it considers a certain number of characters as a record. Fixed records can be up to 4,096

characters each.

floating-point unit (FPU)
A printed circuit board that speeds up computations with floating-point numbers (numbers that have a decimal point).

floppy
Another name for a diskette.

foreground console
A second terminal which can run a second CLI and/or other software.

foreground program
A second program that DG/RDOS can run simultaneously with the background program.

form feed
A character (CTRL-L) that tells the printer to stop printing on the current page and start at the top of the next page. Directed to the terminal, a form feed clears the screen.

function key
A key on a terminal keyboard that, either alone or in conjunction with the SHIFT and/or CTRL keys, can represent a command. (Pressing a key is easier than typing a command.) Function keys may or may not have any legends on them, but they are normally referred to as Function Key 1, Function Key 2, and so forth, and function keys on some keyboards are marked F1, F2, etc. Software products that use function keys normally provide a shaped *template* that fits over them.

hang
See *deadlock.*

initialize
Before you can access directories (including devices such as DJ0), they must be initialized with a DIR or INIT command in order to be recognized by the operating system.

ICOBOL
Interactive COBOL, a COBOL that can run on all Data General computers.

I/O (Input/Output)
The process of reading information from a device into the computer's main memory (input) and writing information from memory (output). The input can come from and the output go to disk files, diskettes, a terminal, telephone lines, or microwave beams.

I/O channel
See *channel.*

ISAM (Indexed Sequential Access Method)
This is a file structure offered by ICOBOL.

kilobyte, KB
1,024 bytes (1,024 characters). A 360-KB diskette holds 368,640 characters.

line (communications)
See *asynchronous line.*

line (of text)
A sequence of ASCII characters that ends with either a NEW LINE, form feed, or null character.

line-sequential record
A type of record that ends with a special, agreed-upon character. Line-sequential is another term for data-sensitive records.

link entry
A file created with the LINK command whose sole function is to indicate the location of another file. A link consumes almost no disk space, yet gives easy access to the file. Links are often used in subdirectories to produce access to often used files in the master directory.

loopback mode
A way of testing communications software by having a computer's transmissions come back to it (instead of going to another system). This allows local problems to be identified quickly. For normal operation, loopback mode is disabled. (These options are offered by the CONFIG program.)

macro
A sequence of instructions or commands that can be accessed by a single name. Macros are primarily timesavers, allowing people to write a series of commands only once, then execute them all by one name.

main memory
Memory inside the computer, directly accessible to it.

Megabyte, MB
In terms of computer memory, a megabyte means 1,048,576 bytes (characters); two megabytes of main memory can hold 2,097,152 characters. In terms of disk storage, a megabyte means 1,000,000 bytes.

modem
A device that connects a remote terminal to a computer over a telephone line. One modem is needed at each site. From the remote site, you dial the destination computer's number, wait for a tone, then connect the modem (either via a switch or by inserting the phone receiver into the modem).

monitor
The system console display screen.

mouse
An input device that you move across a flat surface. Movements are translated as coordinates, which a program then uses to move a cursor or draw a picture.

multiplexor
A board that allows a computer system to manage a communications line, user terminals, and a printer. It sorts the incoming signals for the computer, and ensures that the computer's response goes to the correct line.

NEW LINE
The key that terminates CLI commands.

on line
In direct communication with the computer and under its control. For example, when a terminal is on line, the computer reads from the terminal keyboard and writes to its screen. When a terminal is off line, the computer ignores it.

operating system
A large program that manages and operates devices for users and user programs.

OWNER
A part of the BURST backup utility that determines what file owns a disk block, helping you to recover from new bad blocks.

panic
What happens when an operating system encounters a fatal error condition (an error so serious that the system cannot or dares not recover from it). The system console then prints a panic message, consisting of five groups of six digits.

parity
A method used to ensure that a character has been sent and received correctly.

patch
A correction to a program, applied directly to the program file on disk.

pathname or *directory specifier*
A method of accessing a file or directory that is not in your current directory. Each directory you list in a pathname must be *initialized* to make it available to the operating system.

program
A series of instructions, translated into binary codes, that the computer can execute. Text editors, the CLI, and the operating system itself are programs.

prompt
A character or phrase which indicates that a program is ready to receive a command (or more information). For example, the DG/RDOS CLI prompt is the letter **R.**

protocol
A set of conventions between communicating programs that defines the format and sequence of messages to be exchanged.

queue
A file designed to hold printer requests while the printer prints them. See also *spool.*

RDOS
The predecessor of DG/RDOS.

record
A series of one or more characters written to or read from a file.

Release Notice
A notice that names product files and recent software changes that Data General has not yet been able to include in the product manual(s), supplied with DG/RDOS and other software as a printed listing. The Release Notice also explains how to install the product.

secondary console
A second terminal; see *foreground console.*

secondary partition
A user-created directory whose size is fixed at creation and cannot be enlarged; created with the CPART command. A user-created directory that can enlarge is a *subdirectory.*

Software Trouble Report (STR)
A formal report, made by a customer to Data General through a service area or engineer, about a serious problem that the customer is having with the software. The cause may be a user or DG error. Using information supplied by the user, DG personnel try to duplicate the problem so that it can be solved. Or, instead of reporting errors, an STR can simply offer suggestions.

source file
The file that contains the source statements of a program. If the program is written in a compiled language, the source file must be compiled before the program can be run. In BASIC, you can usually just type and run the source file. In a compiled language, the source file is the most important file (more important than compiled versions, which can easily be recreated by the compiler program).

spool, spooling
A method for storing information on disk while it's being printed. Without spooling, if you use the PRINT command, you cannot use your terminal until the printer has finished. DG/RDOS can spool to a printer connected to any device except a multiplexor. (The word "spool" stands for Simultaneous Peripheral Operation On Line.)

stand-alone program
A program that runs without an operating system, relying on its own device handlers. Usually, stand-alone programs are dedicated to a specific function. The DKINIT formatter is a stand-alone program.

STR
See *Software Trouble Report.*

subdirectory
A type of user-created directory that can grow as needed to hold files within it; created by the CDIR command. See also *secondary partition.*

switch
Aside from conventional meaning, a switch is a slash (/) followed by some characters that modify the execution of a command or macro. For example, the /L switch sends the output of the LIST command to the printer instead of the terminal.

synchronous line
A communications line that uses a synchronous protocol to transmit or receive data. Synchronous lines are frequently used in long distance communication between computer systems.

SYS.DR
A file (not a directory, as its name implies) maintained by DG/RDOS to keep track of files within a directory. A copy of SYS.DR is in each directory.

system buffer
See *buffer.*

system console
The terminal connected directly to the computer. User terminals are not connected directly but to the printer port or multiplexor that sorts incoming commands and sends responses to the correct terminal.

tape
A magnetic medium suitable for file backup and mass storage.

template
The word has two meanings. First, a template is part of a filename, used with one of the template characters (- or *) to access one or more files. For example, the template FILE-.- matches all filenames that begin with the characters FILE in the current directory. Second, a template is a cardboard or plastic shape that fits over the topmost group of keys on the keyboard. These keys are called function keys and the template identifies them.

terminal
An interactive device with a keyboard for input and a screen for display. Sometimes called a console. The system console screen is called a monitor.

text editor
A computer program designed specifically to help people write and edit text. A text editor is related to a word processor.

USAM (Universal Synchronous Asynchronous Multiplexor)
An optional printed circuit board on Desktop Generation systems that can manage one or more asynchronous lines. A USAM line can be used to communicate with another system, run a printer or plotter, or run a terminal (which can be connected directly to the computer or work remotely through a *modem).*

user (as in system user)
Anyone who, in any capacity, uses a computer system.

variable
A variable represents a number or character string. The CLI has variables for certain system values. For example, the CLI variable %DATE% holds the current date.

utility, utility program
A program supplied by Data General to help you use the system; for example, the CLI and DKINIT formatter. Some utilities are included with the operating system; others are optional extras.

warm start
A computer system startup in which computer power has stayed on since the operating system was shut down; usually faster than a cold start, since certain essential programs stay active and need not be reloaded.

Index

U

V

W

X

Y

moisten & seal

CUSTOMER DOCUMENTATION COMMENT FORM

Your Name _____ Your Title _____

Company _____

Street _____

City _____ State _____ Zip _____

We wrote this book for you, and we made certain assumptions about who you are and how you would use it. Your comments will help us correct our assumptions and improve the manual. Please take a few minutes to respond. Thank you.

Manual Title _____ Manual No. _____

Who are you?
☐EDP/MIS Manager ☐Analyst/Programmer ☐Other _____
☐Senior Systems Analyst ☐Operator _____
☐Engineer ☐End User

How do you use this manual? *(List in order: 1 = Primary Use)*

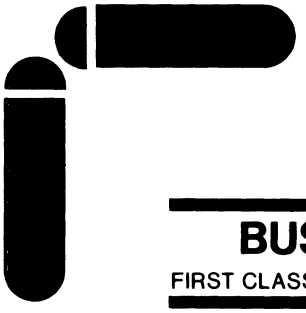
___ Introduction to the product ___ Tutorial Text ___ Other
___ Reference ___ Operating Guide _____

fold

		Yes	No
About the manual:	Is it easy to read?	☐	☐
	Is it easy to understand?	☐	☐
	Are the topics logically organized?	☐	☐
	Is the technical information accurate?	☐	☐
	Can you easily find what you want?	☐	☐
	Does it tell you everything you need to know?	☐	☐
	Do the illustrations help you?	☐	☐

If you wish to order manuals, use the enclosed TIPS Order Form (USA only).

Comments:



NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 26 SOUTHBORO, MA 01772

Postage will be paid by addressee

DataGeneral

Customer Documentation
62 T.W. Alexander Drive
Research Triangle Park, NC 27709-9990

Data General Corporation, Westboro, MA 01580

093-000471-00