Programmer's Utilities Guide

Supplement

RECD WITH 1073-0000-000143

1073-2048-002

Second Edition: February 1987

Introduction

This supplement describes the general-purpose tools provided by Digital Research.. to help driver and application developers write, compile, and maintain programs. The following tools are provided:

In the descriptions that follow, options are shown in square brackets. Generally, you separate options and arguments in the command line with a space. The character | represents "or" and means you can enter either of the arguments shown. Utilities with the -? option have online information available.

Generic Error Messages

The following error messages are generic to most utilities:

Unrecognized option -x

This means you have entered an illegal option for that program; x indicates the illegal option.

Cannot open xxxx

This means the program cannot open one of the files specified; xxxx indicates the file that cannot be opened.

The error messages unique to each tool are presented in the tool description.

BANNER

Form BANNER [-w] string

Explanation Use BANNER to print a title page or pages. Specify the banner contents on the command line as a string. The maximum string length on 80-column paper is 32 characters; the maximum on 132-column is 52 characters. BANNER accepts uppercase and lowercase characters, spaces, and printable special characters.

BANNER prints the string characters in cells 16 lines high by 8 columns wide. Each character is separated by 2 columns. The cells are centered vertically on the page. On 80-column paper, BANNER prints 8 characters per page. On 132-column paper, BANNER prints 13 characters per page.

The default BANNER paper size is 80 columns wide. Use the -w option to output to paper 132 columns wide.

Examples In the following examples, ø represents a space.

banner SORTERøøVersø1.41/8ø2:15

This command prints three 80-column banner pages with the string divided as follows:

Page 1: SORTER
Page 2: Vers 1.4
Page 3: 1/8 2:15

banner -w SORTERØSourceBETAØReleasedØVersion 1.4

This command prints three 132-column banner pages with the string divided as follows:

Page 1: SORTER Source
Page 2: BETA Release
Page 3: Version 1.4

CAT

Form CAT [-s] [-] [filespec] [filespec] ...

Explanation Use CAT to concatenate files. Separate each file specification with a space. The files are output in the order specified to the standard output device. No characters are inserted between the files in the output.

Use the - argument to specify the standard input device as the input file. CAT also assumes the standard input device if you do not enter any file specifications.

Use the -s argument to suppress error display if a file specified is not present.

Note: Remember to specify a new name for the destination file when you concatenate files into a single file. When you use one of the input files as the destination, for example

```
cat file1 file2 > file1
```

you destroy the original contents of file1.

Examples cat src1 src2

Assuming the standard output device is the console, this command displays the files src1 and src2 on the screen.

cat src1 src2 src3 src 4 > dest

This command reads the files src1, src2, src3, and src4 and concatenates them in that order in the file dest.

CMP

Form CMP [-l] | [-s] filespec1 filespec2

Explanation Use CMP to compare binary files. Enter a - in place of the first file specification to compare input from the standard input device with a file

Default CMP output shows the location of each difference by line number and byte offset from the beginning of the file. Nothing is displayed if the files match. If the two files match up to a point, CMP indicates that one file is a subset of the other rather than listing each subsequent line as a difference.

Enter the -l argument to print the conflicting bytes after the line location and offset. Enter -s to suppress the display and generate one of the following exit codes:

0 Files are identical
1 Files have differences
2 Command syntax error

See the BATCH IF command description in the FlexOS User's Guide for the description of your options with respect to the exit code.

Examples

cmp - filex

This command takes input from the standard input device and compares it with the contents of the file filex. Where there are differences, CMP displays the location's line number and byte offset from the beginning of the file. If there are no differences, nothing is displayed.

cmp -s file1 file2

This command compares the contents of file1 and file2. If there are differences, the exit code 1 is returned. If there are no differences, exit code 0 is returned. Otherwise, there is no program output.

Errors

eof reached in xxxx

The file xxxx is a subset of the other file specified.

CUT

Form

CUT -c[list] [filespec]

CUT -flist [-dchar] [-s] [filespec]

Explanation

Use CUT to strip characters on each file line according to column or character specifications. You designate the characters to save by their number or field. If you do not specify a file, CUT uses the standard input device.

The options are defined as follows. You must enter either the -clist or -flist option.

-clist Save characters by column. Enter the columns by number or, using a dash between numbers, by range. Separate each column selection with a comma. The following examples demonstrate column specifications:

- -c1-35,72 saves the characters in columns 1 through 35 and 72.
- -c-25,35- saves the characters in columns 1 through 25 and 35 to the end of line.
- -c8,9,11,12 saves the characters in columns 8, 9, 11, and 12.

Enter -c alone, without a list, to save the characters in columns 1 through 72.

-flist Save characters by field number. Specify the fields by their number in the line. For example, the command -f1,3-5 saves the first and third through fifth field in each line. By default, CUT uses the tab character as the field delimiter. To specify a different delimiter, use the -d option. Lines with no delimiters are saved intact unless you enter the -s option.

-d Specify field delimiter. Enter the character to use as the field delimiter immediately following the option. Space characters and characters with special meaning to the shell must be enclosed in quotes.

-s Strip lines with no delimiter (-f option only). Normally, lines without a delimiter are saved. Use this command to strip them from the output.

Errors

Bad list for c/f option

The -c or -f list specification contains one or more invalid characters.

Line too long

A line in the input file is too long for CUT.

No fields

The -f option specification does not include a field list.

No delimiters

The -d option specification does not specify a character.

DIFF

Form

DIFF [-bb|-b] filespec1 filespec2

DIFF [-?]

Explanation

Use DIFF to compare two ASCII files. DIFF output indicates how to make the files match.

Each difference is called out as either an addition, deletion, or replacement in the form:

(filespec1 line numbers) type (filespec2 line numbers)

where type is the character a (for addition), d (for deletion), or c (for change/replace). The following examples illustrate the three types of output.

n3 a n5,n6	Add filespec2 lines 5 and 6 after filespec1 line 3.
n1,n2 d n3	Delete filespec1 lines 1 and 2.
n1,n2 c n3,n4	Replace filespec1 lines 1 and 2 with filespec2 lines 3 and 4.

After each callout, the lines from each file are listed. Lines preceded with < are from filespec1; lines preceded with > are from filespec2.

The -bb and -b tell DIFF how to handle tabs and spaces. Enter -bb to have blanks and tabs ignored. Enter -b to have strings of blanks and tabs treated as equivalents and trailing blanks ignored.

The DIFF exit code indicates whether or not differences were found as follows:

0 No differences found
1 Differences found
2 Command error

Errors

Out of buffer space

Too many differences were found.

DUMP

Form

DUMP [radix] [display] [+[b]offset,offset] [filespec]

DUMP [-?]

Explanation

Use DUMP to display file contents. You control the radix and the display format with the following options:

Radix		Display	
-o	octal	-b	bytes(default)
-d	decimal	-w	16-bit words
-s	signed decimal	-l	32-bit words
-x	hexadecimal (default)	-c	characters
		-m	mixed -b and -c

Radix and display options are mutually exclusive.

Enter - instead of a file specification to read from the standard input device.

Use the + option to specify a starting and ending offset. Add the -b option to specify the offset in blocks (one block equals 1024 bytes). DIFF interprets the offset entries as byte values when -b is not entered. Specify the offset with a leading zero to indicate an octal value or terminate the offset with a period to indicate a decimal value. Otherwise, DIFF interprets the offset as a hexadecimal value.

Examples

dump -w 23,0xc0 xxxx

This command displays the contents of file xxxx from locations 23H to 0C0H as hexadecimal, word values.

dump -o -b -

This command reads the standard input device and displays the input as octal values in byte form.

DUP

Form DUP count [-num] string

Explanation Use DUP to print a string multiple times. The arguments are defined as follows:

count	The number of lines of string to print
-num	The number of times within a line to print the string
string	The characters to print

DUP interprets count and num as decimal values. The default num is 1. Strings can contain C language conversion specifications in which the arguments set the number of strings per line and the string contents.

Example In the following example, ␣ represents a space.

dup 4 -4 "␣␣␣␣%03d␣%3x" > chart

This command writes a decimal-to-hex conversion chart 4 lines long with 4 entries per line in the file CHART. Each entry begins with 4 blank spaces. The chart appears as follows:

```
000   0    001   1    002   2    003   3
004   4    005   5    006   6    007   7
008   8    009   9    010   A    011   B
012   C    013   D    014   E    015   F
```

Errors **Must specify count**

The duplication count was omitted.

ECHO

Form ECHO [-n] [-q] arguments

Explanation Use ECHO to write the following characters to the standard output device:

\a	bell
\b	backspace
\c	print line without new line
\f	form feed
\n	new line (carriage return and line feed)
\r	carriage return
\s	space
\t	tab
\v	vertical tab
\\	backslash
\---	octal ASCII character code (1 to 3 digits)
\x--	hexadecimal ASCII character code

ECHO also accepts C-like escape conventions. Separate each character in the argument with a space. The arguments are terminated when a new line is received. ECHO automatically includes the new line in the output.

The ECHO options are defined as follows:

-q	Suppress output
-n	Suppress the automatic new line

FGREP

Form FGREP [-n] string [filespec] [filespec] ...

Explanation Use FGREP to search a file or files for a string. Leave out the file specification or enter - to have FGREP search the standard input device. Use quotes to include spaces in the string.

FGREP prints each line in which the string appears. When more than one file is specified, FGREP precedes the line with the file name.

Use the -n option to print the string's line number, relative to the beginning of the file, at the beginning of each line.

The FGREP exit code indicates whether or not matches were found as follows:

0 Matches found
1 No matches found
2 Command error

GREP

Form GREP [options] [expression] [filespec] [filespec] ...

Explanation Use GREP to search a file or files for an expression. Where GREP finds a match, it prints the whole line and, when more than one file is specified, it precedes the line with the file name.

You define the expression with the following characters:

^	Beginning of line
$	End of line
.	Any single character
[xyz]	Character specification: GREP searches on characters listed (here, x, y, or z is a match).
[a-z]	Character range specification: GREP searches on all characters in the range (here, lowercase characters between a and z are a match).
[^xyz]	"Other-than" character specification: GREP searches on all characters besides those listed.
*	Closure: End of a part of the expression (use .* as a wildcard)
\s	Space
\t	Tab
\b	Backspace
\n	New line (carriage return and line feed)
\r	Carriage return

The following options are available:

-c Print only the count of matching lines

-e'exp' Search on the following expression (use when the expression starts with a -). Do not include quotes.

-f'file' Search on the pattern in file. Do not include quotes.

-i Do not distinguish between uppercase and lowercase characters

-l Print only the file names of files with matching lines

-n Precede each line with the line number.

-s Suppress "cannot open file" error message

-v Print only lines that do not have a matching expression

Separate options with a space.

The GREP exit code indicates whether or not the command conditions were met as follows:

0 Matches found
1 No matches found
2 Command error

Example

```
grep -i -n ^..*[a-z0-9]*\t yyy
```

This command searches the file yyy for an expression with the following characteristics:

- starts at the beginning line
- begins with any two characters
- has a letter or number after the two initial characters
- has a tab after the letter or number

Because the -i option is specified, the letter can be either uppercase or lowercase. The number of each line with the match precedes the line in the output.

Errors

Cannot open expression file

Could not open the file in -f specification.

LC2UC

Form LC2UC [file]

Explanation Use LC2UC to translate all characters in lowercase to uppercase. Leave out the file specification to use the standard input device for input.

MAKE

Form MAKE [-f makefile] [-i] [-n] [-s] [names]

Explanation Use MAKE to build programs according to dependencies based on file time-date stamps. MAKE takes its instructions from two files: MAKE.INI and MAKEFILE. MAKE.INI is optional and, if present, is always read first. MAKE searches the current path (see the PATH description) for the MAKE.INI file. MAKEFILE, however, must be in the current directory on the current or designated disk.

MAKE command line options are defined as follows:

-f tells MAKE that the following argument is the name of a makefile to be used instead of the default (MAKEFILE).

-i tells MAKE to continue even if an error is encountered while executing a command.

-n tells MAKE not to execute the commands, but write the ones that should be executed to the standard output. This is useful for creating batch files, for example.

-s tells MAKE not to echo commands. Only text echoed from an invoked program appears on the screen.

names File name or names of the dependencies to test in MAKEFILE. If no names are entered, MAKE tests all of the dependencies.

MAKE.INI and MAKEFILE (or its equivalent) are built from three types of entries: commands, comments, symbol definitions, and dependencies.

A **command** line starts with a tab or space and consists of the command name followed by its arguments. When commands are encountered, MAKE uses the current PATH and ORDER definitions to find the command in the same manner as the shell.

Commands are used in conjunction with the dependency lines to create conditional build instructions in the MAKE.INI and MAKEFILE files. You can enter more than one command after a preceding dependency line.

Command lines can have any combination of the following characters to the left of the command:

@ MAKE does not echo the command line.

- MAKE ignores the exit code of the command, i.e. the ERRORLEVEL. Without this, MAKE terminates when a nonzero exit code is returned.

+ MAKE uses the shell to execute the command. You must use this if the command is a shell built-in command, a batch file, or if you use I/O redirection with < or > or | .

A **symbol definition** has the form

```
SYMBOL = value
```

MAKE replaces all occurances of SYMBOL that follow the definition with the value defined. For example, if the following line is entered:

```
FOO = foo.c, foo.h, foo.inp
```

Make replaces the symbol reference $FOO or $(FOO) in all command and dependency lines with foo.c, foo.h, foo.inp.

To indicate a single "$" character, precede it with another "$". For example, the MAKEFILE command line

```
echo $$test
```

is executed as

```
echo $test
```

Any line starting with a # character is a **comment** line and is ignored by MAKE. MAKE also ignores blank lines.

A **dependency** line has the form

```
target : dependent
```

Both the target and dependent values can be composed of one or more files; separate multiple entries with a space or tab. The dependent value is optional; if none is specified, the command that follows is always executed. Special allowance is made for the colons used in drive specifications. For example, the following line works as intended.

```
c:foo.obj : a:foo.c
```

The command or commands immediately following the dependency are executed only if the time-date stamp of the dependent is more recent than the target.

If any of the target files does not exist, MAKE creates the dependent files. MAKE processes dependencies so that the lowest order files are processed first--MAKE is recursive in this sense. To ensure proper processing order, put the higher order dependencies first in the file. The following dependencies illustrate proper high-to-low order sequencing.

```
foo.286 : foo.obj
        (link command)
foo.obj : foo.c
        (compile command)
```

After the dependent file or files have been processed, MAKE makes the target-dependency date comparison. As above, if any of the dependent are more recent than the target, the next command line is executed.

There is a special sort of dependency line that allows for wildcards in the file names; it has the form:

```
*.obj : *.c
        hc $*
```

This command tells MAKE: whenever a file of the form *.obj needs to be made (for example, foo.obj), then, provided foo.c exists, MAKE creates foo.obj using the command:

```
hc foo
```

As above, MAKE makes the files on the dependent side first if they too need to be made.

The special symbol $* is defined to be the name of the target file without its extension. It is valid only while MAKE is processing a wildcard dependency.

The backslash character "\" also has special meaning; MAKE ignores the backslash and the end-of-line characters (carriage return and line feed) following it so that long lines can be constructed conveniently in the MAKEFILE. The example illustrates use of the backslash.

Example

The following MAKE.INI and MAKEFILE files create a version of MicroEMACS from a set of C files using the Lattice.. C compiler. For brevity sake, the XOBJ group does not include all MicroEMACS object modules.

MAKE.INI

```
# This is the default MAKEFILE for Lattice
# "D" model programs (D means small code, large data)

LC1FLAGS = -i\lc\ -md -s -dLATTICE
LC2FLAGS = -v

# Note: -v causes the code generator to omit stack
# overflow checking.

*.obj : *.c
        lc1 $* $(LC1FLAGS)
        lc2 $* $(LC2FLAGS)

*.obj : *.asm
        masm $*,$*.nul,nul
```

MAKEFILE

```
# Important directories

MLIB =          \mlib\d
HOME =          \xemacs

# Objects generated from files in the HOME directory

XOBJ =          basic.obj \
        buffer.obj \
        cinfo.obj \
        display.obj \
        echo.obj \
        extend.obj \
        file.obj \
        kbd.obj \
        line.obj \
        main.obj \
        fileio.obj \
        spawn.obj \
        ttyio.obj \
        tty.obj \
        ttykbd.obj

# Objects from the MLIB library

MOBJ =          $MLIB\bcopy.obj $MLIB\dosio.obj
        $MLIB\keyboard.obj \$MLIB\video.obj

OBJ =           $(XOBJ) $(MOBJ)

# Link it

xemacs.exe : $(OBJ) xemacs.inp

# All C objects depend on three header files. But it's
# a nuisance recompiling everything for tiny changes.
# This recompiles only when a few terminal-dependent
# things in ttdef.h change.

display.obj tty.obj ttyio.obj ttykbd.obj : ttydef.h
```

PASTE

Forms

PASTE [-dlist] filespec1 filespec2 ...

PASTE [-s] [-dlist] filespec

Explanation

Use PASTE to concatenate files on a line by line basis or to concatenate succeeding lines of one file. Enter a - instead of a file specification to use the standard input device as input.

The default PASTE concatenation option merges the files in columnar fashion into a single file. For example, if the file alpha was composed of the following lines

```
aaa
bbb
ccc
```

and the file numbs was composed of the following lines

```
111
222
333
```

the command

```
paste apha numbs
```

would produce the following output file:

```
aaa     111
bbb     222
ccc     333
```

Unless you specify otherwise, PASTE separates the lines from each file with a tab and puts a new line character at the end of each line.

Use the -dlist option to specify the character used to separate lines. The following escape sequences are available:

\n	New line (carriage return and line feed)
\t	tab
\\	backslash
\0	nothing (no separation)

PASTE uses the characters one at a time and returns to the first character in the list after using the last. For example, the command

```
paste -d\\\n\0 numbs alpha
```

merges the alpha and numbs files shown above as follows:

```
111\aaa
222
bbb
333ccc
```

PASTE cycles through the -dlist entries, returning to the first separater after using the last. For example, if the files alpha and numbs had another line, PASTE would use the backslash to separate the two lines.

Use the -s option to merge all lines in a file into one line. For example, the command

```
paste -s alpha
```

merges the lines in the alpha file shown above as follows:

```
aaa     bbb     cccc
```

PASTE inserts a tab between lines unless you specify otherwise. Use the -dlist option as described above to select different line separaters. PASTE always puts a new line character after the last line merged.

Errors

dlist too long

There were too many characters specified in -dlist.

Bad dlist

The -dlist entry contains an invalid entry.

Too many files specified max is n

There are too many files specified in the command line; n indicates the maximum.

Too many files open max is n

There are too many files specified in the command line than can be opened simultaneously; n indicates the maximum.

Line too long

An output line is too long.

PR

Form PR [options] [filespec] [filespec] ...

Explanation Use PR to print files. Leave out file specifications or enter - to use the standard input device as input. PR withholds error messages until all files are printed when you specify the terminal as the standard output device.

PR default characteristics are as follows:

- Output is printed 66 lines per page with a 5-line header and 5 blank lines at the bottom of the page.
- Page header contains the page number, date and time, and file name.
- Columns are equal width, separated by a space.
- Lines are truncated at 72 character columns

PR provides the following options. You can enter multiple options in a single command. See the examples for the option input syntax.

+k	Start printing from page k.
-k	Print file contents in k columns per page (see examples for explanation)
-ak	Print file lines k per output line (see examples for explanation)
-m	Merge files one per column (as in PASTE above). PR ignores -k and -a arguments when you use -m.
-d	Insert a blank line between lines (doublespace)

-eck Set tabs to column positions. Replace c with the output tab character to be used (tab is the default). Replace k with the number of character positions per tab column (eight is the default).

-nck Print the file with k-digit line numbers (default number of digits is five). PR prints the number in the first tab column of each output column (where -k is more than one) or each line (where -k is one or -m is selected). Replace c with the output tab character to be used (tab is the default).

-wk Truncate lines to k characters.

-ok Offset each line by k characters. The offset is considered a part of the total line length.

-lk Print pages k lines long (default is 66).

-h Print the next argument in the page header.

-p Pause at the end of each page when output device is a terminal. You enter a carriage return to proceed with the next page.

-f Output a form feed at the end of each page (default is a sequence of line feeds). In addition, -f forces PR to pause before outputting the first page when the standard output device is a terminal.

-r Omit error message from output when PR cannot open a file.

-t Omit page header and 5 blank lines at the bottom of the page and stop output at the end of the file. (Otherwise, PR provides blank lines to the end of the final page.)

-sc Do not truncate lines and separate columns with the character c. The -s default character is a tab. Valid entries for c are any characters and escape sequences \\ (backslash) and \0 (no column separation).

Examples In the following examples, the file xxx consists of the following lines:

```
11
22
33
44
55
66
77
88
99
```

PR -3 xxx

This command prints the file xxx three columns per page. The line sequence reads down each column. That is, the lines are printed as follows:

```
11          44          77
22          55          88
33          66          99
```

PR -a3 xxx

This command also prints xxx three columns per page, however, the line sequence is printed across the page. That is, the lines are printed as follows:

```
11          22          33
44          55          66
77          88          99
```

You can specify options in one of two ways. Enter a single - followed by the option letters. For example, the following command prints xxx double-spaced, 3 columns per page, a backslash between columns, and SOURCES added to the page header.

```
PR -3dhs SOURCES xxx
```

Alternatively, enter options individually, separated by a space. The following example prints xxx without page breaks and headers, with tab columns set at every fourth character position, and lines truncated at 48 characters.

```
PR -t -e4 -w48 xxx
```

Errors

Too many open files max is n

Too many files are specified in the command line than can be open simultaneously; n is the maximum.

Too many stdin files max is n

Too many standard input files are specified in the command line; n is the maximum.

Too many columns max is n

The -k specification is too high; n is the maximum.

SPLIT

Form SPLIT -n [filespec] [newname]

Explanation Use SPLIT to break down a file into equal length files. Specify the length of the new files in the -n argument where n indicates the number of lines. Leave out the file specification or enter - to specify the standard input device.

The file name for the new files has two components: the newname entered and a suffix with the characters aa through zz. The newname specification cannot be longer than 6 characters. The name of the first file output is newname appended with aa, the second is newname appended with ab, and so forth to zy and finally zz. If no newname is entered, SPLIT uses x.

Errors

Exceed the maximum of 676 files

The -n specification results in more than 676 new files.

STRINGS

Form STRINGS [-o] [-sn] filespec [filespec] ...

Explanation Use STRINGS to print all printable characters in a file. The strings are output one string per line. Enter -o to output the string's file offset along with the string. Enter -s and a number to offset the output from column 1 by n character positions. STRINGS interprets the number as a hexadecimal value.

SUM

Form SUM [-r] filespec

Explanation Use SUM to generate a file checksum and block count. The block count indicates the number of 1024-byte blocks in the file. SUM generates the checksum using one of two internal algorithms. To use the alternate algorithm, enter the -r option.

Errors

Read error on xxxx

SUM encountered a file system error while reading the input file.

UC2LC

Form UC2LC filespec

Explanation Use LC2UC to translate all characters in uppercase to lowercase. Leave out the file specification to use the standard input device for input.

WC

Form WC [-clwx] filespec [filespec] ...

Explanation Use WC to get the count of characters, lines, and/or words in a file or files. Leave out the file specification to use the standard input device for input. When you enter multiple files, WC outputs the total for each file separately and the total lines of all files.

The WC default adds characters, lines, and words. (A word is defined as any character string terminated by a space, tab, or new line.) To select one or two file characteristics, enter the abbreviation. For example, the following command adds only the characters and lines in file xxx.

```
wc -cl xxx
```

WC prints the total for each characteristic as a decimal value. Use the -x option to print the sum as a hexadecimal value.

End of Supplement