Introduction to the U/Win Utilities

U/Win is a set of utilities for the Windows environment. These utilities provide file management and support functions for software development. The U/Win utilities perform in a manner similar to utilities found in versions of the UNIX operating system.

Most of the U/Win utilities use the concept of standard input and standard output, and can be connected with pipes. Thus, commands can be chained together to perform more complicated tasks.

U/Win can be run in batch or interactive modes. In batch mode, U/Win is executed with command line parameters. In interactive mode, you enter commands through the U/Win dialog windows.

See also

How to Use the U/Win Utilities
Standard I/O and Pipes
Batch and Interactive Modes
Registration

Index to Help for the U/Win Utilities

General Information

The Utilities

How to Use the U/Win Utilities

U/Win may be run interactively by double-clicking on the uwin.exe file in File Manager, or by double-clicking the U/Win icon in Program Manager.

U/Win may be started in batch mode by entering uwin and a U/Win command line in the Run... option of the Program Manager File menu.

In either case, you must specify one or more U/Win utilities to run. In batch mode, the utility name and its parameters are specified after the uwin command:

uwin ls *.c | wc -w > count.txt

would invoke the ls utility to list all files ending with *.c. This list would be passed to the word counting program, which would write a count of the files to the file count.txt.

In interactive mode, you specify U/Win utilities and their parameters through a series of dialog windows.

See also

U/Win Dialogs
Command Format
Standard and Batch Modes

Standard Input and Output

Theory

The U/Win utilities are designed to accept input and deliver output in a consistent manner. Input and output may be specified by arguments to a utility, or as standard input or output through the pipe and redirection operators.

< Input Redirection
The input redirection symbol specifies that the command on the left side of the symbol will read input from the file named on the right side of the symbol.

> Output Redirection
The output redirection symbol specifies that the command on the left side of the symbol will write output to the file named on the right side of the symbol.

| Pipe
The pipe symbol specifies that the command on the left side of the symbol will write output to the input of the command on the right side of the symbol.

Practice

Each of the U/Win utilities require input, and generate output. The input might be names of files to act on, or text to search, or some other data. The output might be a listing or report, or a revised version of the input data. The only exceptions are utilities that perform some action that does not generate output, such as cp, mv, and touch.

The U/Win utilities accept input in the form of filenames following a command. For example:

wc -w foo.c

will count the words in the input file foo.c.

If no filenames are specified after a command, the command will attempt to use standard input. In U/Win, one form of standard input is the output from a previous command, which has been redirected to the command. This is accomplished by linking the commands with a pipe symbol:

ls *.c | wc -w

The pipe symbol informs U/Win that the output of the ls command should not be displayed, but should be sent to the word counting program. The wc utility will count and report the number of words in the listing generated by ls.

Standard output can also be directed to a file:

ls *.c > foo.txt

will create a list of files ending in .c, and will write the listing to the file foo.txt. The file foo.txt could also be used as input to a command:

wc -w < foo.txt

would count the words in the file foo.txt.

The Clipboard

U/Win supports standard input and output through the Windows clipboard, in addition to the standard input/output methods described above. This allows U/Win to exchange data with other Windows applications.

Use **< &c** to redirect standard input from the clipboard (equivalent to the Paste option in many Windows applications). Use **> &c** to redirect standard output to the clipboard (equivalent to the Cut option in many Windows applications). Note that there is a space before the & character.

ls *.h > &c

would create a listing of all files ending in .h, and would write the output to the clipboard. At this point, another Windows application could paste the listing into its own window.

cat < &c

would display the listing in U/Win, reading the information from the clipboard.

Command Format

U/Win utilities are invoked by specifying their names, options, and arguments as complete commands. These commands can be executed in batch mode by passing them to the U/Win program as arguments, or they may be executed interactively by typing them on the command line of the U/Win dialog window. In either case, you construct commands according to the rules below.

Utility Name

Every command begins with the name of a U/Win utility, followed by any options, and then the arguments to the utility. The utility name, options, and arguments are separated by one or more space characters.

pr -nd foo.txt

In this example, pr is the utility. The options specified are n and d. The only argument is foo.txt.

Options

Options are always introduced with a dash character (with one exception, discussed below). If you are specifying several options, you may precede each with a dash, or you may lump all the options together following one dash.

pr -nd foo.txt **pr -n -d foo.txt**

are equivalent. Some options require additional information. These option "sub-arguments" follow their option letter immediately (before any other options). If the sub-argument contains any spaces, it must be enclosed in single or double quotes.

pr -n -h "This is a sub-argument" -d foo.txt

The space between the option letter and its sub-argument is optional. If an option takes a sub-argument, a sub-argument must be specified.

Only one option is not preceded by a dash, and that is the "plus" option.

pr -ndh"This is a sub-argument" +5 foo.txt

Arguments

Arguments to U/Win utilities are almost always file names or path names. In general file names may be a file specifier only (foo.txt), in which case the drive and directory are assumed to be those currently set in U/Win, or may include a full or partial path (d:\temp\foo.txt , ..\temp\foo.txt). These names may also include the conventional DOS wildcards (* and ?). The directory separator character used is the DOS backslash \, not the UNIX-style backslash /. Also, UNIX-style filename patterns including brackets (e.g. ls *.[ch]) are supported. Support for DOS pathnames of the form d:file (drive followed by file name, without a subdirectory specified) varies from utility to utility. The form d:\file is preferred.

NOTE: At this time, the use of '-' to indicate that arguments will come from standard input,

is not supported by U/Win.

Command Piping and Redirection

Multiple U/Win utilities may be executed in series, with the output from one being used by the input of the next. This "piping" of standard output to standard input behaves in a manner similar to that of the UNIX shell environment.

See Also

- Standard Input and Output
- Interactive and Batch Modes

Batch and Interactive Modes

U/Win has two modes of operation. Batch mode lends itself to operations that do not generate usable output, and to those that must be executed routinely with the same arguments. Interactive mode is useful when information must be obtained from a utility, or when arguments and options change frequently.

Interactive Mode

Interactive mode is the typical way of using the U/Win utilities. By double-clicking on the U/Win icon in Program Manager, or by double-clicking the file uwin.exe in File Manager, the program is executed and the main U/Win dialog is displayed.

To run a U/Win utility, type the utility name, options, and arguments in the command line provided in the main dialog window. If you aren't sure of the options or arguments, double-click on the utility name, where it is listed in the Utility list box. A dialog specifically for that utility will be displayed, allowing you to enter the proper parameters. From the utility dialog, press OK to automatically enter the utility name and parameters back into the command line in the main U/Win dialog. Press Cancel if you decide not to use that particular utility.

Once the proper utility name and parameters have been entered into the command line, press OK to execute the utility. The output from the utility will be displayed in the Output window of the U/Win dialog (unless you redirected it to another U/Win utility).

You may continue to enter U/Win commands on the command line, or press the Exit button to leave the U/Win program.

Batch Mode

To execute a U/Win utility without invoking the U/Win dialog box, select the Run... option from the File menu in Program Manager. At the Run command line, enter **uwin** followed by the U/Win utility name and its arguments. When you press the OK button, U/Win will automatically run the utility without displaying any windows. The program will then terminate without any additional action on your part.

See Also

U/Win Dialog

Standard Input

Standard input normally refers to arguments or data passed to a program. The input may come from a file or may even be the output from another program (through the pipe or redirection operators).

Standard Output

Standard Output refers to output from the program. It is normally displayed in the U/Win Output window but may also be redirected to another program or file.

U/Win Dialog

The main U/Win window is composed of a command line window, a program output window, a scrollable list of the individual U/Win utilities, and a list of files in the current directory. In addition, the system menu (accessed by clicking on the symbol in the upper left-hand corner of the window) contains options for changing the default size of the U/Win window.

Command Line Window

The essential feature of the U/Win main dialog window is the command line. U/Win utilities are executed by entering the utility name and parameters in the command line window, and pressing the OK button (or the ENTER key). Commands may also be entered by selecting the utilities and parameters from other controls in the main dialog window (see below). Either method will create a U/Win command line in the command line window.

Command History

U/Win maintains a history of the commands you enter into the Command window To see the list, click the mouse on the arrow symbol next to the window or press the ALT-Down arrow while in the Command window. To activate an old command, highlight it and click it with the mouse or press the down arrow while in the Command window. Repeatedly pressing the down arrow will scroll back through all the commands you entered during your session.

Utility Window

The Utility window contains a list of the U/Win utility programs. When you select one of the utilities by double-clicking on it with the mouse or by selecting it and pressing Enter, a window is presented which will allow you to chose options for that utility.

Redirection Operators

The pushbuttons on the right side of the main window provide an easy way of entering redirection operators. For a discussion of redirection and its use in U/Win refer to the Standard Input and Output topic. Note, however, that the pipe symbol that should be entered into the command line window is the vertical bar symbol '|', not an exclamation point.

Files Window

The Files window allows you to include filenames in the command line window by selecting them from the list. You may also change the current directory or drive from this window, by double-clicking on a drive or directory.

Output Window

Unless the output is redirected to a file or to another program, it will normally be displayed in this window. The window may be scrolled up or down and from side-to-side as needed to view the output. To scroll up or down, press the PgUp or PgDn keys. To scroll right or left,

press ALT-End or ALT-Home respectively.

Related Topics

Command Format
Standard Input and Output
Batch and Interactive Modes

UNIX

UNIX is a multiuser, multitasking operating system. UNIX has always had an extensive library of utility programs. Some of these programs are the model for the U/Win utilities. UNIX is a registered trademark of UNIX System Laboratories, Inc.

Pipe

Multiple U/Win utlities may be executed in series, with the output from one being used as the input to the next. To do this, the pipe operator (' **|** ') is used to separate the programs. For example:

sort my.fil | grep mytext

would sort the file *my.fil* and pass the sorted lines to the grep program, which would look for lines containing the *mytext* string.

cat

Concatenate files.

Syntax

cat [*file1* [*file2 ...*]]

Description

cat reads lines from the specified file or files, and writes them unchanged to standard output. If no files are specified, cat reads from standard input.

cd

Change directory.

Syntax

cd *pathname*

Description

cd changes the current drive and directory to that specified in the pathname.

cmp

Perform a byte-by-byte comparison of two files.

Syntax

cmp [-ls] *filename1 filename2* [*skip1*] [*skip2*]

Description

The **cmp** program compares the contents of *filename1* and *filename2* on a byte-by-byte basis. If either filename is a '-' then the standard input is used.

If no options are given, **cmp** will print the message 'Files are identical' if the two files compare equally or, if they differ, **cmp** will display the byte number (offset) at which the difference occurred.

The *skip1* and *skip2* options specify an initial byte offset into *filename1* or *filename2* respectively. If either option begins with a '0x', **cmp** assumes the number is a hexadecimal number.

-l specifies that cmp should print the byte number and the differing bytes (in hexadecimal) whenever a difference is encountered.

-s indicates that cmp should not print anything when files compare equally.

cp

Copy files and directories.

Syntax

cp *file1 file2*
cp *file1* [*file2 ...*] *dir*
cp *dir1* [*file1 ...*] *dir2*

Description

cp copies a source file to a target file, or multiple source files and/or directories to a target directory. If the target file or directory does not exist, it is created. Multiple files may not be copied to a target file, nor may directories be copied to a file. If the target is a directory, multiple source files and/or directories may be specified. Conversely, if multiple sources are specified, or if any of the sources are directories, cp assumes that the target must be a directory.

When copying a directory, all files and (recursively) all subdirectories in the source directory will be copied to a subdirectory in the target directory. This subdirectory will have the same name as the source directory.

Examples

Copy bar.txt to c:\newdir\bar.txt:

cp c:\foo\bar.txt c:\newdir

Copy the files and subdirectories in c:\foo into directory d:\newdir:

cp c:\foo*.* d:\newdir

Copy the files and subdirectories in c:\foo into directory d:\newdir\foo:

cp c:\foo d:\newdir

Notes

cp does not support standard input or standard output.
Unlike the DOS copy command, a target must always be specified. Also, a drive name alone is not a valid target. Instead of d:, use d:\.

cut

Cut selected fields from each line of a file.

Syntax

cut -c *list* [*file1* [*file2* ...]]
cut -f *list* [-d *char*] [-s] [*file1* [*file2* ...]]

Description

cut reads the specified files and outputs the selected fields from each line. Fields may be defined by absolute character positions or by a field delimiter. If no files are specified, cut selects fields from the lines in standard input.

A *list* is a set of integers or integer ranges separated by commas. The '-' character indicates a range. These integers represent field numbers (the first field is 1). Field numbers must be specified in ascending order. Example: 1,4,6-9,11 specifies fields 1, 4, 6, 7, 8, 9, and 11.

-c indicates that the fields specified in *list* are character positions. In the output, no delimiters will appear between the selected character fields.

-f indicates that the fields specified in *list* are separated by a delimiter in the file. The default delimiter is the tab character. In the output, the field delimiter will appear between the selected fields. By default, lines containing no field delimiters are output in their entirety.

-d specifies a delimiter character for fields selected with the -f option. If a space or other special character is used as a delimiter, it must be quoted. If the -d option is not included, -f will assume that the tab character is the delimiter.

-s suppresses lines containing no field delimiters, when the -f option is used.

Notes

The -c and -f options are exclusive. Either the -c or the -f option must be specified.

df

Report on the amount of free disk space available.

Syntax

df [-at] *disk ...*

Description

df displays the amount of used and available disk space on the specified disk drives, as well as what percentage of the disk's total capacity has been used. The results are in 1,000's of bytes.

If no options are specified, **df** produces a report like this:

DRIVE	KBYTES	USED	AVAIL	%
c:	33462	33250	212	99

-a specifies that **df** should report on all fixed disks in the system.

-**t** specifies that **df** should also report on the total free space available on all the selected drives.

diff

Compare two files line-by-line and display the differences.

Syntax

diff [-bitw] [-D*string*] *filename1 filename2*

Description

The **diff** program compares two text files and reports on what lines differ between the two files. If either *filename1* or *filename2* is a '-' then the standard input is used. If *filename1* is a directory, a file in that directory whose filename is the same as *filename2* is used (or vice versa).

Output consists of lines from *filename1* flagged by '<' followed by lines from *filename2* flagged by '>'.

-b specifies that trailing blanks should be ignored.

-i indicates that upper- and lower-case letters should be treated equally when comparing lines.

-t expands TAB characters in the output lines.

-w specifies that **diff** should ignore **ALL** blanks (SPACE and TAB characters). The input line 'if (x == y)' will be equal to 'if(x==y)' when -w is specified.

-D*string* creates a merged version of the input files on the standard output with 'C' preprocessor commands included. The commands are placed such that compiling the result without defining *string* is the same as compiling *filename1*. If *string* is defined, then compiling the result will be the same as compiling *filename2*.

Notes

The **-D** option ignores any existing preprocessor commands in the input files and may produce commands with overlapping scope.

Future versions of **diff** will include options to produce scripts similar to UNIX 'ed' scripts.

du

Display the number of bytes used by directories or files.

Syntax

du [-s | -a] [*filename ...*] [*directory ...*]

Description

The **du** program displays the number of bytes (in 1,000's) within each specified *filename* or *directory*. If *filename* or *directory* is not specified then **du** uses the current directory.

-s specifies that du should only display the total bytes used by the *directories* specified.

-a indicates that an entry should be created for each file.

find

Find files by name or by characteristics.

Syntax

find [*pathname ...*] [(*expression*)]

Description

The **find** program searches for filenames which match the *filename* argument. Find searches recursively through any subdirectories it encounters.

If a *pathname* is specified it must precede the search *expression*. The *pathname* can be a drive letter. For example, the argument 'c:' would search all of drive 'c' for the specified *expression*. The argument 'c:\utility\stuff' would begin the search in that directory and search any subdirectories which might be found beneath it.

There are several options for **find**:

-name *filename* specifies the name of the file to search for. The *filename* argument may contain legal DOS filename 'masks' such as '*' and '?'.

-size [+|-]*n* specifies that **find** should only report on files which are equal to the specified size (*n*). Preceding **n** with a **'+'** specifies files larger than **n** and **'-'** specifies files that are smaller than **n**.

-mtime [+|-]*n* indicates that **find** should only report on files which have been modified in the last *n* days. Preceding **n** with a **'+'** specifies files larger than **n** and **'-'** specifies files that are smaller than **n**.

-newer *filename* specifies that find should report on files which have been modified more recently than the specified *filename*.

The **(*expression*)** option is a powerful feature which allows you to specify search arguments to **find.** Only files which match the parameters in the (*expression*) argument will be reported on. The argument uses two options, '**-a**' and '**-o**'. '**-a**' indicates that the filename should be reported if both components of (*expression*) are true. '**-o**' indicates that the filename should be reported when one component or the other of the (*expression*) argument is true.

Examples

Report on any files which match the '*.c' **OR** '*.h' filename masks:

find (-name *.c -o -name *.h)

Report on all '*.c' files which have modified or created within the last 10 ten days **AND** which are more than 100000 bytes in size. The search should begin in the 'c:\cfiles' directory:

find c:\cfiles -name *.c (-mtime -10 -a -size +100000)

Display any '*.h' filenames which have been modified or created since the 'c:\include\stuff.h' **OR** the 'd:\moreincs\stuff.h' files were last modified:

find -name *.h (-newer c:\include\stuff.h -o -newer d:\moreincs\stuff.h)

grep

Search files for a character string.

Syntax

grep [-chilnov] [-e '*search string*'] [*search string*] [*filename ...*]

Description

grep searches the input *filenames,* or the standard input, for lines which match the search string. If the search string is found in a line, the line is printed to the standard output. Most of the options for grep pertain to it's output format. By default, the filename where the lines were found is printed followed by copies of the matching lines.

-c displays a count of matching lines instead of displaying the lines which match.

-e indicates that the argument is a literal character string. Useful if the string contains a '-' or other special character. You must use this option if the search string contains any of the regular expression characters.

-h suppresses the display of file names.

-i tells grep to ignore case when searching for the search string. Upper and lower case letters will be considered equal.

-l displays only the filenames of files which contain matching lines. The matching lines are not displayed.

-n prints the line number of the matching line in addition to the text of the line.

-o specifies an output format similar to the standard UNIX format - that is, the filename is displayed at the start of each matching line. This option is nullified if the **-n** option is used since line numbers will be displayed at the beginning of each line in that case.

-v displays only lines that *do not* match.

Examples

Search a file for a fixed string:

grep findit test.txt

Search all 'txt' files in the current directory for a literal string:

grep -e 'find-it' *.txt

Search all 'c' and 'h' files for the string 'find-it'. Display only the names of files containing this string:

grep -l -e 'find-it' *.c *.h

Notes

Be careful when using the characters '|', '>', '<', and '\' in the search string since these characters have meaning within the command line. If you include these characters or any of the regular expression characters you must enclose the search string in single

quotes '....'.

head

Display the beginning of a file.

Syntax

head [-*count*] [*file1* [*file2 ...*]]

Description

head outputs the first *count* lines of each file specified. If multiple files are specified, head precedes each listing with the file name. If no files are specified, head reads from the standard input. If *count* is not specified, head outputs 10 lines per file.

lpr

Print files.

Syntax

lpr [-b *banner*] [-P *printer*] [*file1* [*file2 ...*]]

Description

lpr prints the specified file or files. If no files are specified, lpr prints the standard input.

-P *printer* specifies an alternate printer for lpr to print to. *Printer* is the printer name as it appears in Control Panel (e.g. Epson 9-pin or PostScript Printer). If *printer* contains spaces, it must be enclosed in double quotes. If the P option is not specified, lpr prints to the Windows default printer.

-b *banner* specifies that a banner page should be printed before the files, and that the title *banner* should be printed on it. If *banner* contains spaces, it must be enclosed in double quotes.

Notes

lpr does not support standard output.
lpr uses the Windows Print Manager if it is enabled.

ls

Lists the contents of a directory.

Syntax

ls [-alp] [*path1* [*path2 ...*]]

Description

ls lists the files and subdirectories present in the specified pathnames. If a pathname is a directory name with no file specifier, ls assumes that all files and subdirectories in the directory should be listed. If a pathname includes only a file specifier, ls assumes that only matching files and subdirectories in the current directory should be listed. If no pathnames are specified, ls assumes that all files and subdirectories in the current directory should be listed.

-a specifies that ls should include hidden files in its listing. By default, ls lists only ordinary files, read-only files, and subdirectories. The '.' and '..' directory entries, as well as files with the DOS system attribute are never listed.

-l specifies that a long listing be created. The long listing contains file attributes, size in bytes, access date and time, and file name. By default, ls only includes file names in the listing.

-p specifies that the file and subdirectory names in the listing be output as full pathnames, including the drive letter.

Notes

ls requires a slash character after a drive specification (a:\ is acceptable, but a: is not). ls does not support standard input.

mv

Move files and directories.

Syntax

mv *file1 file2*
mv *file1* [*file2 ...*] *dir*
mv *dir1* [*file1 ...*] *dir2*

Description

mv copies a source file to a target file, or multiple source files and/or directories to a target directory, and deletes the source files/directories. If the target file or directory does not exist, it is created. Multiple files may not be moved to a target file, nor may directories be moved to a file. If the target is a directory, multiple source files and/or directories may be specified. Conversely, if multiple sources are specified, or if any of the sources are directories, mv assumes that the target must be a directory.

When moving a directory, all files and (recursively) all subdirectories in the source directory will be moved to a subdirectory in the target directory. This subdirectory will have the same name as the source directory.

Examples

Move bar.txt to c:\newdir\bar.txt:

mv c:\foo\bar.txt c:\newdir

Move the files and subdirectories in c:\foo into directory d:\newdir:

mv c:\foo*.* d:\newdir

Move the files and subdirectories in c:\foo into directory d:\newdir\foo:

mv c:\foo d:\newdir

Notes

mv does not support standard input or standard output.
Unlike the DOS copy command, a target must always be specified. Also, a drive name alone is not a valid target. Instead of d:, use d:\.

od

Output dump.

Syntax

od [-cSx] [+ *offset*] [*file1* [*file2 ...*]]

Description

od reads the specified files and writes a dump of them to standard output. By default, the dump is output in "split" hexadecimal and character format. If no files are specified, lpr prints the standard input.

-c specifies that the dump should be formatted in lines of 24 characters each. Each character represents one byte in the file. If a data byte cannot be represented as a displayable character, the period ('.') is output. Characters are separated by two spaces.

-x specifies that the dump should be formatted in lines of 24 hex values each. Each value represents one byte in the file, and is separated from the next value by a space.

-S specifies that the dump should be formatted in lines of 16 hex values and 16 characters each. The hex vaules are output first in each line, followed by the character representation of the same data. This is the default.

+ *offset* specifies that the dump should begin at the indicated *offset* in bytes from the beginning of the file. If multiple files are specifed, the offset applies to all files. If no files are specified, the offset applies to the standard input.

Notes

The -c and -x options may be combined to produce alternating lines of hexadecimal and character output (two lines of output represent one 24 byte line of data). The -S option is exclusive of the -c and -x options, and takes precedence over them if both -S and -c, or -S and -x are specified.

pr

Format files for printing.

Syntax

pr [-dnt] [-e *tablen*] [-h *header*] [-l *length*] [-o *offset*] [+ *page*] [*file1* [*file2 ...*]]

Description

pr formats the specified files in a form suitable for printing. If no files are specified, pr formats the standard input. If no options are specified, the following format defaults are in effect:

Output is single spaced, 66 lines per page.
No indenting or line numbering.
Tabs are expanded to eight character positions.
Each page has a header consisting of two blank lines, a title, and two additional blank lines. The title consists of the page number, file name, and file access date and time. If standard input is being formatted, the file date and time are replaced with the system date and time.
Each page also has a footer consisting of five blank lines.
Output begins with the first page formatted.

-d specifies that the output should be double spaced.

-n specifies that each output line will be consecutively numbered. Line numbers appear on the left side of each line, and are up to six digits long (blank-padded if less).

-t specifies that page headers and footers will be omitted from the output. The form-feed that normally follows the last file formatted will be suppressed. This option overrides the -h option, if specified.

-e *tablen* specifies that tab characters will be expanded to *expansion* character positions.

-h *header* specifies information that will appear in the title line of each page header, in place of the file name, date, and time. If *header* contains spaces, it must be enclosed in double quotes.

-l *length* specifies that *length* lines of output will appear on each page.

-o *offset* specifies that each line of output will be indented *offset* character positions.

+ *page* specifies that the output will begin with the *page* formatted. For example, + 2 would skip the first page of output.

Notes

pr has an input line-length limit of 127 characters. Longer lines will be broken into multiple shorter lines. Also, since pr has no knowledge of the page width of the printer that its output might be sent to, no word-wrap or truncation is performed.

Registration

U/Win is being distributed as Shareware. This allows you to try the program before paying for it. If you find the program useful, please register your copy. The suggested trial period is 21 days. Registration is accomplished by sending the registration form (click on the words registration form to access the form) together with the purchase price to:

The Boolean Group, Inc.
3715 Hampton Blvd.
Royal Oak, Michigan 48073-2105

Problems and suggestions may be submitted on the registration form or through Compuserve mail to.72077,506 or 76370,3353.

License

U/Win is not and has never been public domain software, nor is it free software. The Boolean Group, Inc. retains the copyright and all related privileges to the U/Win program.

Non-licensed users are granted a limited license to use U/Win on a 21-day trial basis for the purpose of determining whether U/Win is suitable for their needs. The use of U/Win, except for the initial 21-day trial, requires registration. The use of unlicensed copies of U/Win by any person, business, corporation, government agency or any other entity is strictly prohibited.

A single user license permits a user to use U/Win only on a single computer. Licensed users may use the program on different computers, but may not use the program on more than one computer at the same time.

No one may modify or patch the U/Win executable files in any way, including but not limited to decompiling, disassembling, or otherwise reverse engineering the program.

A limited license is granted to copy and distribute U/Win only for the trial use of others, subject to the above limitations, and also the following:

1) U/Win must be copied in unmodified form, complete with the file containing this license information.

2) The full machine-readable U/Win documentation must be included with each copy.

3) U/Win may not be distributed in conjunction with any other product without a specific license to do so from The Boolean Group, Inc.

4) No fee, charge, or other compensation may be requested or accepted, except as authorized below:

A) Operators of electronic bulletin board systems (sysops) may make U/Win available for downloading only as long as the above conditions are met. An overall or time-dependent charge for the use of the bulletin board system is permitted as long as there is not a specific charge for the download of U/Win.

B) Vendors of user-supported or shareware software approved by the Software Publishers Association (SPA) may distribute U/Win, subject to the above conditions, without specific permission. Non-approved vendors may distribute U/Win only after

obtaining written permission from The Boolean Group, Inc.. Such permission is usually granted. Please write for details (enclose your catalog). Vendors may charge a disk duplication and handling fee, which may not exceed eight dollars.

Warranty

The Boolean Group, Inc. guarantees your satisfaction with this product for a period of 30 days from the date of original purchase. If you are unsatisfied with U/Win within that time period, return the package in saleable condition to the place of purchase for a full refund.

The Boolean Group, Inc. warrants that this software will perform in substantial compliance with the documentation supplied. If a significant defect in the product is found, the Purchaser may return the product for a refund. In no event will such a refund exceed the purchase price of the product.

EXCEPT AS PROVIDED ABOVE, THE BOOLEAN GROUP, INC. DISCLAIMS ALL WARRANTIES, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, WITH RESPECT TO THE PRODUCT. SHOULD THE PROGRAM PROVE DEFECTIVE, THE PURCHASER ASSUMES THE RISK OF PAYING THE ENTIRE COST OF ALL NECESSARY SERVICING, REPAIR, OR CORRECTION AND ANY INCIDENTAL OR CONSEQUENTIAL DAMAGES. IN NO EVENT WILL THE BOOLEAN GROUP, INC. BE LIABLE FOR ANY DAMAGES WHATSOEVER (INCLUDING WITHOUT LIMITATION DAMAGES FOR LOSS OF BUSINESS PROFITS, BUSINESS INTERRUPTION, LOSS OF BUSINESS INFORMATION AND THE LIKE) ARISING OUT OF THE USE OR THE INABILITY TO USE THIS PRODUCT EVEN IF THE BOOLEAN GROUP, INC. HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

Use of this product for any period of time constitutes your acceptance of this agreement and subjects you to its contents.

Select File Print Topic to print this form.

The Boolean Group, Inc.
3715 Hampton Blvd.
Royal Oak, Michigan 48073-2105

U/Win Version 1.0
Registration Form

Name: _____

Company: _____

Address: _____

City: _____ St: _____ Zip: _____

Phone: (____) _____ Country: _____

Disksize (circle one) 5.25" 3.5"

_____ copies @ $29.95 ea.: _____.____

Michigan residents add 4% sales tax: _____.____

Foreign air shipping (except Canada) @ $9.50: _____.____

Total: _____.____

Please enclose a check or money order payable to **The Boolean Group, Inc.**

Send to: **The Boolean Group, Inc.**
3715 Hampton Blvd.
Royal Oak, MI 48073-2105

Please allow 2 to 4 weeks for delivery.

How did you obtain this copy of U/Win?

Comments (likes/dislikes/problems/suggestions):

Regular Expressions

Regular expression search strings are powerful commands for locating occurrences of text patterns within files. This section will explain the various regular expression operators and provide examples of their use. These expressions can be used with the grep utility.

Wildcards

The simplest search patterns are alphanumeric strings such as 'the lazy brown fox'. You may also include special wildcards in your search patterns. For example:

^	Match only if the search pattern if found at the beginning of an input line.
*	Match zero or more repetitions of the preceding character.
.	Match any character except newline.
[*chars*]	Match any of the enclosed chars. You may specify a range of characters or digits by using the '-' character.
c	Disregard the special meaning of the character *c*. This applies to '^', '$', '[', ']', '*', '|', '~', '!', '@', '#', and '.'.

Metacharacters

|	When used to separate two search patterns this character functions as an 'OR' operator. In other words, there is a match whenever one search pattern OR the other is found.
+	Similar to the '*' wildcard character discussed above except that it matches one or more occurrences of the preceding character.
?	Match zero or more occurrences of the preceding character.
(...)	Use parenthesis to group your search patterns. For instance, (HI)+ would match one or more of the characters 'H' or 'I'.

Examples

Search for characters at the beginning of the line:

grep '^this is my search string' *.txt

Search for certain characters:

grep '[1-9] [CTX]' *.txt

Search for alternative patterns:

grep '(strcpy | strncpy) (division | company)' order.c

rm

Remove files and directories.

Syntax

rm [*file...* | *dir ...*]
rm [-r] *dir* ...

Description

rm removes the specified files and/or directories. By default, directories are removed only if they are empty.

-r causes any directories specified to be removed, even if they contain files or subdirectories. Files and subdirectories are removed recursively from each target directory. The -r option, when specified, applies to all directories in the command line.

Examples

Remove bar.txt:

rm c:\foo\bar.txt

Remove all files and (empty) subdirectories in c:\foo:

rm c:\foo*.*

Remove directory c:\foo and all of the files and subdirectories in c:\foo:

rm c:\foo

Notes

The root directory cannot be removed.
Removing the current directory will cause the current directory to be changed to the parent directory of the current directory.
Read-only files are removed without warning.
rm does not support standard input or standard output.

sort

Sort lines.

Syntax

sort [-bcfru] [-t *c*] [*sort-field ...*] [-bfr] [-o *output file*] [*filename ...*]

Description

The **sort** program sorts and merges lines contained in the named files. Output is written to the standard output or, optionally, to the file named as an argument to the **-o** option. **Sort** can also accept input lines from the standard input.

Lines are normally sorted character-by-character from left to right using the ASCII character set as the collating sequence. Unless you specify otherwise (by using the **-b** argument) leading spaces are considered significant. The lines:

abc
abc

are collated as:

abc
abc

You may also specify starting and ending positions, or fields, within each line. This is accomplished by using the *+sw* (start-word), *-ew* (end-word), and **-t***c* (word delimiter) options described below. If you do not specify a word delimiter (a character which separates fields within the line) then one or more white-space characters (SPACE) signify the end of one field and the start of another.

The sort fields are evaluated in the order they appear on the command line. The next field is checked only when all earlier fields compare equally. If all the fields compare equally, the lines are considered equal.

-b specifies that leading SPACE characters should not be considered when comparing lines. When used with a field specification, it indicates that leading SPACE characters should be ignored when determining the starting position of a field.

-c indicates that **sort** should check the input file according to the parameters specified on the command line. If the file is sorted correctly, **sort** will print a message to that effect. If the file is not sorted correctly, an error message will be printed.

-f (fold to lower case) specifies that upper- and lower-case letters should be treated equally when comparing lines.

-r indicates that the order of the sort should be reversed and lines should be ordered from most-to-least rather than from least-to-most.

-u causes **sort** to output only the first line in each set of lines which compare equally.

-o specifies an output file to use rather than the standard output.

Field Specification Options

-t *c* specifies the character to use as a field delimiter. By default, **sort** uses white-space

as the field delimiter. You may, however, specify any character as the field delimiter. Enter the character following the **-t** option (leave a space between the option and the argument).

sort-field specifies a combination of options that together define a field to sort on. The field is part of the input line and can be defined in either of the following ways:

+*sw* [-bfr]
+*sw* -*ew* [-bfr]

The *sw* parameter is the number of the first word (the first word is '0') to include and *ew* is the number of the last word to include in the field. The *-ew* parameter is optional and, if left out, the field continues to the end of the line. The **-b**, **-f**, and **-r** parameters have the same meaning as above, however, they apply only to this sort field. If they are omitted, the parameters specified separately apply to the entire line. If included, they override any other parameters for this sort field only.

A character offset may also be specified within these parameters. The offset indicates that a field will start or end that many characters into the field (the first character is '0'). For instance, *+w.c* specifies that the character in position *c* within word *w* is the start of the field. In addition, *-w.c* indicates that the end of the field comes just before character *c* in word *w*.

Examples

Sort the input file in descending order:

sort -r input.txt

Sort the contents of input.txt using the first non-blank character of the second word as the key:

sort +1.0 -1.1 -b input.txt

Use the 2nd and 3rd words as one field and a portion of the 5th word as the second field. The sort should be in descending order:

sort -r +1.0 -3.0 +4.2 -4.5 input.txt

Sort and merge all 'txt' files in the current directory into one file:

sort -o output.txt *.txt

Sort lines using the third word as the key. The words are separated by *'s:

sort -t * +2.0 -3.0 input.txt

Notes

This version of **sort** accepts lines as long as 256 characters. Longer lines will be truncated. Future versions will allow you to specify a line length.

tail

Display the end of a file.

Syntax

tail [+ | -*count*] [-l | -b | -c] [-f] [*file1* [*file2 ...*]]

Description

tail reads the specified file or files, and displays the last part of each. If no files are specified, tail reads from standard input. The data displayed for each file depends on the *count* given. The *count* is specified in lines by default. If no *count* is given, the value 10 is assumed. The *count* may start from the beginning of the file (+*count*) or from the end of the file (-*count*).

-l specifies that the data count should be in lines.

-b specifies that the data count should be in 1024-byte blocks.

-c specifies that the data count should be in characters.

-f causes tail to display the end of a file repetitively, until cancelled by the user. This allows the growth of a file to be monitored. When -f is specified, tail will repeat its operation every two seconds until cancelled.

Examples

Display the last ten lines of the file foo.txt:

tail foo.txt

Display all lines in foo.txt after the 25th line of the file:

tail +25 foo.txt

Display the last 40 characters of foo.txt on a repeating basis:

tail -40 -cf foo.txt

touch

Update the access date and time for a file.

Syntax

touch *file1* [*file2 ...*]

Description

touch sets the access date and time of the specified files to the current date and time.

Notes

touch does not support standard input or standard output.

uniq

Remove or display duplicate lines.

Syntax

uniq [-cdu] [+n -n] [*input file*] [*output file*]

Description

uniq compares adjacent lines in the input file (or the standard input). By default, only the first instance of a line is kept, the second and succeeding repeated lines are removed. Lines which are kept are written to the output file (or the standard output). Only adjacent lines are removed.

Lines are compared character-by-character from left to right. Leading spaces are considered significant. The lines:

abc
abc

are not considered equal due to the leading spaces in the second line.

The following options can be used with **uniq**:

-c specifies that each output line should be preceded with a count of the number of times it occurred in the file.

-d writes one copy of just the repeated lines.

-u displays only the lines which are not repeated in the input file.

-n specifies that the first ***n*** fields (words separated by white-space characters) should be ignored when performing the line comparisons.

+n indicates that the first ***n*** characters should be ignored. If the ***-n*** parameter is also present, the fields will be skipped and then the characters will be skipped.

Notes

This version of **uniq** accepts lines as long as 256 characters. Longer lines will be truncated.

wc

Count words in a file.

Syntax

wc [-clw] [*file1* [*file2 ...*]]

Description

wc counts items in the specified files. If no files are specified, wc counts items in the standard input. By default, wc counts the characters, words, and lines in each file. If more than one file is specified, wc outputs a total after the last file count.

-c specifies that the individual characters in the file will be counted.

-l specifies that the number of lines (ending with CR/LF) in the file will be counted.

-w specifies that the number of words (tokens delimited by spaces, tabs, and newlines) in the file will be counted.