Software skills for librarians

Module 1: The Unix shell
Reference material

Filing system commands

`ls` *<pathname>*
List the files in the directory specified by pathname. Alternatively, if you specify a filename with wildcards, then the matching files in the current working directory are shown instead.

Options: `-l` Long format: show the creation date and size of each file too.
`-a` All files: show hidden files, whose names begin with a dot.

`cp` *<source>* *<destination>*
Copy the specifed file from source to destination. The source file name can include wildcards, in which case the destination should be a directory.

Options: `-r` Recursively copy directories and their contents.

`mv` *<source>* *<destination>*
Move the specified files from source to destination. This command works like the copy command, but deletes the source files afterwards.

Options: `-i` Inquire: ask before overwriting any destination files.
`-v` Verbose: print the names of files as they are moved.

`rm` *<filename>*
Remove the specifed files, where wildcards are accepted as part of the filename. Note that there is no way of recovering the files after a successful deletion.

Options: `-r` Recursively remove directories and subdirectories.
`-f` Force: remove files even if they are write protected.

`mkdir` *<dir-name>*
Creates a new directory with the specified name in the current working directory.

Examining files

`wc` *<filename>*
Counts the words in the file specified by filename. Multiple files can be specified, or if no names are given read the standard input.

Options: `-c` Count only the number of characters.
`-l` Count only the number of lines.

`cat` *<filename>*
Prints the contents of the specified files to the screen. If this command is used with output redirection it can be used to concatenate several files into one.

Options: `-v` Also display non-printing characters
`-e` Display newlines too, using $
`-t` Display tabs too, using ^I
`-n` Show line numbers.

`sort` *<filename>*
Sorts the lines of the specified file into order. Several input files may be specified, and by default the output is written to the screen.

Options: `-d` Use dictionary order, that is ignore punctuation characters.
`-e` Ignore case.
`-r` Sort in reverse order.

`head` *<filename>*
Prints the first few lines of the specified files. If the number of lines is not specified it defaults to 10. By default the output is written to the screen.

Options: `-n` Prints the first n lines of each file.

`tail` *<filename>*
Like the head command, but prints the first few lines of the specified files.

Options: `-n` Prints the last n lines of each file.

`hexdump` *<filename>*
Displays the raw data in a file as single byte hexadecimal codes.

Options: `-x` Displays the file in a two byte hexadecimal format.

`diff` *<filename1> <filename2>*
Compares the two files and displays only the differences between them.

Options: `-a` Treat the two files as text regardless of their content.
`-B` Ignore blank lines.

Automated search and replace

`grep` *<expression> <filename>*
Searches the specified files line by line for the pattern given by a regular expression and prints any which match.

Options: `-C` Only count matching lines.
`-E` Use extended regular expressions.
`-i` Ignore case when matching.
`-n` Print line numbers.
`-v` Inverse: print those lines which don't match.
`-w` Match only entire words.

`sed` *<command> <filename>*
Reads the specified files line by line and applies any changes specified by the editing command. The modified text is normally written back to the standard output. The command is often used to provide global search and replace facilities.

Options: `-e` The next argument is an editing command.
`-i` Edit files in place, overwriting the original.
`-r` Use extended regular expressions.

`tr` *<string1> <string2>*
Translate the characters from the first string into the corresponding ones in the second string. Data is read from standard input and written to standard output. Each string can also specify a range of characters like A-Z, or a character class like [:lower:].

Options: `-d` Delete the characters specified by the first string.

Filename wildcards and redirection

When specifying filenames, the shell allows the use of wildcards to specify a number of matching filenames. Typically the name with wildcards is expanded into a list of all matching files before being passed on to the command. The permitted wildcards are:

* Matches any number of any character.
? Matches any single character.
[...] Matches any one of the enclosed characters.

Filenames on Unix systems can include any character except for a slash '/' which separates directory names. If a filename includes a space you must enclose the name in quotation marks. Additionally, if you use double quotes, then characters preceeded by a backslash have a special meaning.

By default, the standard input is the keyboard, and the standard output is to the screen. The following redirection options apply:

> filename Redirects the standard output to a file.
< filename Reads input from a file.
>> filename Append the output to the end of an existing file.
| command Send the output to the input of another command.

Regular expressions

Most characters match themselves as in normal search and replace. If you need to match a special character, put a backslash before it, eg. \$ for a literal dollar-sign, and \\ for a literal backslash. The following is a summary of the special meanings:

Matching a position

^	Begining of string
$	End of string or newlines
\b	Word boundary
\B	Not a word boundary

Alternation and groups

..|..	Match either one or the other
(...)	Grouping and capturing
\1, \2, etc.	Refer back to previous groups

Character classes

\d, \s, \w	Any digit, whitespace character, or word character respectively
\D, \S, \W	The negation of the above classes
[...]	Any of the enclosed characters
[^...]	Anything except the enclosed characters
.	Any character

Quantifiers

?	0 or 1
*	0 or more
+	1 or more
{n,m}	Between n and m