Commonly Used Unix Commands

All of the commands which take filename parameters may use wildcards in those positions, but remember the use of the shell’s single and double quotes to prevent wildcard expansion when parameters contain spaces, tabs, etc.

ps Process status. List of all running processes.
-f full listing of command lines (use -l on some systems)
-e every process on the system
-u particular user’s processes (not Linux), ie. **ps -uroot,fred**
$ **ps [-options]**

mail, mailx, Mail
Electronic mail. Send or read electronic mail.
to send: specify user names on the command line, ie. **mail root fred**
to read: don’t give any user names, ie. **mailx**

ls List directories. List the contents of the given directories.
-l long listing
-a all hidden files (those that start with a dot) are listed also
-F append an attribute character to each filename [see *ls(1)*]
$ **ls [-options] [directory(s)]**

pg Paginate files. Use **h** at the colon prompt for the help screen.
-c clear screen before each page is displayed
$ **pg [-options] [filename(s)]**

cat Read file(s) and send to stdout.
-v To see control characters printed as visible sequences.
$ **cat [filename(s)]**

tee Read stdin and send to stdout. Save a copy of the data in files specified.
$ ls -l | **tee file1 file2** | sort | mail root

grep **G**lobally search for **R**egular **E**xpression and **P**rint.
See page 2 for a summary of Regular Expression syntax.
$ **grep ’regexpr’ [filename(s)]**

cd Change current directory. No parameters returns to $HOME directory.
$ **cd [directory]**

pwd Print full pathname of current directory.
$ **pwd**

vi Full-screen visual editor.
$ **vi [-options] [file(s)]**

rm Remove specified files and/or directories.
-r recursive — removes given file/directory and all subdirectories.
-i interactive — prompt for each file before removing.
$ **rm [-options] file(s)**

chmod
Change the modes (permissions) on files and directories.

Options allow specification of **user**, **group**, or **other** permission fields, whether to **add**, **remove**, or **specify** certain bits, and **read**, **write**, **execute**, and **setid** permission bits.
$ **chmod [ugo][+-=][rwxs] file(s)**

chgrp
Change the group id on files and directories; only the file owner and **root** may do this.
$ **chgrp group file(s)**

chown
Change the owner of files and directories; only **root** may do this.
$ **chown userid file(s)**

cp Copy files from one directory/filename to another.
$ **cp old_file new_file**
$ **cp file(s) directory**

mv Move files from one directory/filename to another.
$ **mv old_file new_file**
$ **mv file(s) directory**

sort Sorts given files; many options for controlling sort key(s).
Reads stdin if no files given; writes to stdout if no output file given [see *sort(1)*].
$ ls -il | **sort -n**

find Locates files by recursively searching a directory list.

-type [bcdflp]	Only particular file type
-name pattern	Only files which match shell pattern **pattern**
-print	Print list of names to stdout
-mtime ±**n**	Only files with certain modification times, **-n** < 0, less than **n** days **+n** > 0, more than **n** days **n** = 0, today
-size ±**n**	Only files with size **n** [see above for details]
-user id	Only files owned by user **id** (alpha or numeric); ids separated by commas
-exec cmd ';'	Execute **cmd** for each filename found
-ok cmd ';'	Same as **-exec**, but prompts before executing. The current filename is substituted for **{}** within **cmd**, ie. **-exec ls -l {} ';'**
<exp> -o **<exp>**	Logical OR of two expressions (default is AND)
'(' **<exp>** ')'	Logical grouping of expressions
! **<exp>**	Logical negation of expression

Regular Expressions

In the following descriptions, an **RE** is any single character unless one of the special metacharacters is used. The metacharacters may be made literal by preceding them with a backslash ("\").

Character	Meaning
[ ]	Match any single character from given set. Allows **-** for range, and ˆ immediately after open bracket for set negation (same as **[** and **]** in the shell).
.	Match any single character [same as **?** in the shell]
*	Match zero or more of preceding **RE** (**.* RE** same as ***** in shell)
+	Match one or more of preceding **RE**
?	Match zero or one of preceding **RE**
$	Match EOL when it appears at end of pattern
ˆ	Match BOL when it appears at beginning of pattern
|	Logical OR between two **RE**s

The following only work in *sed* or *vi*, not *grep*.

\<	Match beginning of word.
\>	Match end of word.
\(, \)	Specifies a subpattern **RE**.
\n	Where **n** is 0 through 9, and specifies subpattern **n** in the replacement string.