

# [FREEENGINEER.ORG](http://freeengineer.org)

---

Learn UNIX in 10 minutes. Version 1.3

## **Preface**

This is something that I had given out to students (CAD user training) in years past. The purpose was to have on one page the basics commands for getting started using the UNIX shell (so that they didn't call me asking what to do the first time someone gave them a tape).

This document is copyrighted but freely redistributable under the terms of the [GFDL](http://www.gnu.org/licenses/gfdl.html) .

Have an idea for this page? [cliff@freeengineer.org](mailto:cliff@freeengineer.org)

Send me patches, comments, corrections, about whatever you think is wrong or should be included. I am always happy to hear from you. Please include the word "UNIX" in your subject.

## **Sections:**

### [Directories:](#)

[Moving around the file system:](#)

[Listing directory contents:](#)

[Changing file permissions and attributes](#)

[Moving, renaming, and copying files:](#)

[Viewing and editing files:](#)

[Shells](#)

[Environment variables](#)

[Interactive History](#)

[Filename Completion](#)

[Bash is the way cool shell.](#)

[Redirection:](#)

[Pipes:](#)

[Command Substitution](#)

[Searching for strings in files: The \*grep\* command](#)

[Searching for files : The \*find\* command](#)

[Reading and writing tapes, backups, and archives: The \*tar\* command](#)

[File compression: \*compress\*, \*gzip\*, and \*bzip2\*](#)

[Looking for help: The \*man\* and \*apropos\* commands](#)

[Basics of the \*vi\* editor](#)

[FAQs](#)

```
*****
Basic UNIX Command Line (shell) navigation : Last revised May 17 2001
*****
```

## **Directories:**

File and directory paths in UNIX use the forward slash "/" to separate directory names in a path.

examples:

```
/          "root" directory
/usr       directory usr (sub-directory of / "root" directory)
/usr/STRIM100 STRIM100 is a subdirectory of /usr
```

## **Moving around the file system:**

```
pwd          Show the "present working directory", or current directory.
cd           Change current directory to your HOME directory.
cd /usr/STRIM100 Change current directory to /usr/STRIM100.
```

```

cd INIT          Change current directory to INIT which is a sub-directory of the current
                  directory.
cd ..           Change current directory to the parent directory of the current directory.
cd $STRMWORK    Change current directory to the directory defined by the environment
                  variable 'STRMWORK'.
cd ~bob        Change the current directory to the user bob's home directory (if you have permission).

```

### Listing directory contents:

```

ls      list a directory
ls -l   list a directory in long ( detailed ) format

```

for example:

```

$ ls -l
drwxr-xr-x    4 cliff    user      1024 Jun 18 09:40 WAITRON_EARNINGS
-rw-r--r--    1 cliff    user     767392 Jun  6 14:28 scanlib.tar.gz
^ ^ ^ ^      ^ ^      ^          ^      ^      ^      ^
| | | |      | |      |          |      |      |      |
| | | |      | owner  group      size  date  time  name
| | | |      number of links to file or directory contents
| | |  permissions for world
| |  permissions for members of group
|  permissions for owner of file: r = read, w = write, x = execute -=no permission
type of file: - = normal file, d=directory, l = symbolic link, and others...

```

```

ls -a        List the current directory including hidden files. Hidden files start
              with "."
ls -ld *     List all the file and directory names in the current directory using
              long format. Without the "d" option, ls would list the contents
              of any sub-directory of the current. With the "d" option, ls
              just lists them like regular files.

```

### Changing file permissions and attributes

```

chmod 755 file    Changes the permissions of file to be rwx for the owner, and rx for
                  the group and the world. (7 = rwx = 111 binary. 5 = r-x = 101 binary)
chgrp user file   Makes file belong to the group user.
chown cliff file  Makes cliff the owner of file.
chown -R cliff dir Makes cliff the owner of dir and everything in its directory tree.

```

You must be the owner of the file/directory or be root before you can do any of these things.

### Moving, renaming, and copying files:

```

cp file1 file2    copy a file
mv file1 newname  move or rename a file
mv file1 ~/AAA/   move file1 into sub-directory AAA in your home directory.
rm file1 [file2 ...] remove or delete a file
rm -r dir1 [dir2...] recursively remove a directory and its contents BE CAREFUL!
mkdir dir1 [dir2...] create directories
mkdir -p dirpath  create the directory dirpath, including all implied directories in the path.
rmdir dir1 [dir2...] remove an empty directory

```

### Viewing and editing files:

```

cat filename      Dump a file to the screen in ascii.
more filename     Progressively dump a file to the screen: ENTER = one line down
                  SPACEBAR = page down  q=quit
less filename     Like more, but you can use Page-Up too. Not on all systems.

```

```

vi filename      Edit a file using the vi editor. All UNIX systems will have vi in some form.
emacs filename   Edit a file using the emacs editor. Not all systems will have emacs.
head filename    Show the first few lines of a file.
head -n filename Show the first n lines of a file.
tail filename    Show the last few lines of a file.
tail -n filename Show the last n lines of a file.

```

## Shells

The behavior of the command line interface will differ slightly depending on the *shell* program that is being used.

Depending on the shell used, some extra behaviors can be quite nifty.

You can find out what shell you are using by the command:

```
echo $SHELL
```

Of course you can create a file with a list of shell commands and execute it like a program to perform a task. This is called a shell script. This is in fact the primary purpose of most shells, not the interactive command line behavior.

## Environment variables

You can teach your shell to remember things for later using environment variables. For example under the bash shell:

```

export CASROOT=/usr/local/CAS3.0      Defines the variable CASROOT with the value
                                       /usr/local/CAS3.0.
export LD_LIBRARY_PATH=$CASROOT/Linux/lib  Defines the variable LD_LIBRARY_PATH with
                                       the value of CASROOT with /Linux/lib appended,
                                       or /usr/local/CAS3.0/Linux/lib

```

By prefixing \$ to the variable name, you can evaluate it in any command:

```

cd $CASROOT      Changes your present working directory to the value of CASROOT
echo $CASROOT    Prints out the value of CASROOT, or /usr/local/CAS3.0
printenv CASROOT Does the same thing in bash and some other shells.

```

## Interactive History

A feature of bash and tcsh (and sometimes others) you can use the up-arrow keys to access your previous commands, edit them, and re-execute them.

## Filename Completion

A feature of bash and tcsh (and possibly others) you can use the TAB key to complete a partially typed filename. For example if you have a file called constantine-monks-and-willy-wonka.txt in your directory and want to edit it you can type 'vi const', hit the TAB key, and the shell will fill in the rest of the name for you (provided the completion is unique).

## Bash is the way cool shell.

Bash will even complete the name of commands and environment variables. And if there are multiple completions, if you hit TAB twice bash will show you all the completions. Bash is the default user shell for most Linux systems.

### **Redirection:**

<code>grep string filename &gt; newfile</code>	Redirects the output of the above grep command to a file 'newfile'.
<code>grep string filename &gt;&gt; existfile</code>	Appends the output of the grep command to the end of 'existfile'.

The redirection directives, > and >> can be used on the output of most commands to direct their output to a file.

### **Pipes:**

The pipe symbol "|" is used to direct the output of one command to the input of another.

For example:

`ls -l | more` This commands takes the output of the long format directory list command "ls -l" and pipes it through the more command (also known as a filter). In this case a very long list of files can be viewed a page at a time.

`du -sc * | sort -n | tail`  
The command "du -sc" lists the sizes of all files and directories in the current working directory. That is piped through "sort -n" which orders the output from smallest to largest size. Finally, that output is piped through "tail" which displays only the last few (which just happen to be the largest) results.

### **Command Substitution**

You can use the output of one command as an input to another command in another way called command substitution. Command substitution is invoked when by enclosing the substituted command in backwards single quotes. For example:

```
cat `find . -name aaa.txt`
```

which will cat ( dump to the screen ) all the files named aaa.txt that exist in the current directory or in any subdirectory tree.

### **Searching for strings in files: The *grep* command**

`grep string filename` prints all the lines in a file that contain the string

### **Searching for files : The *find* command**

```
find search_path -name filename
```

`find . -name aaa.txt` Finds all the files named aaa.txt in the current directory or any subdirectory tree.

`find / -name vimrc` Find all the files named 'vimrc' anywhere on the system.

`find /usr/local/games -name "*xpilot"`  
Find all files whose names contain the string 'xpilot' which exist within the '/usr/local/games' directory tree.

**Reading and writing tapes, backups, and archives: The tar command**

The tar command stands for "tape archive". It is the "standard" way to read and write archives (collections of files and whole directory trees).

Often you will find archives of stuff with names like stuff.tar, or stuff.tar.gz. This is stuff in a tar archive, and stuff in a tar archive which has been compressed using the gzip compression program respectively.

Chances are that if someone gives you a tape written on a UNIX system, it will be in tar format, and you will use tar (and your tape drive) to read it.

Likewise, if you want to write a tape to give to someone else, you should probably use tar as well.

Tar examples:

```
tar xv      Extracts (x) files from the default tape drive while listing (v = verbose)
           the file names to the screen.
tar tv      Lists the files from the default tape device without extracting them.
tar cv file1 file2
           Write files 'file1' and 'file2' to the default tape device.
tar cvf archive.tar file1 [file2...]
           Create a tar archive as a file "archive.tar" containing file1,
           file2...etc.
tar xvf archive.tar extract from the archive file
tar cvfz archive.tar.gz dname
           Create a gzip compressed tar archive containing everything in the directory
           'dname'. This does not work with all versions of tar.
tar xvfz archive.tar.gz
           Extract a gzip compressed tar archive. Does not work with all versions of tar.
tar cvfI archive.tar.bz2 dname
           Create a bz2 compressed tar archive. Does not work with all versions of tar
```

**File compression: compress, gzip, and bzip2**

The standard UNIX compression commands are compress and uncompress. Compressed files have a suffix .Z added to their name. For example:

```
compress part.igs    Creates a compressed file part.igs.Z
uncompress part.igs  Uncompresses part.igs from the compressed file part.igs.Z.
                    Note the .Z is not required.
```

Another common compression utility is gzip (and gunzip). These are the GNU compress and uncompress utilities. gzip usually gives better compression than standard compress, but may not be installed on all systems. The suffix for gzipped files is .gz

```
gzip part.igs       Creates a compressed file part.igs.gz
gunzip part.igs     Extracts the original file from part.igs.gz
```

The bzip2 utility has (in general) even better compression than gzip, but at the cost of longer times to compress and uncompress the files. It is not as common a utility as gzip, but is becoming more generally available.

```
bzip2 part.igs      Create a compressed Iges file part.igs.bz2
bunzip2 part.igs.bz2 Uncompress the compressed iges file.
```

**Looking for help: The man and apropos commands**

Most of the commands have a manual page which give sometimes useful, often more or less detailed, sometimes cryptic and unfathomable discriptions of their usage. Some say they are called man pages because they are only for real men.

Example:

man ls        Shows the manual page for the ls command

You can search through the man pages using *apropos*

Example:

apropos build        Shows a list of all the man pages whose discriptions contain the word "build"

Do a **man apropos** for detailed help on apropos.

### Basics of the vi editor

#### Opening a file

vi filename

#### Creating text

Edit modes: These keys enter editing modes and type in the text of your document.

i        Insert before current cursor position  
 I        Insert at beginning of current line  
 a        Insert (append) after current cursor position  
 A        Append to end of line  
 r        Replace 1 character  
 R        Replace mode  
 <ESC> Terminate insertion or overwrite mode

#### Deletion of text

x        Delete single character  
 dd       Delete current line and put in buffer  
 ndd     Delete n lines (n is a number) and put them in buffer  
 J        Attaches the next line to the end of the current line (deletes carriage return).

#### Oops

u        Undo last command

#### cut and paste

yy       Yank current line into buffer  
 nyy     Yank n lines into buffer  
 p        Put the contents of the buffer after the current line  
 P        Put the contents of the buffer before the current line

#### cursor positioning

^d       Page down  
 ^u       Page up  
 :n       Position cursor at line n  
 :\$       Position cursor at end of file  
 ^g       Display current line number  
 h,j,k,l Left,Down,Up, and Right respectively. Your arrow keys should also work if  
           if your keyboard mappings are anywhere near sane.

#### string substitution

:n1,n2:s/string1/string2/[g]        Substitute string2 for string1 on lines  
                                          n1 to n2. If g is included (meaning global),  
                                          all instances of string1 on each line

are substituted. If `g` is not included, only the first instance per matching line is substituted.

`^` matches start of line  
`.` matches any single character  
`$` matches end of line

These and other "special characters" (like the forward slash) can be "escaped" with `\` i.e to match the string `"/usr/STRIM100/SOFT"` say `"\usr\STRIM100\SOFT"`

Examples:

```
:1,$:s/dog/cat/g      Substitute 'cat' for 'dog', every instance
                      for the entire file - lines 1 to $ (end of file)

:23,25:/frog/bird/    Substitute 'bird' for 'frog' on lines
                      23 through 25. Only the first instance
                      on each line is substituted.
```

### Saving and quitting and other "ex" commands

These commands are all prefixed by pressing colon (`:`) and then entered in the lower left corner of the window. They are called "ex" commands because they are commands of the `ex` text editor - the precursor line editor to the screen editor `vi`. You cannot enter an "ex" command when you are in an edit mode (typing text onto the screen) Press `<ESC>` to exit from an editing mode.

```
:w                Write the current file.
:w new.file       Write the file to the name 'new.file'.
:w! existing.file Overwrite an existing file with the file currently being edited.
:wq              Write the file and quit.
:q               Quit.
:q!              Quit with no changes.

:e filename       Open the file 'filename' for editing.

:set number       Turns on line numbering
:set nonumber     Turns off line numbering
```

### **FAQs**

The USENET FAQs should be the first place you look for an answer to specific questions. You can find most of them at [RTFM](#). The contents of [this directory](#) includes `vi`, `bash`, and `comp.unix.questions` FAQs. Searching USENET archives are very useful too. [google.com](#) has a USENET archive (formerly `Deja.com`'s) . [Advanced Group Search](#) rules.

This document was converted from plain text using [Vim](#) and then hacked. `Vim` is the best version of the one true text editor: `vi`.

Copyright (c) 2000-2006 [cliff@freeengineer.org](mailto:cliff@freeengineer.org)  
 Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.1 or any later version published by the Free Software Foundation; with Invariant Section: Preface, with Front-Cover Texts, and with no Back-Cover Texts. A copy of the license can be found on the GNU web site [here](#).

---

[\*\*FREEENGINEER.ORG\*\*](http://FREEENGINEER.ORG)