

# Terse UNIX/LINUX Primer

Lowell M. Boone

Updated December 13, 2001

The intention of this primer is not to elevate you to the status of super-geek over night. This is a goal that is only attainable through time and hard work. However, this primer should give you the basic productivity and information tools you will need to follow your own path down that long dark road. One word of advice before we start: Learning UNIX is a bit of a paradox. It is not a linear process. Thus, you will probably have to read this primer several times before certain aspects make sense. Don't panic.

## UNIX Philosophy

UNIX was my first encounter with the notion of commands as little programs and not just options in a larger interface program. Much of the time, when you enter a command at the prompt, you are actually executing little (or not-so-little) autonomous programs. Often, these commands can be fitted together either directly or indirectly to allow the user to “construct” a more tailored command. This is in direct opposition to the philosophy that drives the “Microsoft Empire”: One huge program that does a lot of different things and *nothing* else. The former derives its power from its flexibility, while the latter is often more simple to use because its rigidity forces a standardization. You gain a lot from moving to a modular environment of simple little programs that do one thing well, but you must first learn to accept that this requires flexibility in your thinking.

## The Shell

The shell is the interface between you and the machine. It is a program that accepts input from you, and relays it to the system. There are a number of different shells—each with its own language. I use the “new and improved C shell”, called `tcsh` (The regular old “C” shell is just called `csh`). I assume you're using `tcsh` in this primer. You can change shells with the `chsh` command—ask for the `/bin/tcsh` shell.

You can set shell preferences and environment variables with `set` and `setenv` respectively. To see a list of things that have been “set”, type `set`. For your current list of environment variables, type `setenv`. When you login, the shell reads your `.cshrc` file and executes all the commands in that file. It is quite common to see a bunch of `set` and `setenv` commands in this file. It is also common to set up aliases that you use frequently in this file.

One very important environment variable is the “path”. This is a list of all possible directories in which the shell might find the commands you type. When you type a

command, the shell looks through all the directories in your path *in order*, until it finds the command you have requested. If you are not able to run a program, check to see where it is with `locate` command, and then check your path with `echo $PATH` to make sure the directory in which the command resides is also included in your path. If not, you need to add that directory to your path. To add `/dev` to your path, use `setenv PATH ${PATH}:/dev`. This will append `/dev` to the *end* of your path. The command `setenv PATH /dev:${PATH}` preappends `/dev` to the beginning of your path. If there is more than one version of the command you are trying to execute, asking *which* `<command>` will tell you which version you are running. You can always override this by typing the explicit path of the command (ie, `/bin/ls` ensures that you are using the `ls` located in `/bin`, and not some other variant).

## Basic Commands and Safety

To list the contents of a directory, use `ls`. To see all the hidden files, use `ls -a`. To see a listing of all the properties of all the files, use `ls -l`. Of course, you can do both at the same time with `ls -la`. You might also try `ls -pF`, which appends little characters to the end of the files depending upon what they are (directories, soft links, executables, etc).

To remove a file, use `rm filename`. To copy a file use `cp oldname newname`. And to move a file use `mv oldname newname`. For directories, use `rmdir`, `cp -R`, and `mv` respectively. It's often nice to have the machine prompt you before it overwrites stuff. To do this you can use `set noclobber` (see section on The Shell for more on `set`), and alias the three commands above with `alias rm '/bin/rm -i'`, `alias cp '/bin/cp -i'`, and `alias mv '/bin/mv -i'`. `alias` replaces the first string with the rest of the line. I would strongly encourage you to put these aliases in your `.cshrc` file (see section on “The Shell”).

## Passwords

These aren't complicated, but it's important to do them right. Your password should be hard to guess. `sunnyday`, `loverboy`, and `studmuffin` are bad because any password cracking program will guess them (and yes, these are actual examples of real-live passwords). Try to include variations in capitalization, and non-alphanumeric characters. Examples of good passwords are `Wh1deD$` and `S@b0*Ag3`. These may seem hard to remember (and they are at first), but you'll get used to them. It is certainly a lot better to spend a couple of days memorizing your password than to have your account cracked and used to threaten the President of the United States. To change your password, type `passwd`. You'll be asked for your old password, then asked to type your new password twice.

## Navigation

Navigating the UNIX file system can be a bit confusing at first—especially for those used to a graphical interface such as the MacOS (or its ugly cousin, Windows). The idea is much the same, but you have to pay attention to directory names rather than pictures. To find out where you are, ask the system to “print the working directory” with `pwd`. To “change directories”, use `cd newdirectory`. The directory `..` is short for the

directory immediately above you, and `.` is short for the current directory. Thus if you `cd /etc/rc.d` and do `pwd`, you will get `/etc/rc.d`. Then, if you `cd ../` and again `pwd`, you will get `/etc`. Your home directory is special, and you can usually return to it with a simple `cd` (no arguments).

Pay attention as you move around the directory tree. Especially when you are getting to know a new system, use `pwd` a lot to make sure you know where you are. Sometimes you can `cd` into a shortcut (called a “soft-link”) that will take you to an entirely different directory—specifically, one that is not below the directory you came from. In these cases, `cd`’ing into the soft-link and then trying to get back out with `cd ..` may *not* bring you back to the same place. The UNIX file system is full of these kinds of rabbit holes—just be aware.

LINUX machines have a `locate` utility for finding files. To find all files that contain the string `dev`, try `locate dev`. This will be a pretty long list, so you might want to `locate dev | more` (see section on Output Manipulation for more on this).

## Productivity / Efficiency

The `tcsh` shell has a number of conventions to help you get things done more quickly. The shell keeps track of each command you’ve typed in a “history”. To look at your history, type `history`. If you hit the up and down arrows, you can scroll through your history and re-execute any of your previous commands.

If you know a command is in your history, you can access it by typing the first few characters of the command and then hitting `<esc>-p`. This will bring up the most recent entry in your history that matches what you typed. Hitting `<esc>-p` again will bring up the next most recent, etc.

If you want to change some aspect of a command, use the right and left arrows to navigate back and forth through the command line. To skip to the beginning of the line, use `<control>-a`. And to delete the next word, use `<esc>-d`. You *don’t* have to be at the end of the line to hit return and execute the command! As soon as you hit return, regardless of the cursor position, the command will be executed.

If you have “tab completion” enabled and you type the first few characters of the command, try hitting `<tab>`. The shell will guess the rest of the command for you and complete the line as best it can. This is a *big* time saver. To turn on tab completion, type `set filec`. To have the system present possible options when it becomes confused, type `set autolist`. `set nobeep` keeps the shell from beeping at you each time you use tab completion. Each of these set commands can go into your `.cshrc` file (see section on “The Shell”).

If you are running a command and you don’t want to wait around for it to complete before regaining control of the terminal, background the process by hitting `<control>-z`. This suspends the command and returns control of the session to you. If you then type `bg`, it will send the process into the background, and it will run while you do other things. To bring the job back to the foreground, use `fg`.

Wild cards are devices that allow you to reference more than one file at a time. The two most common are the asterisk (`*`), and the question mark (`?`). The asterisk expands to *any number* of any type of consecutive characters. So `rm *.ps` will remove all files that

end in `.ps`. Likewise, `cp run*.raw data/` will copy all files that begin with `run` and end with `.raw` to the `data` directory. The question mark expands to a *single* character. So `cp run?.raw data/` will copy files like `runA.raw` and `run2.raw` to `data`, but not `run11.raw`.

## Output Manipulation

Many commands you execute will produce output. There are two main things you will do with this output (besides read it). You can redirect it, or pipe it. These are very similar, but their differences are important.

Piping output from one command to another literally takes the output of the first, and uses it as the input of the second—effectively creating a single command pipeline. Wouldn't it be nice if the output of `ls /dev` were paged, and didn't just fly by on the screen? Try piping the output of the list into the `more` command like so: `ls /dev | more`. One of the most common uses for the pipe is to paginate output with `more`.

Output may be redirected to somewhere other than the screen with the `>` operator. To dump a list of the `/dev` directory to a text file called `thingy.txt`, try `ls /dev > thingy.txt`. If `thingy.txt` already exists, `ls /dev >! thingy.txt` will overwrite it, or `ls /dev >> thingy.txt` will append to the end of it. `/dev/null` is code for nothing at all, so `ls /dev > /dev/null` directs the listing out of existence!

## Getting System Information

Responsible computing requires a knowledge of the resources that are available to you, and the resources you are using. Each program that you run (even most commands) are CPU processes. And each process you run has an identification number (PID) associated with it. You can find out about your processes with `ps`, but this will only tell you about processes that you have started in your current session. Try `ps -u mothra` to find out *all* the processes that the user `mothra` is running. To kill a process, find its PID with `ps`, and then kill it with `kill -9 23451` where `23451` is the PID in this example.

Disk space is also an important resource that you will want to watch. If you want to know how big your directory is, try `du`. If you want to know how big the entire file system is (including how much space is on various disks), try `df`. Following each of these commands with a `-k` option will display the result in 1024 byte blocks (kilobytes). Following each of these commands with a `-h` option will put the output in “human readable” format (12M, 6.5G, etc.). Finally, `du -s *` will report the total size of each element in the current directory without reporting sizes of subdirectories.

## Some Useful Utilities

To thumb through a text file one page at a time, use either `more` or `less`. `less` has more features than `more`, but is a little more complicated than `more`. The more you use `less`, the more `more` will look less appealing than `less`, and the less you will use `more`. Man pages are generally paged with `less` because it has more features than `more`. Try `more /etc/passwd` to read the password file. Enough of that. In both these programs, the space bar pages through the document, `return` advances line by line, and `q` will quit the program. `cat /etc/password` will print the file to the screen as well, but without any

page breaks. For appending a file to the end of another one, `cat file1 >> file2` works nicely.

To compress a file, use `gzip filename`. To uncompress it, use `gunzip filename`. If you have a number of files you would like to bind into an archive, use `tar`. For example, to tar two files into one archive, use `tar -cf archive.tar fileone filetwo`. To tar an entire directory, use `tar -cf archive.tar directory/`. To untar these archives, use `tar -xf archive.tar`. You can gzip an archive as well. In fact, this is so common that when extracting, tar often understands gzipped tar archives. In other words, `tar -zxvf archive.tar.gz` is equivalent to `gunzip archive.tar.gz`, followed by `tar -xf archive.tar` on many machines.

There are a variety of file formats that you cannot access in a sensible manner with `cat`, `more`, or `less`. To view these, you need to use other utilities. Be aware that not all of these utilities are included with standard OS distributions. To view a postscript file (these files usually have a `.ps` or `.eps` suffix), use `gv` or `ghostscript`. I suggest the former. For example, to read this primer, type `gv primer.ps`. To view a Portable Document Format (PDF) file (usually with a `.pdf` suffix), use `xpdf` or `acroread`. The `xv` package can be used to view most types of image files: JPG, GIF, PostScript, etc.

## Spreading Your Wings

In order to conquer the world, you need to be able to get out of your local machine, and into others. The golden rule for machine to machine connections is: “Never use unencrypted connections”. If the connection is not encrypted, it can be sniffed (someone gets your password), or hijacked (someone takes control of your session and threatens the President of the United States). If you don’t have access to encrypted protocols, ask your sysadmin to install them. Once again: Never never never use unencrypted connections.

There are two types of connections you might make to another machine: logging in, and file transfers. To login, use `ssh` (or `ssh1` if your destination complains of a protocol error). To log into `whoop.nasa.gov`, type `ssh whoop.nasa.gov`. If you want to login as someone else (like `stacee`) use either `ssh whoop.nasa.gov -l stacee` or `ssh stacee@whoop.nasa.gov`. `telnet` and `rlogin` are the unencrypted cousins to `ssh`, so you should what? *Never use them.*

For file transfers, use `scp` (or `scp1` if your destination complains of a protocol error). To upload a file to the `stacee` account on `whoop.nasa.gov`, use `scp somefile stacee@whoop.nasa.gov:`. The file will then appear in `stacee`’s home directory. To upload the directory `boing/` to the `online` account on that same machine, use `scp -r boing online@whoop.nasa.gov:`. The directory `boing/`, and all its contents will then appear in the home directory of the user `online`. To put a file in the `bin/` directory of the `stacee` account, use `scp somefile stacee@whoop.nasa.gov:bin`. You don’t always have to specify destinations with respect to the home directory of the receiving account. `scp somefile stacee@whoop.nasa.gov:/tmp` will place `somefile` in the top-level directory `/tmp`. `ftp` and `rcp` are the unencrypted cousins to `scp`, so you should—say it with me now: *Never use them.*

## Further Information

One of the most feared elements of any UNIX system is the manual system. However, if you know how to use it, it is easily one of the most useful elements. The problem is that many folks never get up the nerve to carefully read the pages. Man pages exist for almost every command available in UNIX, as well as for most functions you will use in your programming. To access the page for a command, type `man command`. You can even try `man man` to find out how to use the system, though this is considered by many to be rather a cruel joke. If you don't know the command name, but have a key word to describe your problem, try `man -k keyword | more`. This will give you a list of all the man page lines that contain that keyword. Practice makes perfect. If you always try checking the man pages before looking elsewhere for answers, they will eventually become a very valuable tool.

## Closing

This is just the tip of the iceberg. Hopefully, this is enough to get you up and running in a semi-competent manner. But you will definitely need to understand the UNIX system better than presented here before too long. Books, man pages, and fellow geeks are the best resources I've found for this. I like the O'Reilly books best—partially for their unique covers, and partially because they are generally well written. Try *Learning the UNIX Operating System* for starters (the “Owl” book) if you are still pretty green. There is also *UNIX in a Nutshell* for a general reference. For more specific help on a particular shell, O'Reilly also offers books specifically on the `bash` and `tcsh` shells (among others).

## An example of a .cshrc file:

```
setenv PRINTER scipp-ptr
setenv ENSCRIPT "-fCourier7 --media=Letter -G2r"
setenv PATH $PATH:/usr/games

#####
#                               Setting Some Variables                               #
#####

umask 077                        # rwx priveleges for user only
umask 022                        # let others see your work
set prompt="% "                  # sets prompt to percent sign
set savehist = 25                # save history after session
set history = 200                # sets limit on history memory
source -h ~/.history             # load saved history
set filec                        # enable filename completion (on TAB)
set autolist                     # make tcsh filename completion nicer
set no beep                      # don't beep on incomplete file names
set fignore=(.dvi .log .ps .o)  # for better filename completion
set cdpath = (~ /home/vhep/)    # make cd smarter
set noclobber                    # don't overwrite files on redirection
set notify                      # notify immediately when jobs terminate
set ignoreeof                   # <control>-D won't logout
unset autologout                # turn off autologout feature
unlimit coredumpsize            # so we can look at coredumps

#####
#                               Utitltiy and Aesthetic Aliases                       #
#####

alias ls "/bin/ls -pF"          # so you can tell directories
alias rmdir /bin/rm -ir        # for removing directories
alias cp /bin/cp -i            # query before clobbering
alias mv /bin/mv -i            # query before clobbering
alias rm /bin/rm -i            # query before removing
alias du "du -k"               # get disk useage in kilobytes
alias spell ispell             # use a better spelling program
alias dvips dvips -t letter    # force letter sized paper
alias root /usr/local/root/bin/root -l
alias paw pawX11
alias ghostview 'ghostview -geometry +0+0'
```