General Linux Tips

The purpose of this section is to give students that are new to the Linux Operating System some general tips on using Linux.

Typing Long Filenames

The student should have noticed by now that a large majority of files in Linux have very long filenames. It is however not necessary to type out the entire filename. The student can use the <TAB> key to complete filenames. For example, if you

Understanding Installation Filenames

Files ending with the extension '.gz', means that it is a gnu-zipped file. This is a form of data compression not unlike WinZip. To create your own gzip files, use the command 'gzip'.

***NOTE:** Please review the Linux Commands section or the man documentation provided in Linux for the various options (sometimes referred to as switches) that can be used with this and all subsequent commands. For example, type "man gzip".

Files ending with the extension '.tar', means that the file has been tar-red. This is also just another form of data compression.

Files ending with the extension '.tgz', means that the file has been tarred and gnu-zipped. This is also just another form of data compression.

Installation Instructions

Most installation files when uncompressed have README or INSTALL files. These provide information on how to install the files.

The 'make' command compiles source files. The 'make install' command installs the compiled source to their appropriate directories.

Virtual consoles

The system's console is the monitor and keyboard connected directly to the system. (Because Linux is a multi-user operating system, you may have other terminals connected to serial ports on your system, but these would not be the console.) Linux provides access to virtual consoles (or VCs), that let you have more than one login session on the console at one time. To demonstrate this, log in to your system. Next, press <ALT>+<F2>. You should see the login: prompt again. You're looking at the second virtual console. To switch back to the first VC, press <ALT>+<F1>. The Linux system

we will be using lets you access six VCs, by pressing <ALT>+<F1> through <ALT>+<F6>. It is possible to enable up to 12 VCs--one for each function key on your keyboard. As you can see, use of VCs can be very powerful because you can work in several different sessions at the same time. While the use of VCs is somewhat limiting (after all, you can look at only one VC at a time), it should give you a feel for the multi-user capabilities of Linux. While you're working on the first VC, you can switch over to the second VC and work on something else.

Changing your password

The command ‘passwd' will prompt you for your old password, and your new password. It will ask you to reenter the new password for validation. Be careful not to forget your password!!

Getting Online Help

Almost every UNIX system, Linux included, provides a facility known as ‘manual pages’, or ‘man pages’ for short. These man pages contain online documentation for all of the various system commands, resources, configuration files, and so on. The command used to access man pages is man. For example, if you're interested in finding out about the other options of the ‘ls’ command, you can type ‘man ls’ and the man page for ‘ls’ will be displayed. Unfortunately, most of the man pages out there are written for those who already have some idea of what the command or resource does. For this reason, man pages usually only contain the hardcore technical details of the command, without a lot of tutorial. However, man pages can be an invaluable resource for jogging your memory if you forget the syntax of a command. You'll notice some of commands won't have man pages. This could be for several reasons. For one, the man pages haven't been written yet. Secondly, the command might be an internal shell command, or an alias, in which case it would not have a man page of its own. One example is ‘cd’, which is a shell internal command. The shell actually processes the ‘cd’---there is no separate program which contains this command.

Using the vi Editor

While using vi, at any one time you are in one of three modes of operation. These modes are known as command mode, insert mode, and last line mode. When you start up vi, you are in command mode. This mode allows you to use certain commands to edit files or to change to other modes. For example, typing ‘x’ while in command mode deletes the character underneath the cursor. The arrow keys move the cursor around the file, which you're editing. Generally, the commands used in command mode are one or two characters long.

You actually insert or edit text within insert mode. When using vi, you'll probably spend most of your time within this mode. You start insert mode by using a command such as

'i' (for insert) from command mode. While in insert mode, you are inserting text into the document from your current cursor location. To end insert mode and return to command mode, press <ESC>.

Last line mode is a special mode used to give certain extended commands to vi. While typing these commands, they appear on the last line of the screen (hence the name). For example, when you type ':' from command mode, you jump into last line mode, and can use commands such as 'wq' (to write the file and quit vi), or 'q!' (to quit vi without saving changes). Last line mode is generally used for vi commands which are longer than one character. In last line mode, you enter a single-line command and press <ENTER> to execute it.

The best way to understand these concepts is to actually fire up vi and edit a file. In the example 'screens' below, we're only going to show a few lines of text, as if the screen was only six lines high (instead of twenty-four). The syntax for vi is

vi <filename>

where <filename> is the name of the file that you wish to edit.

Start up vi by typing

vi test

which will edit the file test. You should see something like

```
_____________________________________________________________
|~_                                                          |
|~                                                           |
|~                                                           |
|~                                                           |
|~                                                           |
|"test"_[New_file]___________________________________________|
```

The column of '~' characters indicates that you are the end of the file.

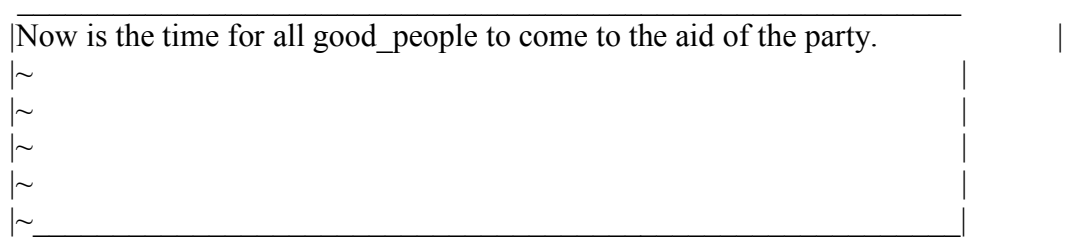
Inserting text

You are now in command mode; in order to insert text into the file, press 'i' (which will place you into insert mode), and begin typing.

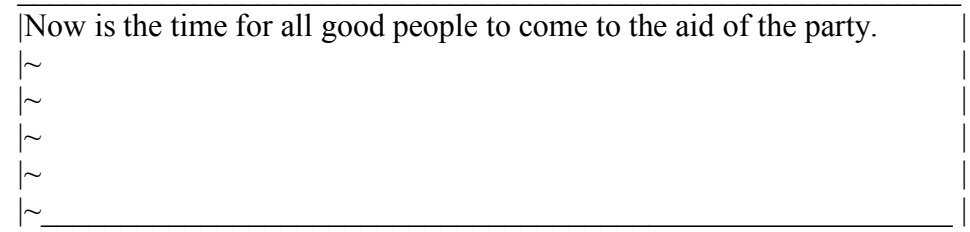
```
 ___________________________________________________________
|Now is the time for all good people to come to the aid of the party._       |
|~                                                           |
```



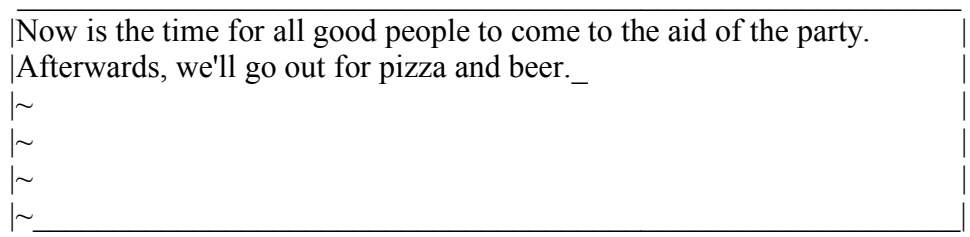
While inserting text, you may type as many lines as you wish (pressing <ENTER> after each, of course), and you may correct mistakes using the backspace key. To end insert mode, and return to command mode, press <ESC>. While in command mode, you can use the arrow keys to move around the file. Here, because we only have one line of text, trying to use the up- or down-arrow keys will probably cause vi to beep at you. There are several ways to insert text, other than using the 'i' command. For example, the 'a' command inserts text beginning after the current cursor position, instead of on the current cursor position. For example, use the left arrow key to move the cursor between the words 'good' and 'people'.



Press 'a', to start insert mode, type 'wo', and then hit <ESC> to return to command mode.



To begin inserting text at the line below the current one, use the 'o' command. For example, press 'o' and type another line or two:



Just remember that at any time you're either in command mode (where commands such as ‘i’, ‘a’, or ‘o’ are valid), or in insert mode (where you're inserting text, followed by <ESC> to return to command mode), or last line mode (where you're entering extended commands, as discussed below).

Deleting text

From command mode, the x command deletes the character under the cursor. If you press <x> five times, you'll end up with:

```
|Now is the time for all good people to come to the aid of the party.  |
|Afterwards, we'll go out for pizza and _                              |
|~                                                                     |
|~                                                                     |
|~                                                                     |
|~_____________________________________________________________________|
```

Now press <a>, insert some text, followed by <ESC>:

```
|Now is the time for all good people to come to the aid of the party.  |
|Afterwards, we'll go out for pizza and Diet Coke._                    |
|~                                                                     |
|~                                                                     |
|~                                                                     |
|~_____________________________________________________________________|
```

You can delete entire lines using the command ‘dd’ (that is, press <d> twice in a row). If your cursor is on the second line, and you type ‘dd’,

```
|Now is the time for all good people to come to the aid of the party.  |
|~                                                                     |
|~                                                                     |
|~                                                                     |
|~                                                                     |
|~_____________________________________________________________________|
```

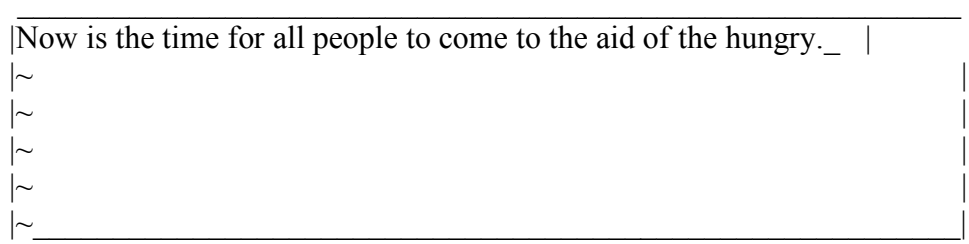
To delete the word which the cursor is on, use the ‘dw’ command. Place the cursor on the word ‘good’, and type ‘dw’.

```
|Now is the time for all people to come to the aid of the party.       |
|~                                                                     |
|~                                                                     |
|~                                                                     |
|~                                                                     |
```

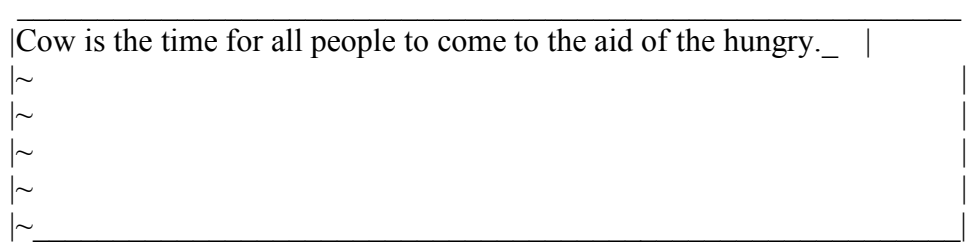


Changing text

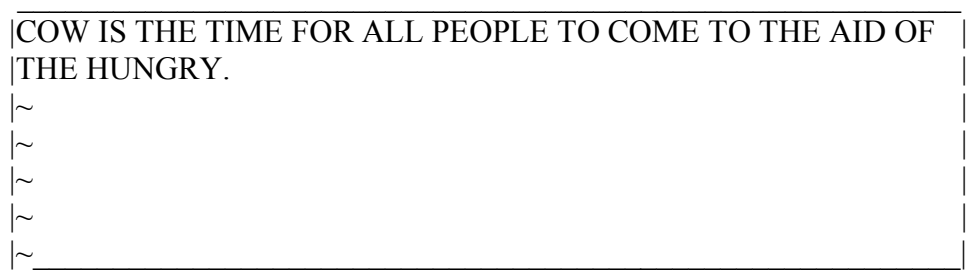
You can replace sections of text using the ‘R’ command. Place the cursor on the first letter in ‘party’, press <R>, and type the word ‘hungry’.



Using ‘R’ to edit text is much like the ‘i’ and ‘a’ commands, but ‘R’ overwrites text, instead of inserting it. The ‘r’ command replaces the single character under the cursor. For example, move the cursor to the beginning of the word ‘Now’, and type ‘r’ followed by ‘C’, you'll have:



The ‘~’ command changes the case of the letter under the cursor from upper- to lower-case, and vise versa, For example, if you place the cursor on the ‘o’ in ‘Cow’, above, and repeatedly press <~>, you'll end up with:



Moving Commands

You already know how to use the arrow keys to move around the document. In addition, you can use the 'h', 'j', 'k', and 'l' commands to move the cursor left, down, up, and right, respectively. This comes in handy when (for some reason) your arrow keys aren't working correctly. The 'w' command moves the cursor to the beginning of the next word; the 'b' moves it to the beginning of the previous word. The '0' (that's a zero) command moves the cursor to the beginning of the current line, and the '$' command moves it to the end of the line. When editing large files, you'll want to move forwards or backwards through the file a screenful at a time. Pressing <CTRLl>+<F> moves the cursor one screenful forward, and <CNTRL>+<B> moves it a screenful back. In order to move the cursor to the end of the file, type 'G'. You can also move to an arbitrary line; for example, typing the command '10G' would move the cursor to line 10 in the file. To move to the beginning of the file, use '1G'. You can couple moving commands with other commands, such as deletion. For example, the command 'd$' will delete everything from the cursor to the end of the line; 'dG' will delete everything from the cursor to the end of the file, and so on.

Saving Files and Quitting vi

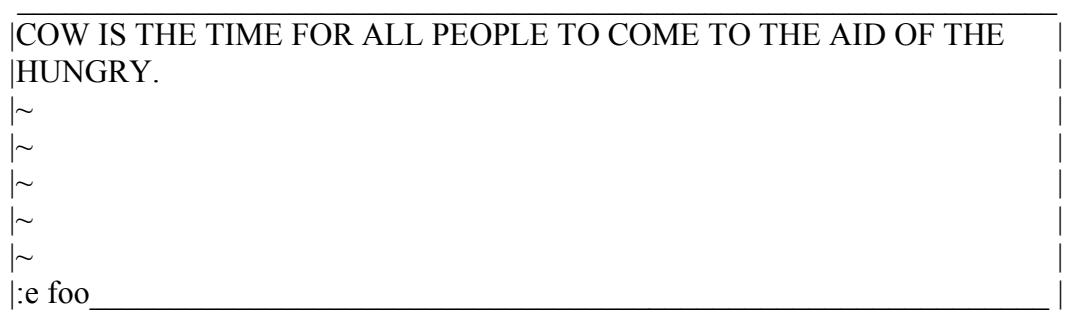
To quit vi without making changes to the file, use the command: 'q!'. When you type the ':', the cursor will move to the last line on the screen; you'll be in last line mode.

```
|COW IS THE TIME FOR ALL PEOPLE TO COME TO THE AID OF THE     |
|HUNGRY.                                                      |
|~                                                            |
|~                                                            |
|~                                                            |
|~                                                            |
|~                                                            |
|:____________________________________________________________|
```

In last line mode, certain extended commands are available. One of them is 'q!', which quits vi without saving. The command ':wq' saves the file and then exits vi. The command 'ZZ' (from command mode, without the ':') is equivalent to ':wq'. Remember that you must press <ENTER> after a command entered in last line mode. To save the file without quitting vi, just use ':w'.

Editing another file

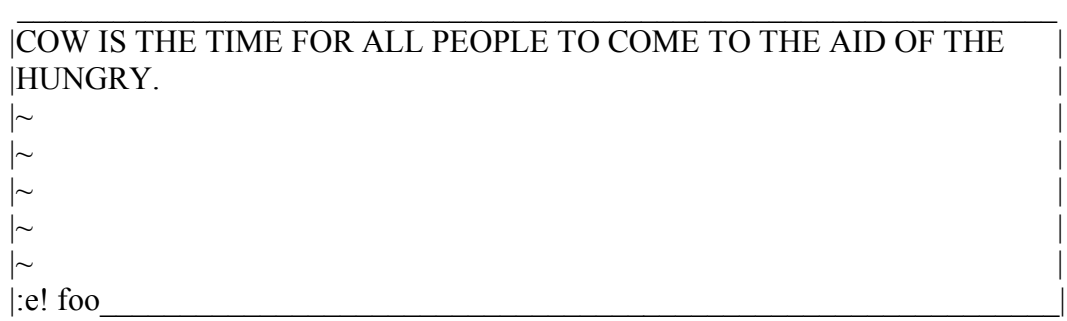
To edit another file, use the ':e' command. For example, to stop editing test, and edit the file foo instead, use the command



If you use ‘:e’ without saving the file first, you'll get the error message

|No_write_since_last_change_(":edit!"_overrides)_____|

which simply means that vi doesn't want to edit another file until you save the first one. At this point, you can use :w to save the original file, and then use ‘:e’, or you can use the command



The ‘!’ tells vi that you really mean it---edit the new file without saving changes to the first.

Including Other Files

If you use the ‘:r’ command, you can include the contents of another file in the current file. For example, the command

:r foo.txt

would insert the contents of the file foo.txt in the text at the current cursor location.

Running shell commands

You can also run shell commands from within vi. The ‘:r!’ command works like ‘:r’, but instead of reading a file, it inserts the output of the given command into the buffer at the current cursor location. For example, if you use the command

:r! ls -F

you'll end up with

```
 _____________________________________________________________________
|COW IS THE TIME FOR ALL PEOPLE TO COME TO THE AID OF THE             |
|HUNGRY.                                                              |
|letters/                                                             |
|misc/                                                                |
|papers/_                                                             |
|~                                                                    |
|~____________________________________________________________________|
```

You can also 'shell out' of vi, in other words, run a command from within vi, and return to the editor when you're done. For example, if you use the command

:! ls -F

the 'ls –F' command will be executed, and the results displayed on the screen, but not inserted into the file which you're editing. If you use the command

:shell

vi will start an instance of the shell, allowing you to temporarily put vi 'on hold' while you execute other commands. Just logout of the shell (using the exit command) to return to vi.

Linux Commands

Here are some of the basic commands that will help the new Linux user.

***NOTE:** Be aware that commands are case-sensitive in Linux.

apropos	See all the commands relating to a word Syntax: apropos <word> (e.g., apropos user | less)
cd	Change the current working directory Syntax: cd <directory>, where <directory> is the name of the directory you wish to change to.
ls	Displays information about the named files and directories Syntax: ls <files>, where consists of the filenames or directories to list
cp	Copies one or more files to another file or directory

Syntax: cp <files> <destination>, where <files> lists the files to copy and <destination> is the destination file or directory

mv — Moves one or more files to another file or directory. This is the equivalent of the cp command followed by the deletion of the original file. This command can be used to rename files.
Syntax: mv <files> <destination>, where <files> lists the files to move and <destination> is the destination file or directory

rm — Deletes files.
Syntax: rm <files>, where <files> are the files to be deleted

mkdir — Creates new directories
Syntax: mkdir <directoryname>, where <directoryname> is the name of the directory you wish to create

rmdir — This command deletes empty directories. When using rmdir, your current working directory must not be within the directory to be deleted.
Syntax: rmdir <dir1> <dir2> ... <dirN>, where <dir1> through <dirN> are the directories to delete.

man — Displays the manual page for the given command or resource (that is, any system utility which isn't a command, such as a library function.)
Syntax: man <command>, where <command> is the name of the command or resource to get help on.

more — Displays the contents of the named files, one screenful at a time.
Syntax: more <file1> <file2> ... <fileN>, where <file1> through <fileN> are the files to display. ["less" is better than "more"]

cat — Officially used to concatenate files, cat is also used to display the entire contents of a file at once.
Syntax: cat <file1> <file2> ... <fileN>, where <file1> through <fileN> are the files to display.

Handy use: cat - > filename
Type any text to put in file 'filename'
^d [control-d to end typing into file]

grep — Display all of the lines in the named file(s) matching the given pattern.
Syntax: grep <pattern> <file1> <file2> ... <fileN>, where <pattern> is a regular expression pattern, and <file1> through <fileN> are the files to search.
Example: grep loomer /etc/hosts will display all lines in the file /etc/hosts which contain the pattern ``loomer''.

shutdown — Shuts the machine down
Syntax: shutdown –h now

shutdown — Reboots the machine.
Syntax: shutdown –r now

startx — Starts X Windows
Syntax: startx