Unix/Linux Basics

Introduction

The USQ's HPC Facility is Linux based system. The information in this document provides some basic information about command-line commands, editors available and environmental variables.

Unix/Linux Commands

Below is a list of commonly used commands when working on the command-line.

Command	Description
Commonly Used Unix/Linux Commands	
cat / more / less *filename*	It a file on the standard output
cd *dirname*	change directory. Basically 'go' to another directory, and you will see the files in that directory when you do 'ls'.
cp *filename1 filename2*	copy a file and directory
grep a*string filename*	search for a regular expression in a file
ls	list directory contents
man *command*	display the on-line manual page for a command
mkdir *dirname*	make a new directory
mv *filename1 filename2*	move a file, i.e. gives it a different name, or moves it into a different directory
pwd	print name of current/working directory
rm *filename*	remove file
rmdir *dirname*	remove directory

File Compression Commands	
gzip *filename*	compresses files, so that they take up much less space. Gzip produces files with the ending '.gz' appended to the original filename.
gunzip *filename*	uncompresses file compressed by gzip.
tar –czf file.tar.gz files	create a tar with Gzip compression containing files
zcat *filename*	lets you look at a gzipped file without actually having to gunzip it (same as gunzip -c).

Other Useful Commands	
date	shows the current date and time
du *filename*	shows the disk usage of the files and directories in filename
emacs *filename*	edit filename in emacs
last *yourusername*	lists your last logins
ps –u *yourusername*	lists your processes
vi *filename*	edit filename in vi (VIM editor)
wc *filename*	count words in a file
whoami	returns your username

Connect/Copy files to remote host	
ssh user@host	secure login to remote host
scp file user@host:destination	secure copy to remote host

Shortcuts	
Ctrl+C	Halts the current command
Ctrl+Z	stops the current commands, resume with *fg* (foreground) or *bg* (background)
Ctrl+D	log out of current session, similar to *exit* or *logout*

Editors

This section is only intended to provide the minimum amount of information about individual editors, enough to open or close a file, make simply changes and then save or quit. For more information about individual editor review their manual (man) pages or visit their respective web sites.

Vi Editor

Vi is not the most user friendly or powerful of editors though it's extremely useful as it is the standard editor on all Unix systems. If you need more information consult other Unix references or visit Vim home page.

Starting vi

```
vi filename
```

vi operates in two modes, i.e. command and input modes, however only command mode will be discussed here as this is default mode and allows the user to move around a file.

Key Combinations	Results
Creating Text	
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 - made a mistake	
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. Arrow keys should also work if your keyboard mappings are correct.

Saving and quitting and other commands	
: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

Emacs Editor

Emacs is a powerful text editor provided by the GNU Free Software Foundation, a non-profit organisation dedicated to providing high quality public domain software. If you need more information consult other Unix references or visit EMACS home page.

Starting Emacs

```
emacs filename
```

Key Combinations	Results
Quitting	
<CTRL>-x <CTRL>-c	Quit
<CTRL>-g	Pushed the wrong key. Help
Working with Files	
<CTRL>-x <CTRL>-f	Load a file
<CTRL>-x <CTRL>-f	Load a directory
<CTRL>-x <CTRL>-f	New file
<CTRL>-x <CTRL>-s	Save a file
<CTRL>-x s	Save all open files
<CTRL>-x <CTRL>-w	Save a file with a new name

Working with Buffers	
<CTRL>-x b	Switch buffers
<CTRL>-x k	Close buffer
<CTRL>-x 2	Split current buffer
<CTRL>-x 1	Make current buffer the only one on screen
<CTRL>-x o	Switch between the buffers on-screen
Cutting and Pasting	
<CTRL><SPACE>	Set mark
<CTRL>-w	Cut and save text from here to mark
<CTRL>-y	Paste saved text
<CTRL>-k	Cut ext from the cursor to the end of the line

Unix Variables

Variables are a way of passing information from the Unix shell to programs. Programs look "in the environment" for particular variables and if they are found will use the values stored. Some are set by the system, others by the user, yet others by the shell, or any program that loads another program.

Standard UNIX variables are split into two categories, shell variables and environment variables. In broad terms, shell variables apply only to the current instance of the shell and are used to set short-term working conditions; environment variables are those set at login and are valid for the duration of the session. The general convention is, shell variables have lower case and environment variables have UPPER CASE names though this depends on the shell you are using.

The two main shells available are bash and csh or tcsh. The information below relates to the bash shell.

The bash shell does not really distinguish between shell and environment variables. When a shell starts, it reads the information in the table of environment variables, defines a shell variable for each one, using the same name (also uppercase by convention), and copies the values. From that point on, the shell only refers to its shell variables. If a change is made to a shell variable, it must be explicitly "exported" to the corresponding environment variable in order for any forked subprocesses to see the change.

Shell Variables

An example of a shell variable is the USER variable.

```
% echo $USER
```

More examples of shell variables are:

- DISPLAY (the name of the computer screen to display X windows)
- HOME (the path name of your home directory)
- HOSTNAME (name of the host you have logged into)
- LOGNAME (your login name)
- PATH (the directories the shell should search to find a command)
- PROMPT_COMMAND (the text string used to prompt for interactive commands shell your login shell)
- PS1 (display prompt)
- PWD (your current working directory)

Shell variables are defined by assignment statements and are unset by the *unset* command. The format of the assignment statement is:

```
% NAME=value[; export NAME]
```

where there are no spaces around the equal sign (=). The unset command format is:

```
% unset NAME
```

where NAME is the variable name, and value is a character string that is the value of the variable.

Finding out the current values of variables.

SHELL variables can be both set and displayed using the *set* command. In the bash shell the *export* command can be used to export variables. To show the value of all shell variables, type

```
% set | less
```

ENVIRONMENT variables are set using the setenv command, displayed using the printenv or env commands, and unset using the unsetenv command. To show all values of these variables, type

```
% printenv | less
```

So what is the difference between PATH and path?

In general, environment and shell variables that have the same name (apart from the case) are distinct and independent, except for possibly having the same initial values. There are exceptions, however.

Each time the shell variables home, user and term are changed, the corresponding environment variables HOME, USER and TERM receive the same values. However, altering the environment variables has no effect on the corresponding shell variables.

PATH and path specify directories to search for commands and programs. Both variables always represent the same directory list, and altering either automatically causes the other to be changed.

References:

1. Haviland, K., Gray, G., & Salama, B. (1999). *UNIX system programming*. Addison-Wesley Longman Publishing Co., Inc..
2. Chan, T. (1996). *Unix system programming using C++*. Prentice-Hall, Inc..
3. Raymond, E. S. (2003). *The art of Unix programming*. Addison-Wesley Professional.
4. McGilton, H., & Morgan, R. (1983). *Introducing the UNIX system*. McGraw-Hill.