Getting Started with Linux

UNIVERSITY OF Southampton

Navigating the Linux Filesystem

The Linux filesystem is a tree-like hierarchy of directories and files.

ⓘ *When you first login to a Linux machine, you find yourself in your home directory.*

ⓘ *A path is a way you need to follow in the tree structure to reach a given file. An absolute path name is one beginning with the "/" character. A relative path is a path relative to your working directory*

Command	Description
pwd	"Print Working Directory". Shows the current location in the directory tree
cd *dir*	Change the current directory to *dir* Ex: `cd /user/src/redhat`
cd .. **cd -** **cd**	Move one directory up Return to previous directory Return to your home directory
ls	List all files in the current directory
ls -l	List file in "long format", one file per line. This also shows you additional info about the file, such as ownership, permissions, modification date and size
ls -a	List files which are normally hidden
ls -ltr	Sort file by modification time and list in reversed order using "long format". In this way recently modified files will be at the bottom of the list.

Command	Description
tree	List contents of directories in a tree-like format

Working with Files and Directories

Command	Description
mkdir	Make directory
rmdir	Remove an empty directory
cp *source dest* **cp -p …**	Copy a file Copy a file, preserving its attributes like mode, ownership, timestamps
mv *source dest*	Move a file to a new location or rename it.
rm **rm -r**	Delete a file Remove directories and their contents recursively
cat	Display the contents of a file on the screen
head **tail**	Display the fist/last few lines of a file
file	Find out what kind of file it is
more	Display a file or program output one screen at a time.

Command	Description
less	More sophisticated version of more (can scroll backwards and has many more options)
dos2unix	the program that converts plain text files in DOS/MAC format to UNIX format

Getting Help

1. Help on most Linux commands is build into the commands themselves:

```
$ ls --help
```

2. The best source of information for most commands is the online manual pages, known as "man pages" for short:

```
$ man ls
```

ⓘ *To search for a particular word within a man page, type "/word". To quite from a man page just type the "q" key.*

3. Sometimes you might not remember the name of Linux command and you need to search for it. For example, if you want to know how to change a file's permissions, you can search the man page descriptions for the word "permission" like this:

```
$ man -k permission
```

In the output you will find a line that looks like:

```
chmod    (1)    -    change    file    access
permissions
```

ⓘ *Choose the smallest number in the brackets (this number corresponds to manual categories: 1="General Commands",*

Getting Started with Linux

2="System Calls", the rest you may never need).

Now you know that "`chmod`" is the command you were looking for!

Basic Keyboard Shortcuts

- **`TAB`** button is used to complete commands and filenames
- **`Ctrl-c`** breaks/cancels an ongoing operation
- **`Ctrl-z`** pauses (stops) an ongoing operation. Type **"`fg`"** (foreground) to resume it or **"`bg`"** (background) to continue the process in the background
- **`Ctrl-d`** exit a terminal, same as typing "exit"
- **`Up/Down`** arrow keys to scroll through your most recent commands. You can scroll back to an old command hit **`Enter`** and execute the command without having to re-type it.

Creating and editing files

Linux comes with several editors. We show just two examples.

Nano

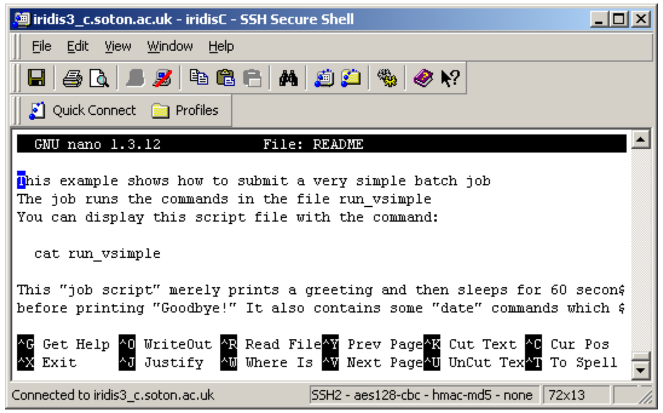
Nano uses very simple key combinations in order to work with files. A file is either opened or started with the command:

```
$ nano filename
```

where *`filename`* is the name of the file you want to open.

When you have the file open in Nano you will notice at the bottom of the terminal window a short list of command key-combinations. The carot symbol (^) means to hold <Ctrl> key. So, when you see ^X, that just means "hold the Ctrl key down and click x". All key combinations for Nano start with the <Ctrl> key:

- Ctrl-x - exits the editor. If you are in the middle of editing a file, Nano will ask you if you want to save your work;
- Ctrl-r - Read a file into your current working file. This enables you to add text from another file;
- Ctrl-w - Search your text;
- Ctrl-g Get help with nano;


iridis3_c.soton.ac.uk - iridisC - SSH Secure Shell
File Edit View Window Help
Quick Connect
Profiles

```
 GNU nano 1.3.12              File: README

This example shows how to submit a very simple batch job
The job runs the commands in the file run_vsimple
You can display this script file with the command:

  cat run_vsimple

This "job script" merely prints a greeting and then sleeps for 60 secon$
before printing "Goodbye!" It also contains some "date" commands which $

^G Get Help  ^O WriteOut  ^R Read File^Y Prev Page^K Cut Text  ^C Cur Pos
^X Exit      ^J Justify   ^W Where Is ^V Next Page^U UnCut Tex^T To Spell
```

Connected to iridis3_c.soton.ac.uk | SSH2 - aes128-cbc - hmac-md5 - none | 72x13

Nano editor home Page: http://www.nano-editor.org/

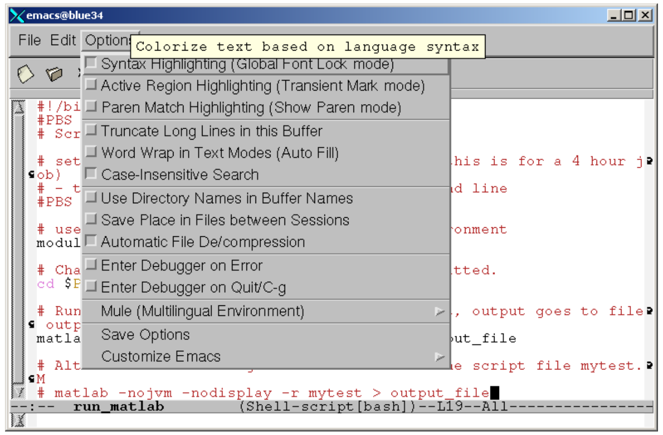
Emacs

In order to use emacs you must have an X-windows server installed and running on your machine. To start emacs:

```
$ emacs filename &
```

This command displays a new window in which you can view and edit the contents of *`filename`*. In this way you can use original window for commands, e.g. compilation. You can see a menu bar on the top of the emacs window, a tool bar just below the menu bar and then most of the remaining space is occupied by the editing buffer (where actual editing is performed). A scroll bar is on the left side of window.

One nice thing about emacs is that it provides color-syntax highlighting and automatic code formatting to help programmers.


emacs@blue34
File Edit Options
Colorize text based on language syntax
Syntax Highlighting (Global Font Lock mode)
Active Region Highlighting (Transient Mark mode)
Paren Match Highlighting (Show Paren mode)
Truncate Long Lines in this Buffer
Word Wrap in Text Modes (Auto Fill)
Case-Insensitive Search
Use Directory Names in Buffer Names
Save Place in Files between Sessions
Automatic File De/compression
Enter Debugger on Error
Enter Debugger on Quit/C-g
Mule (Multilingual Environment)
Save Options
Customize Emacs

```
# matlab -nojvm -nodisplay -r mytest > output_file
--:--  run_matlab        (Shell-script[bash])--L19--All----------------
```

Emacs editor home page: http://www.gnu.org/software/emacs/

Informational commands

Command	Description
ps	List currently running processes
df **df -h**	Report filesystem disk space usage Print sizes in "human-readable" format
du **du -h**	Disk usage in a particular directory Print sizes in "human-readable" format

Getting Started with Linux

Command	Description
top	Display CPU processes in a full-screen GUI. A great way to see the activity on your computer in real time. Type “q” to quit.
free	Display amount of free and used memory in the system
uname -a	Print system information (Kernel version, machine type etc)
w	Show who is logged on and what they are doing

Piping Command Together

The pipe character “|” is used to chain two or more command together. The output of the first command is “piped” into the next program, and if there is a second pipe, the output is send to the third programm, etc. For example:

```
$ ls -la /usr/bin | less
```

In this example we create a list of all files in /usr/bin directory. Because the output of this command is typically very long, we pipe the output to “less” command, which displays the output one screen at a time.

Redirecting Input and Output

Standard Output

There are times when it is useful to save the output of a command to a file instead of displaying it to the screen. For example, if we want to create a file with a list of all txt files in a directory, we can use redirection character “>”:

```
$ ls -l ~/*.txt > my_txt_files
```

To append the standard output to an existing file:

```
$ ls -l ~/*.txt >> my_txt_files
```

Standard Input

Many commands accept input from *standard input*, or *stdin* for short. By default, standard input reads information from your keyboard, but just like standard output, it can be redirected:

```
$ cat < some_input_file
```

Standard Error

This is where error output from your program goes. This normally points at your terminal as well, but you can redirect it:

```
$ cat non_existing_file 2> errors.txt
```

When using cluster, batch control systems like PBS manage Standard Output and Error redirection, so you do not need to worry about it.

Finding things

Command	Description
grep	Search for a pattern in a file or program output `$ grep <pattern> filename`
which	Shows the full path to a command or executable
whereis	Locate the program, source code and manual page for a command (if available)
history	Show your complete command history

Command	Description

find	A very powerful command, but sometimes tricky to use. It can be used to search for files by size, modification times as well as many other types of searches. A simple example is:
	`$ find . -name  *data*`
	This example searches in the current directory “.” and all subdirectories, looking for files with “data” in their names.
	`$ find . -size +500M`
	That would find files that are larger then 500 MB

Special Characters

Char	Description
/	Directory separator, used to separate a string of directory names. Ex: /bin/ls
\	Escape character. If you want to reference a special character, you must “escape” it with a backslash first. Ex: touch name\ with\ space
.	Current directory. Can also “hide” files when it is the first character in a filename.
..	Parent directory
~	Home directory
*	Represents 0 or more characters in a filename, or by itself, all files in a directory.

HPC support team
iSolutions
V1.0

Getting Started with Linux

UNIVERSITY OF Southampton

Char	Description
?	Represents a single character in a filename.
|	"Pipe". Redirect the output of one command into another command. Ex: ls | less
>	Redirect output of a command into a new file. If the file already exists, over-write it. Ex: ls > myfiles.txt
>>	Redirect the output of a command onto the end of an existing file. Ex: echo "all files" >> myfiles.txt
<	Redirect a file as input to a program. Ex: more < myfiles.txt
;	Command separator. Allows you to execute multiple commands on a single line. Ex: cd ~; less myfiles.txt
&&	Command separator as above, but only runs the second command if the first one finished without errors.
&	Execute a command in the background, and immediately get your shell back. Ex: find -name my* > find.results &

Other Utilities

Command	Description
clear	Clear the screen
echo	Display test on the screen. Useful when writing shell scripts. `$echo "Step 1:start execution"`
sort	Sort a file or program output

Command	Description
date	Display the current date and time
time	Runs the command and gives resource usage: `$ time ls`

Environment Variables

All the programs that run under Linux are called processes. When you start a program a new process is created. This process runs within what is called an environment. It looks "in the environment" for particular variables and if they are found will use the values stored. By convention, environment variables have UPPER CASE names. An example of an environment variable is:

```
$ echo $USER
```

To see all environment variables:

```
$ printenv
```

To set an environment variable:

```
$ export SVN_EDITOR=nano
```

Further reading

Link address
https://hpc.soton.ac.uk/community/projects/iridis/wiki
http://linux.co.uk/documentation/tutorials/unix-tutorial-0/
http://linux.org.mt/article/terminal

Contact Us

Email hpc@soton.ac.uk if you have any question regarding this document.