Getting Started with Linux

UNIVERSITY OF Southampton

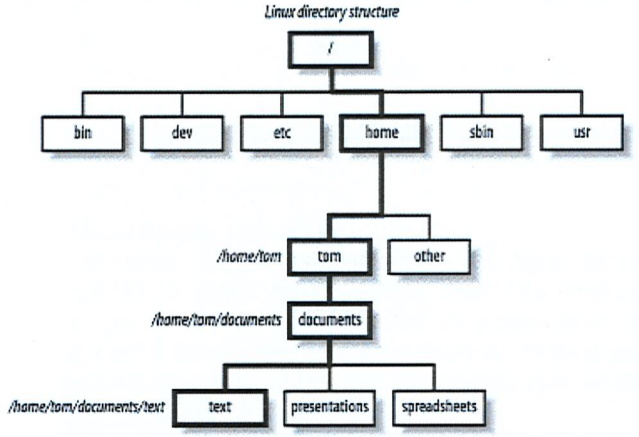
Linux Command Structure

$ command [-option(s)] [argument(s)]

- commands must be entered after prompt $ symbol and followed by <Return> key
- commands are case sensitive
- options and arguments may be optional
- options must come after the command and before any argument
- all options are preceded by a hyphen
- options may be grouped after the hyphen
- options are order independent
- the order of arguments may be important

Navigating 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 all files including files which are normally hidden (starting with ".")
ls -ltr	Sort files 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.
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

Command	Description
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 (FOREVER!) 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
more	Display a file or program output one screen at a time.
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`

Getting Started with Linux

UNIVERSITY OF Southampton

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", 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: vi/vim, nano, emacs, nedit, gedit *etc.*. We suggest you to start with **nano.**

Nano

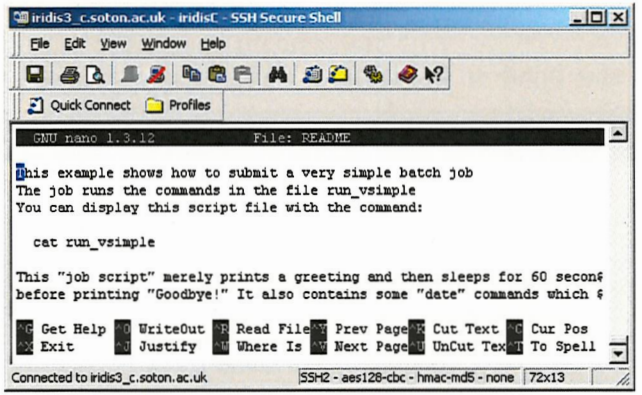
Nano uses very simple key combinations in order to work with files. A file is either opened or started with the command:

```
$ 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 caret 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;



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

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
`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

UNIVERSITY OF Southampton

Command	Description
uname -a	Print system information (Kernel version, machine type etc)
w	Show who is logged on and what they are doing

Piping Commands 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
history	Show your complete command history

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.

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 PATH=$HOME/bin:$PATH
```

this will add bin directory in your home to directories where shell searches for executables.

Further reading - online courses

Link address
Half a day introduction to The Unix Shell http://swcarpentry.github.io/shell-novice/

Free course from EDX covering many different aspects of Linux https://www.edx.org/course/introduction-linux-linuxfoundationx-lfs101x-1
The Ultimate Linux Newbie Guide http://linuxnewbieguide.org/

In addition, there is a number of Linux courses on Lynda.com

top → Jobs on node

Contact Us

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

> → FORWARDS OUTPUT OF COMMAND

nano x.txt to make file

cat to see entire file

less x.txt to parse page by page

ps -alF | grep [illegible] - all the processes you have running

quickreference

HPC support team
iSolutions
V2.0

Using the Batch System on Iridis 4

qsub -q gpu

UNIVERSITY OF Southampton

Batch system info

Iridis4 uses TORQUE (Terascale Open-source Resource and QUEue Manager) based on OpenPBS as resource manager to execute batch jobs on compute nodes. Moab serves as a job scheduler for Torque. In other words, Moab decides when a new job will run based on requested resources, system utilization and user's previous jobs. Users interact with batch system via Moab or Torque client commands.

Compute resources

Login to Iridis:

ssh	iridis4_a.soton.ac.uk
	iridis4_b.soton.ac.uk
	iridis4_c.soton.ac.uk

ⓘ *The login nodes are functionally equivalent, so it does not matter which one you use.*

ⓘ *Please note all CPU/memory intensive jobs MUST USE batch system. Login nodes should be used ONLY to access batch system, compile your code, debug and test simple scripts.*

Hardware:

name	Total number	# cores	Memory
green0NNN	750	16	64 GB
magenta0N	3	32	256 GB
yellow0N	12	16(Xeon Phi)	64 GB
indigoNN	12	16 (GPU)	64 GB

ⓘ *More information about GPU and Xeon Phi nodes can be found on Iridis wiki.*

File Store and Quota:

Path	Size	inodes
/home/$USER	100 GB	100000
/scratch/$USER	1000 GB	400000

ⓘ *The /home directory is fully backed-up. The files in /scratch are **never** backed up.*

ⓘ *To check your disk usage:*

```
$ mmlsquota -block-size g home scratch
```

mmlsquota home scratch --block-size g

Basic TORQUE/MOAB commands

Command	Description
qsub *<script>* **qsub -q test** *<script>* **qsub -q highmem** *<script>*	Submit job script to the scheduler in batch (default), test or highmem queue
qstat	Show status of batch jobs
qdel *<jobid>*	Delete a job
qdel all	Delete all jobs of a particular user from the queue
showstart *<jobid>*	Display the estimated start time of the job
showq (add/pipe grep)	Display information about all job on the system
checkjob -v *<jobid>*	Display detailed job state information
man *<command>*	Gives more details about the above commands

Resource Limits

Resource	Limit
ppn	16
nodes	32
idle jobs in queue	192
Wall-time	60:00:00 (60 hours)

ⓘ *If these limits would be breached by starting a new job then that job will not be allowed to start and will be marked as "blocked" by* **showq** *or* **checkjob -v**.

Job Script

The best way to control execution of your job is through the use of `#PBS` commands embedded in the job script. The job script is any shell script you normally run to execute your programs.

PBS Command	Meaning
`#PBS -l walltime=10:00:00`	Job will be allowed to run for a maximum of 10 hours
`#PBS -l nodes=n`	Allocate n nodes

check limits -
mmlsquota home scratch --block-size g.

qstat -n = show node jobs running on
q.stat -f

qsub.t?

HPC support team
iSolutions
Revision date: 23/02/2016

Using the Batch System on Iridis 4

UNIVERSITY OF Southampton

PBS Command	Meaning
`#PBS -l nodes=n:ppn=p`	Number of nodes and CPUs per node. For MPI job *n*p* defines the number of MPI threads
`#PBS -l mem=20gb`	Request 20 GB of memory
`#PBS -N myjob`	Set the job name

ⓘ *PBS commands must be at the top of the script and before any other command.*

ⓘ *Note: PBS starts from your home directory. Adding a* `cd` *command to take you to the directory where your files are is usually a good idea:*

```
cd $PBS_O_WORKDIR
```

each thread is 4gb, so using 4 = 16gb

Using an interactive session

For debugging purpose, it is often useful to run an interactive job. From a user's prospective, an interactive session is not very different from logging directly into and working on a compute node. However, there are few essential differences:

- User can ask for Interactive session on multiple nodes/processors
- PBS will automatically attempt to group jobs: if there is another user's job already running on a node and sufficient resources are available, interactive job will start on the same node.

To request an interactive session:

```
$ qsub -I -X
```

It can be grouped with other PBS directives:

```
$ qsub -I -X -l walltime=4:00:00
```

Examples

There are several examples of batch scripts for some common applications. To see a list of what is available type:

```
$ copy_example

No example specified, valid examples are:

amber ansys castep cfx comsol course
fluent gaussian matlab mpi openbugs
simple starccm very_simple
```

To copy desired example:

```
$ copy_example fluent
```

Every example contains a `README` file with detailed explanations.

Module commands

Iridis4 uses Environment Modules package to manage user's software

Command	Description
`module av`	Show currently available modules
`module list`	List currently loaded modules
`module load` *<module name>*	Load a module
`module unload` *<module name>*	Unload a currently loaded module
`module show` *<module name>*	Show what is changed by the module
`module help` *<module name>*	Give helpful usage information
`module initadd` *<module name>*	Module will be automatically loaded every time you login

ⓘ *If there are several versions of the same module you can choose the right one by specifying the full name:*

```
$ module load comsol/5.2
```

Further reading

Iridis4 wiki:

https://hpc.soton.ac.uk/community/projects/iridis/wiki

Iridis4 Forums:

https://hpc.soton.ac.uk/community/projects/iridis/boards

Note

All information in this Quick Reference Guide is valid as of February, 206. For most up-to date information, please, consult our web pages.

Contact Us

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

HPC support team
iSolutions
Revision date: 23/02/2016