Particle Physics Linux Systems quick reference

Logging in:

We currently have two general purpose interactive login machines, pplxint8 and pplxint9, which are both running the current Scientific Linux Six.

These are intended for general desktop type work (e.g. running text editors, compilers, interactive root jobs, short tests for large calculation jobs); long-running CPU intensive jobs should run on the batch system (see below).

- From a Linux machine or a Mac:
  - Open a terminal window and do 'ssh *username*@pplxint9.physics.ox.ac.uk'
- From a Windows PC:
  - Start PuTTY, and enter '*username*@pplxint9.physics.ox.ac.uk' in the 'Host Name' box. To run graphical applications, start eXceed first, then PuTTY, and make sure the 'Connection->SSH->X11->Enable X11 forwarding' option is turned on; step by step details are here: http://www2.physics.ox.ac.uk/it-services/ppunix/ppunix_remote_access_1

Basic command-line navigation:

- Where am I, what's here:
  - `pwd`: Displays the path to the directory you're currently in. Immediately after login this will be your home directory.
  - `ls`: list the contents of the current directory. Use 'ls -l' ('-l' = 'long') to get lots more detail.
  - `cd`: Changes the current directory. You can either give it:
    - a relative path: 'cd subdirectory'
    - an absolute path: 'cd /data/group/subdirectory'
    - or use it on its own to jump straight to your home: 'cd'
- Looking at stuff:
  - `less`: Previews the content of text files: 'less myfile.txt'
  - `cat`: Displays a file on the terminal; unlike less it doesn't let you scroll though long ones.
  - `clear`: Simply clears the terminal.
- Searching:
  - `locate`: find a file on the **local** system with a given name
  - `grep`: look for a text pattern in a file
- Chaining commands together with a pipe '|'
  - The output of a command can be used as an input to another. E.g. looking at the help for “rm” to find out how to force a file removal and ignore any prompts:

```
$ rm --help | grep force
-f, --force           ignore non-existent files, never prompt
```

- Making changes:
  - `mkdir`: Makes a new subdirectory, e.g: 'mkdir root_files'
  - `rmdir`: Deletes (empty) directories
  - `rm`: Deletes files: 'rm unwanted_file' Use 'rm -rf' to delete entire directories and all their contents. Carefully.

- Simplest way to edit files:
  - `gedit myfile.txt` (requires and “X11” graphical environment as above)
  - `nano myfile.txt` (does not require graphics)
- Getting help:
  - http://www2.physics.ox.ac.uk/it-services/ppunix
  - Quick help: “*command* --help” e.g. “rm --help”
  - Extensive help, use “man *command”,* e.g. for the remove command “man rm” (use q to quit)
  - And of course, google

- Exercise:
  - A very basic run through of bash, creating and running simple scripts and programs. Sections 1-6, 9-18 come up quite frequently. http://linuxconfig.org/Bash_scripting_Tutorial

Particle Physics software:

Your experiment may have its own recommended list of programs to use in analysis. Use those if they are available.

We also have centrally installed copies of some basic tools, including “root”. Since different groups often have different requirements we don't include them in the path by default, but have setup scripts that create the right environment. Instructions on using them are given in the 'message of the day' file that is displayed on log-in. You can look at it at any time by doing:

```
$ cat /etc/motd
```

Less commonly used programs are not advertised in the 'message of the day', but you can find these programs in a number of ways.

It is easy to get a list, and then load recommended programs.

```
$ module avail
$ module load root
$ root
```

Further programs (installed by collaborators from Oxford “Advanced Research Computing” centre but untested on our systems) can be accessed using

```
$ module load otheros/generic-arc
$ module load otheros/el6-arc
```

If you use programs in the otheros category and they work well or don't work at all please let us know.

Ownership, permissions and websites:

Programs and scripts in Unix need to be executable to run. You can use the chmod command to make the script executable.

```
$ ./myscript.sh
-bash: ./myscript.sh: Permission denied
$ chmod +x myscript.sh
$ ./myscript.sh
OK
```

In the spirit of good collaboration, all files on the Linux cluster are by default world-readable but writable only to yourself. The chmod command allows you to control access rights simply to restrict access to yourself or your group.

To change a file or directory from it's default permission set so that it is inaccessible to anyone outside your research group:

```
$ chmod o-rxw groupreadablefile.txt
```

Which means “modify the permissions for other people so that it is not readable, executable or writable”

To make a file or directory private:

```
$ chmod go-rxw privatefile.txt
```

Which means “modify the permissions for other people and people in the file's group so that it is not readable, executable or writable”

Personal web-space

We also run a webserver which can be used to make **any world-readable file accessible from the internet**. The easiest (but not only) way to use the webserver is by copying or linking a file you want to be internet-accessible into your “public_html” directory:

```
$ /bin/cp someonesfile.txt $HOME/public_html/internet-enabled-file.txt
```

Any files dropped into ~*username*/public_html/ will appear on the web at http://www-pnp.physics.ox.ac.uk/~*username*. This web-space is meant to function as a simple way to expose useful information or make files available to collaborators.

Private web-space is not provided as a standard service, however we can offer some advice on this.

Paths and environment variables:

Most of the common issues we see involve some combination of Linux environment variables being mis-configured, so that the application you think you are running is not quite as the author intended.

When an environment module is loaded, or a bash script is 'sourced', all it is really doing is telling Linux where to look to find files of a given type. The most common three that I see mis-configured are:

PATH – where to look for programs and scripts.

LD_LIBRARY_PATH – where to look for shared libraries to load into my program when it runs.

PYTHONPATH – where to look for python libraries and modules when my python program runs.

Experimental software setup scripts and our system scripts will configure the environment so that it knows where to find the relevant files. However, if two different versions of a piece of software are configured at the same time, it can lead to real and sometimes subtle issues.

When debugging, start with a clean environment (shell) every time and don't be tempted to automatically load programs on login. If you do, and find odd things are happening with your programs, try removing any extra lines from the login scripts and directly calling only the setup scripts that you need.

Using the batch system:

Basics:

As well as our general purpose interactive login machines we have a cluster of nodes intended for running long, CPU intensive calculation jobs. These machines can't be logged into directly, and instead are accessed by submitting jobs to a queuing system, which finds a free job slot (or waits for one to become free), runs the job, and gives the output back.

Jobs on the batch cluster have to be completely non-interactive; you can't pass them parameters and they can't stop and ask for input. The usual way of dealing with this is to prepare a script that sets up everything the job needs and runs it with any required parameters, and then submit that to the batch system. The job script can be any basic script that could run on the interactive machines, and the batch worker nodes have the same access to home directories, /data disk areas and software area as the interactive machines do.

To run a job, create a self contained script that does what you need, e.g (~demo/jobscript):

```
#!/bin/bash

pwd
hostname
sleep 10s
echo Hello World
```

and submit it to the queue with the 'qsub' command, which will give you a job number in return:

```
$ qsub jobscript
390410.pplxtorque05.physics.ox.ac.uk
```

You can get basic information on your jobs using qstat:

```
$ qstat
Job id                  Name             User            Time Use S Queue
------------------   ---------------- --------------- -------- - -----
390410.pplxtorque05   jobscript        demo                   0 R normal
390411.pplxtorque05   jobscript        demo                   0 Q normal
```

In this case the first job is running, and the second is waiting for a free slot (see the 'S'='state' column, 'R'='Running', 'Q'='Queuing'), when a job is finished it will simply disappear from the list.

Details:

The qsub command takes parameters that describe the job's requirements to the batch system. On our cluster the one that's commonly useful is the one to set the job's CPU time requirement, since **the batch system gives a priority boost to shorter jobs**. This takes a time in hh:mm:ss format, e.g. for a one hour job:

```
$ qsub -l cput=01:00:00 jobscript
```

Note: The batch system will kill the job after it's used the specified amount of time, so allow a factor 2-3 safety margin. If you don't specify a time the batch picks a default of a week.

By default, your jobs will send their output back to two files. One is the standard **o**utput and another any **e**rror messages for example jobscript.**o**390410 and jobscript.**e**390410. Useful debug information may be found in this file, for example the actual CPU time requirement is printed out.

```
$ cat jobscript.o.390410
*    Job Terminated at Fri Jun 15 13:29:07 BST 2012
*
*    Job Used
*
*    cput=00:00:14,mem=5260kb,vmem=195264kb,walltime=00:05:33
*
********************************************************************
```

Exercise:

Read through www2.physics.ox.ac.uk/it-services/ppunix/particle-physics-linux-batch-farm.

Other hints and tips:

Using scratch disks in batch jobs

The batch system creates a temporary directory for each job on the worker node's disk. For some jobs that are particularly disk-intensive it can be a good idea to locate files here, rather than accessing them directly over the network. **In your batch script**, add lines like:

```
cp big_input_file ${TMPDIR}
./do_some_physics_on ${TMPDIR}/big_input_file
cp ${TMPDIR}/big_output_file /data/my_experiment/my_job_path
```

The temporary area is cleared automatically at the end of a job, so the job script must save anything that needs to be kept back to the data disk (or home directory). Most jobs don't need to bother with this though.

Transferring files to the Linux cluster

You may find that you have an email or download of a file that you would like to copy to your working area on the cluster.

On **personal Windows** machines, a tool called WinSCP can be downloaded. With this you can log in to pplxint5/6/8/9 in the same way as with PuTTY and be provided with with a view of your Windows “C:” drive on the left and your Linux “home” folder on the right, into which you can drag and drop files and folders.

On **managed Windows desktops** (i.e. those provided by the department) your PP UNIX home disk will also appear at the following path:

Y:\LinuxUsers\pplinux\home\yourname

There is also a '**central Linux'** home directory. When you use a **managed Ubuntu** system, this is what you see in **/home*.**

*Except for Hans Kraus' and Sam Henry's groups

To first connect the Y: drive to your Windows laptop, simply type this into a command prompt on windows:

```
    net use Y: https://winfe.physics.ox.ac.uk/ password /USER:yourname
```

- filling in *yourname* as your physics user name

For other platforms, look into how to use secure webdav

Talk to us

This does not cover how to use everything or even begins to touch on some of the infrastructure that exists to perform less common tasks. I expect a solution will have already been devised to solve most of the computing problems you will encounter. If you are finding that you are struggling with a particular issue, come and talk to us in room 661 or drop us a mail at pp_unix_admin@physics.ox.ac.uk.