Using the bash shell and running software

If you are completely unfamiliar with using the linux command-line interface or with the bash shell, you should probably read the following external document:

- *Learning the shell, a longer narrative document.*

What follows is a bullet-point reference that should be able to get you up to competency with a little bit of study and trial-and-error.

The basics

ls	List directory contents. ls -l long listing with dates and permissions ls -a list all, including hidden files (hidden files start with a period).
cd	Change directory. cd - change to previous directory cd ~ change to home directory cd .. change to parent directory
pwd	Print current (working) directory.
cp	Copy file(s). cp -r somedir somedest Copy recursively (directory and all contents)
mv	Move (or rename) a file (or directory)
rm	Delete (remove) a file. rm -r somedir Delete recursively (directory and all contents) rm -ri somedir Delete recursively, but prompt before each removal rm -rf somedir Delete recursively, never prompt
mkdir	Create directory.
rmdir	Remove (empty) directory.
cat	Print file contents (to console).
less	Scroll through file contents, one page at a time. SPACEBAR to advance a page, B to go back a page, Q to quit, /pattern to search for pattern
who	See what other users are logged in.
htop	See resource utilization on your current machine.
exit	Close shell. If this is your login shell, you will log out.

Editing files

There are a number of commandline programs for editing files. You might want to check out and choose between any of the following three:

- *nano - good for beginners.*
- *vim - very powerful, very flexible.*
- *emacs - very powerful, very flexible.*

Note

You'll know from reading the document on *File Management - Getting your files to and from RC systems* that RC systems share a unified file system. If you edit a file on one machine (say lmc.rc.rit.edu) that change will be reflected on other machines (say werner.rc.rit.edu).

File redirection and pipes

The bash shell interprets some characters in a command to do some pretty wild stuff. In combination, they can be very powerful and make your life much easier.

|	The *pipe*. Redirect the output from one program to another. myprogram | less will take the output of your program and page it through the *less* command.
<	Input file redirection. myprogram < input.txt will read all input from the file *input.txt* instead of from the command-line. Useful for automating experiments.
>	Output file redirection. myprogram > output.txt will save all output from your program to *output.txt* instead of printing it to the command line.

You can see more examples at the following two external links:All about redirectionI/O Redirection

Using software

Research Computing has a number of third-party software packages installed (both with restricted licenses and open source). We use the module s package to mitigate any conflicts between the environments and installations of these different tools.

To see what software is available, you can use the module avail command. For example:

```
[abc1234@brodie ~ []]$ module avail

---------------------------------- /etc/modulefiles ----------------------------------
mpich2-i386   mpich2-x86_64

----------------------------- /tools/Modules/modulefiles -----------------------------

-
ampl/20091123              cuda/2.3                    MapSplice/1.15.2
ampl/20091123-stu(default) cuda/4.0.17                 mathematica/10.0
ampl/20130226              cuda/4.1.28                 matlab/2011b
ansoft/12                  cufflinks/2.0.2b            matlab/2012b
ansys/15.0.7               eigen/3.0.4                 maya/2012
boost/1.48.0               envi/6.3                    module_archive
bowtie/0.12.5              envi/8.0                    omnetpp/4.6
cadence/5.1.41             fluent/13.0.0               pcl/1.4.0
cadence/6.10               fluent/15.0.0               R/2.15.3
cadence/6.13               font_server/1.0.2           R/3.0.0
caldb/caldb-413            g95/0.90                    R/3.0.1
casava/1.7.0               gcc/4.4                     R/3.0.2
ciao/ciao-412              glpk/4.45                   R/3.1.1
cmake/2.8.7                gurobi/4.6.1                samtools/0.1.17
comsol/cheme-35a           gurobi/5.1.0                sentaurus/E-2010.12
comsol/meche-34            harpoon/3.6a                silvaco/multi
comsol/meche-35a           IntelClusterSuite/2013xe    SpliceMap/3.3.5.2
comsol/microsystems-33     knitro/7.0.0                system_defaults
comsol/smfl-40a            lumerical/8.11.318(default) tophat/1.3.1
comsol/smfl-41             lumerical/8.9.269
cplex/12.4                 maple/15
```

We no longer run CentOS 5.5, the following, while vaild, isn't 100% accurate

For instance, the operating system we run (CentOS 5.5) comes with python 2.4 by default (which is pretty old). Many users need access to the latest version of python. You can check which version of python is available in your $PATH and load the newest version with the following:

```
[abc1234@brodie ~ []]$ python -V
Python 2.4.3
[abc1234@brodie ~ []]$ which python
/usr/bin/python
[abc1234@brodie ~ []]$ module load python/2.7.1
[abc1234@brodie ~ []]$ python -V
Python 2.7.1
[abc1234@brodie ~ []]$ which python
/tools/python/2.7.1/bin/python
```

Some packages are a little trickier. For instance, many imaging scientists need access to the idl programming environment. Our module avail command doesn't list idl explicitly, but it *is* contained with the envi package. The following example demonstrates this:

```
[abc1234@brodie ~ []]$ which idl
/usr/bin/which: no idl in (/usr/kerberos/bin:/usr/local/bin:/bin:/usr/bin:/tools/condor/7.4.1/bin:/tools
/bin:/tools/bin:/home/abc1234/bin)
[abc1234@brodie ~ []]$ module load envi
[abc1234@brodie ~ []]$ which idl
/tools/rsi/idl_6.3/bin/idl
[abc1234@brodie ~ []]$ idl
IDL Version 6.3 (linux x86_64 m64). (c) 2006, Research Systems, Inc.
Installation number: 214968-1.
Licensed for use by: Rochester Institute of Technol

IDL>
```

Extras

We also have the following commands installed that are quite helpful, if nonstandard.

resource-therapist	Determine what resources you are consuming.
hardlink.py	Hardlink identical files together. Conserve disk. hardlink.py -n somedir to do a dry-run.