Introduction to UNIX

Manual

Michael Gribskov

10/5/2017
Contents

Connecting to servers ........................................................................................................ 4
PUTTY ............................................................................................................................... 4
SSH ................................................................................................................................. 5

File Transfer .................................................................................................................... 5
scp <local file> <remote file> – secure copy ................................................................. 5
sftp – secure file transfer using FTP ............................................................................... 5
wget <URL> – retrieve a file from web sites .............................................................. 5

Basic UNIX ...................................................................................................................... 6
Unix basics references .................................................................................................. 6
ls – list files in current directory .................................................................................. 6
mkdir <new_directory_name> – create a new directory (folder) .................................. 6
cd <directory_name> – change directory ..................................................................... 7
cp <original_file_name> <new_file_name> – copy a file or files ............................... 7
mv <original_file_name> <new_file_name> – rename a file without creating a copy .... 7
rm <file_name> – delete (remove) a file ........................................................................ 7

UNIX utilities .................................................................................................................. 7
cat <file_name> – print an entire file to the terminal screen ..................................... 7
less/more <file> – file viewer; print a file to the screen one page at a time ........... 7
pwd – print working directory ..................................................................................... 7
wc <one or more files> – word count .......................................................................... 8
head <file name> – print the first lines of a file ......................................................... 8
tail <file name> – print the last lines of a file ........................................................... 8
grep <pattern> <file name> – find information in a file ............................................ 8
More Information .......................................................................................................... 9

UNIX command line .................................................................................................... 9
Redirecting output and input ....................................................................................... 9
Piping .............................................................................................................................. 10
Running in the background ......................................................................................... 10
Symbols ......................................................................................................................... 10
More Information .......................................................................................................... 10

PBS Batch Jobs ............................................................................................................. 11
qsub <job_file_name> – submit a Torque job ............................................................ 11
qstat – examine the status of a queue or job ............................................................. 11
qdel <job_number> – delete a job from a queue ......................................................... 12
nano <file> - preparing job files ................................................................. 12

Module System ................................................................................................. 13

module avail – show modules installed on this server ........................................... 13
module load <module_name> – load a specific module for use .................................. 13
module show <module_name> - show details of installation of a specific module ........ 13
module list – show a list of currently installed modules ........................................ 13
module unload <module_name> - unload a specific module ..................................... 13

Practical 1. Hands-on UNIX .............................................................................. 14
Connecting to servers

To connect to the servers used in this workshop you will either use a direct connection (SSH) or connect through using a terminal emulator, a piece of software that makes your local computer appear to be a terminal directly connected to the server. How you connect is primarily determined by whether your local workstation is a hardwired terminal in a classroom or computer center, an Apple Mac running OsX, or a Windows PC. In this workshop we will use the following servers

- snyder.rcac.purdue.edu

Your username and password is the same as your Purdue career account.

PUTTY

PUTTY is a free open-source terminal emulator available for Windows and UNIX. It is pre-installed on many systems, but if it is not available, you may download it from the URL below. PUTTY gives you a lot of control over the size and color of the screen, you should read the documentation to find out how to modify the appearance of the terminal or other parameters.

webpage: http://www.chiark.greenend.org.uk/~sgtatham/putty/
download: http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html
documentation: http://the.earth.li/~sgtatham/putty/0.64/htmldoc/

When you start PUTTY, you will see a configuration screen (right). Type the name of the server you want to connect to in the box labelled “Host Name (or IP address)”, and click open. The first time you log in to a server, you will usually see a security screen asking if you want to save the server’s rsa2 host key (below, left); you should answer “yes”. You will only see this security screen the first time you connect to a server, otherwise you will proceed directly to a terminal screen (below, right) where you can login with your username and password.
SSH

SSH (Secure Shell) provides a text-based terminal emulator on UNIX computers. The SSH connection is encrypted (hence, it is secure). You will generally use SSH when connecting to a server from another UNIX computer, including from Apple Mac computers. From a Mac, you need to first open a terminal window to get access to the UNIX layer of the MacOs. From a Linux or other standard UNIX system you should simply have to type “SSH <servername>” at the command prompt, where <servername> is the name of the server you are connecting to, for instance, snyder.rcac.purdue.edu. The first time you connect to a server you will be asked to confirm the RSA key fingerprint of the server; answer “yes” (“y” is not sufficient).

The authenticity of host 'snyder.rcac.purdue.edu (128.211.148.51)' can’t be established.
Are you sure you want to continue connecting (yes/no)?

File Transfer

We often have to transfer files between a compute server and a local server, or even a personal laptop. Many programs are available for this, but which ones you can use varies according to how the computers have been set up. Some of the possibilities are listed below.

scp <local file> <remote file> – secure copy
scp <remote file> <local file>

Secure copy is convenient way to copy one or many files between a local computer and a UNIX systems (such as RCAC). The following examples assume you are running scp while logged into UNIX system. If you use a Mac, scp is built in. If you use a Windows computer, there are many free scp clients available. I use WinSCP (http://winscp.net/eng/download.php).

<table>
<thead>
<tr>
<th>(to a remote system from local)</th>
</tr>
</thead>
<tbody>
<tr>
<td>% scp sourcefilename myusername@hostname:somedirectory/destinationfilename</td>
</tr>
<tr>
<td>% scp trinity.pl <a href="mailto:gribskov@ser.genomics.purdue.edu">gribskov@ser.genomics.purdue.edu</a>/. (dot means use same filename)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>(from a remote system to local)</th>
</tr>
</thead>
<tbody>
<tr>
<td>% scp myusername@hostname:somedirectory/sourcefilename destinationfilename</td>
</tr>
<tr>
<td>% scp <a href="mailto:gribskov@ser.genomics.purdue.edu">gribskov@ser.genomics.purdue.edu</a>/trinity.pl .</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>(recursive directory copy to a remote system from local)</th>
</tr>
</thead>
<tbody>
<tr>
<td>% scp NGS/ <a href="mailto:gribskov@ser.genomics.purdue.edu">gribskov@ser.genomics.purdue.edu</a>/NGS/</td>
</tr>
</tbody>
</table>

sftp – secure file transfer using FTP

A secure ftp client, such as SecureFX can also be used for transferring files. If you use SecureCRT for a terminal emulator, just open SecureFX from your personal computer and connect to any one of the RCAC. If you are using PUTTY, you can download PSFTP from the same site. After starting a SFTP client, you will be able

wget <URL> – retrieve a file from web sites

Both data files and programs are often available on the internet for download using http. wget is a simple way to download such files. You will generally get better results downloading such files directly to
the UNIX system on which you wish to use them, than by downloading the files to your personal computer using a web-browser and then transferring them to the UNIX server.

**Basic UNIX**

UNIX is a text oriented operating system that has been around since the 70s, and is the primary operating system used at high performance computing facilities, as well as underlying the Mac OSX graphical operating system. You interact with the computer via a *shell*. There are several UNIX shells, but the most common is probably *bash* (Bourne again shell), but differences between shells are not very important unless you are going to write *shell scripts* to automate your work. In this class we are only introducing the smallest possible set of commands needed to work in a UNIX environment — Some of the references below give much more detail and an introduction to shell scripting.

Unix basics references
- Unix tutorial for beginners, [http://www.ee.surrey.ac.uk/Teaching/Unix](http://www.ee.surrey.ac.uk/Teaching/Unix)
- Introduction to UNIX, [http://www.doc.ic.ac.uk/~wjk/UnixIntro](http://www.doc.ic.ac.uk/~wjk/UnixIntro)

`ls` – list files in current directory
- `-l` long listing, includes file date and size
- `-h` show file sizes in human readable terms
- `-t` show the newest files first

```
% ls
accepted_hits.bam  align_summary.txt  deletions.bed  insertions.bed
junctions.bed      logs            prep_reads.info  unmapped.bam
% ls -lh
total 1.5G
-rw-r--r--  1 mgribsco  mgribsco-genomics 157M Apr  8 16:20 unmapped.bam
-rw-r--r--  1 mgribsco  mgribsco-genomics 603M Apr  8 16:19 accepted_hits.bam
-rw-r--r--  1 mgribsco  mgribsco-genomics 238K Apr  8 16:17 deletions.bed
-rw-r--r--  1 mgribsco  mgribsco-genomics 714K Apr  8 16:17 insertions.bed
-rw-r--r--  1 mgribsco  mgribsco-genomics  199 Apr  8 16:17 junctions.bed
-rw-r--r--  1 mgribsco  mgribsco-genomics  238 Apr  8 16:17 logs
```

`mkdir <new_directory_name>` – create a new directory (folder)

UNIX directory names should not have any spaces. You can specify either a relative path from the current directory (example 1 below) or an complete (absolute) path (example 2 below)

```
% mkdir raw_data
% mkdir /users/mgribsko/magnaporthe/cleaned_data
```
cd <directory_name> – change directory
You can specify either a relative path from the current directory (example 1, below) or a complete path (example 2, below). In relative paths, remember that “..” indicates the parent directory. If you type cd without providing a directory name, you will be returned to your home directory.

% cd raw_data
% cd /users/mgribsko/magnaporthe/cleaned_data
% cd ../raw_data

Example 1

Example 2

cp <original_file_name> <new_file_name> – copy a file or files
The original file remains in its original location, and a new file with the same content is created in the new location. When a file is copied to a different directory, you can omit the file name (or use “.” for the name) and the new file will have the same name as the old file. Remember that you may use wildcards such as “*” (match any number of letters) or “?” (match any one letter) in the original file name to copy large numbers of files.

mv <original_file_name> <new_file_name> – rename a file without creating a copy

rm <file_name> – delete (remove) a file
BEWARE. A file cannot be recovered once it is removed. It is gone forever. rm can be used with wildcards to remove many files at once, or with a list of files. Be careful though, a simply typographical error can change a command that removes all temporary files, e.g. rm *.tmp (remove all files with the suffix .tmp), into a catastrophe that removes all your files rm * tmp (remove all files, *, then remove the file named tmp).

• -R recursively remove all files and directories

% rm wt.R1.fq raw_data/hen3.R1.fq
% rm *.fq
% rm -R tmpdir

UNIX utilities

cat <file_name> – print an entire file to the terminal screen
A simple way to see the entire contents of a file. It is not very useful for large files since they scroll fast too fast to see. Usually you will want to use less or more (see below)

less/more <file> – file viewer; print a file to the screen one page at a time
less and more are very similar file viewers. Every UNIX system has one or the other, or both available. On many systems less and more are equivalenced, so that typing more actually runs less (leading to the UNIX quip “more is less”). You can advance your position in the file one line at a time using enter/CR or a page at a time by typing space. Both less and more let you search for keywords in the text; less is slightly more powerful because it lets you search and move both forward and backward. If available you should probably use less. See the man pages for more detail on less/more.

pwd – print working directory
Simply prints out the complete path to your current working directory. Useful if you forget what directory you have moved to with cd.

% cd src
% pwd
/home/mgribsko/src
wc <one or more files> – word count

Determines the number of lines, words, and bytes in a file. This command is very useful for determining, for instance, the number of reads in a FASTQ file. Every entry in a FASTQ file has four line so the number of reads is the number of lines divided by four. If the number is not evenly divisible by four, you have a problem. The two files, C01.R1.clean.fq and C01.R2.clean.fq, in the example below, each contain 446,660 reads. If you don’t want to see the number of words and bytes, use the option –l.

- -l  print only the number of lines

head <file name> – print the first lines of a file

In the example below, I examine the first lines of a FASTQ file, perhaps to confirm that it really is in FASTQ format. By default, head will show the first 10 lines of the file, but you can control this with –n.

- -n  print the specified number of lines

tail <file name> – print the last lines of a file

tail works just like head, but shows the last lines of a file. Useful for testing whether the file is complete. By default, tail will show the first 10 lines of the file, but you can control this with –n.

- -n  print the specified number of lines

grep <pattern> <file name> – find information in a file

grep is one of the most flexible and useful UNIX utilities. grep reports all of the lines in a file that contain a query pattern. The query pattern can be a file name, a sequence, or anything that you are looking for in a long file. If the query pattern contains spaces or special characters (such as $) you must place it in quotes.

- -c  report the number of matching line s(count)
- -v  report the lines that do not match
- -A  the number of lines after the match to print
- -B  the number of lines before the match to print
- --color=always  color the matching letters (usually in red)
More Information

Most UNIX commands and utilities have many options. You can (usually) find more information by using the `man` utility. `man <command>` displays a description of the command (or utility), a synopsis of options (switches), and often provides a simple example.

**UNIX command line**

Redirecting output and input

UNIX programs typically produce two kinds of output, `STDOUT` and `STDERR`, which are the main result of the program, and any extra messages of error reports, respectively. The author of the program controls the content of these outputs, so the usage is not completely standard; some programs, for instance, may write their main output to `STDERR`. Other programs may write their output to files, whose names you may, or may not, be able to specify.

By default, `STDOUT` and `STDERR` are sent to your terminal screen. This is not very convenient if you want to save the output; perhaps, the output is the assembled scaffold sequences of your genome! In this case you will want to redirect the output to a file. In UNIX this is done with the “>` operator. `STDOUT` and `STDERR` must be redirected separately and UNIX shells handle this slightly differently. The examples below are for the `bash` shell.

Input to a UNIX program is often read from `STDIN`, which is, by default, the terminal keyboard. When running on a server or compute cluster, you will often need to read the input for a program from a file. Redirecting `STDIN` is similar to redirecting `STDOUT` and `STDERR`, and is done using the “<” operator. For much more detail see [http://www.catonmat.net/download/bash-redirections-cheat-sheet.pdf](http://www.catonmat.net/download/bash-redirections-cheat-sheet.pdf).
Piping

One of the reasons people like UNIX is that it allows the output of one program (STDOUT) to be sent directly into the input of another program (STDIN). This is called piping and is a powerful tool to combine simple commands to accomplish complex goals. Piping is used extensively in NGS analysis. The example below shows a simple example of piping; `grep` is used to identify matches to the beginning of an adapter sequence, and the result is piped to `head` to display the first ten matches.

```
grep --color=always AATGATACGGCG SRR1265029.fastq | head
```

Running in the background

Normally, when you run a UNIX program or command, you must wait for the command to finish before you can use the terminal again. This is called running a command in the foreground. Alternatively, UNIX can start the command, and return control of the terminal to you while the command continues running until it finishes. This is called running a command in the background. To run a command in the background, include the operator “&” at the end of the command. Every running job on the computer, consume resources (CPU time and memory); if you start many background jobs simultaneously, your computer may become very slow and some jobs may even crash. If you run jobs from a command file on a server or compute cluster, you must realize that, after starting the background command, the PBS system will immediately move to the next command. If there are no further commands, the PBS job will terminate, along with all of the background commands it has started. To avoid this, include the command `wait` as the last command in the job file, which will cause the system to wait for all background commands to complete before proceeding.

Symbols

Symbols are a shorthand way to refer to complicated paths and other information. Usually, symbols are a short string, often in all capital letters, introduced by $. Symbols are defined in `.bash_profile` in your root directory. If the server uses the module system, one of the main functions of a module is to define any symbols and paths used by a particular piece of software. As this information is completely dependent on the program, having a common definition is a great convenience. In `bash` you can see your currently defined symbols by typing `set -o errexit`. Some common symbols include

- `$HOME` – the user’s home directory
- `$SCRATCH` – a scratch directory for temporary storage. RCAC defines `$RCAC_SCRATCH` instead.
- `$HOSTNAME` – the name of the server
- `$SHELL` – the shell being used

More Information

There are many online tutorials and print books that explain how to use UNIX in more detail. The description above barely touches the surface and tries to cover just those commands that we will use immediately. If you plan to continue performing NGS analysis, you should eventually become more familiar with UNIX – this will also speed up your work as well as give you much greater flexibility in doing novel analyses. You may also need to learn a scripting language such as Perl or Python in order to make your own programs or pipelines. A few suggestions for additional study are listed below.
Unix basics

- Unix tutorial for beginners, [http://www.ee.surrey.ac.uk/Teaching/Unix](http://www.ee.surrey.ac.uk/Teaching/Unix)
- Introduction to UNIX, [http://www.doc.ic.ac.uk/~wjk/UnixIntro](http://www.doc.ic.ac.uk/~wjk/UnixIntro)

Bash shell


PBS Batch Jobs

Compute servers, especially large cluster systems, usually require you to create a command file and run your jobs by submitting the command file to a queue. Job queues have different characteristics such as which research groups can run jobs on the queue, limits on memory and/or CPU time, number of processors, and access to storage. Many computer centers, including RCAC at Purdue, use Torque (Terascale Open-source Resource and QUEue Manager) as a queuing system to manage computer resources.

Torque provides commands for defining the resources that a job requires, submitting jobs to specific queues, and monitoring job queues and job status.

qsub <job_file_name> – submit a Torque job

A job file is simply a text file that contains the same commands that you would type on an interactive system. The Torque environment also provides a few useful symbols that make it easier to run jobs (see preparing job files). The command line options below can also be embedded in the job file (see preparing job files).

- `-q <queue_name>` specifies the queue the job is to run on. At Purdue, if you do not specify this, your job will run on a queue called `standby`.
- `-N <string>` specifies the name that will be shown for the job in the queue. This is completely optional.
- `-l walltime=HH:MM:SS` maximum CPU limit for the job. At Purdue, if you do not specify a time limit, the limit will be set to 00:30:00 (30 minutes).
- `-l nodes=1:ppn=<int>` specifies the number of processors to be used for the job. The requested number of processors must be equal or smaller than the number of processors per node available on the server. On the Rice (Scholar) and Snyder servers at Purdue there are 20 processors per node.

qstat – examine the status of a queue or job

qstat is used to examine the status of jobs that are queued (waiting) or running. The second to the last column, labeled S show the job status. Q, R, H, C, E which mean queued, running, held, completing, or exiting, respectively. A job with status H is waiting for an event specified when it was submitted, usually for another job to finish.

- `-a <queue_name>` - show all entries in the queue
- `-u <username>` - show all jobs owned by the specified user
qdel <job_number> – delete a job from a queue
Kills the job if it is running, or simply deletes it if it is queued.

nano <file> - preparing job files
To prepare a job file, you must learn to use a text editor. While you can prepare a file on your local computer or laptop and transfer it to the server, it is usually more convenient to create and edit the file on the server itself. There are many editors available (for instance vi, emacs, pico and nano) and if you are familiar with an editor you can use the one you are already familiar with. If you do not already know a UNIX editor, we recommend that you use nano. One of the advantages of nano is that the commands to exit the editor and write out a file are shown at the bottom of the screen (^ indicates Ctrl should be held down while typing the letter).

The basics of nano are simple (see figure):
- type nano at the command prompt to start the editor
- type your commands into the screen
- move around with the arrow keys if necessary
- save your file with the ^o key, and provide a name for the file
- quit the editor with ^x

The commands that specify the job name, queue, number of processors and time limit can be included in the job file. I find this very convenient. An example is shown in the nano figure, below. The first line is called the shebang line and specifies that this job file is a bash script. Don’t worry about this for now, just include it. The #PBS commands must come after the shebang line and before any other commands (blank lines are OK). In the job file above, the symbol SPBS_O_WORKDIR is used to move to the directory from which...
the job is submitted. Your job files are usually in the directory where you are working, often in a scratch area, so this is the normal practice.

Module System

Many servers use the module system to simplify the task of running complex software. Often, software must be installed in a particular location, with particular directory structures, and symbols must be defined to access software and data directories. Some of these operations can only be performed by system administrators. The module system allows the system administrator to install the software and makes it easy for users to find and correctly initialize packages.

module avail - show modules installed on this server

Depending on the diversity and number of users on the server, the list of installed modules can be very long. Often, especially in computational biology, there will be more than one version of a specific program. Generally, but not always, the most recent version will be the default and you need not specify a specific version number unless you want another version (for instance, if you want to repeat a prior analysis). In the example below, only a partial list of modules available on my system is shown.

module load <module_name> - load a specific module for use

module show <module_name> - show details of installation of a specific module

This information may be necessary to find data files or utilities that are outside the main program. In also includes pointers to documentation and test files, as well as details about the program version.

module list - show a list of currently installed modules

module unload <module_name> - unload a specific module
Practical 1. Hands-on UNIX

1. Log on to server.
2. Use `pwd` to determine the path to your home directory
3. Use `ls` to examine your home directory
4. Move to your scratch directory (`$RCAC_SCRATCH`) using `cd`
5. Create a working directory for the class project.
6. Data
   a. Download the adapter.fa datafile from the wiki and upload to the server.
   b. The sequence datafiles will be made accessible in a common location on the server.
7. Use `wc` to determine how many reads are in each datafile
8. Use `head` and `tail` to examine the first and last few reads in the data file
9. Practice redirecting files by sending the output from `head` to a file
10. Practice piping by piping the output from `cat` to `wc`