Introduction to UNIX

Manual

Michael Gribskov

V2018.3

24 August 2018
## Contents

Connecting to servers ........................................................................................................... 4
  Windows ............................................................................................................................... 4
   MobaXterm (recommended) .............................................................................................. 4
   PUTTY ............................................................................................................................... 5
OS X/MacOS/Linux .................................................................................................................. 6
  SSH ................................................................................................................................... 6
  Cyberduck .......................................................................................................................... 6
   Edit server files .............................................................................................................. 7
   Transferring files ............................................................................................................ 7
Purdue University Specific ....................................................................................................... 7
  SecureCRT/SecureFX ........................................................................................................ 7
  ThinLinc ............................................................................................................................ 7
UNIX .................................................................................................................................... 9
   Unix basics references .................................................................................................... 9
UNIX command line .............................................................................................................. 9
   Redirecting output and input ......................................................................................... 9
   Piping ............................................................................................................................... 9
   Running in the background ............................................................................................. 10
   Symbols ........................................................................................................................... 10
   More Information ........................................................................................................... 10
    Unix basics ..................................................................................................................... 10
     Bash shell ..................................................................................................................... 10
Organizing your work Directories .......................................................................................... 11
   ls – list files in current directory .................................................................................. 11
   mkdir <new_directory_name> – create a new directory (folder) .................................... 11
   cd <directory_name> – change directory ....................................................................... 11
   cp <original_file_name> <new_file_name> – copy a file or files .................................... 11
   mv <original_file_name> <new_file_name> – rename a file without creating a copy ........ 12
   rm <file_name> – delete (remove) a file ......................................................................... 12
UNIX utilities ......................................................................................................................... 12
   cat <file_name> – print an entire file to the terminal screen ........................................... 12
   less/more <file> – file viewer; print a file to the screen one page at a time ...................... 12
   pwd – print working directory ...................................................................................... 12
   wc <one or more files> – word count ........................................................................... 12
head <file name> – print the first lines of a file ................................................................. 13

tail <file name> – print the last lines of a file ..................................................................... 13
grep <pattern> <file name> – find information in a file ......................................................... 13

Customize your Unix environment ....................................................................................... 14

More Information .................................................................................................................... 14

File Transfer .............................................................................................................................. 14

scp – secure copy, securely transfer files between computers ................................................. 14
sftp – secure file transfer using FTP ...................................................................................... 14
wget – retrieve a file from a web site ..................................................................................... 15

Example: wget ......................................................................................................................... 15

Globus – large file transfer ....................................................................................................... 15

Torque/PBS Batch Jobs ........................................................................................................... 15

qsub <job_file_name> – submit a Torque job ....................................................................... 15
qstat – examine the status of a queue or job ......................................................................... 16
qpeek <job_number> ................................................................................................................. 16
qdel <job_number> – delete a job from a queue ..................................................................... 16

nano <file> - preparing job files .............................................................................................. 16

Fortress large data archive ................................................................................................... 17

Module System .......................................................................................................................... 18

module avail – show modules installed on this server .......................................................... 18
module load <module_name> – load a specific module for use ............................................ 18
module show <module_name> - show details of installation of a specific module ............... 18
module list – show a list of currently installed modules ......................................................... 19
module unload <module_name> - unload a specific module ................................................ 19
module purge – unload all modules ....................................................................................... 19
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 server

- scholar.rcac.purdue.edu

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

Windows

MobaXterm (recommended)
Website: https://mobaxterm.mobatek.net

Download: https://mobaxterm.mobatek.net/download-home-edition.html

Manual: https://mobaxterm.mobatek.net/documentation.html

MobaXterm supports a number of different kinds of communication between your local computer and a remote server (usually a linux OS computer). The main communication mode that we will use is called SSH or Secure Shell. Two other communication modes which are used for graphical interfaces (X11/XWindows) and file transfer (SFTP) will be discussed below.

Installation. MobaXterm is a single executable program so you should be able to download and use it even if you do not have system administration privileges. To download go to the MobaXterm web site, / . Select ‘Download’ on the main menu and select the free ‘Home Edition’. I suggest you select the Installer edition unless you plan to use MobaXterm from a flash drive (in which case you would get the Portable edition). The downloaded file is a Zip archive; click to open it and extract the two files to a suitable directory (I usually choose the desktop). Then click on the MobaXterm installer, accept the user agreement, and select the directory for installation. If you do not have system admin privileges, you will not be able to install in the default directory (C:\Program Files (x86)\Mobatek\Mobaxterm). Change the target directory to one where you have write privileges such as your desktop (I used C:\Users\michael\Desktop\MobaXterm). After installation, click on the desktop icon to open the MobaXterm application.

SSH. Select SSH from the session menu (upper left corner). Enter the name of the remote host (usually scholar.rcac.purdue.edu), and if you like, enter your username (Purdue career ID) and check the box ‘Specify username’. Adding your username is not essential, it just saves having to type it every
time you log in. The default settings should work for your initial work, you can read the online manual to find out more about the options. After the first time you login, a shortcut to your server session will appear under ‘sessions’ in the left menu bar, and in the central screen. The SSH session window is a text window that allows you to issu Unix commands on the server.

**File transfer.** It is often necessary to copy files between your local computer and the server. After you login to the server using MobaXterm, you should automatically see a panel labelled ‘SFTP’ in the left panel (Secure File Transfer Protocol). If the panel does not open automatically, click on the tab labelled ‘sftp’. This

**X11/XWindows.** Certain applications incorporate a graphical interface. Examples are the IGV genome browser or R Studio. In order to use these applications the graphics must be transferred to your local computer using the X11/Xwindows system. Fortunately, MobaXterm incorporates an X11 server to support this. The X11 server is automatically started for MobaXterm SSH sessions so all you should need to do is to run the desired program from the SSH session window (usually by typing the program name). Note that when you run a program in this way it has access to the directories and files on the server, not the files on your local computer.

**PUTTY**
*Website:* 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/

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.

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.
Many mac users do not know that OS X incorporates (or is built on top of) a Unix operating system. Unix commands can be used by simply opening a terminal (found in applications).

**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 simple 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).

Cyberduck

**Website:** https://cyberduck.io/

**Download:** See main web page

**Documentation:** https://trac.cyberduck.io/wiki/help/en

If you use a Mac you may prefer to use Cyberduck for file editing and file transfer rather than one of the applications listed below. Cyberduck can be installed for a fee through the app store, but can be installed without charge from the website.

One of the advantages of Cyberduck is that you do not have to learn a unix editor (although I think you should so that you can work directly on unix systems). With Cyberduck you can use the built-in TextEdit application or another editor such as TextWrangler (recommended).

**Installation**

- Choose “download cyberduck for Mac”
- Open the downloaded Zip file
- Drag the uncompressed application to the desired location (probably Applications folder)
- You can allow cyberduck to be the default application for FTP and SFTP
Connecting to the server

First, you must add the RCAC cluster to your list of connections. Select the “Open Connection” button on the upper left. Enter the information for the RCAC server you are using, e.g., scholar.rcac.purdue.edu, and your username and password. Using the pulldown menu above the Server name, select “SFTP (SSH-File Transfer Protocol)”. The port should automatically change to 22. You don’t have to worry about the file “SSH Private Key” for now, leave it as “None”. Click “Connect”. You should then get a message about “Unknown fingerprint” – this is normal, check “Always” and select “Allow”. After the first time you connect, the server information should be filled in. You can Bookmark the server if your like.

After connecting, you will see a file browser that shows your files on the server.

Edit server files
- select a file
- click on “Edit” in the top menu. The file will then open in your chosen editor.
- When finished save the file. Keep in mind that when you save the file it will be saved ON THE SERVER, not on your mac.

Transferring files
- Connect to the server, as described above.
- Click on the Gear icon labelled “Action” and choose “Download” to copy a file from the server to your mac, or “Upload” to copy files from your mac to the server. You will be presented with an a file browser to locate the files you want to transfer, and to select the destination directory.

Purdue University Specific

SecureCRT/SecureFX
Website: https://www.vandyke.com/products/securecrt/
Download: https://communityhub.purdue.edu/storefront/browse/securitysoftware
login with Purdue credentials required

Documentation: http://securecrttutorial.blogspot.com/

For students and staff at Purdue, the commercial program pair SecureCRT/SecureFX (Van Dyke Software) is available without charge through ITAP (for both home and Purdue use). This software is stable and reliable and works well, especially if you are already used to it. Since it is not available without charge at most sites, I recommend MobaXterm. SecureCRT can be configured to support X11 sessions, but it requires slightly more than MobaXterm. SecureCRT/FX are available for both Mac and Windows which could be an advantage if you switch between the two OS.

ThinLinc
Website: https://www.cendio.com/thinlinc
Download: https://www.cendio.com/thinlinc/download (see Download ThinLinc Clients section)
Documentation: in app help
ThinLinc is an application that allows you to connect to a Unix server using a graphical interface. In theory, the response of data/graphics intensive programs (such as IGV) should be better than with MobaXterm. There are two ways to use ThinLine:

**Client version:** The maker of ThinLink, Cendio, provides a custom client. This must be installed (see download site, above), which probably requires administrator privileges. If you do not have admin privileges, you can use the browser version (see below). The client application is probably a little faster than the browser version. The client version can also be configured so that the files on your local computer can be seen from within the Unix session. This is convenient for moving files between your personal computer and the server.

**Browser version:** To use the browser version, all you have to do is to enter the URL [https://desktop.snyder.rcac.purdue.edu](https://desktop.snyder.rcac.purdue.edu) into your browser.

Whether you use the client app or browser to connect, you must login with your Purdue credentials. The login process can be somewhat slow, but shouldn’t take more than about a minute. Then you will be presented with a somewhat clunky desktop. This interface is most useful if you are using a program with a graphical interface, but can be used for everyday tasks if you have a very hard time adjusting to the Unix command line interface, or CLI, (I strongly encourage you to get used to the CLI). Note that, just as with a OS X computer, you can start a terminal application within ThinLinc to use the CLI. The example above shows a ThinLinc session with the default (Thunar) file browser open to my default directory.

When you close your ThinLinc session you are not automatically logged out. This is convenient if you only need to disconnect for a short period of time, but may result in your session being slowed (throttled) due to excessive resource use. You should therefore always use the logout option (found in the applications menu at the upper left).
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 command line interface called a shell. There are several UNIX shells, but the most common is called bash (Bourne again shell), but differences between shells are not very important unless you are going to write shell scripts to automate your work. At Purdue, your account will be set up with the Bash shell. 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
- Introduction to UNIX, http://www.doc.ic.ac.uk/~wjk/UnixIntro

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.

| % grep "">" adapter.fa > adapter.name.txt |
| % filter_reads < wt.R1.fq >wt.R1.filtered.fq |
| % filter_reads < wt.R1.fq >wt.R1.filtered.fq 2>wt.R1.filtered.log |
| % filter_reads < wt.R1.fq &>reads_and_errors.txt |

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

| % grep --color=always AATGATACGGCG SRR1265029.fastq | head |
| TGT<br>TGT<br>AATGATACGGCG SRR1265029.fastq | head |
| GCC<br>GCC<br>AATGATACGGCG SRR1265029.fastq | head |
| GCC<br>GCC<br>AATGATACGGCG SRR1265029.fastq | head |
| GCC<br>GCC<br>AATGATACGGCG SRR1265029.fastq | head |
| GCC<br>GCC<br>AATGATACGGCG SRR1265029.fastq | head |
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.

**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<cr>. 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 both 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**

Organizing your work Directories

Work in computational biology frequently produces a large number of files, sometimes millions, and you will often perform multiple analyses within the same project. It is essential that you organize your work into folders in order to control the chaos. Many programs do not allow you to control the names of the files they produce — in this case, if you run the program twice on different inputs, the second result will overwrite and obliterate the first.

**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 -lth
-rw-r----- 1 mgbribsko mgbribsko-genomics 157M Apr 8 16:20 unmapped.bam
-rw-r----- 1 mgbribsko mgbribsko-genomics 603M Apr 8 16:19 accepted_hits.bam
-rw-r----- 1 mgbribsko mgbribsko-genomics 131M Apr 8 16:12 junctions.bed
-rw-r----- 1 mgbribsko mgbribsko-genomics 256K Apr 8 16:17 deletions.bed
-rw-r----- 1 mgbribsko mgbribsko-genomics 714K Apr 8 16:17 insertions.bed
-rw-r----- 1 mgbribsko mgbribsko-genomics 199 Apr 8 16:17 align_summary.txt
-rw-r----- 1 mgbribsko mgbribsko-genomics 119 Apr 8 16:05 prep_reads.info
drwxrwxrwx   2 mgbribsko mgbribsko-genomics 8.0K Apr 8 16:02 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 a 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
```

**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

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.

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 treads 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 last 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 lines (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)
--no-group-separator do not print "- -" between groups of matching lines (see example, below)
Customize your Unix environment

One of the features of Unix is that it is completely customizable. You can make shortcuts, change the names of commands, or run commands automatically to any degree you like. Be cautious, however, it becoming too reliant on your personalized shortcuts because they may not be available when you use another computer. When you log into a Unix system, several different initializations take place. Many of these are at the system level and you have no control over them, but there are two command files that are completely under your control. Both of these files are found in your home directory, but will not be shown in directory listings unless you type “ls –a”. If either file is absent it is just skipped.

~/.bash_profile – this file is run when login from a terminal or terminal emulator and start a new session login shell). Jobs that are run under Torque (see Torque/PBS batch jobs, below) are considered to be login shells so the commands in .bash_profile are executed. One of the most important things to put in your .bash_profile is the command “module load bioinfo”, which will set up your access to the bioinformatics modules (see Module System, below).

~/.bash_rc – this file is run when you start a subshell of your login shell, for instance, by typing “bash”

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.

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 – secure copy, securely transfer files between computers**

Examples:

```
scp  <local file>  <remote file>
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 runnning 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>
<th>(from a remote system to local)</th>
<th>(recursive directory copy to a remote system from local)</th>
</tr>
</thead>
<tbody>
<tr>
<td>% scp  sourcefilename  myusername@hostname:somedirectory/destinationfilename</td>
<td></td>
<td></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>
<td></td>
<td></td>
</tr>
<tr>
<td>% scp myusername@hostname:somedirectory/sourcefilename  destinationfilename</td>
<td></td>
<td></td>
</tr>
<tr>
<td>% scp <a href="mailto:gribskov@ser.genomics.purdue.edu">gribskov@ser.genomics.purdue.edu</a>/trinity.pl  .</td>
<td></td>
<td></td>
</tr>
<tr>
<td>% scp NGS/  <a href="mailto:gribskov@ser.genomics.purdue.edu">gribskov@ser.genomics.purdue.edu</a>/NGS/</td>
<td></td>
<td></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. SFTP is built into MobXterm,
as explained above. With MobXterm, a SFTP session is automatically opened in the left panel of your SSH session.

**wget – retrieve a file from a web site**


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.

**Globus – large file transfer**

see: [https://www.rcac.purdue.edu/knowledge/rice/storage/transfer/globus](https://www.rcac.purdue.edu/knowledge/rice/storage/transfer/globus)

When transferring large files, such as sequence data files, the file may be corrupted if there is an interruption in the network connection, or due to other events (such as your laptop battery dying). Globus provides a fast and secure method for file transfer (faster than SFTP) in which the completeness and integrity of the files is confirmed. This is particularly useful for very large files and for archiving files on the Fortress long-term archive system.

Globus uses a client application or a web interface to transfer between what it refers to as “endpoints”. An endpoint can be thought of as a particular directory folder on a particular computer (for instance, the home directory on my laptop, or my scratch directory on Snyder.rcac.purdue.edu). To use Globus, enter the URL `https://transfer.rcac.purdue.edu` into your web browser and login using your Purdue credentials. The two panels represent the two endpoints of your transfer. Click on the left one to choose the origin of the transfer (type Purdue to see a list of Purdue resources). You can create endpoints on your own computer by installing the Globus Connect Personal client (requires admin privilege). Click on the link ‘Get Globus Connect Personal’ on the upper middle of the transfer screen (see above).

**Torque/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.

```
% qsub -N example -q mgribsko -c -l walltime=1:00:00 -l nodes=1:ppn=16 example.job
5846659. carter-ade.rca, purdue.edu:

% qstat -a mgribsko-c
carter-adm.rca, purdue.edu:

Job ID   Username   Queue   Jobname   SessID   NDS   TSK   Memory   Time    S   Time
--------  ---------   ------   --------   -------   ----   ----   -------   ------   ----  ----
6809412. carter-ade.rca  tkljopn  mgribsko JobApr17_raxml-p  47800   1   1   --  300:00:00   R  227:37:27
6845700. carter-ade.rca  gcore    mgribsko abyss_batch_50-h  447    1   16   --  300:00:00   R  09:34:33
6845701. carter-ade.rca  gcore    mgribsko abyss_batch_60-n  6420   1   16   --  300:00:00   R  09:38:25
6845704. carter-ade.rca  gcore    mgribsko abyss_batch_50-n  5054   1   16   --  300:00:00   R  04:34:06
6846659. carter-ade.rca  mgribsko mgribsko example             --   1   16   --  01:00:00   Q  --

% qdel 6846659

% qstat -a mgribsko-c
carter-adm.rca, purdue.edu:

Job ID   Username   Queue   Jobname   SessID   NDS   TSK   Memory   Time    S   Time
--------  ---------   ------   --------   -------   ----   ----   -------   ------   ----  ----
6809412. carter-ade.rca  tkljopn  mgribsko JobApr17_raxml-p  47800   1   1   --  300:00:00   R  227:37:49
6845700. carter-ade.rca  gcore    mgribsko abyss_batch_50-h  447    1   16   --  300:00:00   R  09:43:35
6845701. carter-ade.rca  gcore    mgribsko abyss_batch_60-n  6420   1   16   --  300:00:00   R  09:38:47
6845704. carter-ade.rca  gcore    mgribsko abyss_batch_50-n  5054   1   16   --  300:00:00   R  04:34:28
6846659. carter-ade.rca  mgribsko mgribsko example             --   1   16   --  01:00:00   C  --

% qstat -a mgribsko-c
carter-adm.rca, purdue.edu:

Job ID   Username   Queue   Jobname   SessID   NDS   TSK   Memory   Time    S   Time
--------  ---------   ------   --------   -------   ----   ----   -------   ------   ----  ----
6809412. carter-ade.rca  tkljopn  mgribsko JobApr17_raxml-p  47800   1   1   --  300:00:00   R  227:38:31
6845700. carter-ade.rca  gcore    mgribsko abyss_batch_50-h  447    1   16   --  300:00:00   R  09:44:17
6845701. carter-ade.rca  gcore    mgribsko abyss_batch_60-n  6420   1   16   --  300:00:00   R  09:39:29
6845704. carter-ade.rca  gcore    mgribsko abyss_batch_50-n  5054   1   16   --  300:00:00   R  04:35:10
```

**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

**qpeek <job_number>**

Shows the current STDOUT of the job. qpeek –e shows STDERR. This is sometimes useful if you think the job may have gotten stuck.

**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 $PBS_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.

Fortress large data archive

The fortress system is a multiple petabyte archival system found at fortress.rcac.purdue.edu. Because it is a tape based system, fortress is not suitable for storing data you are actively using. However, since the scratch filesystem is only for active data (i.e, used in the last 60 days), fortress is perfect for storing your original data, and inactive or completed projects. Usually you can backup whole directory trees using a single “htar” command. Points to consider:

- Your fortress account is automatically created whenever an account is activated for you on any server
- All servers have access to the same files on fortress (this makes it one way to transfer between servers).
- Fortress is designed to be efficient for large files, not for many small files. Your files need to be aggregated in some way, usually by using htar.
- Htar will not work for very large files. In this case you need to run tar and transfer the file use globus (preferred) or his (cryptic).
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. Because bioinformatics has such a large amount of installed software, you must always use “module load bioinfo”. Put this command in your bash profile and you will never have to type it!

<table>
<thead>
<tr>
<th>module avail</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>BBTools/34.25</td>
<td>bowtie/1.1.1</td>
<td>picard-tools/1.93</td>
<td></td>
</tr>
<tr>
<td>BBTools/34.26</td>
<td>bowtie2/2.0.0-beta5</td>
<td>picard-tools/1.94</td>
<td></td>
</tr>
<tr>
<td>BBTools/34.26</td>
<td>bowtie2/2.0.0-beta6</td>
<td>picard-tools/1.95</td>
<td></td>
</tr>
<tr>
<td>BBTools/34.33</td>
<td>bowtie2/2.0.0-beta7</td>
<td>picard-tools/1.96</td>
<td></td>
</tr>
<tr>
<td>BBTools/34.38</td>
<td>bowtie2/2.0.1</td>
<td>pplacer/v1.1_alpha3</td>
<td></td>
</tr>
<tr>
<td>BBTools/34.41</td>
<td>bowtie2/2.0.2</td>
<td>primer/v1.1.4</td>
<td></td>
</tr>
<tr>
<td>BBTools/34.46</td>
<td>bowtie2/2.0.3</td>
<td>primer2/2.3.4</td>
<td></td>
</tr>
<tr>
<td>BBTools/34.48</td>
<td>bowtie2/2.0.4</td>
<td>primer3/2.3.6</td>
<td></td>
</tr>
<tr>
<td>BBTools/34.49</td>
<td>bowtie2/2.0.5</td>
<td>prottest/3.3-20130716</td>
<td></td>
</tr>
<tr>
<td>BBTools/34.56</td>
<td>bowtie2/2.0.6</td>
<td>pykab/2.15</td>
<td></td>
</tr>
<tr>
<td>BBTools/34.64</td>
<td>bowtie2/2.1.0</td>
<td>pysam/0.6</td>
<td></td>
</tr>
<tr>
<td>BBTools/34.64</td>
<td>bowtie2/2.1.1</td>
<td>pysam/0.7</td>
<td></td>
</tr>
<tr>
<td>BBTools/34.72</td>
<td>bowtie2/2.2.0</td>
<td>quake/0.3</td>
<td></td>
</tr>
<tr>
<td>BBTools/34.72</td>
<td>bowtie2/2.2.1</td>
<td>qualimap/0.6</td>
<td></td>
</tr>
<tr>
<td>BBTools/34.79</td>
<td>bowtie2/2.2.2</td>
<td>qualimap/1.0</td>
<td></td>
</tr>
<tr>
<td>BBTools/34.83</td>
<td>bowtie2/2.2.3</td>
<td>quast/2.1</td>
<td></td>
</tr>
<tr>
<td>BBTools/34.86</td>
<td>bowtie2/2.2.4</td>
<td>quest/2.4</td>
<td></td>
</tr>
<tr>
<td>BBTools/34.86</td>
<td>bowtie2/2.2.5</td>
<td>quip/1.1.1</td>
<td></td>
</tr>
<tr>
<td>BEDTools/2.16.2</td>
<td>bowtie2/2.2.5</td>
<td>quip/1.1.1-3-g02d3228...</td>
<td></td>
</tr>
<tr>
<td>BEDTools/2.16.2</td>
<td>bowtie2/2.2.5</td>
<td>quip/1.1.3-3-g02d3228...</td>
<td></td>
</tr>
<tr>
<td>BEDTools/2.17.0</td>
<td>bowtie2/2.2.5</td>
<td>quip/1.1.4</td>
<td></td>
</tr>
<tr>
<td>BEDTools/2.20.1</td>
<td>bowtie2/2.2.5</td>
<td>quip/1.1.5</td>
<td></td>
</tr>
<tr>
<td>BEDTools/2.21.0</td>
<td>bowtie2/2.2.5</td>
<td>quip/1.1.6</td>
<td></td>
</tr>
<tr>
<td>BESS/1/1.2</td>
<td>bowtie2/2.2.5</td>
<td>quip/1.1.7</td>
<td></td>
</tr>
</tbody>
</table>

| % module avail |
|---|---|---|
| % module load bowtie2 |
| % module show bowtie2 |

```bash
% module avail
...
BBTools/34.25 bowtie/1.1.1 picard-tools/1.93
BBTools/34.26 bowtie2/2.0.0-beta5 picard-tools/1.94
BBTools/34.26 bowtie2/2.0.0-beta6 picard-tools/1.95
BBTools/34.33 bowtie2/2.0.0-beta7 picard-tools/1.96
BBTools/34.38 bowtie2/2.0.1 pplacer/v1.1_alpha3
BBTools/34.41 bowtie2/2.0.2 primer/v1.1.4
BBTools/34.46 bowtie2/2.0.3 primer2/2.3.4
BBTools/34.48 bowtie2/2.0.4 primer3/2.3.6
BBTools/34.49 bowtie2/2.0.5 prottest/3.3-20130716
BBTools/34.56 bowtie2/2.0.6 pykab/2.15
BBTools/34.64 bowtie2/2.1.0 pysam/0.6
BBTools/34.64 bowtie2/2.1.1 pysam/0.7
BBTools/34.72 bowtie2/2.2.0 quake/0.3
BBTools/34.72 bowtie2/2.2.1 qualimap/0.6
BBTools/34.79 bowtie2/2.2.2 qualimap/1.0
BBTools/34.83 bowtie2/2.2.3 quast/2.1
BBTools/34.86 bowtie2/2.2.4 quest/2.4
BBTools/34.86 bowtie2/2.2.5 quip/1.1.1
BBTools/34.86 bowtie2/2.2.5 quip/1.1.1-3-g02d3228...
BBTools/34.86 bowtie2/2.2.5 quip/1.1.3-3-g02d3228...
BBTools/34.86 bowtie2/2.2.5 quip/1.1.4
BBTools/34.86 bowtie2/2.2.5 quip/1.1.5
BBTools/34.86 bowtie2/2.2.5 quip/1.1.6
BBTools/34.86 bowtie2/2.2.5 quip/1.1.7

% module load bowtie2
% module show bowtie2

/posts/group/bioinformatics/modules/bowtie2/2.2.5:
module-whatis Fast and sensitive read alignment
module load gcc
module load perl
prepend-path PATH /group/bioinfo/apps/apps/bowtie2-2.2.5
prepend-path PATH /group/bioinfo/apps/apps/bowtie2-2.2.5/scripts

Notes:
Bowtie2 version 2.2.5
Fast and sensitive read alignment
Manual: /group/bioinfo/apps/apps/bowtie2-2.2.5/MANUAL

% module list
Currently Loaded Modulefiles:
1) gcc/4.7.2 3) tidyp/1.04 5) ImageMagick/6.8.9-8 7) perl/5.20.1 9) bowtie2/2.2.5
2) gnuplot/4.6.6 4) gsl/1.15 6) expat/2.1.0 8) java/7

% module list
Currently Loaded Modulefiles:
1) gcc/4.7.2 3) tidyp/1.04 5) ImageMagick/6.8.9-8 7) perl/5.20.1 9) bowtie2/2.2.5
2) gnuplot/4.6.6 4) gsl/1.15 6) expat/2.1.0 8) java/7

% module list
Currently Loaded Modulefiles:
1) gcc/4.7.2 3) tidyp/1.04 5) ImageMagick/6.8.9-8 7) perl/5.20.1 9) bowtie2/2.2.5
2) gnuplot/4.6.6 4) gsl/1.15 6) expat/2.1.0 8) java/7
% module list
Currently Loaded Modulefiles:
1) gcc/4.7.2 3) tidyp/1.04 5) ImageMagick/6.8.9-8 7) perl/5.20.1 9) bowtie2/2.2.5
2) gnuplot/4.6.6 4) gsl/1.15 6) expat/2.1.0 8) java/7
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
module purge – unload all modules