**Basics involved in working and programming with command line**

Posted on [February 28, 2019](http://www.zoology.ubc.ca/~rieseberg/RiesebergResources/?p=38541) by [Cassandra Elphinstone](http://www.zoology.ubc.ca/~rieseberg/RiesebergResources/author/cassandra-elphinstone/)

Topics briefly touched upon are:

Servers, Command line basics, tmux sessions, interactive job allocations, sbatch, conda installation on Compute Canada-miniconda2, virtual environments, dDocent installation, demultiplexing paired end reads for dDocent

**Introduction to High Performance Computing**

Download from:<http://bit.ly/introhpc>  
  
If you are just starting in Unix or Compute Canada I recommend downloading the slides from the link above and scrolling through the first slides to begin (e.g. slides 1-14).   
There are also these getting started videos: <http://bit.ly/2sxGO33>  
and this is the Compute Canada YouTube channel <http://bit.ly/2ws0JDC>

**Servers**

Compute Canada has two main servers: Cedar and Graham  
[username@cedar.computecanada.ca](mailto:%75ser%6eame%40c%65da%72%2ecompu%74%65canada%2eca)  
<https://docs.computecanada.ca/wiki/Cedar>

[username@graham.computecanada.ca](mailto:u%73%65r%6eame@gra%68am.co%6dput%65ca%6e%61%64%61%2e%63a)  
<https://docs.computecanada.ca/wiki/Graham>   
  
Note: The instructions below all relate to Compute Canada not specifically to the lab servers.

**Login to the server – using the ssh command**

Home$ ssh username@cedar.computecanada.ca

Note: Your password will not be shown but you are still typing.   
See for more info on login for a remote computer:   
[http://oliverelliott.org/article/computing/ref\_unix/26](http://oliverelliott.org/article/computing/ref_unix/26/)  
  
Also see the slides from the link above if you are using Windows and need to set up a terminal port:  
<https://docs.computecanada.ca/wiki/Connecting_with_MobaXTerm>

**Command line basics**

The two links below give you information regarding commands useful to know when using unix and/or linux operating systems (command line).  
<http://oliverelliott.org/article/computing/ref_unix/>     
I would suggest reading through these lists of commands, if you are new to command line. As well, I have found it is good to know the following commands:  
Ctrl-C -> Cancels the code running  
Tab -> will complete the text you are typing if that file or program exists  
All Unix commands are in lowercase and have the following general layout:  
Command  –options  file\_name\_input    file\_name\_output  
Examples can be found at:  
<https://www.geeksforgeeks.org/gzip-command-linux/>  
<https://www.cyberciti.biz/faq/copy-folder-linux-command-line/>

In the following, the symbol  # separates the command line text (left side) from my comments (right side).

**Sessions – using tmux**   
The benefit of tmux session tabs are that you can work in multiple windows from one computer and run multiple processes at once.

$tmux list-sessions  # will list all running sessions  
$tmux new-session name\_session  #will create a new session  
$tmux attach-session –t name\_session  #will reopen a current session

Use ‘Ctrl+b’ together then press ‘d’ to detach from a session and go back to the main tab  
‘Ctrl+b’ together then press ‘)’ to go to the next session  
‘Ctrl+b’ together then press ‘(’ to go to a previous session  
See other ctrl-b commands here to move around sessions: <http://oliverelliott.org/article/computing/ref_unix/65/>

Note: Cedar has two parts cedar1 and cedar5. Your tmux sessions will exist in one of these parts. If you log in and randomly get assigned to a different part you will be unable to access your tmux sessions that are still running on the other part.

**Interactive job allocations – using salloc**  
Salloc starts an interactive node. It is a good place to test scripts before running them with sbatch where they could be queued for days. Unlike just running the commands in your folders, it will continue running when you log off (at least until your time allocation runs out)

$salloc -c 16 --time 3:00:00 --mem 64G --account xxxx

In the above command, the options are (note the double dash for some options):

--time     # should be in days-hours:minutes:seconds and requests the amount of time you want the resources allocated for.  
-c     #is the number of CPUs to allocate for your tasks  
--mem      # is the amount of memory allocated to your tasks

#A good general guideline is 4Gb/1CPU. This ratio can be found by calculating the ratio of the memory  needed to the number of CPUs required. The number of CPUs or separate Central Processor Units (CPU) would correspond to the number of separate calculations you want to run in parallel. The memory is the amount of storage required for those calculations.

--account # is the account that will be charged for your

allocation of resources.

In general, it is good practice to run any tasks that might take longer than a couple minutes on an interactive node. You can create a [run\_program] session using the tmux command above and in the [run] session call salloc to allocate a certain amount of memory and CPU to that run session for some amount of time. This will allow you to run your code uninterrupted on a designated node until you allocated time runs out.

Some useful diagnostics if you want to know about the resources left on your resource allocation

$man sacct        # ‘man’ provides the manual and info about the command  
$sacct     # command will tell you about the jobs running on your node  
$time       # time left on allocation  
$getrusage     # <http://man7.org/linux/man-pages/man2/getrusage.2.html>

If needed, you can ssh directly into your interactive job session using:

$ssh username@cdr490.int.cedar.computecanada.ca

This will only work if you have set up automatic password identification. To set up automatic password identification see:   
<http://oliverelliott.org/article/computing/ref_unix/27/>

**Using sbatch to submit jobs**

Only send jobs that are sure to work as this sends your jobs to Slurm where they could be queued for days.  
See more: <https://docs.computecanada.ca/wiki/Running_jobs>   
and: [https://slurm.schedmd.com/sbatch.ht](https://slurm.schedmd.com/sbatch.html)

**Installing miniconda2**

The code below will install miniconda2 on your home directory in a folder called miniconda2. This only needs to be done once for your account on a particular Compute Canada server.

~$  curl "https://repo.anaconda.com/miniconda/Miniconda2-latest-Linux-x86\_64.sh" -o miniconda.sh  
#follow prompts  
#when asked to modify ~/bashrc   say ‘no’  
$ chmod +x miniconda.sh  
$  ~/miniconda.sh  
$  rm miniconda.sh  
$ ~/miniconda2/bin/conda

See more about ‘curl’: <https://curl.haxx.se/>

**Virtual environments using conda**

A virtual environment allows you a space where a certain set of programs are installed and allows you to keep track of the exact version of the program you are using.    
Note that when using the Compute Canada modules, they can change the version number without you knowing and cause an error in your pipeline down the road.  
Using conda you can create a virtual environment to work in and know the versions of the programs installed.   
Here is how to create a virtual environment with conda:

$ ~/miniconda2/bin/conda create --name gbs     
# this calls conda and allows us to create our own virtual environment called [gbs] here  
$ source ~/miniconda2/bin/activate gbs      
#this activates the environment so we can work in it using the programs installed  
$source ~/miniconda2/bin/deactivate gbs      
#this deactivates the environment so the programs and their versions are no longer available

Also see on virtual environments: <http://oliverelliott.org/article/computing/ref_unix/99/>

**Installing dDocent using conda**

After activating the gbs environment, within the gbs environment we can install bioconda ddocent. dDocent is a useful program for denovo GBS assembly (when you have no reference genome). dDocent only needs to be installed once in a new environment.

Username ~$ source ~/miniconda2/bin/activate gbs      
[gbs] Username ~$ conda install -c bioconda ddocent

**Fixing issues with dDocent samtools installation**

$LD\_LIBRARY\_PATH="$LD\_LIBRARY\_PATH:/home/celphin/miniconda2/envs/gbs/lib" samtools  
$pushd ~/miniconda2/envs/gbs/lib  
$\*  
$ls -l libcrypto.\*  
$ln -s libcrypto.so.1.1 libcrypto.so.1.0.0  
$samtools  
$source ~/miniconda2/bin/deactivate gbs

Fix from: <https://github.com/bioconda/bioconda-recipes/issues/12100>

**To start using dDocent**

$source ~/miniconda2/bin/activate gbs  
$dDocent

-follow the prompts and instructions here:  
<http://ddocent.com/UserGuide/>