An Introduction to Linux and Bowtie

Cavan Reilly

April 17, 2018

Table of contents

Introduction to Linux

In order to use the latest tools in bioinformatics you need access to a linux based operating system (or Mac OS X).

So we will give some background on how to use a linux based system.

The linux operating system is based on the UNIX operating system (by a guy named Linus Torvalds), but is freely available (UNIX was developed by a private company).

There are a number of implementations of linux that go by different names: e.g. Ubuntu, Debian, Fedora, and OPENsuse.

Introduction to Linux

Most beginners choose Ubuntu (or one of its offshoots, like Mint).

I use Ubuntu or Fedora on my computers and centos is used on the server we will use for this course.

You can install it for free on any PC.

Such systems are based on a hierarchical structure for organizing files into directories.

We will use a remote server, so we need a way to communicate between the computer you are using and this remote machine.

Introduction to Linux

To connect to a linux server from a Windows machine you need a Windows program that allows you to log on to a terminal and the ability to transfer files to and from the remote server and your Windows machine.

WinSCP is a free program that allows one to do this.

You can download the data from the WinSCP website and it installs like a regular Windows program: just allow your computer to install it and go with the default configuration.

To get a terminal window you can use a program called PuTTY.

So download the program `PuTTY` (which is just the file called PuTTY.exe) and create a subdirectory in your Program Files directory called PuTTY.

Then save PuTTY.exe in the directory you just created.

Once installed start WinSCP and use `boris.biostat.umn.edu` as the hostname along with your user name and password.

You can then get a terminal window using putty: just click on the box that shows 2 computers that are connected right below the session tab.

Then disconnect when you are done (using the option in the session tab).

Introduction to Linux

Much of the time we will use plain text files: these are files that have no formatting, unlike the files you use in a typical word processing program.

To edit text files you need to use a text editor. I like vi and vim (which has capabilities to interact with large files).

Sometimes we will encounter files we can't read with a text editor, for example binary files and compressed files.

The editor pico is simple to use and installed on the server we will use for this class.

To use this editor you type pico followed by the file name-if the file doesn't exist it will create the file otherwise it will open up the file so that you can edit it.

For example

```
[user0001@boris ~]$ pico temp1
```

will open the file temp1 for editing. Commands at the bottom of the window tell you how to conduct various operations.

Introduction to Linux

You can create subdirectories in your home directory by typing

```
[user0001@boris ~]$ mkdir NGS
```

This would create a NGS subdirectory where you can put files related to your NGS project. Then to keep your microarray data separate you could do

```
[user0001@boris ~]$ mkdir microarray
```

To move around hierarchical directory structure you use the command `cd`. We enter the NGS directory with

```
[user0001@boris ~]$ cd NGS
```

Introduction to Linux

To go back to our home location (i.e. the directory you start from when you log on) you just type `cd` and if you want to go up one level in the hierarchy you type `cd ..`.

To list all of the files and directories in your current location you type `ls`.

To copy a file you use `cp`, to move a file from one location to another you use `mv` and to remove a file you use `rm`.

One can modify the behavior of these commands by using options.

Introduction to Linux

For example, if I want to see the size of the files (and some additional information) rather than just list them I could issue the command

```
[user0001@boris ~]$ ls -l
```

Other useful commands include `more` and `less` for viewing the contents of files and `man` for information about a command (from the manual). For instance

```
[user0001@boris ~]$ man ls
```

Will open a help file with information about the options that are available for the ls command.

Introduction to Linux

Wild cards are characters that operating systems use to hold a variable value. These are extremely useful as it lets the user specify many tasks without having to do a bunch of repetitive typing.

For example, suppose you had a bunch of files that ended with .txt and you wanted to delete them. You could issue commands like

```
[user0001@boris ~]$ rm file1.txt
[user0001@boris ~]$ rm file2.txt
[user0001@boris ~]$ rm file3.txt ...
```

Or you could just use

```
[user0001@boris ~]$ rm file*txt
```

or even

```
[user0001@boris ~]$ rm *txt
```

Introduction to Linux

Most linux configurations will check that you really want to remove the files, but if you are removing many files, the process of typing yes repeatly is unnecessary. Just type

```
[user0001@boris ~]$ rm -f *txt
```

When you obtain programs from the internet that run on linux systems these programs will typically install many subdirectories, so if you want to get rid of all the contents of a directory, first enter that directory,

```
[user0001@boris ~]$ cd oldDir
```

then issue the command

```
[user0001@boris oldDir]$ rm -rf *
```

then go back up one level

```
[user0001@boris oldDir]$ cd ..
```

and get rid of the directory using the rmdir command

```
[user0001@boris ~]$ rmdir oldDir
```

If you issue a command to run a program but wish you hadn't and want it to stop you push the Ctrl and C keys simultaneously.

Installation of Bowtie

Recall: Bowtie is a program that aligns short reads to an indexed genome.

Bowtie2 is a more up to date version that allows gapped alignments and Smith Waterman type scoring, Bowtie just allows a couple of nucleotides to disagree between the read and the genome.

To use Bowtie we need a set of short reads (in fastq format) and the index of a genome and of course the program itself.

We will download a compressed version of the program (an executable) and after we uncompress it, we can use the *executable*, i.e. a program that does something.

Bowtie has the capability to build indexes if we have the genome sequence.

Installation of Bowtie

First set up a directory to hold all of this work

```
[user0001@boris ~]$ mkdir RNAseq
```

then create a directory to hold all of the programs we will download and set up, we will call it bin (for binaries).

```
[user0001@boris ~]$ mkdir bin
```

Then enter this directory so we can install some programs.

```
[user0001@boris ~]$ cd bin
```

Installation of Bowtie

In order to know what version of software you need to install you need to check the architecture of the computer. To do so we issue the following command

```
[user0001@boris bin]$ uname -m
x86_64
```

So this is a 64 bit machine (32 bit machines will report something like i386). So get the appropriate version of Bowtie using the wget command

```
[user0001@boris bin]$ wget

http://sourceforge.net/projects/bowtie-bio/files/bowtie2/2.2.6/bowtie2-2.2.6-linux-x86_64.zip
```

Installation of Bowtie

then unpack it

```
[user0001@boris bin]$ unzip bowtie2-2.2.6-linux-x86_64.zip
Archive:  bowtie2-2.2.6-linux-x86_64.zip
creating:  bowtie2-2.2.6/
creating:  bowtie2-2.2.6/scripts/
inflating:  bowtie2-2.2.6/scripts/build_test.sh
inflating:
bowtie2-2.2.6/scripts/make_a_thaliana_tair.sh
...
```

This will generate a subdirectory with the name `bowtie2-2.2.6`, so enter that subdirectory so that you can have access to the executable called `bowtie` in this subdirectory.

Aligning reads to an indexed genome

The program ships with some data from the bacteriophage Enterobacteria phage lambda, so we will check the installation with this.

We will first use a fasta file that ships with Bowtie to construct the index files for this organism.

To use the program, you type the name of the executable, then give the location and name of the fasta file, then give the index name.

So to test the installation type the following

```
[user0001@boris bowtie2-2.2.6]$ ./bowtie2-build
example/reference/lambda_virus.fa lambda_virus
```

Aligning reads to an indexed genome

Then you get a bunch of output written to the screen about the index creation process and it generates 6 new files: `lambda_virus.1.bt2`, `lambda_virus.2.bt2`, `lambda_virus.3.bt2`, `lambda_virus.4.bt2`, `lambda_virus.rev.1.bt2` and `lambda_virus.rev.2.bt2`.

After that we are ready to align reads to the reference genome using the index we constructed.

Here the syntax is to give the name of the index after -x, the files with the reads after -U and here we specify that we want the output in the sam file format using -S.

Aligning reads to an indexed genome

```
[user0001@boris bowtie2-2.2.6]$ ./bowtie2 -x lambda_virus -U
example/reads/reads_1.fq -S eg1.sam
10000 reads; of these:
  10000 (100.00%) were unpaired; of these:
    596 (5.96%) aligned 0 times
    9404 (94.04%) aligned exactly 1 time
    0 (0.00%) aligned >1 times
94.04% overall alignment rate
```

We can also look at the output.

```
[user0001@boris bowtie2-2.2.6]$ head eg1.sam
```

Aligning reads to an indexed genome

Illumina stores prebuilt indices for many commonly studied organisms at the location

`http://support.illumina.com/sequencing/sequencing_software/igenome.html`

If one has a set of index files one can use the program `bowtie2-inspect` to recreate the fasta files.

The Illumina site has the fasta files for these organisms in the same location.

Altering your $PATH

Note that we need to be in the same directory level as our executable in order to use that executable thus far-this is not necessary.

To have access to an executable anywhere in your system you should alter your `PATH` variable in the linux environment to include where you keep your executables then write the executables to your path.

Altering your $PATH

If you look at your path variable you will see that the bin directory we created is already on the path, so we can just copy executables to that location to use them.

Here is how to examine the PATH variable.

```
[user0001@boris bowtie2-2.2.6]$ echo $PATH
/usr/lib64/qt-3.3/bin:/usr/kerberos/sbin:/usr/kerberos/bin
/usr/local/bin:/bin:/usr/bin:/usr/local/sbin:/usr/sbin:
/sbin:/export/home/courses/ph7445/user0001/bin
```

Copying executables to bin

Many NGS programs use bowtie2, and for this to work they need to find the program in their path.

So here we copy all of the executables to the bin directory:

```
[user0001@boris bowtie2-2.2.6]$ cp bowtie2
$HOME/bin/bowtie2
[user0001@boris bowtie2-2.2.6]$ cp bowtie2-align-l
$HOME/bin/bowtie2-align-l
[user0001@boris bowtie2-2.2.6]$ cp bowtie2-align-s
$HOME/bin/bowtie2-align-s
[user0001@boris bowtie2-2.2.6]$ cp bowtie2-build
$HOME/bin/bowtie2-build
[user0001@boris bowtie2-2.2.6]$ cp bowtie2-build-l
$HOME/bin/bowtie2-build-l
```

Copying executables to bin

and copy the rest too.

```
[user0001@boris bowtie2-2.2.6]$ cp bowtie2-build-s
$HOME/bin/bowtie2-build-s
[user0001@boris bowtie2-2.2.6]$ cp bowtie2-inspect
$HOME/bin/bowtie2-inspect
[user0001@boris bowtie2-2.2.6]$ cp bowtie2-inspect-l
$HOME/bin/bowtie2-inspect-l
[user0001@boris bowtie2-2.2.6]$ cp bowtie2-inspect-s
$HOME/bin/bowtie2-inspect-s
```

Altering your $PATH

Bowtie 2 uses environmental variables to find the executables-just set these as follows.

```
[user0001@boris bowtie2-2.2.6]$ export

BT2_HOME=/export/home/courses/ph7445/user0001/bin/bowtie2-2.2.6
```

and check this with

```
[user0001@boris bowtie2-2.2.6]$ echo $BT2_HOME
```

Altering environmental variables

To test that this works let's go back to our home directory and issue our Bowtie2 commands again (note that we need to tell Bowtie where to find the fastq and index files)

```
[user0001@boris  ]$ bowtie2-build
$BT2_HOME/example/reference/lambda_virus.fa
lambda_virus
[user0001@boris  ]$ bowtie2 -x lambda_virus -U
$BT2_HOME/example/reads/reads_1.fq -S eg1.sam
```

and it works fine.

Producing SAM files

As we have seen, Bowtie2 can also produce a special file type that is widely used in the NGS world, a SAM file.

If we open our `eg1.sam` file we see that the first 3 rows are the header (since they start with @).

Then there is a row for each read and 11 mandatory tab delimited fields per line that provide information about the alignment, for example the 4th column gives the location where the read maps.

For the full specification consult `samtools.github.io/hts-specs/SAMv1.pdf`.

SAMtools

The SAM (for sequence alignment/map) file type has a very strict format.

These files can be processed in a highly efficient manner using a suite of programs called SAMtools.

Here is how to setup these tools

Download the latest version in your bin directory `samtools-1.2.tar.bz2`.

```
[user0001@boris bin]$ wget

http://sourceforge.net/projects/samtools/files/samtools/1.2/samtools-1.2.tar.bz2
```

SAMtools

then unpack

```
[user0001@boris bin]$ tar -jxvf samtools-1.2.tar.bz2
```

then there is a directory full of C programs called samtools-1.2, so go in there and type

```
[user0001@boris samtools-1.2]$ make
```

and write the executable to your path

```
[user0001@boris samtools-1.2]$ cp samtools
$HOME/bin/samtools
```

SAMtools example

To use this for variant calling, we need an example with at least 2 subjects, so let's use the example that ships with SAMtools.

We will return to this example latter, but for now we will just take a look at the "pileup" which represents the extent of coverage.

Here is how to use SAMtools to index a FASTA file:

```
[user0001@boris samtools-1.2]$ samtools faidx
examples/ex1.fa
```

BAM and SAM files

then create a BAM file from the SAM file (BAM files are binary, so they are compressed but not readable by eye)

```
[user0001@boris samtools-1.2]$ samtools import examples/ex1.fa.fai
examples/ex1.sam.gz ex1.bam
```

should get

```
[sam_header_read2] 2 sequences loaded
```

BAM and SAM files

then index the BAM file

```
[user0001@boris samtools-1.2]$ samtools index ex1.bam
```

then we can view the alignment to visually examine the extent of coverage and the depth of coverage

```
[user0001@boris samtools-1.2]$ samtools tview ex1.bam
examples/ex1.fa
```

and hit q to leave that visualization.

Visualization of the pileup