Portfolio 6: Linux Command Line

Kieran Morris

The command line is the interface to the operating system, and is in fact a language itself, in fact on linux it has a name: bash. It's a great tool that allows you to be much better involved in your computers operations, plus it looks really cool. In this portfolio we will explore a few fundamental commands along with some bonus stuff I've found on my own. We will be making use of `{bash}` code chunks to run these commands.

Genomic Data

Navigating and Creating Directories

I have a file called `ncbi_dataset.zip` which contains a dataset of genomic data. I am currently in the correct directory.

```
ls
```

```
## ncbi_dataset.zip
## Portfolio_6.pdf
## Portfolio_6.rmd
```

But I need a new directory to store the new files I will create, so we create a new directory called `genomic_data` by running:

```
mkdir genomic_data
```

then we can see that the directory has been created by running `ls` again.

```
ls
```

```
## genomic_data
## ncbi_dataset.zip
## Portfolio_6.pdf
## Portfolio_6.rmd
```

I will then place my zipped file into this directory by running:

```
mv ncbi_dataset.zip genomic_data
```

Let's navigate to the `genomic_data` directory and check we have indeed moved our file:

```
cd genomic_data
ls
```

```
## ncbi_dataset.zip
```

Nice!

Unzipping files

Now we need to unzip the file, we can do this by running:

```
cd genomic_data
unzip ncbi_dataset.zip
```

```
## Archive:  ncbi_dataset.zip
##   inflating: README.md
##   inflating: ncbi_dataset/data/data_summary.tsv
##   inflating: ncbi_dataset/data/assembly_data_report.jsonl
##   inflating: ncbi_dataset/data/GCA_009858895.3/GCA_009858895.3_ASM985889v3_genomic.fna
##   inflating: ncbi_dataset/data/GCF_009858895.2/GCF_009858895.2_ASM985889v3_genomic.fna
##   inflating: ncbi_dataset/data/dataset_catalog.json
```

This will unzip the file and we can check that it has been unzipped by running:

```
cd genomic_data
ls
```

```
## ncbi_dataset
## ncbi_dataset.zip
## README.md
```

We actually have a few files in here, but we specifically care about the `.fna` file:

```
cd genomic_data
cd ncbi_dataset/data/GCA_009858895.3
ls
```

```
## GCA_009858895.3_ASM985889v3_genomic.fna
```

So lets shift this file to the `genomic_data` directory:

```
cd genomic_data
cd ncbi_dataset/data/GCA_009858895.3
mv GCA_009858895.3_ASM985889v3_genomic.fna ../../../
```

Processing Data

Let's get a good look at our data by using the `head` command:

```
cd genomic_data
head GCA_009858895.3_ASM985889v3_genomic.fna
```

```
## >MN908947.3 Severe acute respiratory syndrome coronavirus 2 isolate Wuhan-Hu-1, complete genome
## ATTAAAGGTTTATACCTTCCCAGGTAACAAACCAACCAACTTTCGATCTCTTGTAGATCTGTTCTCTAAACGAACTTTAA
## AATCTGTGTGGCTGTCACTCGGCTGCATGCTTAGTGCACTCACGCAGTATAATTAATAACTAATTACTGTCGTTGACAGG
## ACACGAGTAACTCGTCTATCTTCTGCAGGCTGCTTACGGTTTCGTCCGTGTTGCAGCCGATCATCAGCACATCTAGGTTT
## CGTCCGGGTGTGACCGAAAGGTAAGATGGAGAGCCTTGTCCCTGGTTTCAACGAGAAAACACACGTCCAACTCAGTTTGC
## CTGTTTTACAGGTTCGCGACGTGCTCGTACGTGGCTTTGGAGACTCCGTGGAGGAGGTCTTATCAGAGGCACGTCAACAT
## CTTAAAGATGGCACTTGTGGCTTAGTAGAAGTTGAAAAAGGCGTTTTGCCTCAACTTGAACAGCCCTATGTGTTCATCAA
## ACGTTCGGATGCTCGAACTGCACCTCATGGTCATGTTATGGTTGAGCTGGTAGCAGAACTCGAAGGCATTCAGTACGGTC
## GTAGTGGTGAGACACTTGGTGTCCTTGTCCCTCATGTGGGCGAAATACCAGTGGCTTACCGCAAGGTTCTTCTTCGTAAG
## AACGGTAATAAAGGAGCTGGTGGCCATAGTTACGGCGCCGATCTAAAGTCATTTGACTTAGGCGACGAGCTTGGCACTGA
```

Lots of A,C,T and G's! Let's count the number of lines in the file:

```
cd genomic_data
wc -l GCA_009858895.3_ASM985889v3_genomic.fna
```

```
## 375 GCA_009858895.3_ASM985889v3_genomic.fna
```

This command is pretty simple, lets build a more complex one, lets count the amount of A,C,G and Ts in the file:

```
cd genomic_data
grep -o 'A' GCA_009858895.3_ASM985889v3_genomic.fna | wc -l
grep -o 'C' GCA_009858895.3_ASM985889v3_genomic.fna | wc -l
grep -o 'G' GCA_009858895.3_ASM985889v3_genomic.fna | wc -l
grep -o 'T' GCA_009858895.3_ASM985889v3_genomic.fna | wc -l
```

```
## 8954
## 5492
## 5863
## 9594
```

One codon (sequence of base pairs) that we care about is called Methionine, which is the start of a sequence, it has the designation `ATG`. Let's count the number of times this codon appears in the file. First we remove the header from the file:

```
cd genomic_data
tail -n +2 GCA_009858895.3_ASM985889v3_genomic.fna > GCA_009858895.3_ASM985889v3_genomic_no_header.fna
```

Then we count the number of times the codon appears:

```
cd genomic_data
grep -o 'ATG' GCA_009858895.3_ASM985889v3_genomic_no_header.fna | wc -l
```

```
## 709
```

We can also save chunks of this file to a new file, say we want to save the first 100 lines of the file to a file called `first_100_lines.fna`:

```
cd genomic_data
head -n 100 GCA_009858895.3_ASM985889v3_genomic_no_header.fna > first_100_lines.fna
```

and by using `ls` we see that we have sucessfully created the file.

```
cd genomic_data
ls
```

```
## first_100_lines.fna
## GCA_009858895.3_ASM985889v3_genomic.fna
## GCA_009858895.3_ASM985889v3_genomic_no_header.fna
## ncbi_dataset
## ncbi_dataset.zip
## README.md
```

Bonus: CRON

So I have a few pet projects that I run in my spare time, one of them is a hunger games style bot which emails me and my friends at regular intervals with updates on an automated battle which goes on in a python program. I faced a problem, that this program needed to run constantly if it were to work, but I didn't want to constantly have a python program running on my laptop. The solution was to use CRON, a linux tool that allows you to schedule tasks to run at certain times.

You can access the `cron` table by typing `crontab -e` into the command line. This will open up a text editor where you can write your cron jobs.

```
crontab -e
```

The syntax for a cron job is as follows:

```
* * * * * path_to_file_to_run
```

where the asterisks represent the time and date at which the job will run. The first asterisk represents the minute, the second the hour, the third the day of the month, the fourth the month, and the fifth the day of the week. So for example, if you wanted to run a job at 3:30pm every day, you would write:

```
30 15 * * * path_to_file_to_run
```

I used this to run my python program every 3 hours, so I wrote:

```
0 */3 * * * /usr/bin/python3 /home/et18646/Documents/BattleIsland/Python_Code/Battle_Sim.py
```

To save, you can press `ctrl + x` and then `y` to confirm, then press `enter` to exit. It shows you that you have installed a new job via

```
crontab: installing new crontab
```

and you can additionally check by entering `crontab -l` to list all the cron jobs you have installed. It's a really great tool that has led to a lot of fun so I thought I'd include it here! I'm sure it has practical uses too.