Programming for Bioinformatics | BIOL7200

Week 5 Exercise

September 20, 2022

The goal of these exercises is to get you used to working with some basic UNIX commands and their options. Try and think about what each command does, and where it can be used. Reading the `man` page for each of these is recommended.

We're going to cover the following commands:

Command	Explanation
`sort`	Sort some data
`uniq`	Find unique lines in input
`tee`	Create a tee junction
`if, else, elif`	Conditional operators
`for, while, until`	Loops
`select, case`	Conditional on loops
`getopts`	Command line argument building utility

Instructions for submission

- Prepare two solution sheets for this exercise - one for submission and another for your own reference. Submit the submission sheet on Canvas. The solution sheet you create for your reference should help you going forward - make it as detailed/brief as you'd like for your own learning style. For the submission sheet, copy the question and write the correct answers below the question.
- You also need to submit the script you write for Mini Challenge 3 on Canvas.

To be clear – you will submit two files to Canvas: (1) your solution sheet for the exercises and (2) your script for Mini Challenge 3.

Exercises

We start our shell scripting discussion this week. Most of the concepts in this exercise will be covered in the class. There are a couple of additional commands that are used in this exercise but will not be discussed in the class. Part of the homework will be understanding what these commands are and how to use them.

sort and uniq

1. More piping with sort and `uniq` (This is a repeat from a previous assignment, just a little refresher)
    1. Create the following file:
    `perl -e 'foreach(1..100){print $_."\n".($_ / 2)."\n"}' > uniq.txt`
    2. sort the file numerically and output it to another file without using `>` and count the number of lines
    3. Pipe the result of the sort to `uniq`. What happened? How many lines are there?
    4. Pipe the result of the sort to `uniq` and discard all lines that appear more than once. I.e., I don't want the lines (e.g., 1, 2, etc.) which occur more than once.
    5. Pipe the result of the sort to `uniq` and count the number of times each number appear

Basic Shell scripting

2. Write shell scripts for the following:
    1. Write a shell loop that adds up all the numbers from 1-100
    2. Write a shell loop that prints each letter in the word `Hello` separately. This can be done using a `for` loop iterating through the letters `H e l l o`. This is a 5 line script; don't make it harder than that.
    3. Have variable `x` increase from 0 to 100 and variable `y` decrease from 100 to 0 by 1 unit at a time and print `"x and y are equal"` when the variables are equal
    4. Repeat the above (c) but this time print the difference between `x` and `y` (`$x-$y`) when they are not equal and print `"x and y are equal"` when they are equal
    5. Write a `getopts` block with 3 options including one that requires an argument
    6. Write a script utilizing `case` that asks the user the day of the week and prints the day number based on user input
    `Monday = 1; Tuesday = 2; ...; Sunday = 7`

    Example terminal output (lines that start with `>` are output from the script, lines that start with `$` are user input):

```
./scriptName.sh
> Please enter a day:
$ Monday
> Monday is day number 1
```

Mini challenge part 1

3. We are going to do an exercise for fun, but it will enforce shell concepts and will help you prepare for this week. To make things easier for you, I have provided you with stepwise

instructions on how to proceed. You will add one element each time, making the loop a little more complicated as compared with the step before.

1. Write a `for` loop to create 10 files by the name `seq1.fasta`, `seq2.fasta`, `seq3.fasta`, and so on. You can create a file using the touch command.
2. Modify this loop to now do two more things:
    i. Delete the `seq1.fasta`, `seq2.fasta`, `seq3.fasta`, etc. if they exist
    ii. Create new `seq1.fasta`, `seq2.fasta`, `seq3.fasta`, etc. files with a FASTA description line in it. The FASTA description lines needs to be like this: `>seq1 for seq1.fasta`, `>seq2 for seq2.fasta`, `>seq3 for seq3.fasta`, etc.
3. Add another element to this loop: the loop now also adds a random DNA sequence along with the FASTA description line. The following command will give you a random DNA string of 50x10 letters:

```
cat /dev/urandom | tr -dc 'ACGT' | fold -w 50 | head
```

Mini challenge part 2: Let's try nesting the loops

4. Nesting loops is really having a loop within a loop. Instead of creating 10 single sequence files, as in part 1, we will create 10 multiple sequence FASTA files (aka, 10 multi-FASTA files) with 8 sequences in each.

   The FASTA descriptors for file 1 will go like this: `>seq1_1`, `>seq1_2`, `>seq1_3`, …; the FASTA descriptor for file2 will go like this: `>seq2_1`, `>seq2_2`, `>seq2_3`, … ; etc.

Mini challenge part 3: Now add `getopts` to the beginning of the script

5. For the final part of this series of questions, I want you to add `getopts` to the script so that it takes take three options:

   - Option `n` => will take a number as an input. This argument describes the number of files to be created (which until now was 10)

   - Option `m` => will take a number as an input. This argument describes the number of sequence to be added in each file (which until now was 8)

   - Option `v` => verbose mode, i.e., I want the script to print out every single action it is doing (e.g., what file is the script currently working on? what sequence number?). This is a flag, so no input is expected with it.

- Again, you should submit a script for the 3rd Mini Challenge as part of your submission on Canvas.