Introductory Linux Tutorial for Life Sciences

Session 9: Exercises

- The following seven exercises are ideally solved in pairs.
- To solve these exercises, it is assumed that the knowledge gained from the previous tutorial sessions is used.

Each solution should be a shell script that makes use of variables where appropriate, and that is documented with moderate use of comments.

Data files for the exercises that require these are available in the directory `~/data/pairEx` on the E-learning virtual machine.

Scripts should be stored in the `~/myLinuxProject/scripts` directory and if an exercise requires you to create an output file, this should be written to the directory `~/myLinuxProject/results/courseDay_03` unless the exercise says something else (create these directories if they don't already exist).

The point of this is twofold:

1. Everyone has the same directory conventions, and
2. It shows that you are able to work with directories.

Use the `man` command to get a reference manual for commands that you are unsure of how to use. For example, use `man ls` to read the manual for the `ls` command.

Exercise 1

Create a short shell script called `date.sh` using the text editor of your choice. The script should start with the following:

```
#!/bin/bash -eu
```

The rest of the script should print out the current date and time using the `date` command.

Make the script executable and run it.

Exercise 2

Write a shell script called `name.sh` that takes arguments from the command line. Given three arguments, a first name, a surname, and a country, the script should use these strings to form a reply.

Running the script may look like this:

```
$ ./name.sh Jan Gustafsson Sweden
My first name is Jan.
My surname is Gustafsson.
I'm from Sweden.
```

Exercise 3

Write an interactive shell script named `interactive.sh` that has a simple dialog with the user, possibly something like this (the blue text is given by the user running the script):

```
$ ./interactive.sh
What is your name? Dagobert
Hello Dagobert!
What is your Surname? Duck
I'm very pleased to meet you, Dagobert Duck!
```

This would probably require you to use the `read` command. Experiment with using the command's `-p` option to set the prompt, and you may possibly want to test using a timeout with e.g. `-t 3` (or similar), or read secrets using the `-s` option (or combinations thereof).

Feel free to make up funnier or more useful dialog.

Exercise 4

The columns in the file `anno.gff` are delimited by `::` (double colon).

```
Scaffold1::GeneWise::CDS::1641987::1642110::.::-::0::Parent=Pad_R000152;
Scaffold1::GeneWise::CDS::1642411::1642512::.::-::0::Parent=Pad_R000152;
Scaffold1::GeneWise::CDS::1655666::1655868::.::-::2::Parent=Pad_R000152;
Scaffold1::GeneWise::CDS::1664843::1666751::.::-::0::Parent=Pad_R000152;
```

Write a script that reads this specific file (which is quite large, the above only shows a limited number of lines) and writes it to `anno-corrected.gff` with all double colons replaced by tabs. The location of both input and output files should be stored in variables in the script.

The final result (given only the line shown above) written to `anno-corrected.gff` should look something like

---

```
Scaffold1	GeneWise	CDS	1641987	1642110	.	-	0	Parent=Pad_R000152;
Scaffold1	GeneWise	CDS	1642411	1642512	.	-	0	Parent=Pad_R000152;
Scaffold1	GeneWise	CDS	1655666	1655868	.	-	2	Parent=Pad_R000152;
Scaffold1	GeneWise	CDS	1664843	1666751	.	-	0	Parent=Pad_R000152;
```

(but should obviously include all data from the original file).

Extra task for this exercise

Would you be able to write a script that instead of a specific infile and outfile used the names given as two arguments on the command line of the script as input and output filenames?

What about a script that just reads using an input redirection (`<`) and writes to some output using an output redirection (`>`) so that one would be able to say e.g.

```
$ ./myscript.sh <anno.gff >anno-corrected.gff          # or...

$ cat anno.gff | ./myscript.sh >anno-corrected.gff     # same thing
```

(I've left out the paths to the input and output files in the above text for brevity.)

Exercise 5

The file `nameList.txt` contains a number of names of people. Some of these names are duplicated, but most are not.

Write a script that extracts the unique names to a new file called `names-unique.txt`, and that also extracts the duplicated names to a new file called `names-duplicates.txt`.

For your reference, the duplicated names are Anna, Lina, and Tyler. There are further 30 unique names.

Exercise 6

The file `animalSounds.tab` is a tab-delimited file with two columns. The first column contains the names of animals while the second column contains the sounds they make.

Write a script to figure out what sound is the most common sound listed in the file. Also extract the number of times this particular sound is listed.

For your reference, the sound is `grunt`, and it occurs three times.

Exercise 7

The file `myCDF.fa` contains multiple DNA sequences in fasta format. Each sequence is written on a single long line, which means that the file is made up by alternating header lines (prefixed with `>`) and sequence lines.

Write a script that

- Counts the number of sequences in the file and output this number in the terminal,
- Folds the long sequence lines to 60 characters using the `fold` command (use the command `man fold` to read how this command is used; you can assume that no header line in the file is longer than 60 characters),
- Change all lowercase sequence letters to uppercase (for an extra challenge, try doing this without affecting the sequence header lines, even though they are already upper-case (pretend they're not!)), and
- Writes the modified sequence, with the sequence headers, to a new file called myCDS-fixed.fa.

For your reference, the first sequence in the file should come out as

---

```
>MLOC_7.2
ATGAAAAATGTGGAGGTCGAGTCTAATTGCCAGACACTAATAAAAGCGCTCCAAAGTTCA
GTTTATGACAGGGCACCAGAAGGAATTTTATTGAGAGAGATTTGGGACTTCGCTCGTCTA
AGTTTCAGTGCTTGCACGTTTTATTTTGCCCCCAAGGCATGTAATAACTTGGTGCATGCT
CTTGCAGCTTTTGGTGCCTCTCAACAGGCAGGCCTACATCTGTGGTTAGAGGACCTGCCT
GACAAGGTACTTGTT
```
