Unix/Linux Tutorial for Beginners
Session VIII – Pairwise exercises

Solve the following tasks by using simple shell scripts. Define the paths to the input data, as well as the output files as variables. Use comments to document your script.

The input files mentioned below should be located (unless otherwise stated) in your ~/myLinuxProject/myData/raw/pairEx folder. The shell scripts should be stored in the folder ~/myLinuxProject/scripts, while the result files should be stored in the folder ~/myLinuxProject/results/2017-03-28/pairExShell.

Create the folders if they don't exist and use the command man <command> to get help and see all available options.

1. Create a simple shell script called 'date.sh' using a command line editor of your choice. Your script should include the Bash script header

```
#!/bin/bash
set -e
set -u
```

   and print out the current date using the command date. Make the script readable, writable, and executable for the user and the group using the command chmod. Check the permissions to make sure that they are set correctly. Execute the file by one of the methods presented in the slides.

   Shell script:

2. Write a shell script named name.sh, which takes arguments from the command line and returns the following text back:

```
$ ./name.sh Jan Gustafsson Sweden
My first name is Jan.
My surname is Gustafsson.
I'm from Sweden.
Total number of arguments is 3.
```

   Shell script:

3. Write an interactive shell script named interactive.sh, which will have the following dialog with the user.

```
$ ./interactive.sh
What is your name?
  Dagobert
Dagobert, a very nice name. And your surname?
  Duck
Hello Dagobert Duck! I'm excited to meet you.
Are you sure you're up for this challenge?
  Yes
Your answer is: Yes, but I'm not convinced. How old are you?
  18
You are to young with 18. See you later aligator.
```

The answers provided by the user should be provided interactively via the keyboard. Use your own answers. You don't need to give the same answers as shown above. Only the questions should be included in your shell script. Feel free to create your own dialog, if the one above is to boring. Use the -s and -t options of the command read to make it more funny.

Shell script:

4. The columns in the file *anno.gff* are separated by '::' instead of tabs.

```
C10002475::GeneWise::mRNA::3::104::56.14::-::.::ID=Pad_R000001;Source=ENSTGUT00000006161;
C10002475::GeneWise::CDS::3::104::.::-::0::Parent=Pad_R000001;
C10002475::GeneWise::mRNA::16::291::65.91::+::.::ID=Pad_R000002;Source=ENSGALT00000035625;
C10002475::GeneWise::CDS::16::136::.::+::0::Parent=Pad_R000002;
C10002475::GeneWise::CDS::239::291::.::+::2::Parent=Pad_R000002;
...
```

Write a shell script which replaces all occurances of the string '::' with a tab and write the output to the file *anno_corrected.gff*.

Answer: The new created file should look as follow:

```
$ ls ~/myLinuxProject/results/2017-03-28/pairExShell/
anno_corrected.gff

$ less anno_corrected.gff
C10002475  GeneWise  mRNA  3    104  56.14  -  .  ID=Pad_R000001;Source=ENSTGUT00000006161;
C10002475  GeneWise  CDS   3    104  .      -  0  Parent=Pad_R000001;
C10002475  GeneWise  mRNA  16   291  65.91  +  .  ID=Pad_R000002;Source=ENSGALT00000035625;
C10002475  GeneWise  CDS   16   136  .      +  0  Parent=Pad_R000002;
C10002475  GeneWise  CDS   239  291  .      +  2  Parent=Pad_R000002;
```

Shell script:

5. Extract all duplicated and unique identifier from the file *nameList.txt* and save the results in the files *duplicated_names.txt* and *uniq_names.txt*. How many unique and how many duplicated identifier have you found?

   Answer: There are 3 duplicated names (Anna, Lina, and Tyler) and 30 unique ones.

   Shell script:

6. The tab-separated file *animalSounds.tab* list animals and the sounds they make. Extract the second column and determine which sound is the most common among the listed animals and how often it occurs.

   Answer: The most common sound in the animal list is *'grunt'* and it occurs 3 times.

Shell script:

7. Count how many sequences are stored in the file *myCDS.fa* and print the number to the standard output using the command echo. In a next step reverse complement the sequences (not the sequence identifier) and convert at the same time the lowercase letters to uppercase letters. The complements of the nucleotides A, C, G, T are T, G, C, A. Use the command paste to obtain a standard fasta file (with both, the sequence identifier and sequence). Write the output in your result folder in a file named *myCDS_revCompl.fa*.

Answer: There are 10 sequences stored in the fasta file *myCDS.fa*. The reverse complemented and to uppercase letters converted file looks as follow:

```
$ less ~/myLinuxProject/results/2017-03-28/pairEx/myCDS_revCompl.fa
>MLOC_7.2
AACAAGTACCTTGTCAGGCAGGTCCTCTAACCACAGATGTAGGCCTGCCTGTTGAGAGGCACCAAAAGCTGCAAGAGCATGCACCAA
GTTATTACATGCCTTGGGGGCAAAATAAAACGTGCAAGCACTGAAACTTAGACGAGCGAAGTCCCAAATCTCTCTCAATAAAATTCC
TTCTGGTGCCCTGTCATAAACTGAACTTTGGAGCGCTTTTATTAGTGTCTGGCAATTAGACTCGACCTCCACATTTTTCAT
>MLOC_16.1
GATTGATACTGCAAGACTACTGATAAATCTGCCATAACTAAGCTTCAATTCCTGCCAATTTCCAACATCAACCAATCCTTCCTCATG
TCCATTGGAAGTGCGTGCCACAAGGGATTCGAACTTGCGGTCATTGCTGGTGAGAATACAGTAGGCCCTCAGCAGATCACCCCGTCC
TAAGTTTGGTATCTCTTGAAGTGCCGCAAAGATTTCCTTCGAAGATGCATTCGCTGGTTCCTCTCGAGGTTCATTTGGCTTCACAGA
TATCTGAAATGAATTATTGATGTTCCCAGACATGCAAAAGAGTGCATCATCTGTTCTAAACCTCTTTCTCCTCCTGGCCTGTAGAGT
GGGTTGCGTTGATGTCGGCGTTCCAACCTGGTTTTGACTGCTAGGTTCAGATGCTGAAGGTGAATTTTGCATAAGCTCCTGGACATC
TTCAGCACCCTTCAATGCCATTTCTTCGGCATTCTCCAT
....
```

Shell script:

Exercises are in part derived by material from ©Software Carpentry (`http://software-carpentry.org`, license: CC BY 4.0) that was adapted from me for this course. Another part is from a BILS course given by Martin Dahlö and used here by his kind agreement. Remaining exercises by M. Martis.