Unix/Linux Tutorial for Beginners
Session IX – Shell scripts

Keep in mind that tilde (∼) can’t be used together with a path in a bash script. Use $HOME followed by the path or the absolute path instead.

1. Write a shell script called fileBasename.sh which iterates through the directory *∼/data/fasta* and prints out the names of all fasta files without the path information. The path to the directory to iterate through should be saved in a variable. Hints: use a for-loop and the command basename.

2. Write an interactive shell script named login.sh, which will request the following information from the user: *name, surname, username, password, age, and affiliated university*. The user will provide this informations via keybord. The *’password’* should not be visible while typing it in the terminal. Furthermore, you need to test if the password has a certain length (minimal 8 characters). If the length of the password is below this value, then the user will be asked to type the password again.

   The length of a string can be determined by using the command ${#string}, while the word *’string’* should be replaced with the name of your variable. After collecting all information, return the following message to the user:

```
The registration was successful! We have received the following data from you:
name: name surname
age:  age
university: university
user: username
password: password.
Please check your password. Does it correspond to what you had in your mind?
```

3. The next task involves the empty files stored in the folder *~/data/mzML*. The files should be renamed, filled with information, and copied to your result folder *~/myLinuxProject/results/mzML*.
   First rename the files by removing the prefix **’0_’** (now: 0_C1.mzML, new: C1.mzML). Second, copy the files to your results directory, and third write in each file the following message: **”I’m the file number N”**. Please replace ’N’ with the respective serial number of the file. A variable can be incremented by initialising it first with the value 0 (var=0) and by using arithmetic expansion ((var+=1)) in the loop.
   To test things first, use the command echo to display your commands (e.g.

'mv $file' with 'echo mv $file'). Use the commands grep, ls, and wc -l to check if the number of files copied to your result directory corresponds to the number of files in the folder /data/mzML, as well as wheather the information was written in all files or not.

4. Write a shell script called lineByline.sh which reads line by line the file *~/data/others/Peters_shoppingsList.txt* and prints out only those lines which start with a capital C. Hint: use the *if-statement* with a regular expression comparison (=~) to identify those lines.

5. Write a shell script called fastqc.sh, which runs the program FastQC on all FASTQ files whose names start with 'S' in the folder *~/data/fastq*. Redirect the results to the directory *~/myLinuxProject/results/fastqc*. Create this folder using the command mkdir within the script. The FastQC tool checks the quality of your FASTQ files. The tool can be invoked by just typing the command fastqc either on the command line or in your shell script. Use fastqc --help to learn more about how to run the FastQC tool. It is not necessary to decompress the FASTQ files. As soon as your job finishes, inspect the produced results.

6. Explain what the script below does when run as ./guess.sh:

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

file="$HOME/data/fasta/brachy_CDS.fa"

for i in A C G T
do
  var=$(grep -v ">" $file | grep -o $i | wc -l)
  echo "$i: $var"
done
```

7. Extend the script *guess.sh* described above, so that it calculates the total amount of base pairs in the given fasta file and its GC content.

8. Write a shell script called longest.sh that goes through all *.txt located in the directory ~/data/others, and prints out the name of the file with the most lines.

9. Write a shell script called reverse_complement.sh that goes through the directory *~/data/fasta/subset* and reverse complement all fasta files (rev, tr). Write the output to a new folder which you create within your script. Use the command basename together with the suffix .fa to extract the base name of your file. Add to the basename of your file the suffix '_revCompl.fa' and save it in the new created output directory. Hint: you can use a *for-loop* to iterate through the files, a *while-loop* to read each file line by line and a *if-statement* to check which line has been processed (description or sequence line).

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.