Exercises

Commandline tools

I. TAR & GZIP

1. Use gzip to compress the file P12931.txt
2. Decompress the resulting file P12931.txt.gz (revert previous command)
3. Use tar to create an archive containing all fasta files in the current directory into an archive called "fastafiles.tar"
4. Use gzip to compress the archive "fastafiles.tar".
5. How can you achieve the two previous steps "using tar to create archive" and "gzip the archive" in one command?
6. Test (list the contens of) the compressed archive "fastafiles.tar.gz"
7. Download the compressed PDB file for entry 1Y57 from rcsb.org (eg. "wget http://www.rcsb.org/pdb/files/1Y57.pdb.gz") and decompress it.

II. GREP

1. Which of the DNA files ENST.* contains 'TATATCTAA' as part of the sequence?
2. Which of the DNA files ENST.* contains 'CAACAAA' as part of the sequence?
3. Considering the previous example, would you consider grep a suitable tool to perform motif searches? Why not? Try to find the pattern 'CAACAAA' by manual inspection of the first two lines of each sequence.
4. Count the number of ATOMs (lines starting with 'ATOM') in the file 1Y57.pdb.
   Does this number agree with the annotated number of atoms (Search the REMARKs for 'protein atoms')

III. SED

1. Use sed to print only those lines that contain "version" in the files P12931.txt and P04062.txt
2. Use sed to change the text 'sequence version 3' to 'sequence version 4' in the files P12931.txt and P04062.txt (without actually changing the files, just printing)
3. Use sed to update the text 'entry version 3' to 'entry version 4' in the files P12931.txt and P05480.txt (this time, make the changes directly in the files)
4. Replace all occurrences of 'r' by 'l' and 'l' by 'r' (at the same time) in the file PROTEINS.txt (so that 'structural' becomes 'stluctular')

IV. AWK

1. Use awk to print only those lines that contain "version" in the files P12931.txt and P05480.txt and compare the procedure to sed.
2. For all FASTA files print only the second item of the header (split on "|")
   eg. for ">sp|P12931|SRC_HUMAN Proto-oncogene", print only "P12931"
3. The file 'P12931.csv contains phosphorylation sites in the protein P12931. (If the file 'P12931.csv' does not exist, use 'wget' to download it from "http://phospho.elm.eu.org/byAccession/P12931.csv").
   a. Column three of this file lists the amino acid position of the phosphorylation site. You are only interested in position 17 of the protein. Try to use 'grep' to filter out all these lines containing '17'.
   b. Now use awk to show all lines containing '17'.

c. Next try show only those lines where column three equals 17 (Hint: The file is semicolon-separated…).
d. Finally print the PMIDs (column 6) of all lines that contain '17' in column 3.

Quoting and Escaping

1. Familiarize yourself with quoting and escaping.
   a. Run the following commands to see the difference between single and double quotes when expanding variables:

```
# echo "$HOSTNAME"
# echo '$HOSTNAME'
```

   b. Next, use ssh to login to a different machine to run the same command there, again using both quoting methods:

```
# ssh teach01@pc-atcteach01 'echo $HOSTNAME'
```

```
# ssh teach01@pc-atcteach01 "echo $HOSTNAME"
```

   Closely inspect the results; is that what you were expecting? Discuss this with your neighbour.

Basic scripting

1. **gzip** can use nine different levels of compression. Read 'gzip –help' or 'man gzip' to find out how to compress a file while keeping the original file unchanged and how to use these different compression levels. Then write a **for**-loop to iterate over all possible levels, compressing the file 1Y57.pdb and compare the sizes of the resulting nine compressed files.
2. Write a script that combines the two files "P00523.fasta" and "P12931.fasta" into one file called "twofiles.fasta" and run /g/software/bin/clustalw2 on these files. (OPTIONAL) Finally, view the resulting alignment "twofiles.aln" using /g/software/bin/clustalx.
3.
   a. Write a script that tests the existence of a directory and if it exists, prints out the content list as a nice bulleted list (using "*" as bullets). If the directory does not exist, then print out the error message of "ls <directory>".
   b. Modify the above script, so that the directory can be given as commandline argument. Print out an error message, if no commandline argument is given or if the given argument is not a directory.
   c. Modify the script, so that the directory must be given as an option (e.g. myscript.sh -d <directory>)
   d. If you used file testing expressions, then modify the script to only use "ls" and its exit status to test for the directory existence. "ls" should only be run once (!) in the course of the whole script (Hint: Use temporary files)
4. (OPTIONAL) Write a for loop to get all (unique) PDB ids (column 13) from file P12931.csv and retrieve the corresponding PDB file using this URL scheme: "http://www.rcsb.org/pdb/files/*ID*.pdb"
   eg. "wget http://www.rcsb.org/pdb/files/1Y57.pdb"