LAB2_Tips

SURVIVAL TIPS

Absolute and relative pathnames in UNIX

A path is a unique location to a file or a folder in a file system of an OS. An absolute path-name starts at the root directory (/) and work down, writing a slash (/) after every directory name (last one is optional).

Relative path is defined as the path related to the present working directly(**pwd --> see first line in useful bash commands**). It starts at your current directory and **never starts with a /** .

`.(a single dot)` - this represents the current directory. `..(two dots)` - this represents the parent directory.

Examples:

1. **Changing directory with relative path concept:**

```
pwd
/home/kt
cd abc
pwd
/home/kt/abc
```

2. **Changing directory with absolute path concept:**

```
pwd
/home/kt
cd /home/kt/abc
pwd
/home/kt/abc
```

Useful bash commands

pwd --> outputs the **global path** of directory in which you are

`cd dir_name` --> moves inside the path specifyed by dir_name. E.g. `cd Documents` or `cd /home/marta/Documents` or `cd ..` (to move to the previous directory)

`mkdir` --> creates a new folder

`ls` --> lists files and folders in the actual directory (hidden files not showed)

`l` --> lists all files and folders in the actual directory

`touch pippo.txt` --> creates a file and names it pippo.txt

`less pippo.txt` --> outputs the content of pippo.txt (press ‘q’ to exit from file visualization)

`cat pippo.txt` --> outputs the content of pippo.txt

`more -d pippo.txt` --> displays pippo.txt inside terminal

`rm` --> deletes files

`vim pippo.txt` --> advanced editor inside a terminal (press ‘i’ to edit, ‘esc’ and the ‘:wq’ to exit file visualization saving all changes made). If you are not a PRO, we suggest you to use an external text editor to edit your files

`nano pippo.txt` --> editor inside a terminal (press 'Ctrl + X' to exit file visualization). If you are not a PRO, we suggest you to use an external text editor to edit your files

`cp path1/file1.txt path2/file2.txt` --> copies file1.txt (which is located in directory path1) into directory path2 and names it file2.txt

`mv path1/file1.txt path2/file2.txt` --> moves file1.txt (which is located in directory path1) into directory path2 and names it file2.txt

`grep 'word' filename` --> Search any line that contains the word in filename. For a recursive search use: `grep -r ‘word’ filename`

`find path -name filename` --> search for filename in directory defined by path. E.g. `find . -name ‘pippo.txt’`

`head pippo.txt` --> displays the beginning of pippo.txt file. `head -n 10 pippo.txt` displays the first 10 lines of pippo.txt

`tail pippo.txt` --> displays the end of pippo.txt file. `tail -n 10 pippo.txt` displays the lasts 10 lines of pippo.txt

`wc -l pippo.txt` --> displays the total number of lines of `pippo.txt`

Create and run a bash script

A Bash script is a plain text file which contains a series of commands. Anything you can run normally on the command line can be put into a script and it will do exactly the same thing. Similarly, anything you can put into a script can also be run normally on the command line and it will do exactly the same thing.

Create your file (e.g. `my_first_script.sh`) in the directory you prefer with vim or a text editor.

• The script must start with:
```
#!/bin/bash
```
• Variables are defined in this way:
```
variablename = value
```
• Values from variables are extracted with $ simbol
```
$variablename
```
• `$1, $2, $3`, … respectively reads arguments from the command line (`$1` pass the first argument and so on. You can have N arguments to be passed to the script)
• echo print messages to the final user or values of variable
```
echo “message”
echo $variablename
```

Finally, to run the script in your bash shell, move to the directory in which `my_first_script.sh` has been saved and then type:
`sh scriptname` (e.g. `sh my_first_script.sh`)

Example: BWA bash script

```
#!/bin/bash
source activate Bioinfo_labs
echo "BWA initializing..."

BWA_index="/home/marta/Documents/BIOINFORMATICS/BioInfoCourse/LAB_alignmen
t/tools/bwa_index/bwa_index"
mate_1=$1
mate_2=$2
out=$3

echo "BWA aligner is currently running. This step could require time, be
patient! "
bwa mem $BWA_index $mate_1 $mate_2 > $out

echo "Alignment is finished! You can now check your results."
```

Run the script

(the following two lines must be typed on the same input line in the terminal)

```
sh example_script.sh mate_1_file_with_path mate_2_file_with_path
output_filename_with_path
```

INSTALL BWA

With the following steps you will install the two tools on your laptop. If you encounter problems in the installation process, feel free to contact us to solve the problem.

Open your terminal (for MacOS and Linux users) or Ubuntu 18.04 LTS app (for Windows) and type:

```
conda activate Bioinfo_labs (or source activate Bioinfo_labs)
conda install -c bioconda bwa
```

Download reference files (chr10 and chr18) and mate_1.fq and mate_2.fq

In this lab we will use **real human data**. However, the human reference genome is 3 GB long and using it in its entirety requires a lot of computational efforts. So, for convenience, we will use only two chromosomes (chr10 and chr18), but the same reasoning can be extended on all DNA.
Download the two files from here:
ftp://ftp.ensembl.org/pub/release-92/fasta/homo_sapiens/dna/Homo_sapiens.GRCh38.dna.chromosome.18.fa.gz

ftp://ftp.ensembl.org/pub/release-92/fasta/homo_sapiens/dna/Homo_sapiens.GRCh38.dna.chromosome.10.fa.gz

Move the two files to your working folder and unzip them:
```
gzip -d *.fa.gz
```

Then, concatenate the two fasta files into one file named `reference_chr10_chr18.fa`

```
cat *.chromosome.*.fa > reference_chr10_chr18.fa
```

mate_1.fq and mate_2.fq are fastq files coming from a RNA-seq sequencing of human brain, and specifically of cerebral cortex. **The original experiment has produced more than 12 GB of data**!! Therefore, to allow you to experience a real aligner on real human data, the files have been truncated. Please, download them from the course website.