

TP - prise en main de `unix` et `bash`

Master parcours SSD - UE Gestion de Projet

L'objectif de ce TP est de prendre en main certaines fonctionnalités élémentaires de `unix` et `bash`. Pour cela nous travaillerons en ligne de commande, en utilisant un terminal "natif" si vous travaillez dans un environnement Linux, ou (par exemple) l'outil `git-bash` que l'on peut télécharger ici: <https://gitforwindows.org/> si vous êtes sous Windows.

Commencer par télécharger et décompresser le fichier `tp-unix.zip` dans votre répertoire personnel, ce que devrait vous créer un répertoire `tp-unix` contenant 5 fichiers (`ard.csv`, `file1.txt`, `file2.txt`, `genome.fasta` et `kmers.fasta`).

1 Exercice 1: principes de base

Dans ce premier exercice, nous allons voir et manipuler quelques principes de bases liés à l'environnement `unix`.

1.1 Commande `cd`

Ouvrir un terminal, exécuter les commandes suivantes et commenter les résultats obtenus:

- `$ pwd`
- `$ cd tp-unix`, puis `$ ls` et `$ pwd`
- `$ mkdir test-dir`, puis `$ ls` et `$ pwd`
- `$ cd ..` et `$ pwd`
- `$ cd /usr/bin`

Pour aller plus loin: trouver la commande permettant de retourner dans le répertoire `test-dir` que vous avez créé précédemment à partir du répertoire `/usr/bin` où vous vous trouvez.

1.2 Commande `ls`

Ouvrir un terminal, retourner dans le répertoire `tp-unix` précédent et exécuter les commandes suivantes:

- `$ ls`
- `$ ls file*`
- `$ ls -a`
- `$ ls -l`
- `$ ls -lh`

Pour aller plus loin: trouver dans la documentation de la fonction `ls` (que l'on obtient en tapant la commande `$ man ls`) l'option (ou les options) à ajouter pour afficher les fichiers et leurs attributs, en les triant par taille croissante.

1.3 Commande cat

Ouvrir un terminal, retourner dans le répertoire **tp-unix** précédent et exécuter les commandes suivantes:

- `$ cat file1.txt`
- `$ cat file1.txt > file3.txt` puis `$ cat file3.txt`
- `$ cat file1.txt >> file3.txt` puis `$ cat file3.txt`
- `$ cat file1.txt > file3.txt` puis `$ cat file3.txt`

Pour aller plus loin: écrire le contenu du fichier `file2.txt` à la suite du fichier `file1.txt` dans un nouveau fichier, en n'utilisant qu'un seul appel à la fonction `cat`.

1.4 Commandes cp et mv

Ouvrir un terminal, retourner dans le répertoire **tp-unix** précédent et exécuter les commandes suivantes:

- `$ cp file*.txt ./test-dir` puis `$ ls ./test-dir`
- `$ cp -r test-dir test-dir-2` puis `$ ls test-dir*`
- `$ rm test-dir/file*` puis `$ ls test-dir*`
- `$ mv test-dir-2/file* test-dir` puis `$ ls test-dir*`
- `$ rm -r test-dir-2`

1.5 Variables d'environnement

- Afficher vos variables d'environnement avec la commande `$ env`.
- Afficher le contenu de la variable `PATH`.
- Vérifier que la commande `ls` est dans votre path en utilisant la commande `$ which ls`.
- Modifier votre fichier de configuration `.bashrc` pour définir un alias appelé `LL` permettant de lister les fichiers avec leurs attributs en les triant par taille croissante, comme vu précédemment.

2 Exercice 2: éditeur de texte vim

Nous allons à présent voir comment faire quelques manipulations élémentaires de fichier avec l'éditeur de texte **vim**. Pour cela, commencer par ouvrir un terminal et retourner dans le répertoire **tp-unix** précédent.

- Créer un nouveau fichier
 1. ouvrir l'éditeur avec la commande `$ vim`.
 2. passer en "mode insertion" en tapant sur la touche `"i"`, écrire du texte et quitter le mode insertion en tapant sur la touche `echap`.
 3. passer en "mode commande" en tapant sur la touche `":"` et sauvegarder le fichier en tapant la commande `w my_file`.
 4. quitter `vim` en tapant la commande `:q`.

Noter que l'on peut réaliser les deux dernières étapes d'un seul coup avec la commande `:wq my_file`.

- Naviguer dans et éditer un fichier existant
 1. ouvrir le fichier `kmers.fasta` avec la commande `$ vim kmers.fasta`.

2. se rendre à la ligne 527 en tapant `:527`.
 3. la supprimer en tapant deux fois sur la touche "d".
 4. aller ensuite à la ligne 123 et inclure la chaîne de caractères "-TEST" à la fin de la ligne.
 5. quitter `vim` sans sauvegarder le fichier en tapant la commande `:q`.
- Rechercher / remplacer
 1. ouvrir le fichier `kmers.fasta` avec la commande `$ vim kmers.fasta`.
 2. retourner à la première ligne du fichier si ce n'est pas déjà le cas.
 3. rechercher les occurrences du motif `kmer` en tapant la commande `/kmer`.
 4. se déplacer dans les occurrences suivantes et précédentes avec les touches "n" et "N".
 5. remplacer toutes les occurrences de `kmer` par `KMER` en passant en "mode commande", et en tapant la commande `%s/kmer/KMER/gc`. A quoi servent les termes "g" et "c" à la fin de cette commande?
 6. quitter `vim` sans sauvegarder le fichier.
 - Comparer deux fichiers
 - taper la commande `$ vim -d file1.txt file2.txt` pour comparer visuellement les deux fichiers.

3 Exercice 3: utilitaires unix

Dans ce premier exercice, nous allons voir comment manipuler quelques utilitaires clé de l'environnement `unix` (`wc`, `head`, `tail`, `cut`, `sort`, `grep` et `sed`), et comment les combiner via le mécanisme de redirection de "pipes", symbolisé par l'opérateur "|". Pour cela, commencer par ouvrir un terminal et retourner dans le répertoire `tp-unix` précédent.

- Commande `wc`
 1. utiliser les commandes `ls` et `wc` pour compter le nombre de fichiers ou de sous-répertoires du répertoire courant.
 2. faire de même pour compter le nombre de fichiers commençant par la lettre "f".
- Commandes `head` et `tail`
 1. utiliser la commande `head` afficher les 10 premières lignes du fichier `kmers.fasta`.
 2. afficher uniquement la 10ème ligne du fichier `kmers.fasta`.
- Commandes `cut` et `sort`
 1. utiliser la commande `cut` pour extraire la deuxième colonne du fichier `ard.csv`
 2. proposer deux manières d'extraire le 4ème champ de la 10ème ligne du fichier `ard.csv`.
 3. combiner les commandes `cut`, `sort` et `wc` pour compter le nombre de valeurs distinctes de la 4ème colonne du fichier `ard.csv`. Combien il y en a t'il ?
 4. combiner les commandes `cut`, `sort` et `uniq` (avec l'option `-c`) pour compter le nombre d'occurrences de chacune de ces valeurs.
- Commandes `grep` et `sed`
 1. utiliser la commande `grep` pour extraire du fichier `kmers.fasta` les lignes contenant le motif `kmer` (qu'il soit écrit en minuscules ou non). Combien il y en a t'il ?
 2. faire de même pour extraire les lignes ne contenant pas ce motif.
 3. utiliser la commande `sed` pour remplacer le motif `kmer` (qu'il soit écrit en minuscule ou en majuscule) par le motif `feature`.

4 Exercice 4: script shell

L'objectif de cet exercice est d'écrire un script shell visant à mettre en forme un fichier au format fasta, un format classique pour manipuler les données génomiques. Un fichier fasta est structuré de la manière suivante:

- il contient plusieurs séquences nucléiques, faites essentiellement de caractères **atgc**.
- chacune de ces séquences est précédée d'une ligne (ou "header") commençant par le symbole ">" et structuré comme ">sequence-id comments". Ce "header" permettant d'associer un identifiant (sequence-id) à chacune de ces séquences, ainsi que diverses informations en commentaires.

Nous travaillerons pour cela sur le fichier `genome.fasta`, dont les 30 premières lignes se présentent ainsi:

```
>573.33136.con.0121   SRR9858953_contig_121   [Klebsiella pneumoniae strain NR6227 | 573.33136]
nnngcctgtaaccactgtgaagatccggcctgcaccaaggtctgtccgagcggcgcgatg
cataagcgcgaagacgggtttgtggtggtcaacgaagaggtatgcattggctgccgctac
tgccatatggcctgcccgtacggcgccccgcagtacaatgccgacaaaggccatatgacc
aagtgcgatggctgtcacgagcgtgtcgccgaggggaaaaagccgatctgcgtcgagtcc
tgcccgtgcggcggttgatttcgggcccgttgcgaactgcgcgctaagcacggccag
ctggccgcggtagcgccgctgccgtcgccgcactttaccggccgagtatcgttatcaaa
cctaacgccaatgcccggcgtgtggcgataaccacgggttacctggcgaacccgaaggag
gtgtgagatgggaaacggatggcatgaatggccgctggtgctgtttacggtgctgggtca
gtgcgcgctggnn
```

```
>573.33136.con.0122   SRR9858953_contig_122   [Klebsiella pneumoniae strain NR6227 | 573.33136]
wbtccatccctcgtctgctgctaattgccccattcgccagctgccgtgatctgtccgaa
actggttgtagctgttggtaccccagctgtagccgctgacgttctccatctgcatgttggc
gaaagagtaattgccgtccaccacgcccggaggcagcctgcgatgttgactgagccccga
tgaagaaaaatggctgtaggcactggagaaggctgctcccatgctcttcacgatcccca
cgacaaaggcggatcatcacggcaatatagcccgggtggtcgcaatatccgagttttt
gagctgaacctgcgacaactcagagagaacgactgggacgccattttgcttcgcatagta
ggccattgcgctgttcaggatggcgaacagtaatggccacgtctgcagccagataagcgc
aaacacatacccccttttagcacctgcaganz
```

```
>573.33136.con.0123   SRR9858953_contig_123   [Klebsiella pneumoniae strain NR6227 | 573.33136]
gccatcccgcgtctgctgctcatgcctccattcgccagctgccgcacatctgtccgaa
actggttgtagctgttggtaccccagctgtagccgctgacgttctccatctgcatgttggc
aaacgagtaattgccatccaccacgcccgaagccgctgcgatgttgaccaagcccgga
agaagaaaaatggctgtaggcactggagaazrtzggcggtcccatgctacgatcccca
cgacaaaggcggatcatcacggcaatataacccgggtggtcgcaatatccgagttttt
gagctgaacctgcgacaactcagagagaacaaccgggacgccattttgcttcgcatagta
ggccattgcgctgttcaggatggcgaacagtaatggccacgtctgcagccagataagcgc
aaacacataccccctttaacacctgcagc
```

Elles contiennent donc 3 séquences ayant pour identifiants `573.33136.con.{0121/0122/0123}`.

Pour commencer:

1. afficher les 30 premières lignes du fichier pour prendre en main le format du fichier.
2. compter le nombre de séquences présentes dans le fichier en utiliser la commande `$grep ">" genome.fasta`. Attention à bien inclure les guillemets autour du caractère ">" au risque de perdre le contenu du fichier.

Ensuite: écrire un script shell visant à mettre en forme le fichier de la façon suivante:

1. supprimer les champs de commentaires dans les "headers" pour ne garder que les identifiants des séquences.
2. mettre systématiquement les séquences nucléiques en majuscules.
3. supprimer les lignes vides.

Bien qu'il y ait de nombreuses manières de faire, une manière de procéder peut consister en:

1. parcourir le fichier ligne à ligne
2. si la ligne est vide, passer à la suivante
3. si la ligne est un header, ne conserver que son identifiant (i.e., supprimer les commentaires)
4. sinon, passer le texte de majuscule avec la commande `tr` (pour translate): `tr [:lower:] [:upper:]`

Pour aller plus loin: complexifier le script pour passer les séquences en majuscules, en remplaçant tous les caractères autres que `atgc` par des "?". Cette dernière opération pourra par exemple se faire en utilisant la commande `sed`, via une expression régulière.