

Commandes Unix: pour les débutants

D. Puthier TAGC/Inserm, U1090, denis.puthier@univ-amu.fr

Matthieu Defrance, ULB, matthieu.dc.defrance@ulb.ac.be

Stéphanie Le gras, Igbmc, slegras@igbmc.fr

Christophe Blanchet, IFB, Christophe.BLANCHET@france-bioinformatique.fr

 aviesan
alliance nationale
pour les sciences de la vie et de la santé

 Aix-Marseille
université

umr U 1090
 TAGC
technological advances for genomics and clinics 

 Inserm

 Abims

 ifb

Le bureau MATE

Demo

Tour d'horizon rapide.



Installation:

<http://www.france-bioinformatique.fr/?q=fr/core/cellule-infrastructure/documentation-cloud>

Dashboard:

<https://cloud.france-bioinformatique.fr/cloud/instance/>



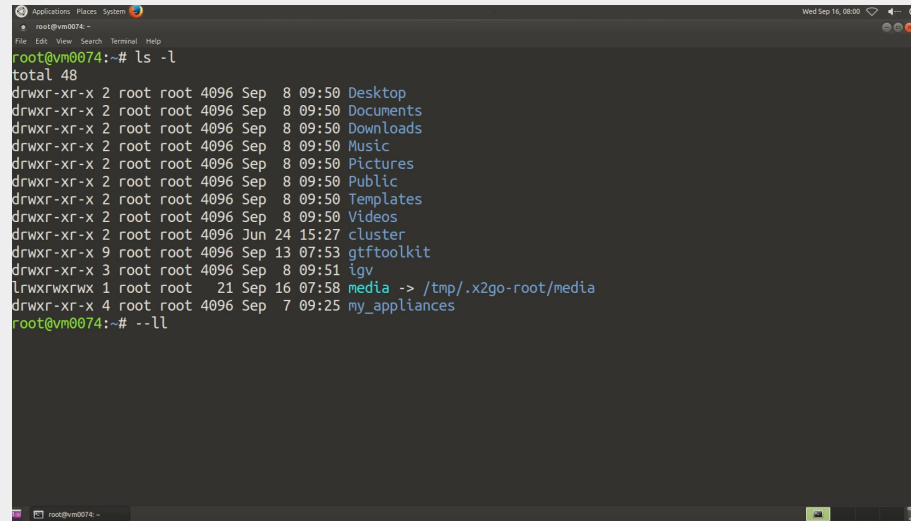
Le terminal...

Demo

Tapez 'ls' dans le terminal
(lister les fichiers)

lister les fichiers

```
root@vm: ls
```



```
root@vm0074:~# ls -l
total 48
drwxr-xr-x 2 root root 4096 Sep  8 09:50 Desktop
drwxr-xr-x 2 root root 4096 Sep  8 09:50 Documents
drwxr-xr-x 2 root root 4096 Sep  8 09:50 Downloads
drwxr-xr-x 2 root root 4096 Sep  8 09:50 Music
drwxr-xr-x 2 root root 4096 Sep  8 09:50 Pictures
drwxr-xr-x 2 root root 4096 Sep  8 09:50 Public
drwxr-xr-x 2 root root 4096 Sep  8 09:50 Templates
drwxr-xr-x 2 root root 4096 Sep  8 09:50 Videos
drwxr-xr-x 2 root root 4096 Jun 24 15:27 cluster
drwxr-xr-x 9 root root 4096 Sep 13 07:53 gft toolkit
drwxr-xr-x 3 root root 4096 Sep  8 09:51 igv
lrwxrwxrwx 1 root root 21 Sep 16 07:58 media -> /tmp/.x2go-root/media
drwxr-xr-x 4 root root 4096 Sep  7 09:25 my_appliances
root@vm0074:~# --ll
```

Comment converser avec le terminal ?

```
root@vm0074:~# Bonjour mon nom est Denis. Et toi ?  
Bonjour: command not found
```

- Réponse : lui parler en langage **BASH (Bourne Again Shell)** *
 - Le langage BASH est un des nombreux **dialectes** Shell (ksh, csh, zsh,...).
 - Tous ces langages Shell sont extrêmement similaires.
 - Ce langage est basé notamment sur un ensemble de **commandes**.
 - Ces commandes **modulaires** permettent de réaliser des **tâches**.

* Référence (calembour) au premier langage Shell écrit par Stephen Bourne :)

Prototype(s) d'une commande (1)

- Une **commande** réalise une tâche (trier, sélectionner, ouvrir, aligner des reads,...).
- Elle dispose d'un certain nombre d'**arguments** qui peuvent être facultatifs et qui peuvent **modifier son mode de fonctionnement**.
- Ces arguments peuvent ou non prendre des **valeurs**.
- De manière générale une **instruction** dans le terminal commence toujours par une commande
- Dans l'exemple ci-dessous on dira '**moins v**'.

```
# Exemple d'argument sans valeur associée  
# v pouvant signifier verbose, version (ou autre suivant la commande)  
fastqc -v  
  
# Exemple d'argument avec valeur associée  
man -k jpeg
```

Prototype(s) d'une commande (2)

- De manière générale, les arguments peuvent être utilisés sous leurs **formes courtes** ou sous leurs formes **longues** (plus **explicités/lisibles** mais plus longues à taper...).
- Les formes longues sont généralement précédées de deux tirets (dans l'exemple ci dessous on dira '**moins-moins apropos**)

```
# Exemple d'argument sans valeur associée  
fastqc --version
```

```
# Exemple d'argument avec valeur associée  
man --apropos jpeg
```

Trouver de l'aide !

Appeler son collègue ou mieux, chercher sur internet ou utiliser la commande **man** (manuel)

```
# Demo
```

```
root@vm: man ls      # obtenir de l'aide sur la commande ls
```

```
root@vm: man man    # obtenir de l'aide sur la commande man ...
```

Raccourcis dans l'aide:

/truc : pour chercher le terme 'truc'.

n : (**n**ext) pour chercher la prochaine occurrence de 'truc'.

p: (**p**revious) pour chercher l'occurrence précédente de 'truc'.

q : pour **q**uitter.



Notre première
commande: ls



La commande `ls` et ses arguments

- La commande `ls` peut prendre un certain nombre d'arguments.
- Parmi les arguments principaux:
 - `-l` : (**long**) donne beaucoup d'informations sur les fichiers.
 - `-a` (**all**) montre tous les fichiers y compris ceux qui sont cachés*.
 - `-1` : présente les fichiers sous la forme d'**une** colonne.
 - `-t` (**time**) trie par date de modification.
 - `-r` (**reverse**) inverse l'ordre du tri.
- On peut combiner les arguments
 - `ls -l -a`
 - `ls -la`

* Sous Linux les fichiers cachés commencent par un point (e.g '`.lefichier.txt`').

La commande ls et ses arguments

```
# Demo
root@vm: ls      # on liste les fichiers
root@vm: ls -a   # on liste les fichiers y compris ceux qui sont cachés *
root@vm: ls -l   # on veut beaucoup d'informations sur les fichiers
root@vm: ls -l   # on liste les fichiers sur une colonne
root@vm: ls -t   # on liste les fichiers par date de modification **

# En combinant les arguments
root@vm: ls -rtl # beaucoup d'informations, par date (du plus ancien au
plus récent)
```

* **ATTENTION** aux espaces. L'instruction commence par une commande. La commande `ls-a` n'existe pas !

** Le comportement par défaut est de trier par ordre alphabétique en tenant compte de la casse (i.e majuscule minuscule).

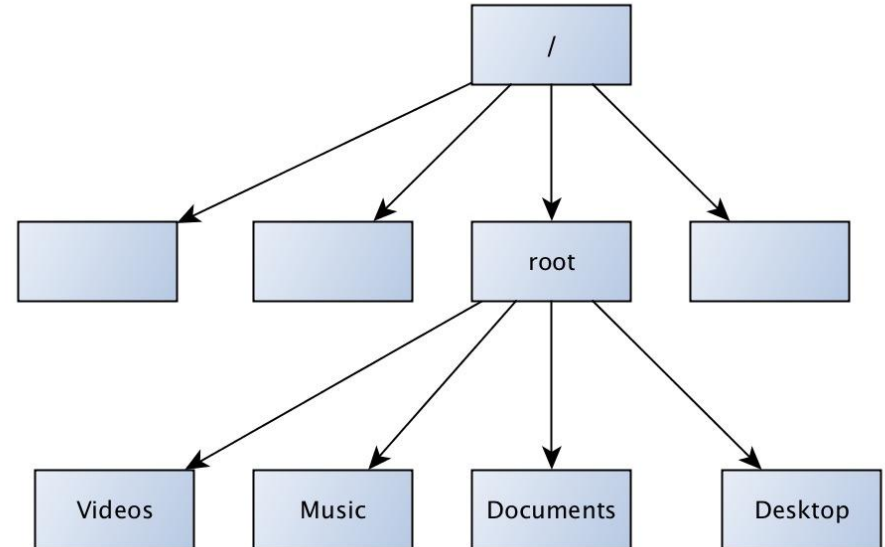


Créer des répertoires et se déplacer



L'arborescence du système de fichier

- Le système de fichier peut être vu comme un **arbre** dont les **feuilles** sont des dossiers et fichiers. On peut se **déplacer** dans cet arbre.
- Cet arbre contient une racine, le dossier **/**
- Le dossier **/** contient notamment
 - un dossier **root** *
 - qui lui même contient un dossier **Documents**



* Sur les VM de l'IFB, vous êtes administrateur de la machine (cet utilisateur est l'utilisateur appelé root dans le monde Unix).

Comment faire référence à un dossier ou fichier

- 1) En spécifiant **un chemin depuis la racine**. On parle de **chemin absolu**
e.g; /root/Documents /root/Music
- 2) En se référant au répertoire courant (celui dans lequel on se trouve). C'est le **chemin relatif**.

Syntaxe pour l'écriture relative

Le répertoire au dessus du répertoire courant ('point point')

..

Deux répertoires au dessus ('point point slash point point')

../..

trois répertoires au dessus

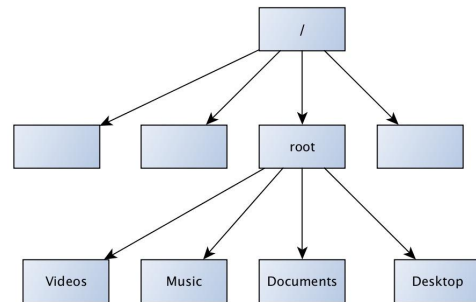
../../..

Le répertoire courant

./

L'arborescence: Demo

On utilise ci-dessous la commande **pwd** (print working directory) et la commande **cd** (change directory). *

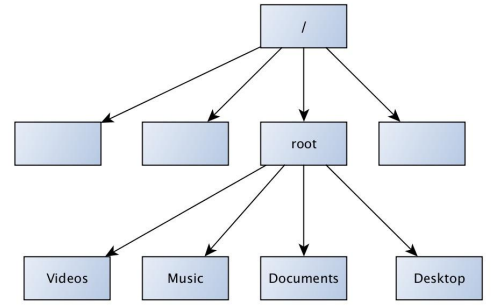


```
root@vm: pwd # Le répertoire courant (/root)
root@vm: cd /root/Documents # On se déplace dans Documents
root@vm: pwd # /root/Documents
root@vm: cd .. # on remonte d'un répertoire (/root)
root@vm: cd /root/Music # On se déplace dans le répertoire Music
root@vm: pwd # /root/Music
root@vm: cd ../.. # on est à la racine
root@vm: ls # Le dossier root est dans le répertoire courant
root@vm: cd /root/Music # On se déplace dans le répertoire root/Music
root@vm: cd ../Documents # On se déplace dans Documents
```

* Utilisez la **complétion** pour les noms les noms de fichier (**touche tab**) et éventuellement les noms de commandes

L'arborescence quelques astuces

- Vous êtes l'utilisateur root. Le dossier qui stocke vos **documents** est par défaut **/root**
 - i.e 'dossier utilisateur' ou dossier **home**.
- Plutôt que d'écrire /root vous pouvez écrire **~** (tilde).



```
root@vm: cd / # On est à la racine
root@vm: pwd # /
root@vm: cd ~/Documents # Le dossier Documents du dossier home.
root@vm: cd ~ # On se déplace dans son dossier home
root@vm: cd /usr/local/bin # On se déplace dans /usr/local/bin
root@vm: ls ~ # on liste les fichiers dans le répertoire 'home'
root@vm: cd ~/Music # On se déplace dans le dossier Music
root@vm: cd # équivalent de cd ~
```


Créer des répertoires

- On utilisera la commande **mkdir** (make directory).

```
root@vm: mkdir projet_roscoff      # On crée le dossier
root@vm: cd ./projet_roscoff       # équivalent de cd projet_roscoff *
root@vm: mkdir rna-seq             # on crée un dossier
root@vm: mkdir chip-seq dna-seq    # on crée plusieurs dossiers
root@vm: ls -l                    # trois dossiers
root@vm: cd chip-seq              # équivalent de cd ./chip-seq
root@vm: pwd                      # /root/projet_roscoff/chip-seq
root@vm: cd ../..                 # Retour à la maison
```

* L'utilisation de ./ est souvent facultative.

Exercices

- 1) Déplacer vous dans le répertoire `~/projet_roscoff/chip-seq`
- 2) Depuis ce répertoire créez un répertoire *annotations* dans le dossier `~/projet_roscoff/`
- Déplacez vous dans le répertoire `annotation`
- Revenez dans votre home.

Exercices

- 1) Déplacer vous dans le répertoire `~/projet_roscoff/chip-seq`
- 2) Depuis ce répertoire créez un répertoire *annotations* dans le dossier `~/projet_roscoff/`
- Déplacez vous dans le répertoire annotation
- Revenez dans votre home.

Solution

```
root@vm: cd ~/projet_roscoff/chip-seq
```

```
root@vm: mkdir ../annotations
```

```
root@vm: cd ../annotations
```

```
root@vm: cd
```



Manipuler des fichiers



Le fichier hg19_exons.bed

Contient les coordonnées (début/fin) des exons humains au format BED.

Le format bed (Bed6) (<http://genome.ucsc.edu/FAQ/FAQformat.html#format1>) *

Format tabulé (les colonnes sont séparées par des tabulations)

Chromosome Start End Name Score Strand (Others...)

chr17	19281773	19281943	NM_002749	0	+
chr17	19282208	19282445	NM_002749	0	+
chr17	19283094	19283260	NM_002749	0	+
chr17	19283920	19284999	NM_002749	0	+
chr17	19285093	19285779	NM_002749	0	+
chr17	19286125	19286259	NM_002749	0	+
chr17	19286390	19286857	NM_002749	0	+
chrX	75648045	75651746	NM_020932	0	+
chr1	155158299	155158685	NM_001204289	0	-
chr1	155159700	155159850	NM_001204289	0	-
chr1	155159930	155160052	NM_001204289	0	-
chr1	155160197	155160334	NM_001204289	0	-

* Positions Start et End sont toujours données par rapport au sens 5'/3' du brin +. Les coordonnées sont 'zero-based, half-open'.

Visualiser le contenu d'un fichier

- On utilisera **less** ou **more** (qui font + ou - la même chose *dixit* JvH) pour parcourir le fichier ligne à ligne (logiciels de type 'pager').
- On utilisera **head** ou **tail** pour voir les **n premières** ou **n dernières** lignes d'un fichier.
- La commande **cat** permet de renvoyer tout le contenu d'un fichier sur la sortie standard (l'écran). <ctrl> + c (cancel) pour arrêter.
- Les **raccourcis clavier dans less** sont les mêmes que pour la commande **man**.

Raccourcis dans less:

↑ : se déplacer vers le haut.

↓ : se déplacer vers le bas.

> : Aller à la première ligne.

< : Aller à la dernière ligne.

/truc : pour chercher le terme 'truc'.

n : (next) pour chercher la prochaine occurrence de 'truc'.

p: (previous) pour chercher l'occurrence précédente de 'truc'.

q : pour quitter.

Exercices

- 1) Utilisez la commande **head** pour regarder les 10 premières lignes du fichier `hg19_exons.bed`
- 2) Utilisez la commande **tail** pour regarder les 10 dernières lignes du fichier `hg19_exons.bed`
- 3) Promenez vous dans le fichier `hg19_exons.bed` en utilisant la commande **less**.
- 4) Renvoyer le contenu du fichier à l'écran avec **cat**.

Exercices

- 1) Utilisez la commande **head** pour regarder les 10 premières lignes du fichier `hg19_exons.bed`
- 2) Utilisez la commande **tail** pour regarder les 10 dernières lignes du fichier `hg19_exons.bed`
- 3) Promenez vous dans le fichier `hg19_exons.bed` en utilisant la commande **less**.
- 4) Renvoyer le contenu du fichier à l'écran avec **cat**.

Solution

```
root@vm: head -n 10 hg19_exons.bed
root@vm: tail -n 10 hg19_exons.bed
root@vm: less hg19_exons.bed
root@vm: cat hg19_exons.bed
```

Compter les lignes d'un fichier

Utiliser la commande **wc** (**w**ord **c**ount) avec l'argument **-l** (**l**ine).

```
root@vm: wc -l hg19_exons.bed # 484127 exons
```

Extraire des colonnes

- Pour extraire des colonnes on utilisera la commande **cut** avec l'argument **-f** (field)
- Les colonnes du fichiers doivent nécessairement être séparées par une **tabulation** (sinon utiliser l'argument **-d** pour 'delimiter')

```
root@vm: cut -f1 hg19_exons.bed           # extraire la colonne 1
root@vm: cut -f1,2 hg19_exons.bed         # extraire la colonne 1 et 2
root@vm: cut -f3-5 hg19_exons.bed         # extraire la colonne 3 jusqu'à 5
root@vm: cut -f3- hg19_exons.bed          # extraire la colonne 3 jusqu'à la fin de la ligne
```

Trier un fichier

- Il faut utiliser la commande **sort** (tri alphabétique par défaut).
 - **-k (key)**: e.g
 - **-k1,1** utiliser les caractères de la colonne 1 à 1 pour le tri.
 - **-k2,2nr** utiliser les caractères de la colonne 2 à 2 pour faire un tri **numérique** en inversant l'ordre (**reverse**).
 - **-k2,2g** pour effectuer, sur la colonne 2, un tri sur des valeurs décimales.

Exemple: Trier le fichier `hg19_exons.bed` par chromosomes (tri alphabétique) puis par coordonnées génomiques:

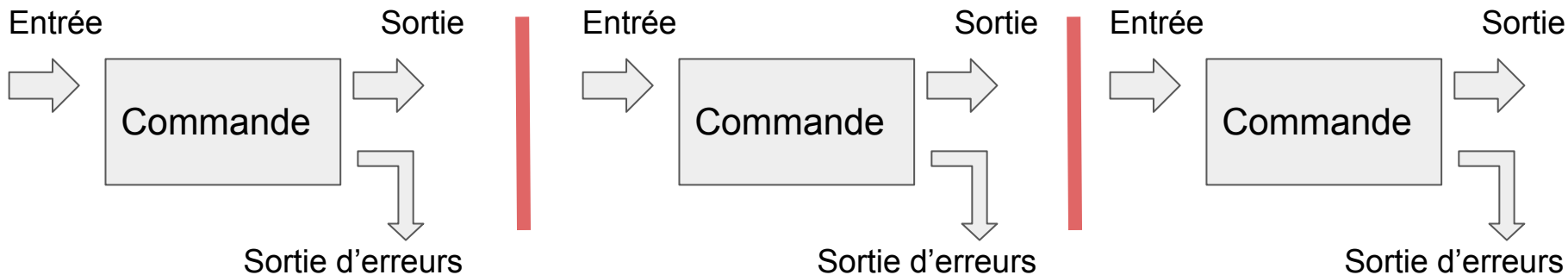
```
root@vm: sort -k1,1 -k2,2nr hg19_exons.bed # Trier les lignes par coordonnées
```



Redirections



Enchaînement de commandes



- Entrée: **un fichier** ou du texte (**un flux** de texte).
- Sortie standard : par défaut, **l'écran**.
- Sortie d'erreur: peut être capturée si nécessaire.



Demo: enchaînements de commandes

Obtenir la liste de chromosomes présents dans le fichier

```
root@vm: cut -f1 hg19_exons.bed | sort | uniq      # La liste non-redondante  
des chromosomes
```

Obtenir la liste des chromosomes présents dans le fichier et leur nombre

```
root@vm: sort hg19_exons.bed | uniq -c           # -c pour 'count'
```

Compter le nombre de transcript non codant (contenant 'NR_').

```
root@vm: cut -f4 hg19_exons.bed | grep "NR_" | sort | uniq | wc -l #11675
```

Note: La commande **uniq** permet d'éliminer les doublons dans un fichier **trié**.

Note: la commande **grep** permet de chercher une chaîne de caractères.

Exercices (notés)

- Combien y-a-t-il **d'exons sur le chromosome 22** ?
- Quel est le **tuple chrom-start-end le plus représentés** ?
 - i.e l'exon le plus représentée.

Exercices (notés)

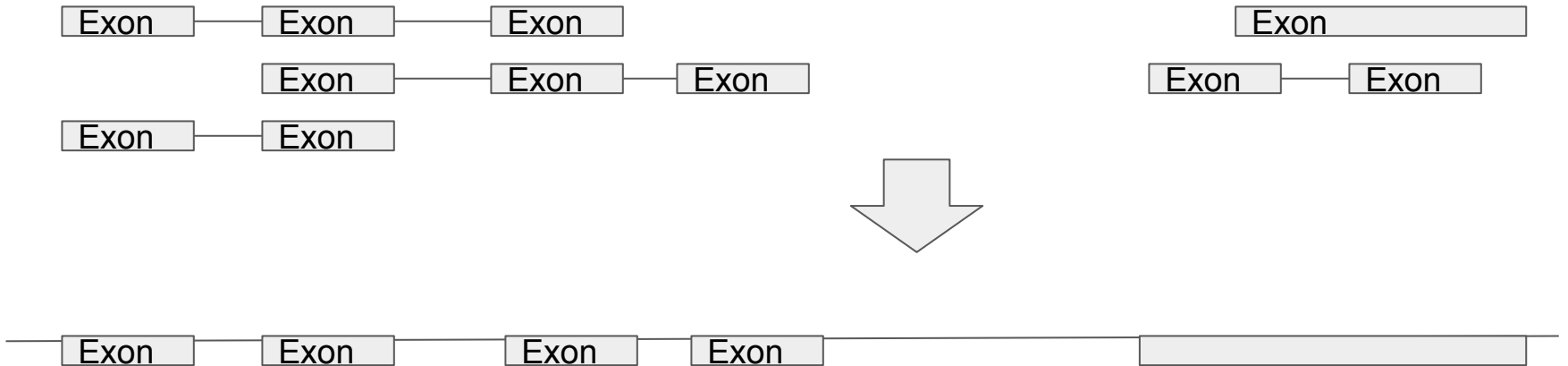
- Combien y-a-t-il **d'exons sur le chromosome 22** ?
- Quel est le **tuple chrom-start-end le plus représentés** ?
 - i.e l'exon le plus représentée.

Solution

```
root@vm: grep -w chr22 hg19_exons.bed | wc -l # n = 259
root@vm: cut -f1-3 hg19_exons.bed | sort | uniq -c | sort -n | tail -n 1 # 77 chrY
```

Exercice

- Quelle est la fraction du génome couverte par les exons ?
 - Comment réaliser l'opération ci-dessous.



- Nous allons répondre à cette questions dans les slides suivantes



Utiliser Bedtools



Bedtools

- Une suite d'outils pour effectuer des opérations arithmétiques sur des coordonnées génomiques.
 - <http://bedtools.readthedocs.org/en/latest/content/overview.html>
- Exemples d'utilisation:
 - Etendre des régions.
 - Comparer des régions.
 - Fusionner des régions.
 - Convertir des formats.
 - ...
- La commande **bedtools** est associée à un certain nombre de **sous-commandes**.



Exercice avec bedtools

- Utiliser la commande bedtools avec l'argument -h.
 - Qu'observez vous ?
- Demandez de l'aide sur la commande **merge** (**bedtools merge -h**)
 - Analysez les arguments.
 - Qu'indique la **note** à la fin de l'aide sur cette commande ?



Exercice avec bedtools

- Utiliser la commande bedtools avec l'argument -h.
 - Qu'observez vous ?
- Demandez de l'aide sur la commande **merge** (**bedtools merge -h**)
 - Analysez les arguments.
 - Qu'indique la **note** à la fin de l'aide sur cette commande ?

Solution

```
root@vm: bedtools -h          # l'ensemble des sous commandes
```

```
root@vm: bedtools merge -h    # utiliser l'argument -i
```

```
# la note indique que les régions génomiques doivent être triées au préalable.
```



Exercice

- Utilisez bedtools sort et bedtools merge pour fusionner les regions génomiques chevauchantes.



Exercice

- Utilisez bedtools pour fusionner les regions génomiques chevauchantes (merge).

```
root@vm: bedtools sort -i hg19_exons.bed | bedtools merge
```


Comment sauvegarder le résultat dans un fichier ?

- Utilisez l'opérateur de re-direction `>`.
 - Ecrase le fichier si il existe.
- Notez que `>>` permet d'ajouter des lignes dans un fichier existant.

```
root@vm: bedtools sort -i hg19_exons.bed | bedtools merge >  
hg19_exons_merged.bed  
root@vm: ls # Un nouveau fichier est apparu.
```





Un peu d'arithmétique avec awk



Awk

- Awk est une commande disponible sur la plupart des systèmes Unix.
- Awk est une commande qui dispose de son propre langage (awk).
- Awk permet de réaliser des traitements ligne à ligne
- Le prototype d'une commande awk est le suivant:

```
awk 'BEGIN{ } END{ }' fichier
```

- Chaque jeu d'accroches a un rôle particulier:

```
BEGIN{quelque chose à faire avant d'ouvrir le fichier}
```

```
{quelque chose à faire sur chacune des lignes}
```

```
END{quelque chose à faire après avoir lu les lignes}
```

Awk

- Awk dispose d'un certain nombre de variables qui lui sont propre (on parle de **variables spéciales**)
- Exemple de variables spéciales

FS: Field Separator. Le séparateur de colonne (e.g tabulation)

OFS: Output Field Separator. Le séparateur de colonnes utilisé en sortie.

NR: Number of Row. Le numéro de la ligne courante.

NF: Number of **Field**. Le numéro de colonne.

\$0: le contenu de la ligne courante

\$1,\$2,\$3 (...): le contenu de la colonne 1,2 ou 3 (...) de la ligne courante

Exemple

Imprimer la colonne 2 et 1

\t correspond au caractère tabulation

```
root@vm: awk 'BEGIN{FS="\t"}{print $2,$1}' hg19_exons.bed
```

Imprimer la colonne 2 et 1 avec une tabulation (\t) en sortie

```
root@vm: awk 'BEGIN{FS=OFS="\t"}{print $2,$1}' hg19_exons.bed
```

Imprimer la colonne 2 et 1 et le numéro de ligne

```
root@vm: awk 'BEGIN{FS=OFS="\t"}{print NR,$2,$1}' hg19_exons.bed
```

Calculer start - end pour chaque ligne

```
root@vm: awk 'BEGIN{FS=OFS="\t"}{print $3-$2}' hg19_exons.bed
```

Exemple

```
# Calculer à chaque ligne la somme cumulée de la taille des fragments
# Notez que les ";" permettent de séparer des instructions
# s est une variable que l'on déclare à 0
# 75861726
root@vm: awk 'BEGIN{FS="\t"; s=0}{s=s+$3-$2; print s}' hg19_exons_merged.bed

# Ou encore
awk 'BEGIN{FS="\t"; s=0}{s=s+$3-$2}END{print s}' hg19_exons_merged.bed

# A vos calculettes (vous pouvez utiliser R).
# 75861726/3.2e9*100
# ~ 2.37 % du génome couvert
```



Merci

