Montpellier Omics Days

Workshop : Introduction à Linux et Bash

1er Février 2024

L'objectif de cet atelier est de fournir des informations sur les commandes les plus essentielles à exécuter dans un terminal pour une gestion efficace d'un ordinateur sous système Linux. De plus, il vise à enseigner la création de scripts système en utilisant le langage bash et intégrant ces commandes.

1 Manipulation du terminal

Lancez un terminal :

- En cliquant sur l'icône du terminal : 
- En cliquant sur l'item "terminal" dans un sous-menu "système" dans le menu principal de votre bureau.

Lors de la création d'un terminal, il y a la création d'un processus exécutant un interpréteur de commandes (ou shell). L'interpréteur de commandes est le programme qui va "comprendre" vos commandes. Il existe plusieurs implémentations possibles d'interpréteurs de commandes, les deux les plus fréquentes étant bash et csh. Pour connaître l'interpréteur de commandes que vous utilisez par défaut, vous effectuerez la commande suivante dans votre terminal : echo $SHELL.

Nous allons donc utiliser des commandes bash sur notre ordinateur aujourd'hui.

2 Premières commandes : gestion de fichiers et de répertoires

2.1 Éléments de base

Structure de base d'une commande linux : `commande [ -options ] [ arguments ]`

Pour lire la documentation relative à une commande (affiche les pages du manuel de la commande donnée en argument) : `man [commande]`

Raccourcis clavier utiles : complétion avec touche Tab →, annulation avec `ctrl+C`, copier-coller dans terminal `ctrl+shift+ C` (copier) ou `V` (coller)

2.2 Commandes utiles

Commande	Fonction	Utilisation
pwd	Afficher le nom et le chemin du répertoire courant	pwd
date	Affiche la date et l'heure	date
ls	Afficher le contenu d'un répertoire	ls <répertoire>
cd	Se déplacer dans le répertoire donné	cd <répertoire>
mkdir	Créer un répertoire	mkdir <répertoire>
rmdir	Supprimer un répertoire	rmdir <répertoire>
touch	Créer un fichier	touch <fichier>
rm	Supprimer un fichier	rm <fichier>
mv	Déplacer un fichier	mv <fichier> <destination>
cp	Copier un fichier	cp <fichier> <destination>

2.3 Exercice 1

Utilisez les commandes Bash pour accomplir diverses tâches liées à la gestion de fichiers et de répertoires.

1. Accéder au répertoire de départ : Vous vous trouvez actuellement dans un répertoire inconnu. Utilisez une commande pour afficher le chemin complet de ce répertoire.

2. Créer un fichier texte : À l'intérieur de ce répertoire, créez un fichier texte nommé "**mon_fichier.txt**".

3. Créer un répertoire : Créez un nouveau répertoire portant le nom **Workshop** à l'intérieur du répertoire actuel.

4. Copier un fichier texte : Copiez le fichier "**mon_fichier.txt**" à l'intérieur du répertoire que vous venez de créer.

5. Explorer le contenu du répertoire : Déplacez vous dans le répertoire **Workshop** et affichez son contenu.

6. Renommer un fichier : Renommez le fichier "mon_fichier.txt" en "**nouveau_fichier.txt**".

7. Déplacer un fichier : Déplacez "**nouveau_fichier.txt**" dans le répertoire parent (le répertoire où vous vous trouviez initialement).

8. Supprimer un fichier : Revenez dans le répertoire de départ (étape 1) et supprimez le fichier "**nouveau_fichier.txt**" que vous avez créé.

3 Manipulation de fichiers

3.1 Commandes utiles

Commande	Fonction	Utilisation
cat	Concaténer le contenu de plusieurs fichier	cat <fichier1> <fichier2>
less	Afficher le contenu d'un fichier page par page	less <fichier> (q pour quitter)
head	Afficher les n premières lignes d'un fichier dans le terminal	head -n <fichier>
tail	Afficher les n dernières lignes d'un fichier dans le terminal	tail -n <fichier>
wget	Télécharger le contenu de l'adresse web donnée	wget <url>
grep	Chercher l'expression donnée dans un fichier	grep "expression" <fichier>
wc	Afficher le nombre de lignes, mots et octets dans un fichier	wc <fichier>
cut	Supprimer des parties de chaque ligne d'un fichier	cut -f1,2 <fichier>
gzip	Compresser un fichier	gzip <fichier>
gunzip	Décompresser un fichier	gunzip <fichier>
zcat	Afficher le contenu d'un fichier compressé dans le terminal	zcat <fichier>
zgrep	Chercher l'expression donnée dans un fichier compressé	zgrep expression <fichier>

3.2 Exercice 2

Utiliser les commandes Linux pour effectuer différentes opérations de manipulation de fichiers et de texte.

1. Accéder au répertoire de départ : Déplacez vous dans le répertoire **Workshop**.

2. Téléchargement d'un fichier : Utilisez une commande pour télécharger le fichier disponible à l'adresse suivante : `https://raw.githubusercontent.com/FGoutieras/data_mods/main/sequence.fasta.gz`

3. Décompresser un fichier : Utilisez une commande pour décompresser le fichier téléchargé.

4. Afficher le contenu d'un fichier : Utilisez une commande pour afficher le contenu du fichier "**sequence.fasta**" à l'écran.

5. Lire un fichier page par page : Utilisez une commande pour afficher le contenu du fichier "**sequence.fasta**" à l'écran.

6. Afficher les premières lignes d'un fichier : Affichez les 5 premières lignes du fichier "**sequence.fasta**" à l'écran.

7. Afficher les dernières lignes d'un fichier : Affichez les 5 dernières lignes du fichier "**sequence.fasta**" à l'écran.

8. Rechercher un mot dans le fichier : Utilisez une commande pour rechercher les occurrences de la lettre "A" dans le fichier "**sequence.fasta**".

9. Compter les lignes dans un fichier : Utilisez une commande pour compter le nombre de lignes dans le fichier "**sequence.fasta**".

10. Extraire une colonne de données : Téléchargez le fichier "**donnees.csv**" à partir de l'adresse suivante : `https://raw.githubusercontent.com/FGoutieras/data_mods/main/donnees.csv` Affichez le dans votre terminal pour constater qu'il contient plusieurs colonnes de données séparées par des tabulations. Utilisez une commande pour extraire la troisième colonne de données.

4 Sauvegarde de la sortie d'une commande dans un fichier

4.1 Rediriger la sortie d'une commande vers un fichier

Le chevron (>) permet de rediriger la sortie d'une commande vers un fichier au lieu de l'afficher dans le terminal. Si le fichier n'existe pas, il sera créé et si le fichier existe, cela efface le contenu et remplace par la sortie de la commande.
Utilisation : `Commande > Fichier`

Le double chevron (>>) permet d'écrire le contenu de la sortie à la fin d'un fichier existant, sans effacer le contenu du fichier.
Utilisation : `Commande >> Fichier`

4.2 Enchaîner plusieurs commandes

La commande pipe "|" est utilisée pour rediriger la sortie d'une commande vers l'entrée d'une autre commande. Cela permet de chaîner plusieurs commandes ensemble.
Utilisation : `Commande_1 | Commande_2`

4.3 Exercice 3

Utilisez **une seule** ligne de commande pour :

- afficher le fichier "**sequence.fasta**" avec cat
- filtrer les lignes contenant ">" avec grep
- afficher les 5 premières lignes des résultats avec head
- écrire la sortie de cette commande dans un fichier "**sortie.txt**"

5 Scripts en bash

Maintenant que vous maîtrisez les commandes de base, il est temps d'explorer une puissante fonctionnalité de Linux : les scripts. Un script est une série d'instructions enregistrées dans un fichier sous la forme de commandes. Cela permet de créer des programmes simples d'utilisation et facilement réutilisables afin d'automatiser les tâches répétitives. Les commandes seront ici écrites en bash, dans un fichier comportant l'extension .sh.

5.1 Premier script

Pour cela, il faudra suivre les étapes suivantes (utilisez les commandes apprises précédemment pour les mémoriser !) :

1. Créez un fichier dont le nom finit par ".sh". En linux, préciser l'extension n'est pas absolument nécessaire mais reste une bonne pratique pour organiser ses fichiers.

2. Ouvrez-le dans un éditeur de texte. Si vous n'avez pas de préférence, utilisez emacs : `emacs <script.sh>`

3. Sur la 1ère ligne, écrivez le shebang : `"\#!\bin\bash"`. Cette ligne permet de définir le programme qui va interpréter le script, ce qui est utile si le script est amené à être lancé depuis un terminal utilisant un autre langage que bash.

4. Sur les lignes suivantes, écrivez le script. Pour ce premier exemple, nous allons nous faire un programme qui donne la date :

   ```
   echo "Bonjour, la date du jour est:"
   date
   ```

   La commande echo permet d'afficher du texte dans le terminal.

5. Enregistrez le script (File -> Save)

6. Donnez le droit d'exécuter le script : `chmod +x <script.sh>`. Par défaut, Linux ne laisse pas à l'utilisateur faire ce qu'il veut avec tous les fichiers afin d'éviter les erreurs de manipulation.

7. Exécutez le script : `./<script.sh>`

5.2 Pour aller plus loin : script avancé

L'objectif de cette partie est l'écriture d'un script bash interactif qui demande à l'utilisateur d'entrer un nombre et qui affiche si le nombre est positif, négatif ou nul. Cet exercice fait appel a des structures conditionnelles. Sous bash et comme dans la quasi totalité des langages de programmation, le mot réservé **if** introduit une expression conditionnelle dans une structure de programmation qui permet de conditionner l'exécution d'un bloc d'instructions.
Structure conditionnelle dans un script :

```
if (<expression conditionnelle>)
then
    <bloc d'instructions>
else
    <bloc d'instructions>
fi
```

Cet exercice utilise également la récupération d'une valeur au clavier. Il faut utiliser la commande **read** tel que :

```
read <entree>
```

Voici les instructions pour réaliser le script :

1. Utilisez un éditeur de texte pour créer un fichier appelé "detecter_nombre.sh".

2. Commencez par écrire un script qui demande à l'utilisateur d'entrer un nombre avec la commande `echo`. Puis il faut indiquer au programme de récupérer cette valeur avec la commande `read nombre`.

3. Ensuite, utilisez des structures conditionnelles pour vérifier si le nombre est positif, négatif ou nul. Par exemple pour vérifier que le nombre est positif la condition est :
   `if (($nombre > 0))`

4. Ajoutez le bloc d'instruction pour chaque conditions de telle sorte que cela affiche sur le terminal l'information sur le nombre. Cela s'effectue avec la commande suivante pour le nombre positif : `echo "$nombre est un nombre positif."`.

5. Rendez le script exécutable et testez-le en entrant différents nombres. Assurez-vous que le script fournit la sortie correcte en fonction du nombre entré.