CPE Lyon - 4ETI - Année 2022/2023

Introduction à l'Administration Système Linux

TP 1 - Bash : rappels

Préparation de l'environnement de TP

Pour ces trois TP, certaines des commandes nécessitent des droits d'administrateur, auxquels vous n'avez pas accès sur les postes de l'école. C'est pourquoi vous utiliserez une machine virtuelle VirtualBox, au format `.ova` (disponible dans le dossier `/sync/VMs` sur les postes Linux, ou directement à partir de VirtualBox sur les postes Windows). Ce fichier est plus exactement une *archive*, qu'il faut importer dans VirtualBox (Menu **`Fichier > Importer un appareil virtuel...`**)

Important !

Un utilisateur du nom de **`tp`** a été créé sur cette machine, et son mot de passe est également **`tp`**. Nous verrons dans le TP 3 comment créer et gérer d'autres utilisateurs.

Rappels de 3ème année

Bash désigne :

- un *interpréteur de commandes* (cf. ce TP),
- un *langage de scripts* reconnu par cet interpréteur (cf. TP 2).

Les commandes reconnues par Bash peuvent être de différente nature :

- quelques commandes sont *natives* à Bash, c'est-à-dire que ce sont des commandes *fournies par Bash lui-même* (on parle aussi de *commandes internes*). Ce sont par exemple **`cd, pwd, echo, alias, if...`**
- des commandes *externes* : ce sont tous les programmes installés sur la machine pouvant être invoqués *via* la ligne de commande (**`cat, ping, ip, vim...`** et même **`bash`** lui-même !).

A retenir !

La liste des commandes *internes* est disponible avec la commande **`help`**, qui permet également d'obtenir de l'aide sur l'une de ces commandes (**et uniquement sur les commandes internes**), en saisissant **`help`** ***`commande`*** (par exemple **`help cd`**).

Les commandes *externes*, quant à elles, sont généralement fournies avec une documentation complète sous forme d'une *page de manuel*, à laquelle on accède en tapant **`man`** ***`commande`*** (par exemple **`man ls`**). Très souvent, une aide succincte est également disponible en tapant ***`commande`*** **`--help`**.

Attention ! les commandes internes n'ont pas de page de manuel, comme l'indique le message obtenu en tapant (par exemple) **`man cd`**.

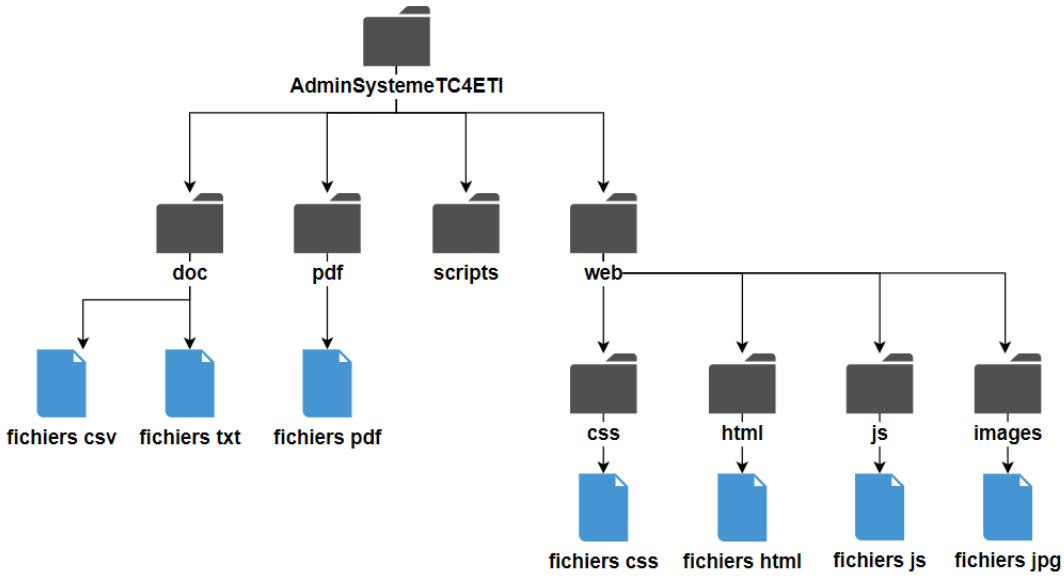
A retenir (aussi) !

Les commandes externes peuvent être de deux natures :

- des ***scripts***, c'est-à-dire des suites de commandes dans un langage spécifié (pas nécessairement Bash : ce peut être Python, Perl...), modifiables dans un éditeur de texte, et qui nécessitent un *interpréteur* ;
- des ***binaires***, c'est-à-dire des programmes *compilés* (écrits en C, C++...), et *exécutables directement par le processeur* (et donc beaucoup plus rapides).

Exercice 1. Navigation dans l'arborescence de fichiers

1. A quoi correspond le symbole `~` dans le chemin `~/admin/tp/tp1.txt` ?
2. Parmi les propositions suivantes, lesquelles désignent des chemins *absolus* ? Comment qualifie-t-on les autres chemins ?
   - `/usr/bin`
   - `../`
   - `/etc/conf`
   - `~/2021/notes`
   - `/bin/../var`
   - `./dossier`
3. Utilisez la commande `git clone` (vue en 3ème année) pour rappatrier le dépôt suivant dans votre répertoire personnel : https://github.com/GregMorel/AdminSystemeTC4ETI.
4. A l'aide des commandes `cd`, `rm`, `mkdir`, `mv`, `cp`... modifiez l'arborescence téléchargée pour obtenir celle-ci :



   Autrement dit, le dossier `site` doit être renommé en `web` et ses fichiers répartis dans les dossiers indiqués. Le dossier `scripts` doit être créé. Le dossier `docs` doit être renommé `doc` et les fichiers `pdf` qu'il contient doivent être déplacés dans un nouveau dossier `pdf`.
5. Déplacez vous dans le dossier `pdf` de l'arborescence créée, puis tapez simplement `cd` ? Que se passe-t-il ?
6. Saisissez à présent la commande `cd -`. Que fait-elle ?
7. Quelle commande donne le chemin complet du dossier courant ?
8. Essayez de vous déplacer dans le dossier `/root` (attention, il y a un bien un `/` au début !). Que se passe-t-il et pourquoi ?
9. Revenez dans votre dossier personnel, puis tapez successivement les commandes `ls` puis `la` ; que peut-on en déduire sur les fichiers commençant par un point ?
10. Essayez la commande `ll`. Existe-t-il une entrée de manuel pour cette commande ? Utilisez la commande `alias` pour en savoir plus sur la nature de `ll`.
11. Essayez la commande `du` (pour *disk usage*), qui permet d'obtenir la taille de tous les dossiers présents dans le chemin courant. On peut lui demander de faire une synthèse en spécifiant un niveau de profondeur maximal avec l'option `-d`. Par exemple `du -d1 -h` affiche la taille de chaque sous-dossier, dans un format lisible par un humain (option `-h`).

> **Important !**
>
> C'est très utile pour identifier les dossiers volumineux quand on dépasse son quota disque !

Exercice 2. Obtenir un peu d'aide : le Manuel Linux

1. Quand on ignore le rôle d'une commande, le plus simple est de demander à **`whatis`**. Utilisez-la pour déterminer ce que fait la commande **`head`**.
2. Pour avoir un aperçu rapide des arguments et options que peut prendre une commande, on la suffixe (généralement) avec **`--help`**. Essayez avec la commande **`head`**.
3. Pour avoir la documentation exhaustive d'une commande, on se tournera vers le manuel. Ouvrez la page du manuel de la commande **`head`**.
4. On peut chercher un terme dans une page de manuel ouverte en tapant **`/terme`**. Cherchez les occurrences du mot **`file`** dans la page de manuel de **`head`**.

   > A savoir !
   >
   > Toutes les occurrences trouvées dans la page sont surlignées. On peut passer à la suivante en appuyant avec **`n`** et à la précédente avec **`N`**. Les autres commandes peuvent être trouvées avec **`h`**.

5. Expliquez ce que vous obtenez en tapant **`man -k count words`** (ou son équivalent **`apropos count words`**).
6. Le manuel est divisé en *sections*. A quoi correspondent-elles ? Quelle est la différence entre les commandes **`man passwd`** et **`man 5 passwd`** ?
7. Comment affiche-t-on la *page d'introduction* de la section 6 ? De quoi traite cette section ?
8. Que fait la commande **`man man`** ?
9. La commande **`find`** permet de rechercher des fichiers (par nom, par taille, par date de modification, etc.). Utilisez le manuel pour déterminer les options vous permettant de rechercher tous les fichiers situés dans **`/usr`** et dont le nom contient **`passwd`**. Recherchez ensuite tous les fichiers situés dans votre dossier personnel et pesant plus d'1 kio (= 1024 octets).

   > A savoir !
   >
   > La commande **`find`** parcourt le disque à la recherche des fichiers demandés, ce qui peut être très long. La commande **`locate`** (qu'il faut installer manuellement) permet de trouver des fichiers presque instantanément, grâce à une base de données de fichiers indexés.

Exercice 3. Traitement de fichiers texte

> Pipelines de commande
>
> Dans cet exercice, vous allez découvrir toute la puissance et la flexibilité des outils en ligne de commande. Certains argumenteront qu'il est plus simple et plus rapide de cliquer sur une icône en forme de loupe pour chercher un mot dans une page que de taper une commande à la syntaxe ésotérique dans un terminal. Mais les très nombreuses options des commandes permettent de réaliser des tâches bien plus complexes que ce qu'il est possible de réaliser avec une interface graphique. Et surtout, il est possible d'intégrer les commandes dans un script pour *automatiser* des tâches.
>
> Mieux encore, en *combinant* plusieurs de ces commandes les unes à la suite des autres à l'aide de la syntaxe **`commande1 | commande2`**, on crée un *pipeline* (d'ailleurs, le caractère **`|`** se prononce *pipe*, et symbolise le fait qu'on crée un "tuyau de connexion" permettant de relier la sortie de **`commande1`** et l'entrée de **`commande2`**). Et on peut même réaliser des pipelines de 3, 4, ou 10 commandes !

Lire un fichier

1. A l'aide de la commande **`cat`**, affichez le contenu du fichier **`small.txt`** situé dans le dossier **`doc`** (cf. Exercice 1). Quelle option de **`cat`** permet de préfixer chaque ligne par un numéro ?

   > A savoir !
   >
   > La commande **`nl`** (pour *Number Lines*) permet d'effectuer la même chose, mais avec beaucoup plus de possibilités, et ne numérote pas les lignes vides par défaut.

2. Affichez à présent le fichier **`small.txt`** à l'aide de la commande **`tac`**. Que constatez-vous ?

   > A savoir !
   >
   > Comme leur nom l'indique, **`tac`** fait l'inverse de **`cat`**...

3. Tapez la commande **`wc small.txt`**. Qu'affiche-t-elle ?
4. Affichez à présent le fichier **`big.txt`**. On constate que celui-ci est trop long pour tenir à l'écran. Pour mettre en pause l'affichage et permettre un défilement page par page, on peut utiliser la commande **`more`**, ou sa version plus moderne et plus complète, **`less`** (oui, les développeurs adorent les jeux de mots...).

   > A savoir !
   >
   > **`more`** est un outil historique assez rudimentaire ; on passe à la ligne suivante avec la touche **`<Entrée>`** et à la page suivante avec **`<Espace>`**. **`less`** est plus complet, et permet d'utiliser les touches du clavier, de revenir en arrière...

5. Donnez la commande permettant d'afficher les 5 premières lignes du fichier **`big.txt`**.
6. Donnez la commande permettant d'afficher les 8 dernières lignes du fichier **`big.txt`**.
7. A l'aide d'un pipeline, affichez les lignes 10 à 20 (**incluses**) du fichier **`big.txt`**, en numérotant les lignes du fichier d'origine.
8. Complétez la commande de la question précédente pour déterminer le nombre de mots dans le résultat.

Chercher dans un fichier

9. La commande **`grep`** est un outil précieux pour tous ceux qui utilisent la ligne de commandes. Elle permet de rechercher un mot dans un fichier. Recherchez toutes les occurrences du mot **`Harry`** dans le fichier **`big.txt`**. Consultez le manuel pour afficher le numéro de chaque ligne du résultat.
10. En réalité, la commande **`grep`** permet d'effectuer des recherches beaucoup plus poussées que la simple recherche de mots. En particulier, elle permet de spécifier un *motif* (appelé ***expression rationnelle*** (*regular expression* ou *regex* en anglais)) que doivent vérifier les résultats (voir encadré page suivante).

`grep` comprend deux syntaxes : les regex *basiques* (syntaxe par défaut) et les regex **`étendues`** (avec l'option **`-E`**, ou en utilisant **`egrep`**). Dans la version "GNU" de **`grep`**, les deux syntaxes ont la même puissance ; mais avec la syntaxe basique les caractères **`., +, ?, *...`** sont interprétés comme des caractères ordinaires, et il faut les préfixer avec un
pour obtenir leur signification spéciale.

Expression rationnelles

Une expression rationnelle est une chaîne de caractères qui décrit, selon une syntaxe précise, un ensemble de chaînes possibles. Par exemple, on peut désigner :

- `^debut` : toutes les lignes commençant par `debut`
- `fin$` : toutes les lignes se terminant par `fin`
- `^[AEIOU]` : toutes les lignes commençant par une majuscule voyelle
- `[a-z]` : n'importe quelle lettre minuscule
- `[1-5]` : n'importe quel chiffre entre 1 et 5
- `.` : n'importe quel caractère
- `a+` : le caractère `a` *au moins une fois*
- `b?` : éventuellement le caractère `b` une fois
- `c*` : le caractère `c` *zero, une, ou plusieurs fois*
- `[^bf]` : n'importe quel caractère autre que `b` ou `f`
- `[[:alpha:]]` : n'importe quelle lettre (majuscule ou minuscule)
- `[[:lower:]]` : n'importe quelle lettre minuscule
- `[[:upper:]]` : n'importe quelle lettre majuscule
- `[[:alnum:]]` : n'importe quel caractère alphanumérique

Affichez toutes les lignes de dialogues (ce sont celles commençant par un tiret `-`), puis toutes les lignes se terminant par un point (attention, vérifiez bien que le résultat est celui attendu !).

11. Donnez une commande permettant d'obtenir **uniquement** et **exactement** les mots du fichier `big.txt` contenant au moins deux `a`.
12. La commande `sed` est un autre outil fondamental sous Linux. Elle permet de supprimer des lignes selon un numéro ou une expression rationnelle, ou encore de réaliser des substitutions. Par exemple, la commande `sed 's/old/new/g' fichier` affiche le contenu de `fichier` en remplaçant toutes les occurrences de `old` par `new`. Affichez le fichier `small.txt` en remplaçant toutes les occurrences de Jordan par Harry. Recommencez sans le `g` final présent dans la chaîne de substitution. Que constatez-vous à la ligne 9 ?

A savoir !

Dans la chaîne `'s/old/new/g'`, `s` signifie *substituer* et `g` signifie *de manière globale*, c'est-à-dire toutes les occurrences si le terme recherché apparaît plusieurs fois sur une ligne. Sans ce `g`, seule la première occurrence est remplacée.

Travailler avec des fichiers tabulaires

Les fichiers tabulaires sont des fichiers contenant des données sous forme de tableau (une liste de passagers d'un avion, un annuaire téléphonique, une liste de commandes passées sur un site de vente en ligne, etc.). Naturellement, on connaît tous les fichiers Excel, mais pour représenter des données tabulaires dans un simple fichier texte, on a besoin de spécifier un *délimiteur* (le plus souvent, un simple caractère) entre deux colonnes. L'utilisation fréquente de la virgule comme délimiteur a conduit à nommer de tels fichiers `CSV` (pour *comma separated values*, mais on peut aussi utiliser les caractères `:`, `;`, des espaces ou des tabulations.

13. Dans le dossier `doc` de l'arborescence du TP, vous disposez d'un fichier nommé `ventes.csv`. Utilisez les commandes vues précédemments pour déterminer le nombre de lignes et de colonnes qu'il contient. Quel est le délimiteur utilisé ?
14. Sous cette forme, le fichier `ventes.csv` n'est pas très lisible. Utilisez la commande `column` (et les options adéquates) pour afficher le fichier sous forme de tableau.
15. Le résultat obtenu n'est pas encore réellement satisfaisant : les lignes trop longues sont renvoyées à la ligne suivante. Combinez la commande de la question précédente avec `less -S`, et vous devriez obtenir désormais un vrai tableau.

16. Nous souhaitons à présent extraire seulement certaines colonnes du fichier. La commande **`cut`** permet de faire cela, en spécifiant avec l'option **`-f`** les numéros des colonnes à conserver. Essayez de n'afficher que les informations suivantes : numéro de commande, nom du client et pays de résidence.
17. On constate que dans le fichier, les commandes ne sont pas classées par numéro croissant. Heureusement, la commande **`sort`** existe ! Triez les commandes par numéro croissant.
18. Triez le fichier **`ventes.csv`** par pays d'expédition *décroissant*
19. A l'aide de la commande **`grep`**, affichez seulement les commandes qui n'ont pas été expédiées (la colonne **`STATUS`** ne contient pas **`Shipped`**). Comment les dénombrer sans les compter à la main ?
20. La commande **`uniq`** permet de supprimer les répétitions de lignes **consécutives**. Utilisez-la pour afficher la liste des clients. Dans un second temps, affichez également pour chacun d'eux le nombre de commandes passées.

Exercice 4. Redirections de flux d'entrées / sorties

Flux d'entrées / sorties

Toute commande possède un *flux d'entrée* (là où elle prend ses données d'entrée), un *flux de sortie* (là où elle écrit son résultat) et un *flux d'erreur* (là où elle écrit les éventuels messages d'erreur).

Par défaut, le flux d'entrée correspond au clavier, et les flux de sortie et d'erreur correspondent à l'écran. Mais il est possible, et très simple, de *rediriger ces flux*, par exemple de ou vers un fichier, ou même de ou vers une autre commande (c'est en fait ce qu'on a déjà fait avec les pipelines).

Rappel de quelques syntaxes (cf. cours) :

- **`commande < fichier`** : **`commande`** lit ses données depuis **`fichier`** au lieu de de les lire au clavier
- **`commande > fichier`** : la sortie de **`commande`** est écrite dans **`fichier`** (Attention ! Si **`fichier`** existe déjà, il est supprimé sans avertissement !)
- **`commande >> fichier`** : la sortie de **`commande`** est écrite **`à la fin`** de **`fichier`**
- **`commande 2> fichier`** : les messages d'erreur de **`commande`**sont écrits dans **`fichier`** (la syntaxe **`2>>`** existe aussi, et fonctionne comme décrit précédemment).

1. Extrayez du fichier **`ventes.csv`** les données sur les clients (trois dernières colonnes) et enregistrez le résultat dans le fichier **`clients.csv`**
2. A l'aide de la commande **`find`** vue précédemment, recherchez tous les fichiers nommés **`passwd`** présents sur la machine.
3. Vous avez dû obtenir un certain nombre de messages d'erreurs liés au fait que vous n'avez pas le droit d'explorer certains dossiers réservés à l'administrateur. Modifiez la commande précédente pour rediriger les messages d'erreur dans un fichier nommé **`erreurs.txt`**
4. Modifiez la commande précédente pour rediriger aussi la sortie standard, dans un fichier nommé **`passwd-files.txt`**
5. La commande précédente a l'inconvénient qu'on ne voit plus les résultats à l'écran. Utilisez la commande **`tee`** (cf. cours) pour afficher les résultats **à la fois** à l'écran et dans le fichier **`passwd-files.txt`**.

Exercice 5. Quelques autres commandes à connaître

Il existe de nombreuses autres commandes utiles à un administrateur système, et il est bien entendu impossible de les découvrir toutes dans un TP de 4h. Néanmoins, les commandes suivantes sont à retenir :

1. **`htop`** : c'est l'équivalent du Gestionnaire des tâches de Windows. On y trouve des informations sur l'activité des processeurs, la mémoire disponible, les programmes en cours d'exécution, etc.
2. **`date`** : affiche la date et l'heure. **A quoi sert la commande `time` ?**

3. `file` : permet de connaître le type d'un fichier (fichier texte, fichier binaire, dossier, etc.)
4. `curl` : elle permet de télécharger une ressource sur un réseau informatique (par exemple, une page web sur Internet) : très utile pour réaliser du *web scraping* ou automatiser des téléchargements ! **Essayez de télécharger la page d'accueil du site `www.example.org`.**

Exercice 6. Découverte de l'éditeur de texte nano

Nano est un éditeur de texte rudimentaire, où toutes les commandes évidemment se font au clavier.

Les raccourcis clavier les plus courants sont affichés en bas de l'écran, mais sous une forme peut-être inhabituelle :

- ^G signifie **Ctrl + G**
- M-U signifie **Alt + U**

Quelques raccourcis utiles :

F1 ou Ctrl + G	Affichage de l'aide
Ctrl + X	Quitter nano / Fermer une fenêtre / Exécuter une commande
Ctrl + R	Ouvrir un fichier
Ctrl + O	Enregistrer sous
Ctrl + S	Enregistrer
Ctrl + K	Couper
Ctrl + U	Coller
Ctrl + W	Rechercher
Ctrl + \	Remplacer
Ctrl + C	Afficher des informations sur la position du curseur (numéro de ligne, de colonne)
Alt + U	Annuler
Alt + E	Refaire

1. Créez une copie du fichier `big.txt` fourni, puis ouvrez-la avec nano en mode *numérotation des lignes*.
2. Remplacez toutes les occurrences du mot *élèves* par le mot *sorciers*.
3. Déplacer les 10 premières lignes à la fin du fichier.
4. Annulez cette action.
5. Enregistrez le fichier avant de quitter nano.

Exercice 7. Personnalisation du shell (pour les plus rapides)

Le shell par défaut est plutôt austère, mais il existe de nombreux moyens de le personnaliser, en modifiant le fichier `~/.bashrc`.

1. Commencez par créer une copie de ce fichier, que vous appellerez `.bashrc_bak`
2. Editez le fichier `.bashrc` avec `nano` et décommentez la ligne `force_color_prompt=yes` pour activer la couleur. Enregistrez le fichier et quittez `nano`.
3. Le fichier `.bashrc` est lu au *démarrage* du shell ; pour le recharger, il faudrait donc se déconnecter puis se reconnecter ; mais il existe un autre moyen : la commande `source .bashrc`. Testez-la, l'invite de commande devrait immédiatement passer en couleurs.
4. Les couleurs par défauts (surtout celle du dossier courant) ne sont pas très visibles. Dans `.bashrc`, cherchez les lignes commençant par `PS1=` ; elles indiquent la mise en forme de l'invite de commande (selon que l'on est en couleurs ou non).

   Sur cette ligne, on peut distinguer un certain nombre de raccourcis :

\u	Nom de l'utilisateur
\h	Nom de la machine (*hôte*)
\d	Date
\t	Heure avec les secondes
\A	Heure sans les secondes
\w	Chemin complet du dossier courant

Remarquez la séquence particulière : `\[\033[1;32m]\u@\h\[\033[00m\]` C'est cette instruction qui indique d'affiche le nom de l'utilisateur et de la machine en vert clair. Plus précisément :

- un code couleur se place entre `\[\033[` et `\]`
- on peut remplacer `\033` par `\e` (ce n'est pas forcément toujours plus lisible…)
- un code couleur se termine toujours par la lettre m
- le code couleur 00 efface toute mise en forme
- on peut combiner différents attributs (couleur, souligné, clignotant, etc.) en les séparant par des points virgules

La page suivante vous donne les différents codes couleurs possibles : https://misc.flogisoft.com/bash/tip_colors_and_formatting

Modifiez l'invite de commande pour qu'elle s'affiche sous la forme suivante :

[heure] - user@host:chemin_courant$

où l'heure est affichée en violet et entre crochets, et le chemin du dossier courant en cyan