TP2 Module 2

Notions

1) Redirections

La plupart des commandes utilisent trois flots d'entrée - sortie : **STDIN**, l'entrée standard, habituellement les caractères entrés au clavier ; **STDOUT**, la sortie standard, ce qui est affiché sur la console ; **STDERR**, la sortie d'erreur, qui est également affichée sur la console. Il est possible de rediriger l'entrée standard d'une commande pour qu'elle lise un fichier en ajoutant < *fichier* à la fin de la commande ; et la sortie standard ou d'erreur pour qu'elles écrivent dans un fichier en ajoutant respectivement > *fichier* et `2>` *fichier* qui créent un fichier ou **effacent** un fichier existant pour écrire dedans. Si l'on ne souhaite pas effacer le fichier, mais ajouter à la fin : remplacer > par >>.

Exemple
La commande `cat` utilisées sans arguments à pour comportement de recopier sur **STDOUT** ce qu'elle lit à partir de **STDIN**.
Si on tape la commande *cat>fic*, celle-ci enregistre les caractères tapés au clavier dans le fichier *fic* du répertoire courant à condition qu'on termine la saisie en appuyant simultanément sur les touches CTRL et D. Pour ne pas effectuer l'enregistrement, on tape simultanément sur les touches CTRL et C.

Exercice 1.

- Dans votre répertoire personnel exécuter la ligne de commandes suivante :

echo "Bonjour linux">f1 ; echo "Heureux de te connaitre">f2 ;echo "À bientôt" >f3

- Taper la commande

cat f1 f2 f3

- Utiliser la dernière commande pour créer un fichier f contenant le contenu du fichier f1, suivi du contenu de f2 et enfin le contenu f3

- Ajouter le texte « A+ » à la fin du fichier f

- Ajouter au fichier f le chemin du répertoire personnel

- Taper les commandes suivantes :

cat f1

^f1^f2

^f1^f3

Exercice 2.

1. Utilisez *`cat`* pour créer un fichier `f1` contenant cinq ou six lignes de votre choix.
2. Utilisez *`cat`* pour créer un fichier `f3` contenant `/etc/passwd`, `f1`, puis encore `/etc/passwd`. Essayer de le faire en une seule commande.
3. Créez un fichier contenant la liste de **tous** les fichiers se trouvant sous votre répertoire principal.
4. Ajoutez au fichier précédent la sortie de la commande `env`.

Il existe un fichier spécial `/dev/null` agissant comme un puit sans fond pour les données vers lequel on redirige tout ce dont on veut se débarrasser.

1. Exécutez *`cat /etc/passwd /etc/toto`*.
2. Comment faire pour que cela n'affiche rien ?

2) Filtres

Un processus qui lit des données sur l'entrée standard et produit des données sur la sortie standard est appelé **filtre**.

2.1 La commande sort

La commande **sort** permet d'effectuer des tris sur des lignes de texte dans l'ordre numérique (**-n**), lexicographique (par défaut) ou selon le dictionnaire (**-d**). Les champs sont délimités par défaut par le caractère de tabulation mais il est possible de spécifier un autre caractère avec l'option "**-t**". Il est également possible de trier sur un champs particulier avec l'emploi de l'option **-k**. L'option "**-r**" permet d'inverser l'ordre de tri. Le tableau 2 donnent quelques exemples.

Commande	Action
$ sort -n < /etc/passwd	tri le fichier /etc/passwd par ordre numérique
$ sort -nt : -k 3 < /etc/passwd	tri le fichier passwd par ordre numérique sur le 3ème champ avec **:** comme délimiteur de champs
$ sort -nrt : -k 3 < /etc/passwd	même type de tri en présentant les résultats inversés

2.2 La commande grep

La commande **grep** permet la recherche dans des fichiers d'une expression particulière. Les options basiques sont **-n** qui permet d'afficher les numéros de ligne, **-i** qui permet de ne pas tenir compte des majuscules et minuscules et **-v** qui affiche les lignes ne contenant pas l'expression.

Commande	Action
$ grep -i "home" < /etc/passwd	affiche les lignes contenant *home* sans tenir compte des majuscules et minuscules
$ grep -v "home" < /etc/passwd	affiche les lignes ne contenant pas *home*

2.3 La commande wc

La commande **wc** permet de compter le nombre de lignes, de mots et de caractères dans un fichier. Parmi les options, il y a **-l** qui affiche le nombre de lignes, **-w** qui affiche le nombre de mots et **-c** qui affiche le nombre de caractères. Le tableau 4 vous donne deux exemples d'utilisation de cette commande.

Commande	Action
$ wc -l < /etc/passwd	compte le nombre de lignes dans */etc/passwd*
$ wc -c < /etc/passwd	compte le nombre de caractère dans */etc/passwd*

2.4 La commande cut

Cette commande extrait des colonnes (option **-c**) ou des champs (option **-f**) des lignes d'un fichier ou de l'entrée standard. Dans le cas de l'option **-f**, il est possible de lui spécifier le délimiteur à chercher en utilisant l'option **-d**. Le délimiteur par défaut est la tabulation. Quelques exemples figurent dans le tableau 5.

Commande	Action
$ cut -f3,7 -d : /etc/passwd	filtre les champs 3 et 7 de chaque ligne de *passwd* en considérant le caractère **:** comme délimiteur
$ date | cut -c1-3	filtre les caractères 1 à 3

2.5 La commande head

Cette commande permet d'éditer le début d'un fichier (ou de l'entrée standard) en spécifiant le nombre de lignes (option **-n**) ou le nombre de caractères (option **-c**) souhaités.

Commande	Action
$ head -c 1000 /etc/passwd	édite à l'écran les 1000 premiers caractères du fichier
$ head -n 10 /etc/passwd	édite les 10 premières lignes du fichier

2.6 La commande tail

Cette commande permet de donner la fin d'un fichier ou de l'entrée standart. Comme avec la commande head, il est possible de spécifier un nombre de caractères (option **-c**) ou de lignes (option **-n**). Il est aussi possible de donner un nombre de blocs (512 octets) avec l'option **-b**.

Commande	Action
$ tail -c 15 /etc/passwd	édite les 15 derniers caractères de */etc/passwd*
$ tail -n 5 /etc/passwd	édite les 5 dernières lignes de */etc/passwd*
$ tail -n +5 /etc/passwd	édite la fin de */etc/passwd* à partir de la 5ème ligne

2.7 La commande tee

La commande **tee** permet une dérivation à l'intérieur d'un tube vers un fichier. Par exemple, si vous souhaitez obtenir un fichier "f1" contenant la liste de votre répertoire et un autre fichier "f2" contenant cette même liste triée, vous taperez :

ls | tee f1 | sort > f2

La sortie de la commande **ls** sera copiée dans **f1** et dirigée dans la commande **sort**. L'option **-a** permet d'écrire dans le fichier en concaténant (mode append).

3) Expressions rationnelles

Les expressions rationnelles sont utilisées par plusieurs commandes très utiles pour filtrer (reconnaître) des mots. Elles ressemblent aux expressions contenant des métacaractères du shell, mais elles n'ont pas la même syntaxe. Le caractère point . permet de filtrer un seul caractère quelconque. L'étoile est un opérateur postfixé signifie une répétition quelconque de l'expression que l'on trouve avant. Le chapeau ^ et le dollar permettent d'ancrer

l'expression respectivement en début ou en fin de ligne. On peut rajouter des parenthèses précédées d'un backslash pour grouper les expressions.

Exemple :

`^ll*$` permet de reconnaître les lignes qui sont composées d'une séquence non vide de la lettre `l`.

L'expression `ab\(ab\)*` permet de trouver les lignes contenant une suite `abababababab`.

Exemples d'expressions rationnelles

```
. (point) représente un caractère quelconque, sauf \n
* (astérisque) répétition du caractère précédent
+  au moins une occurrence de l'expression rationnelle
?  au plus une occurrence de l'expression rationnelle
[...] (crochets) l'un des caractères de l'ensemble.
[^..] en début de crochets recherche dans le complémentaire de l'ensemble
^ début d'une ligne
$ fin d'une ligne
\ annule l'interprétation du caractère suivant, pour jouer le rôle du caractère
usuel
{n,m} indique le nombre de répétitions attendus du caractère précédent
|  joue le rôle de "ou" entre 2 expressions rationnelles.
```

Attention : pour que les caractères spéciaux ne soient pas interprétés par le shell, il est *conseillé* de toujours mettre les expressions rationnelles entre apostrophes.

Exercice 2.

On a la possibilité de rediriger l'entrée et la sortie standard, non plus vers un fichier, mais vers un autre programme. Les programmes conçus pour être utilisés de cette manière sont appelés *filtres*. Nous en avons vu quelques-uns (`cat`, `grep` qui peut être utilisé comme un filtre). Pour effectuer la redirection, on utilisera un *pipe* noté `|`. Par exemple, la ligne `cat /etc/services | grep 25 | more` permet d'afficher, page par page, toutes les lignes du fichier `/etc/services` contenant le mot `25`.

Lisez les pages de *man* des commandes *wc*, *sort*, *cat*, *uniq* et répondez aux questions suivantes :

1. A quoi servent ces fonctions ?
2. Que doit on faire avant d'utiliser `uniq` ?

Quels sont les enchaînements de commandes (une seule ligne) nécessaires pour effectuer les opérations suivantes :

1. Compter le total des lignes des fichiers dans `/etc`
2. Compter le nombre de lignes contenant 25 dans /etc/services
3. Trier le fichier /etc/hosts
4. Afficher le nombre de fichiers n'appartenant pas à root dans /etc.

Exercice 3.

On utilisera, dans cet exercice, les commandes `cut, tail` et `head`.

Les fichier **/etc/passwd** contient les informations relatives aux utilisateurs autorisés. Tandis que le fichier **/etc/group** contient des informations concernant leurs groupes d'appartenance. Utiliser la commande man connaitre la structure de chacun de ces deux fichiers.

Pour les questions qui suivent, trouvez la ligne de commandes permettant de faire l'opération voulue.

1. Afficher le 5ème caractère de chaque ligne du fichier **/etc/group**
2. Afficher du 5ème caractère au 10ème caractère du fichier **/etc/group**
3. Afficher le 5ème et le 10ème caractère du fichier **/etc/group**
4. Afficher à partir du 5ème caractère (jusqu'à la fin) le contenu du fichier **/etc/group**
5. Afficher le 1er et le 3ème champs sachant que le caractère '**:**' est le délimiteur utilisé dans le fichier **/etc/group**
6. Afficher le nom de login et le répertoire de connexion des utilisateurs appartenant au groupe **root**.
7. Combien y'en a-t-il ?
8. Afficher seulement les noms d'utilisateurs commençant par **ro** et leurs répertoires de connexions

Exercice 4.

La commande `find` permet de retrouver des fichiers selon divers critères spécifiés en tant que paramètres pour la commande. Par exemple, la commande `find` *`rep`* `-name` *'nom'* permet de retrouver tous les fichiers, situés dans l'arborescence du répertoire nommé *`rep`*, dont le nom est spécifié par *`nom`*. *`nom`* peut contenir des métacaractères identiques à ceux du shell (mais pas des expressions rationnelles). Cependant, c'est à `find` d'interpréter les caractères et non au shell, donc il est nécessaire de placer *`nom`* entre apostrophes.

5. Affichez tous les fichiers et sous-répertoires de votre l'arborescence du système du fichier.
6. En utilisant l'option `-type` (cf. le manuel), n'affichez que les fichiers. Puis recommencez avec que les répertoires.
7. Comptez le nombre de fichiers spéciaux du répertoire `/dev`.

La commande `find` permet également d'exécuter une commande sur chacun des fichiers qu'elle trouve. On utilise la syntaxe suivante :

`find repertoire -diverseOptions -exec commande {} \;` où la commande *`commande`* va être exécutée pour chaque fichier trouvé en remplaçant `{}` par le nom du

fichier.

Exemple : `find /etc/ -type f -exec wc -l {} \;` permet de compter les lignes de tous les fichiers sous `/etc`.

3. Donnez le rôle de chacune des commandes suivantes :
   $ find . –name ”*.c” –type f –print
   $ find sousrep –name ”*.c” –type f –exec cat {} \; -print
   $ find sousrep –name ”*.o” –type f –exec rm {} \; -print

4. Affichez les informations de la sortie de `ls -l` pour tous les fichiers et répertoires à partir de votre répertoire principal.
5. N'affichez maintenant que la taille et le nom du fichier.
6. Sauvegardez le résultat précédent trié en utilisant la commande `sort`.
7. Affichez tous les fichiers de `/etc/` contenant `host`.
8. Affichez tous les fichiers de `/etc/` contenant le nom de la machine sur laquelle vous êtes.

Exercice 5.

Nous donnons la session ci-dessous ouverte par un utilisateur dont le login est toto et tel que son répertoire de connexion est initialement vide.

$ pwd
/home/toto
$ umask 077
$ mkdir dossier toto
$ touch fichier1
$ chmod u-x dossier ; chmod 500 toto
$ ls –l >fichier ; touch fichier2 ; ls –l >>fichier
$ cd **.././**toto/**././**toto/**./../**
$ grep ’/bin/bash$’ /etc/passwd >passwd
$ cat passwd
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/bin/bash
daemon:x:2:2:daemon:/sbin:/bin/bash
news:x:9:13:News system:/etc/news:/bin/bash
uucp:x:10:14::/var/lib/uucp/taylor_config:/bin/bash
toto:x:500:100:Utilisateur toto:/home/toto:/bin/bash

En continuant la session précitée, donner les résultats ou les messages d’erreurs de chacune des lignes de commandes avant de les tester.
1) pwd
……..

2) cd ; cd dossier/

……..

3) ls –l | grep –v '^–' | wc –l

……..

4) ls –la toto/ | wc –l

……..

cat <passwd | tail –2l | cut –f1 –d'1'

……..

……..

6) find **./** –type b –o type f –exec grep '^–.*$' {} \; 2>/dev/null | wc –l

……..