Ecole Internationale des Sciences
du Traitement de l'Information
Cycle Ingénieur – Première Année

T.P. N° 2 – COMPLEMENTS COMMANDES LINUX DE BASE (corrigé)

OBJECTIFS :

1	Utilisation de flow, tunnels et redirections
2	Utilisation de la commande find
3	Commandes de filtres : find, grep, cut, tr, sed
4	Utilisation des expressions régulières

Les commandes GNU et UNIX
(GNU & Unix commands)

· Ligne de commande
· Gestion de textes avec filtre
· Gestion de fichiers
· Utilisation de flow, tunnels et redirections
· Utilisation des expressions régulières
· Utilisation de l'éditeur vi

PREMIERE PARTIE : REDIRECTIONS

Redirection de la sortie standard et des erreurs

La redirection de la sortie standard permet de ranger ce qui est normalement affiché à l'écran dans un fichier. Elle est symbolisée par la syntaxe :

commande > fichier
commande >> fichier

Dans le premier cas, on crée le fichier ou on l'écrase s'il existait déjà ; dans le deuxième cas on ajoute en fin de fichier.

Redirection de l'entrée standard

Elle est symbolisée par la syntaxe :

commande < filename

Redirection entre commandes

La redirection entre commandes est appelée pipe ; elle est symbolisée par le caractère | ou ^

DEUXIEME PARTIE

Commandes de filtres : grep, cut, tr, sed

Généralités

- Un filtre est une commande qui lit les données sur l'entrée standard, effectue des traitements sur les lignes reçues et écrit le résultat sur la sortie standard.
- Bien sûr les entrées/sorties peuvent être redirigées, et enchaînées avec des tubes.
  A noter que le caractère d'indirection < en entrée n'est pas obligatoire pour les filtres
  Ainsi, dans `# cat /etc/*.conf > tous.conf` cat va bien lire les fichiers qui correspondent au modèle `/etc/*.conf` et les contaténer dans le fichier `tous.conf`

- Les principaux filtres utilisés dans le monde UNIX sont :

☐ cat, more et less
☐ grep
☐ cut
☐ tr
☐ sed

Les commandes cat, more et less

```
cat f1 f2 .. > f concatène f1, f2 .. dans le nouveau fichier f
less f1 f2 .. > f  concatène les fichiers f1 f2 .. en un seul fichier f
(comme cat)
less f3 >> f    ajoute le contenu du fichier f3 à la suite du contenu du
fichier f
```

La commande grep : sélection de lignes

Cet utilitaire (*General Regular Expression Parser*, analyseur général d'expression régulière) sélectionne toutes les lignes qui satisfont une expression régulière (ou rationnelle).

Syntaxe

`grep [options] expreg [fichiers]`
Cette commande recherche dans les fichiers ou sur son entrée standard des lignes de texte qui satisfont l'expression régulière `expreg` indiquée.

Sa sortie peut être redirigée dans un fichier.

Options :

-c donne seulement le nombre de lignes trouvées obéissant au critère
-l donne seulement le nom des fichiers où le critère a été trouvé
-x ligne correspondant exactement à la chaîne
-v donne les lignes où le critère **n'a pas** été trouvé
-i ne pas tenir compte de la casse (ne pas différencier majuscules minuscules)
-n pour n'afficher que les numéros des lignes trouvées
-w pour imposer que le motif corresponde à un mot entier d'une ligne

Constructions :

grep est souvent inclus dans un tube qui lui fournit en entrée le fichier à étudier.

```
Exemple : quelle est la question posée ?
cat /etc/passwd | cut -d: -f1 | grep -w "jean" > sortie
```

Expressions reconnues

Grep ne reconnait pas toutes les expressions rationnelles étendues.
Voici la liste des symboles utilisables par `grep` : . * [ ] [^] ^ $

- ^ début de ligne
- **.** un caractère quelconque
- ***** répétition du caractère situé devant
- **$** fin de ligne
- **x*** zéro ou plus d'occurrences du caractère **x**
- **x+** une ou plus occurrences du caractère **x**
- **x?** une occurrence unique du caractère **x**
- **[...]** plage de caractères permis
- **[^...]** plage de caractères interdits
- **\{n\}** pour définir le nombre de répétition **n** du caractère placé devant

Attention :

Pour éviter une confusion entre les interprétations de ces symboles spéciaux par grep ou par le shell, il est indispensable de "verrouiller" `expreg` en plaçant l'expression entre guillemets " " (et non entre quotes !).

Exemples

Etudier et commenter les commandes suivantes :

1. *cherche dans fichier, les lignes dont la 1ère lettre est qcq et la 2ème doit être o*
   grep "^.o" fichier

2. *cherche dans le fichier passwd les lignes commençant par t*
   grep "^t" /etc/passwd
3. *cherche les lignes ne commençant pas commençant par t*
   grep -w "^t" /etc/passwd
4. *cherche les lignes contenant les mots suivant le modèle T.t.*
   grep "T.t." /etc/passwd
5. *cherche dans le fichier des groupes, ceux qui commencent par a ou b .. ou j*
   less /etc/group | grep "^[a-j]"
6. *pour lister les s-répertoires du rép. /etc*
   ll /etc | grep "^d"
7. *compter les lignes saisies au clavier qui se termine par a*
   grep -c "a$"
8. *afficher les lignes des fichiers essai?.txt qui contiennent a, b ou c*
   grep [abc] "essai?.txt"
9. *détourne le flot de sortie du moniteur pour l'envoyer sur l'entrée de wc pour ?*
   grep [abc] "essai?.txt" | wc -l

Exercices

1. donner une version sans `cut` de la commande précédente
2. *Comment ne sélectionner que "root" avec*
   `cat /etc/passwd | cut -d : -f 1 | grep  "r"`
3. on sait que `ps aux` donne la liste des processus. La commande `/etc/X11/X` est celle qui lance le serveur X. Il est nécessaire de connaitre son **`PID`**, en cas de plantage du serveur X Ecrire la commande qui retourne la ligne correspondante.
4. Se placer dans `/etc`. En une seule commande, faire calculer et afficher le nombre de sous-répertoires de /etc, sous la forme :
   "Il y a 33 répertoires dans /etc"
5. Créer un fichier `essai1` contenant quelques lignes dont des lignes vides Avec **`grep`** générer le fichier essai2 à partir de essai1 sans ligne vide
   `cat essai1 .....  >essai2`

cut : sélection de colonnes

La commande **`cut`** présente 2 formes suivant que l'on sélectionne des colonnes de caractères ou qu'on distingue des champs séparés par un caractère précis.

sélection_colonne

`cut -c(sélection_colonnes) [fichiers]`

Exemples

- *affiche le 5ième caractère*
  cut -c5 fichier
- *affiche du 5ième **au** 10ème caractères*
  cut -c5-10 fichier

- *affiche le 5ième **et** le 10ème caractères*
  cut -c5-10 fichier
- *affiche à partir du 5ième (jusqu'à la fin)*
  cut -c5- fichier

sélection champs

cut -d(séparateur) -f(sélection_champs) [fichiers]

Exercices

Etudier les commandes suivantes

1. `cut -d":" -f1,6 /etc/group`
2. *Que réalise la ligne suivante ? Vérifiez*
   `grep "^st" /etc/passwd | cut -d":" -f1,3-4,6`

La commande wc

Exemples

```
cat /etc/paswwd | grep /bin/bash/ | wc -l
    pour compter les titulaires d'un compte pouvant se connecter avec le
login shell
less /etc/paswwd | grep -vc /bin/bash/
    négation de la question précédente (revoir les rôles ds options -c et -v)
```

La commande tr

tr=Translate, est un filtre ne reconnaissant pas les expr. régulières.
Cette commande est le plus souvent associée à des redirections
Les caractères entrés sont traités et le résultat est envoyé sur la sortie standard
On peut utiliser les intervalles du type `a-z` et les codes ASCII des caractères en notation octale `\0xx`

Syntaxe

1. **tr [options] ch1 ch2 <fich1 >fich2**
   Remplace toutes les occurrences de TOUS les caractères de ch1 par le caractère de ch2, de même rang, dans le flot d'entrée.
2. Exemple
   ```
   # pour convertir et afficher la ligne saisie au clavier en minuscules
   read ligne; echo $ligne | tr 'A-Z' 'a-z'
   ```
3. `tr` **`-c`** `chaine car` remplace tout caractère NON INCLUS dans la chaine `chaine` par le caractère `car`
   ```
   # remplace supprime tous les caractères différents de a,b, ..z par un
   espace
   echo $ligne | tr -c a-z ' '
   ```
4. `tr` **`-d`** `chaine` supprime tout caractère entré, appartenant à la chaine `chaine`
   ```
   # supprime toutes les minuscules non accentuées
   echo $ligne | tr -d a-z
   ```
5. `tr` **`-s`** `chaine` supprime toute répétition des caractères contenus dans `chaine`
   ```
   # supprime les espaces multiples entre les mots
   echo $ligne | tr -s ' '
   ```

L'utilitaire sed

Il s'agit d'un utilitaire (*sed = "Stream EDitor"*) qui sélectionne les lignes d'un fichier texte (ou d'un flot provenant d'un pipe) vérifiant une expression régulière et qui leur applique un traitement ou un remplacement.

Syntaxe

sed [-n] [-e script] [-f fichier-commandes] fichier-source

- L'option -n empêche la sortie à l'écran du résultat (souvent associé à l'option p)
- Le fichier source est traité ligne par ligne conformément à la liste des commandes (-e) ou au fichier de commandes (-f)

Commande de substitution

- La commande s permet d'effectuer des substitutions suivant la syntaxe :
  `sed [adresse]s/expr-régulière/remplacement/options`
- Attention ! contrairement à ce que l'on pourrait attendre, cette commande laisse passer toutes les lignes et ne sélectionne pas celles qui ont satisfait l'expression régulière et donc subi la substitution. Pour sélectionner, voir la commande de destruction.
- Options
  - Sans précision, la commande ne s'applique qu'à la 1ère occurence de chaque ligne.
  - 0...9 : indique que la substitution ne s'applique qu'à la nième occurence
  - g : effectue les modifications sur toutes les occurences trouvées.
- Exemple : `sed s/moi/toi/g fich.moi > fich.toi`
  le fichier `fich.moi` est parcouru, à chaque occurrence de "moi", ce mot est remplcé par "toi" et le nouveau fichier est sauvegardé sous le nom `fich.toi`
- Délimitateur
  Le slash **/** étant très utilisé au niveau du shell comme séparateur de niveau de répertoire, il est possible d'utiliser à la place tout autre caractère comme **#**
  `sed s#/home#/rep_perso#g /etc/passwd > /tmp/passwd.new`

Destruction ou sélection

- Cette option permet de filtrer les lignes qui satisfont une expression régulière. Ces lignes ne sont pas détruites dans le fichier d'origine, mais ne sont pas transmise en sortie.
- Comment modifier alors le fichier à traiter ?
  ```
  cp fichier copie
  sed /.../d copie
  ```
- Par exemple, pour détruire toutes les lignes vide d'un fichier :
  `sed /^$/d`

Ajout, insertion et modification

Pour utiliser ces commandes, il est nécessaire de les saisir sur plusieurs lignes

```
sed [adresse] commande\
expression
```

La commande peut être :

```
a pour ajout ;
     i pour insertion ;
     c pour modification.
```

TRAVAIL A EFFECTUER

Exercices 1 – Redirections

1. Créez un fichier dont chaque ligne commence par un chiffre, suivi d'un slash (`/`), puis d'un ou plusieurs mots.
    1. Affichez les lignes de ce fichier triées en ordre croissant, suivant le nombre placé en début de ligne;
    2. Éliminez de chaque ligne le chiffre et le caractère «`/`»;
    3. Triez ces lignes par ordre alphabétique inverse.
3. Combien de fichiers de configuration avez-vous ?
4. Combien de répertoires de configuration avez-vous ?
5. Comment mettre dans un fichier la liste de tous les fichiers de l'arborescence à partir du répertoire courant ?
6. Créez un fichier `liste` contenant la liste de tous vos fichiers, avec leur taille, leurs droits, etc.
7. Comment afficher uniquement les fichiers du répertoire courant qui sont des liens symboliques ?
8. Combien de lignes contiennent le mot «*file*» dans la page de man de `syst` ?
9. Quels sont les dix plus gros fichiers de `/usr/bin/` ?
10. Pour chaque ligne du fichier `/etc/hosts`, affichez :

- Le cinquième caractère;
- Les caractères 5 à 10, et le treizième;
- Tous les caractères à partir du quinzième.

Exercices 2 - Grep

1. Créer un répertoire **essai-grep** dans votre home directory. Dans ce répertoire créer les fichiers suivants:
tomate poire pomme cerise Fraise fraise courgette POMME3 afraise

Editez les fichiers (sortie de la commande **ls** redirigée vers **grep**) avec les critères sur leur nom suivant:

Critère 1	Le nom doit être Fraise ou fraise
Critère 2	se est en fin de nom
Critère 3	ai est présent dans le nom
Critère 4	Nom contenant un chiffre numérique
Critère 5	Nom contenant la chaîne mm ou MM

2. Copiez le fichier **/etc/passwd** dans votre home directory. Editez la ligne commençant par votre nom de login.

3. Dans le fichier **passwd** qui est dans votre home directory, éditez les lignes commençant par des noms de login ne contenant pas de chiffre.

4. Editez les lignes du fichier **passwd** commençant par des noms de login de 3 ou 4 caractères.

Exercices 3 - Les filtres

1. Copiez le fichier **/etc/passwd** dans votre home directory. Editez uniquement les champs contenant le login et la home directory.

2. Triez **passwd** sur le nom

3. Extraire les nom de login et UID puis triez suivant les UID, le tout en une seule commande, vous redirigerez le tout vers un fichier

4. Dans le fichier de résultat précédent remplacer les ":" par des " " (espace).

5. Editez les cinq dernières lignes du fichier.

6. Editez les cinq premiers caractères du fichier.

Exercices 4 - find

1. Cherchez dans toute l'arborescence les fichiers dont le nom se termine par **.c**, redirigez les erreurs vers le fichier poubelle **/dev/null**
Commençant par **X** ou **x**.
Dont les noms ne contiennent pas de chiffre.

2. Chercher dans **/usr** les fichiers dont la taille dépasse 1Mo (2000 blocs de 500Ko) et dont les droits sont fixés à 755 (-rwxr-xr-x).

3. Combien il y a de fichiers dans toute l'arborescence vous appartenant et ayant les droits fixés à 666 (-rw-rw-rw-).

4. Trouver tous les fichiers **core** dans l'arborescence et supprimez les après confirmation.

TROISIEME PARTIE : CREATION ET SURVEILLANCE DES PROCESSUS

Exemple :

1	Affichez la liste de tous les processus qui vous appartiennent	`ps -ef | grep votre_username`

Exercices :

1. Affichez la liste des processus associés à votre terminal. Affichez la liste des processus dont vous êtes propriétaire. Recommencez en utilisant les options `-l` et `-f`. À quoi correspondent les colonnes `PID` et `PPID` ?
2. Utilisez la commande `nice` pour lancer des commandes ayant une faible priorité.
3. Interprétez la hiérarchie des processus qui vous appartiennent.
4. La commande `ps | wc` compte deux processus en plus de ceux qui existent réellement lorsqu'on lance la commande. Pourquoi ?
5. Donner deux commandes pour reprendre l'exécution d'une instruction interrompue par un `^Z`.

QUATRIEME PARTIE : LES LIENS

Exercices

6. Vous avez chez vous un répertoire `tmp/` qui contient un fichier `bidon`. Créez un lien physique sur `tmp/bidon` appelé `blo`, dans votre répertoire d'accueil (`HOME`). Comparez les contenus de `tmp/bidon` et de `blo`. Que contient `blo` ?
7. Même question avec un lien symbolique.
8. Quelles sont les différences entre les liens durs et les liens symboliques ?
9. Dans quel cas ne peut-on pas faire de lien physique ? Que faut-il faire ?
10. Quel est l'effet de `chmod` sur un lien ?

CINQUIEME PARTIE : UTILISATION DES EXPRESSIONS REGULIERES

Exercices

1. Vous avez chez vous des fichiers appelés `essai1`, `essai2`, `essai3` et `essai4`. Comment les effacer en une seule ligne de commande ?
2. Dans mon répertoire d'accueil, j'ai un certain nombre de fichiers avec un suffixe `.c`. Je désire les regrouper dans un répertoire que j'appelerai `C/`. Quelles sont les commandes que je dois taper ?
3. Vous désirez regrouper dans un répertoire `Rangement` les fichiers dont le nom contient un caractère minuscule suivi d'un caractère majuscule. Quelle(s) est/sont la/les commande(s) à donner ?
4. Même chose avec les fichiers dont le nom contient trois voyelles à la suite.
5. En utilisant `ls` et `grep`, affichez la liste des fichiers dans `/bin` dont le nom :

- Commence par «a» et dont la deuxième lettre est «s» ou «t»;
- Contient «un» et se termine par «t»;
- Contient «gre» ou «st».
- Contient exactement deux lettres «m»;
- Contient au moins deux lettres «m»;
- Contient au moins quatre caractères et aucun chiffre;
- Est constitué de deux lettres exactement;
- Commence et finit par un chiffre.

11. Comment éliminer les lignes vides dans un fichier ? Comment éliminer les lignes ne contenant que des blancs ?