EMSI RABAT – Ingénierie Informatique & Réseaux

EMSI 3 – 3°IIR 3 – INFORMATIQUE

Système d'Exploitation UNIX & Programmation Système
*===> Études Avancée des SE UNIX & GNU/Unix & LINUX

Professeur : **Mme. L. ELHALOUI**

Étudiants :
Youssef MAHTAT
Ibrahim MANNANE
Anoir EL-AROUSSE
Imane BOULHIMEZ

Système d'Exploitation UNIX

&

Programmation Système

–

Études Avancée des SE UNIX : GNU/Unix & LINUX

Notes de Cours et de Résolutions
en
Informatique :

Révision du cours
& Résolutions des TPs
(UNIX & GNU/Unix & LINUX)

Système d'Exploitation UNIX

GNU/Unix & LINUX


slackware
CentOS
fedora
ubuntu
Mandriva
debian
openSUSE
PCLinuxOS
redhat
pardus
archlinux
Live GNU/Linux System
KNOPPIX

Partie 1 :

CHAP 1 & 2 : Notions & Commandes de Bases
CHAP 3 : Droits d'accès des utilisateurs

→ Part 1 – Notions & Commandes de Bases :

** → Caractéristiques UNIX :

Unix est le système d'exploitation des grands serveurs par excellence. Ce système est multitâche, multi-utilisateur, multi-Platform, réparti et sécurisé.

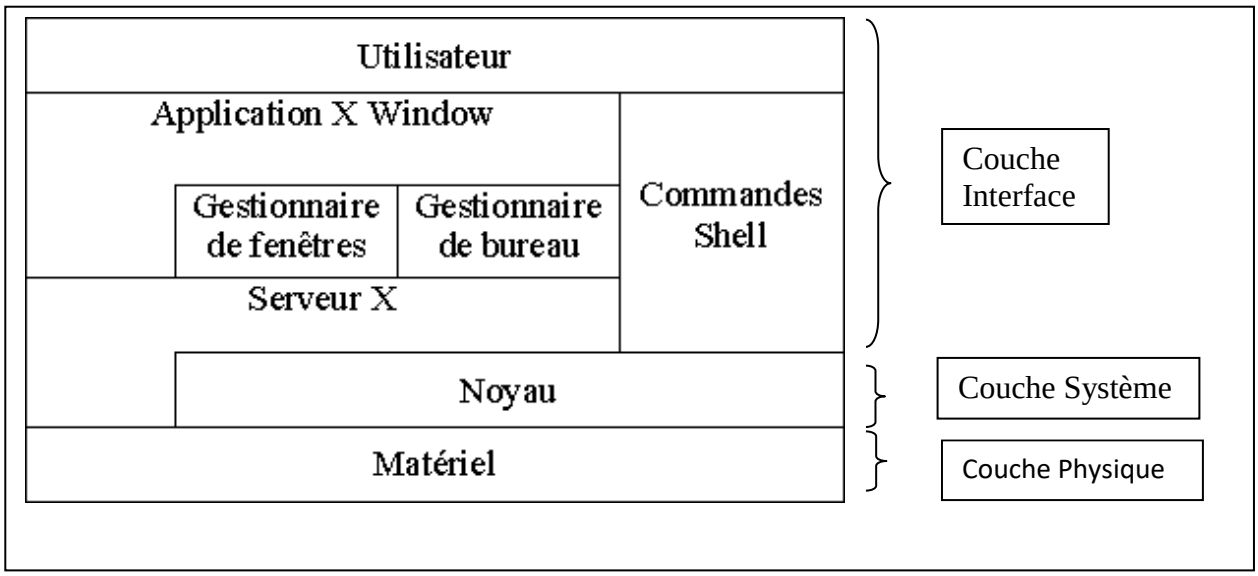
** → Système de Gestion des Fichier (SGF) du UNIX :

→ En UNIX, tout est fichier.
→ Sous Unix un fichier est :

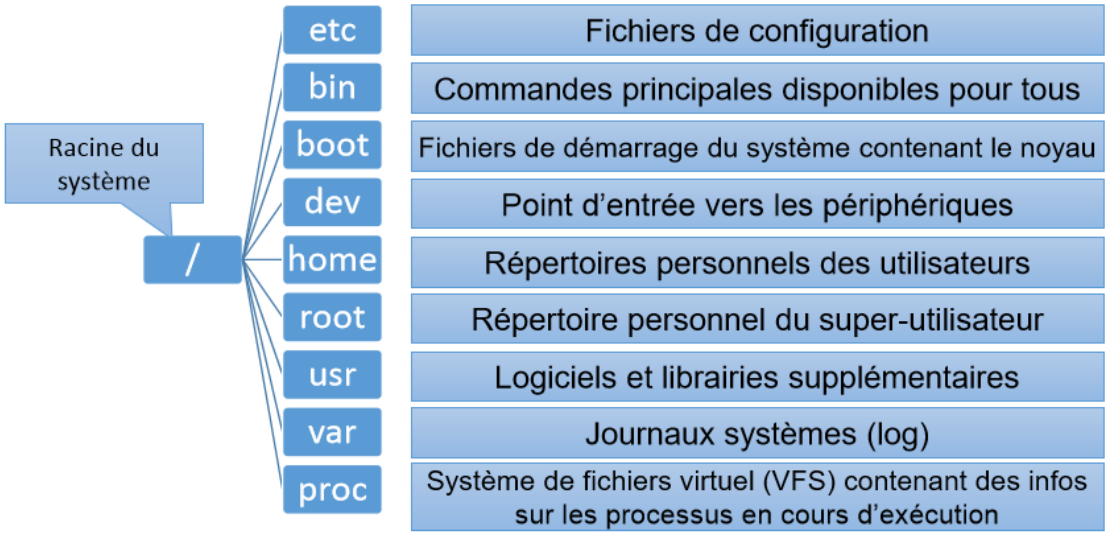
- Toujours désigné par un nom.
- Possède un unique i-node (certaines informations concernant le fichier).
- Possède les fonctionnalités suivantes :
  - Ouverture.
  - Fermeture.
  - Lecture (consultation).
  - Écriture (modification)

- Un fichier peut être :
  - Ordinaire (on utilise parfois le terme "normal") (-)
  - Répertoire (d)
  - Lien ou Lien symbolique (l)
  - Fichier de Périphérique

** → Architecture du SE LINUX :



** → Arborescence LINUX :



→ Symboles Associés à l'arborescence :

- Le « . » : répertoire courant ;
- Le « .. » : répertoire parant ;
- Le « ~ » : répertoire personnel de l'utilisateur ;
- Le « / » : répertoire racine ;

** → Système de Gestion des Utilisateurs (SGU) du UNIX :

- Unix est multi-utilisateurs :
  - Nécessité d'avoir un nom particulier aux yeux du système ("login") ainsi qu'un mot de passe pour la sécurité ;
- Types d'utilisateurs :
  - L'utilisateur "root" : super utilisateur -> possède tous les droits sur la machine ;
  - Les autres utilisateurs (possèdent des droits restreints) (Utilisateurs Humains, virtuel, ...) ;
- Caractéristiques des utilisateurs :
  - Login (nom d'utilisateur) ;
  - Password ;
  - UID (identifiant unique de l'user dans le système) ;
  - GID (identifiant du groupe dont appartient l'utilisateur) ;
  - Type d'interpréteur Shell utilisé par l'utilisateur ;
  - Répertoire personnel de l'utilisateur ;

RmQ : Toutes ces informations sont stockées dans le fichier "/etc/passwd"

** → Commandes LINUX de bases :

cd	Change de Répertoire

ls	Afficher les fichiers <chemin>
-m	Séparés par virgule
-t	Triés par date (le plus récent)
-lu	Triés par date de dernier accès et indique cette date (le plus récent)
-F	Indiquer les types : **f/** (répertoire), **f*** (exécutable), **f@** (lien)
-S	Triés par taille (décroissant)
-X	Triés par extension
-r	Triés par alphabet (inverse)
-R	Arbre de fichiers
-l	Longue format (droit, appartenance, taille, date/heure modification, nom)
-L	Si lien symbolique afficher les informations du fichier référencé
-i	i-node (emplacement, propriétaire, droits, taille, date création, date modif)
-d	Ne pas lister le contenu des répertoires

cp	Copier des fichiers dans un répertoire <fichier> <répertoire>
-i	Avertit l'existence d'un fichier du même nom et demande de le remplacer
-f	Forcer la copie
-l	Lien dur
-s	Lien symbolique
-p	Préserver toutes les informations
-r	Copier récursivement

mv	Déplacer fichier <fichier> <répertoire> / Changer nom <f1> <f2>
-b	Sauvegarde de fichier avant le déplacer
-i	Demande la permission pour chaque fichier & dossier
-u	Ne pas remplacer si la date de modification >= de celle du remplaçant

rm	Supprimer fichier <fichier1> <fichier2> …
-d	Supprimer les dossiers vides
-r	Supprimer un répertoire et ses sous-répertoires récursivement
-f	Supprimer fichiers protégés en écriture et dossier sans confirmation
-i	Demande la permission pour chaque fichier & dossier

mkdir	Créer répertoire <répertoire1> <répertoire2> …
Ø	Créer répertoire seulement si parents existent déjà, sinon erreur
-p	Créer répertoire et ses parents s'ils n'existent pas

rmdir	Supprimer répertoire <répertoire1> <répertoire2> …
Ø	Supprimer répertoire seulement s'il est vide
-p	Supprimer répertoire et sous-répertoires (récursivement) s'ils sont vides

touch	Créer fichier <fichier>

cat	Concaténer fichier
<f1>	Concaténer f1 à la fin de stdout (afficher le fichier f1)
<f1> <f2>	Concaténer f2 à la fin de f1
> <f1>	Ecrase f1 avec le stdin à la fin de <ctrl+D>
>> <f1>	Concaténer stdin à la fin de f1 <ctrl+D>

grep	Chercher dans fichier <motif1> '|' <motif2> <fichier1> <fichier2>
-c	nombre de lignes contenant l'expression
-a	traiter fichier binaire comme fichier texte
-R, -r, -recursive	tous fichiers du répertoire récursivement
-v	lignes ne contenant pas l'expression
-i	ignorer la case
-w	les lignes contenant le mot donné comme motif

sort	Trier lignes fichier <fichier>
-b	ignorer les blancs au début des lignes
-d	tri téléphonique : ignorer tous sauf lettres, chiffres et blancs
-f	minuscules équivaux aux majuscules (non accentuées)

head / tail	Afficher premières/dernières lignes <fichier>
-n, --lines	N lignes
-q, --quiet, --silent	ne pas afficher les en-têtes mentionnant les noms de fichiers
-v, -verbose	toujours afficher les en-têtes mentionnant les noms de fichiers
--version	précéder par un numéro de version

diff	Comparer 2 fichiers <source> <cible>
-a	traiter comme fichiers texte et comparer ligne par ligne
-b	ignorer les espaces blancs
-B	ignorer les lignes blanches
--brief	indiquer si différents seulement

find	Chercher fichier <répertoire> <critère1> <critère2> …
-name	sur nom du fichier
-perm	sur droits d'accès du fichier
-links	sur nombre de liens du fichier
-user	sur propriétaire du fichier
-group	sur groupe auquel appartient le fichier
-type	sur type du fichier (d=répertoire, c=caractère, f=fichier normal, l=lien)
-size	sur taille du fichier en blocs (bloc = 512 octets)
-atime	sur date de dernier accès en lecture du fichier
-mtime	sur date de la dernière modification du fichier
-ctime	sur date de création du fichier

chmod	Modifier droits <droits> <fichier/répertoire>
-R	appliquer à tous fichiers et répertoires d'un répertoire
rwxrwxrwx	3 lettres pour chacun : propriétaire, groupe du propriétaire, autres
ajouter/retirer/ retirerTous&Ajouter droit	concerné#droit : concerné (u, g, o, a), #(+,-,=), droit(r,w,x) -> concerné#droit : [multipliable séparable par virgule] -> concerné : [multipliable non séparable]
modifier droits	(3bits)(3bits)(3bits) ⇔ 3 chiffres octaux (ex : 676)

umask	Changer droits par défaut <valeur>
par défaut	Fichier : 666, répertoire : 777
Ø	Retrancher de droits par défaut la valeur donnée comme argument

chown	Changer propriétaire du fichier <propriétaire> <fichier>
chgrp	Changer groupe du fichier <groupe> <fichier>

cut	Supprimer une partie de chaque ligne <fichier1>
-d	Séparateur des champs
-f	Liste des champs
-c	Sélection sur le rang du caractère
-b	Sélection sur le numéro d'octet
-s	(Avec -f) supprime les lignes vides

Commandes simples	
id	Informations d'id d'utilisateur
whoami	Nom d'utilisateur
users	Utilisateurs connectés
who	Users plus détaillée
passwd	Changer mot de passe
groups	Groupes auxquels l'utilisateur appartient
newgrp <groupe>	Changer le groupe
su <username>	Changer l'id de l'utilisateur
lastlog	Date de dernière connexion
cd <chemin>	Changer de répertoire
pwd	Répertoire courant
date	Date actuelle
cal	Calendrier
more <fichier>	Contenu du fichier (espace : suivant, b : précédent, q : quit)
less <fichier>	Contenu du fichier (espace : suivant, b : précédent, q : quit)
vi <fichier>	Éditer le fichier
nano <fichier>	Éditer le fichier
gedit <fichier>	Éditer le fichier en mode graphique
man <commande>	Manuel de la commande
du	Taille de l'arborescence
wc	Nombre de lignes, mots et caractères (-l, -w, -c)
paste	Regroupe les fichiers ligne par ligne <fichier1> <fichier2> …
echo "texte"	Afficher du texte

Caractères spéciaux			
*	Suite de caractères	;	Séparateur des commandes
?	Un seul caractère	[ ]	Choix entre alternatives
esapce	Séparateur des paramètres	.	Répertoire courant
..	Répertoire parent	~	Répertoire personnel
{ , }	Ensemble de fichiers	[!]	N'importe quel caractère sauf ces alternatives

Caractères spéciaux pour la commande grep			
.	Un seul caractère	^	Début du mot
[ ]	Choix entre alternatives	$	Fin du mot
[^]	N'importe quel caractère sauf ces alternatives		

** → Résolution du TP1 :

Exercice 1 :

1) Dans votre répertoire courant, créez en une commande les fichiers suivants: **fiche1 fiche2 Fiche4 fiche45 fichier41 Fichier510** en utilisant la commande **touch**
2) Lister tous les fichiers
   a. se terminant par 1
   b. commençant par fiche4
   c. commençant par fiche4 et de 7 caractères maximum
   d. commençant par fiche et se terminant par 2 chiffres numériques
   e. contenant la chaîne « hier »
   f. commençant par f ou F

3) Créer les répertoires **Files** et **tmp** dans votre répertoire courant, en une commande déplacez les fichiers précédemment créés dans le répertoire **Files**.

4) Copier les fichiers dont l'avant dernier caractère est un 4 ou 1 dans le répertoire **tmp** en une seule commande.

→ Résolutions :

Q1/

```
touch fiche1 fiche2 Fiche4 fiche45 fichier41 Fichier510
```

Q2/

```
a/   ls *1
b/   ls fiche4*
c/   ls fiche4?
d/   ls fiche*[0-9][0-9]
e/   ls *hier*
f/   ls [Ff]*
```

Q3/

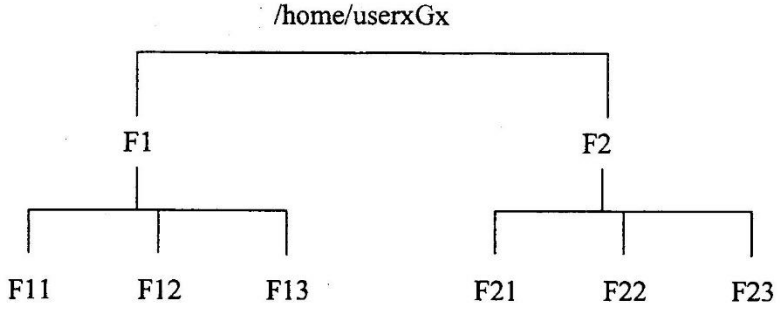
```
mkdir Files tmp
mv [fF]ich* Files
```

Q4/

```
cp Files/*[41]? tmp/
```

Exercice 3 :

Soit l'arborescence suivante :



1) Dans votre répertoire personnel, crée en une seul commande les répertoires F11, F12, F13, F21, F22 et F23 en utilisant la commande mkdir avec l'option –p
2) Créer le fichier ch1 dans le répertoire F22
3) Copier ch1 dans F11 sous le nom ch2
4) Copier ch2 dans F12 sous le nom ch3
5) Lier ch2 à ch4 dans F13
6) Lier ch1 à ch5 dans F13
7) Modifier le fichier ch2 en utilisant la commande cat
8) Afficher le contenue du fichier ch4
9) Supprimer le fichier ch2
10) Copier ch1 dans F1 sous le nom ch6
11) Quel est le nombre de liens pour chacun des fichiers suivants : ch1, ch3, ch4, ch5, ch6
12) Supprimer de deux façons les deux arborescences suivantes : F1 et F2

→ Résolutions :

```
Q1/   mkdir -p F1/{F11,F12,F13} F2/{F21,F22,F23}

Q2/   touch F2/F22/ch1

Q3/   cp F2/F22/ch1 F1/F11/ch2

Q4/   cp F1/F11/ch2 F1/F12/ch3

Q5/   ln F1/F11/ch2 F1/F13/ch4

Q6/   ln F2/F22/ch1 F1/F13/ch5

Q7/   cat <<FIN>> F1/F11/ch2

      out[]: > Je suis un EMSISTE càd the best
      out[]: > Si tu ne me crois pas vient me défier
      out[]: > Je suis pret et j'ai pas peur
      out[]: > FIN
 // ou bien : cat >> F1/F11/ch2
 // ou bien : vi F1/F11/ch2
```

Q8/ cat F1/F13/ch4

// Normalement si le Fichier ch4 est bien lier avec ch2 il va
// afficher le contenu qu'on vient de taper

Q9/ rm F1/F11/ch2

Q10/ cp F2/F22/ch1 F1/ch6

Q11/ ls -l F2/F22/ch1

Q12/

methode 1 : rm -R F1 F2

methode 2 : rm F1/*/* F2/*/* & rmdir -p F1/* F2/*

** → Résolution du TP2 :

Exercice 1 :

1. Créer un fichier fruit.price dans votre répertoire tpunix et écrire le texte suivant (en utilisant la commande cat) :

apples : 09
apricot : 7
bergamot : 12
blackberry : 39
cantaloupe :10
cherry :6
citron :5
elderberry :79
grape :59
grapefruit :67
huckleberry :66
Indian mulberry :99
kiwi :19
lemon :8
mango :34
melon :50
noni :56
nut :19
orange :14
peach : 17
pear :29
sloe :34
strawberry : 80

2. Visionner votre fichier avec la commande more.
3. À l'aide de **grep**, faire afficher les lignes de ce fichier :
   a) contenant l'expression **berry**
   b) contenant l'expression **appLES** en ignorant la casse (distinction minuscule / majuscule)
   c) ne contenant pas l'expression **apples**
   d) dont le fruit commence par la lettre **s**
   e) dont le fruit commence par une voyelle minuscule
   f) dont le fruit commence par la lettre **l** (et pas le chiffre 1) ou une lettre comprise entre **a** et **g**
   g) dont le fruit ne commence pas par une voyelle
   h) dont le fruit ne commence pas par une lettre comprise entre **a** et **m**
   i) dont le prix se termine par **79**
   j) dont le prix ne se termine pas par **9**
   k) dont le prix se termine par 9 mais pas par **39**, **69**, **79**, ni **89**
   l) dont le prix ne se termine pas par **09**, ni par **39**, ni par **79**
4. en une seule commande supprimer votre répertoire courant (travail).

→ Résolutions :

```
Q1/   cat >> fruit.price
      (ctrl + D)


Q2/   more fruit.price
   //ou cat fruit.price


Q3/

       a/   grep berry fruit.price

       b/   grep -i appLES fruit.price

       c/   grep -v apples fruit.price

       d/   grep ^s fruit.price

       e/   grep ^[aeiou] fruit.price

       f/   grep ^[la-g] fruit.price

       g/   grep -i ^[^aeiou] fruit.price

       h/   grep -i ^[^a-m] fruit.price

       i/   grep 79$ fruit.price

       j/   grep -v 9$ fruit.price

       k/   grep [^3678]9$ fruit.price

       l/   grep -v [037]9$ fruit.price


Q4/   rm -R
```

Exercice 2 :

1. Dans votre répertoire de travail personnel, créer un répertoire que vous appellerez **tpunix**.
2. Copier le fichier **/etc/passwd** dans votre répertoire **tpunix** sous le nom **des_lignes**.
3. Afficher le contenu de ce fichier
4. Faire afficher la première ligne de **des_lignes**
5. Utiliser un nombre négatif en argument de l'option -n, faire afficher **des_lignes** mais pas ses deux dernières lignes
6. Faire afficher les deux dernières lignes de **des_lignes**
7. Utiliser un nombre positif en argument de l'option -n, faire afficher **des_lignes** mais pas ses cinq premières lignes
8. Faire afficher les deux premières lignes de tous vos fichiers d'extension **.txt** (de tpunix), sans en-tête

→ Résolutions :

```
Q1/   mkdir tpunix

Q2/   cp /etc/passwd tpunix/des_lignes

Q3/   cat tpunix/des_lignes

Q4/   head -n 1 tpunix/des_lignes

Q5/   head -n -2 tpunix/des_lignes

Q6/   tail -n 2 tpunix/des_lignes

Q7/   tail -n +6 tpunix/des_lignes

Q8/   head -qn 2 *.txt
```

Exercice 3 :

1. Rechercher le fichier ordinaire fiche.txt dans l'arborescence de votre répertoire personnel.
2. Rechercher le répertoire tpunix dans l'arborescence de votre répertoire personnel.
3. Rechercher tous les fichiers ordinaires vous appartenant dans toute l'arborescence du système de fichiers (c'est à dire depuis la racine).
4. Afficher les références des fichiers ordinaires vous appartenant contenus dans l'arborescence du système de fichiers et qui ont été modifiés il y a moins d'une semaine,
5. Afficher tous les noms des fichiers répertoires commençant par une lettre majuscule.
6. Afficher tous les fichiers normaux ayant pour taille 150Ko ?
7. Afficher tous les répertoires ayant les autorisations d'accès '755'.

→ Résolutions :

```
Q1/   find ~ -type f -name 'fiche.txt' -print

Q2/   find ~ -type d -name "tpunix" -print

Q3/   find / -type f -user YMahtat -print

Q4/   find / -type f -user YMahtat -mtime -7 -print

Q5/   find / -type d -name '[A-Z]*' -print

Q6/   find / -type f -size 300 -print (bloc 512 octet -> 1 Ko = 2 blocs)

Q7/   find / -type d -perm 775 -print

RmQ :
      -print peut ne pas être ajoutée ;
```

** → Résolution du TP3 :

Exercice 1 :

1) Dans votre répertoire de travail personnel, créer un répertoire que vous appellerez **catalogue** et les fichiers suivants : **fich1**, **fich2**, **fich3**, **prog.c**, **f1.c**, **f2.c**.
2) Quels sont les droits d'accès attribués à ces fichiers.
3) Placez les valeurs de ces droits d'accès aux fichiers suivants

fich1 : -rwxrw-r--
fich2 : -r—r—r--
fich3 : ---x--xr--

Utilisez les deux formes symbolique et octale.
4) Déplacez-vous dans le répertoire **catalogue** et créez un fichier **fich4**.
5) Comment appeler la commande **umask** pour attribuer à tous les nouveaux fichiers les droits d'accès 'rw-r---w-'.
6) Changer les modes d'accès de tous les fichiers qui ont une extension '**.c** ' en 'rw-r-xr-x'.
7) Supprimer le mode 'x' pour le groupe au niveau du fichier **fich3**.
8) Rajoutez le mode 'w' pour le propriétaire et enlevez 'r' aux autres du **fich3**
9) Changer le propriétaire de ce fichier.

→ Résolutions :

```
Q1/  touch fich1 fich2 fich3 prog.c f1.c f2.c

Q2/  ls -l

Q3/  chmod 764 fich1 ou chmod u=rwx,g=rw,o=r fich1
     Chmod 444 fich2 ou chmod a=r fich2
     chmod 114 fich3 ou chmod ug=x,o=r fich3

Q4/  cd catalogue
     touch fich4

Q5/  umask 024

Q6/  chmod 665 *.c

Q7/  chmod g-x fich3

Q8/  chmod u+w,o-r fich3

Q9/  chown new_user fich3
```

Exercice 2 :

1) Créer un fichier texte **fic1** et changez ses droits d'accès pour que tous ceux de votre groupe puissent écrire dedans.
2) Donnez en une seule ligne le droit d'exécution à tous les utilisateurs d'un fichier **fiche1.bash** qui n'a jusqu'alors que des droits standards (-rw-r--r--).
3) Le fichier **toto** a les droits suivants : -rwxr--r--. Modifiez-en les droits en une ligne de commande de sorte que le propriétaire n'ait plus que le droit de lecture.
4) Modifier les droits du fichier **toto** (-rwxr--r--) de sorte que le groupe et les autres utilisateurs aient les mêmes droits que le propriétaire.
5) Quelle option permet de modifier récursivement les droits d'un répertoire et des fichiers qu'il contient ?
6) Comment appeler la commande pour attribuer à tous les nouveaux fichiers les droits d'accès 'r---w-r--'.

→ Résolutions :

```
Q1/   touch fich1
      chmod g+w fich1

Q2/   chmod u,g,o +x fich1.bash

Q3/   chmod u=r toto ou chmod 444 toto

Q4/   chmod g+wx,o+wx toto ou chmod 777 toto

Q5/   chmod -R

Q6/   umask 242
```

Exercice 3 :

1) Vérifiez les droits de votre répertoire utilisateur. Que peuvent y faire les autres utilisateurs ?
2) Créez un répertoire nommé `rep`. Quels sont les droits de ce répertoire ?
3) Créez un fichier texte `fichier.txt` dans le répertoire `rep`. Changer l'identité de l'utilisateur et essayer de lire le contenu du fichier. Peut-il le supprimer ?
4) Placez-vous dans le répertoire père de `rep` et retirez-vous les droits en lecture sur ce répertoire. Pouvez-vous lister le contenu du répertoire ? Pouvez-vous visualiser le contenu de `fichier.txt` ? Rétablissez le droit en lecture sur `rep`.

5) Créez le fichier `fichier2.txt` dans `rep`. Supprimez les droits en écriture au répertoire `rep` ainsi qu'au fichier `fichier2.txt`. Modifiez `fichier2.txt`. Est-ce possible ? Pourquoi ?
6) Essayez de faire une copie de `fichier2.txt` en `fichier3.txt`.
7) Ajoutez le droit d'écriture et enlevez le droit d'exécution du répertoire `rep`. Essayez de créer un fichier dans `rep`. Essayez de lister le contenu de `rep`. Essayez de vous déplacer dans `rep`. Qu'en concluez-vous sur l'utilisation du droit d'exécution sur un répertoire ?

→ Résolutions :

**→ On suppose qu'il y'a 2 users : fedora(super-user) et EMSI

```
Q1/   ls -l ~ (drwx------)

Q2/   mkdir rep (drwxr-xr-x)

Q3/   cd rep
      touch fichier.txt
      su EMSI
      cat fichier.txt (autorisé)
      rm fichier.txt (non-autorisé)

Q4/   su fedora
      cd ..
      chmod g-r rep
      cd rep
      su EMSI
      ls (non-autorisé)
      cat fichier.txt (autorisé)
      su fedora
      chmod g+r ../rep
```

Q5/ touch fichier2.txt
chmod d-w ../rep
su EMSI

Q6/ cat >> fichier.txt (non-autorisé)
cp fichier2.txt fichier3.txt (non-autorisé)

Q7/ su fedora
chmod g-x+w /rep
su EMSI
ls (non-autorisé)
cd (non-autorisé)

Partie 2 :

CHAP 4 : Redirection des Entrées/Sorties

→ Part 2 – Redirection des E/S :

Redirection de flux	
>	sortie standard vers un fichier en l'écrasant
>>	sortie standard à la fin du fichier
2>	sortie d'erreur vers un fichier en l'écrasant
2>>	sortie d'erreur à la fin du fichier
2>&1	fusionner la sortie d'erreur et la sortie standard
<	lire depuis un fichier
<<	lire depuis le clavier progressivement
|	Chainer les commandes

tr	Substitution ou suppression des caractères <ch1> [<ch2>]				
Ø	remplacer toutes les occurrences de <ch1> par <ch2>				
-s	éliminer les répétitions successives de <ch1>				
-d	supprimer toutes les occurrences de <ch1>				
-c	complément de <ch1>				
[a-z]	segment de a à z	\xyz	code octal xyz	'[:upper:']	majuscule
[a*n]	a…a n fois	'[:lower:]'	minuscule	'[:alnum:']	alphanumérique

uniq	Eliminer les lignes dupliquées successives <fichier>
-d	affiche seulement les lignes dupliquées
-u	affiche seulement les lignes non dupliquées (par défaut)
-c	nombre d'exemplaires de chaque ligne

RmQ : Avant d'utiliser la commande uniq il faut impérativement utiliser la commande sort ;

paste	regroupe les fichiers ligne par ligne <fichier1> <fichier2> …

cut	Supprimer une partie de chaque ligne <fichier1>
-d	séparateur des champs
-f	liste des champs
-c	sélection sur le rang du caractère
-b	sélection sur le numéro d'octet
-s	(avec -f) supprime les lignes vides

wc	Nombre de lignes, mots et caractères (-l, -w, -c)

** → Résolution du TP4 :

Exercice 1 :

1) Dans votre répertoire courant, créez en une commande les fichiers suivants : annee1 Annee2 annee4 annee45 annee41 annee510 en utilisant la commande **touch**
2) Dans votre répertoire courant, créez le répertoire R
3) Dans votre répertoire courant, Copier la liste des fichiers et leurs attribues dont l'avant dernier caractère est un 4 ou 1 dans le fichier ch1 dans R.
4) Afficher le contenue de ch1
5) Copier la date du système dans le fichier ch1 dans R
6) Afficher le contenue de ch1
7) Ajouter au contenu du fichier ch1, la liste des fichiers et leur attribues dont l'avant dernier caractère est un 4 ou 1
8) Que se passe-t-il si vous taper les commandes suivantes :
   - ls –l [aA]* > ch2
   - wc < ch2
   - ls -l [aA]* | wc

   wc (imprime le nombre de lignes, de mots et de caractères fournis à l'entrée standard)

→ Résolutions :

```
Q1/    touch annee1 Annee2 annee4 annee45 annee41 annee510

Q2/    mkdir R
Q3/
       ls -l *[41]? > R/ch1
       ou ls -l *[41]? >> R/ch1

Q4/
       more R/ch1
       ou cat R/ch1
       ou less R/ch1

Q5/
       date > R/ch1

Q6/
       more R/ch1
       ou cat R/ch1
       ou less R/ch1

Q7/    ls -l *[41]? >> R/ch1

Q8/
```

`ls –l [aA]* > ch2 :` insérer la liste des fichiers qui commencent par a ou A dans le fichier ch2 ;
`wc < ch2 :` retourne le nombre de lignes, de mots et de caractères de ch2 ;
`ls -l [aA]* | wc :` retourne le nombre de lignes, de mots et de caractères de la liste fournis par ls ;

Exercice 2 :

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
2) Editez les fichiers (sortie de la commande **ls** redirigée vers **grep**) avec les critères sur leur nom suivant:
   a) Critère 1 Le nom doit être Fraise ou fraise
   b) Critère 2 « se » est en fin de nom
   c) Critère 3 « ai »est présent dans le nom
   d) Critère 4 Nom contenant un chiffre numérique
3) En une seule commande supprimer votre répertoire courant (travail).

→ Résolutions :

```
Q1/ mkdir essai-grep
    touch tomate poire pomme cerise Fraise fraise courgette POMME3 afraise

Q2/ a/ ls | grep ^[Ff]raise
    b/ ls | grep se$
    c/ ls | grep ai
    d/ ls | grep [0-9]

Q3/ rm -Rf
```

Exercice 3 :

Le répertoire **/usr/include** contient les fichiers d'entête standards en langage C (stdlib.h, ...).

1) Créer un répertoire nommé inc dans votre répertoire de connexion (HOME).
   En utilisant une seule commande, y copier les fichiers du répertoire /usr/include dont le nom commence par std.
2) Afficher la liste des fichiers de /usr/include dont le nom commence par a, b ou c.
3) Modifier la commande de la question précédente pour qu'au lieu d'afficher le résultat, celui-ci soit placé dans un fichier nommé "Abc.list" de votre répertoire de connexion.
4) Afficher le contenu de ce fichier en utilisant la commande cat.
   Copier avec cat son contenu dans un nouveau fichier nommé "Copie".
5) Toujours avec cat, créer un nouveau fichier nommé "Double" formé par la mise bout à bout (concaténation) des fichiers "Abc.list" et "Copie".
   Vérifier que le nombre de lignes a bien doublé à l'aide de la commande wc.
6) Créer un fichier nommé "Temp" contenant une ligne de texte.
7) Avec cat, ajouter la ligne *"The end"* à la fin du fichier "Temp".

*) En une seule ligne de commande, faire afficher le nombre de fichiers de /usr/include dont le nom contient la lettre t.

→ Résolutions :

// Supposons qu'on se trouve dans le répertoire de connexion (HOME) ;

```
Q1/   mkdir inc
      cp /usr/include/std* inc
Q2/   ls /usr/include/[abc]*
Q3/   ls /usr/include/[abc]* > Abc.list
Q4/   cat Abc.list
      cat Abc.list > Copie

Q5/   cat Abc.list Copie > Double
      wc -l Abc.list
      wc -l Double
Q6/   cat > temp
      >>> Je suis un étudiant de l'EMSI (LOL)
      >>> ctrl-D
Q7/   cat >> temp
      >>> The end
      >>> (ctrl-D)
*)    ls -d /usr/include/*t* | wc -w (w pour compter le nombre de mots)
```

Exercice 4 :

1) Afficher la liste des répertoires de connexion des utilisateurs déclarés dans le fichier /etc/passwd.
2) On rappelle qu'à chaque utilisateur est associé un interpréteur de commandes (shell) lancé lors de son login. La commande correspondante est indiquée dans le 7ième champ du fichier /etc/passwd.
   Afficher en une ligne de commande le *nombre* d'interpréteurs de commandes différents mentionnés dans /etc/passwd.
3) On dispose d'un fichier texte **telephone.txt** contenant un petit carnet d'adresses.
   Chaque ligne est de la forme "nom : prenom : numerotelephone". Les champs sont séparés par « **:** ».
   Répondre aux questions suivantes en utilisant à chaque fois une ligne de commande shell:

   a) Afficher le carnet d'adresse trié par ordre alphabétique de noms.
   b) Afficher le nombre de personnes dans le répertoire.
   c) Afficher toutes les lignes concernant les "Hicham".
   d) Afficher toutes les lignes ne concernant pas les "Hicham".
   e) Afficher le numéro de téléphone (sans le nom) du premier " Hicham " apparaissant dans le répertoire.
   f) Afficher le numéro de téléphone (sans le nom) du premier " Hicham " dans l'ordre alphabétique

→ Résolutions :

Q1/ `cut -d : -f 6 /etc/passwd`

Q2/ `cut -d : -f 7 /etc/passwd | sort | uniq | wc -l`

Q3/

a/ `more` **`telephone.txt`** | sort

b/ grep -cv ^$ **`telephone.txt`**

c/ `more` **`telephone.txt`** | grep -w hicham

d/ `more` **`telephone.txt`** | grep -v -w hicham

e/ `more` **`telephone.txt`** | grep -w hicham | cut -d : -f 3 | head -n 1

f/ `more` **`telephone.txt`** | sort | grep -w hicham | cut -d : -f 3 | head -n 1

Partie 3 :

CHAP 5 : Commandes de l'éditeur vi

→ Part 3 – Commandes de l'éditeur vi :

**→ TP UNIX 5 :

VI est un éditeur de texte présent en standard sous UNIX. Il présente la particularité de posséder deux modes de travail :
Mode commande : dans lequel l'utilisateur spécifie les requêtes de traitements du fichier ;
Mode insertion : dans lequel tout ce qui est entré au clavier est écrit dans le tampon de mémoire associé au fichier.
Pour ouvrir un fichier existant ou pour créer un nouveau fichier, il suffit d'utiliser la syntaxe suivante : vi nom_fichier.
A l'appel de l'éditeur, on se trouve dans le mode commande. Plusieurs commandes d'insertion de texte permettent de passer en mode insertion, alors que pour passer du mode insertion au mode commande, on tape le caractère d'échappement ESC.

Les principales commandes de l'éditeur vi

1) Les commandes de déplacement

← ↑ ↓ → ou h, k, j, l	Déplacement dans les quatre directions.
0, ou ^	Déplacement en début de ligne.
$	Déplacement en fin de ligne.
ctrl-F	Déplacement à la page suivant.
ctrl-B	Déplacement à la page précédent.
H	Déplacement en début de page.
L	Déplacement sur la dernière ligne de page.
M	Déplacement en milieu de la page.
[n]G	Déplacement sur la $n^{ème}$ ligne du fichier. Par défaut, n est le numéro de la dernière ligne.
[[	Déplacement en début du fichier.
]]	Déplacement en fin du fichier.
w ou W (word)	Déplacement sur le début du mot suivant.
b ou B (Back)	Déplacement sur le début du mot précédent.
e ou E(End)	Déplacement sur la fin du mot courant.
(	Déplacement d'une phrase précédente.
)	Déplacement d'une phrase suivante.
{	Déplacement d'un paragraphe précédent.
}	Déplacement d'un paragraphe suivant.

2) Les commandes d'insertion

a	Insère du texte après le curseur.
A	Insère du texte en fin de ligne courante.
i	Insère du texte à l'emplacement du curseur.
I	Insère du texte en début de ligne courante.
o	Insère de ligne sous le curseur.
O	Insère de ligne au-dessus du curseur.

3) Les commandes de suppression du texte

[n]x	Supprime-le ou les caractères à partir du curseur.
[n]X	Supprime-le ou les caractères précédents le curseur.
D	Suppression de la chaîne de caractères comprise entre le caractère courant et la fin de la ligne.
[n]dw	Supprime-le ou les mots à partir du curseur.
[n]dd	Supprime-le ou les lignes à partir du curseur.
d$	Supprime à partir du curseur jusqu'à la fin de la ligne.
d^	Suppression du texte jusqu'au début de la ligne.
dG	Suppression du texte jusqu'à la fin document

4) Les commandes de remplacement du texte

R<car>	Substitue le caractère pointé par <car>.
R<chaîne>	Substitue le caractère pointé par <chaîne>.
<n>s<chaîne>	Substitue n caractère pointé par < chaîne >.
S<chaîne>	Substitue la ligne entière par < chaîne >.
cw<chaîne>	Remplace le mot courant par < chaîne >.
c$:	Modification de caractère jusqu'à la fin de la ligne.
c^:	Modification de caractère jusqu'au début de la ligne.
cG:	Modification de caractère jusqu'à la fin du document.

5) Les commandes de copier des lignes

yy :	Copie la ligne courante.
y$:	Copie de caractères jusqu'à la fin de la ligne.
y^ :	Copie de caractères jusqu'au début de la ligne.
yw :	Copie du mot courant.

Les lignes copiées ainsi que les lignes supprimées seront mises dans un tampon. Pour le restituer, il suffit de taper sur le caractère p (put).

6) Les commandes de Recherche

/expreg	Recherche une expression régulière à partir du curseur vers la fin du fichier.
? expreg	Recherche une expression régulière à partir du curseur vers le début du fichier.
n	Relance la recherche de l'expression régulière spécifier par les deux commandes ci-dessus dans le même sens.
N	Relance la recherche dans le sens inverse spécifié par les deux commandes ci-dessus.

6 – 1) Utilisation d'un caractère fixe :

/le : Recherche tous les mots qui contiennent la chaîne le.

6–2) Utilisation d'un caractère spécial dans un modèle de recherche :

/[Ss]e : Recherche toutes les occurrences de Se ou se dans le texte.

6–3) Signification des caractères spéciaux :

* : Remplace n'importe quelle chaîne de caractères.
. : un caractère est obligatoire à cet endroit..
^ : Début de la ligne.
$: Fin de la ligne.
[ ] : Remplace une liste ou une plage.
[^] : Remplace tous le caractères nom compris dans la liste ou plage.

6) Les commandes spéciales

:w <nom_fichier>	Sauvegarde le contenu du buffer dans le fichier dont le nom est donné comme argument.
:q	Sortir de vi. Si le contenu du buffer a été modifié, il faut dans ce cas le sauvegarder.
:q!	Sortir de vi sans sauvegarder le contenu du buffer.
:e<nom_fichier>	Edition du fichier <nom_fichier>, possible si le contenu du buffer n'a pas été modifié.
:e!<nom_fichier>	Edition du fichier <nom_fichier>, sans sauvegarder le contenu du buffer.
:r!<nom_fichier>	Insertion du fichier <nom_fichier> à partir de la ligne courante
:!<commande>	Exécution d'une commande Shell.

Exercice :

1) Ouvrer vi et tapé le texte suivant

Comment se connecter sous UNIX ?

Le premier concept important à garder en mémoire avant de travailler avec UNIX ou sa version PC Linux est qu'il s'agit d'un système multi-utilisateur. L'accès à la machine UNIX doit donc être contrôlé. Pour être enregistré sous UNIX, il faut avoir un compte utilisateur (login, password) créé par l'administrateur système. L'administrateur est un super utilisateur qui a les droits de gestion du système (login root ou su).

2) Enregistrer ce fichier sous le nom de fiche1

3) Ajouter le texte suivant :

login **:** (Taper votre nom d'utilisateur)
Le système affiche alors le message :

password : (Vous devez alors rentrer votre mot de passe)
Trouver un bon mot de passe : Voici quelques conseils qui vous permettront de trouver un mot de passe le plus sûr possible.

* utiliser des majuscules et des minuscules
* utiliser des chiffres et des caractères spéciaux
* 7 à 8 caractères de long
* concaténer des mots pour en créer un autre

4) Sauvegarder les modifications.
5) Supprimez la première ligne du fichier, puis enregistrez et quittez Vi.
6) Déplacez le curseur d'occurrence en occurrence de la chaîne de caractères 'le' ou 'la'.
7) Rechercher toutes les lignes qui commence par Le ou La.
8) Rechercher toutes les lignes qui contient des caractères écrits en majuscule.
9) Rechercher dans tout le texte les occurrences de 'utilisateur' et remplacer les par 'user'

Partie 4 :

CHAP 6 : Gestion des processus

→ Part 4 – Gestion des processus :

ps	Afficher les processus en cours d'exécution
-f	format complète
-e	chaque processus
-u …	appartenant à l'utilisateur …
-l	format longue

Commandes processus	
top	afficher avec remise à jour les processus en fonctionnement
commande&	lancer le processus de la commande en background
jobs	afficher les processus en background
kill %...	terminer un processus en background
fg %...	mettre une commande en avant plan
bg %...	mettre une commande en arrière-plan

pstree	Afficher les processus en arborescence
-p	afficher les PID des processus entre parenthèses
-h	mettre en surbrillance le processus actuel et ses ancêtres
-a	afficher les arguments de la commande des processus
-c	désactiver le regroupement des processus en une seule ligne
-u …	appartenant à l'utilisateur …

kill	Envoyer un signal à un processus <option-signal> <PID>
-l	afficher la liste des signaux
-15 ou -term	terminer un processus (SIGTERM)
-9 ou -kill	tuer un processus
-19	stopper un processus (SIGSTOP)
-18	reprendre l'exécution d'un processus suspendu
-2	interrompre le processus (SIGINT)

pkill	Envoyer un signal à un processus en fonction de propriétés <option-signal> <option-motif> <motif>
-u …	les processus appartenant à l'utilisateur …
-n	le processus le plus récent
-x …	les processus dont le nom est …
-G …	les processus appartenant à GUID …
-t …	les processus lancés dans le terminal …
-o …	l'ancien processus issu de la commande …

pgrep	Afficher les IDs des processus en fonction de certains critères
-l …	IDs des processus issu de la commande …
-t …	IDs des processus lancés dans le terminal …
-o …	ID de l'ancien processus issu de la commande …

nice -n [p] commande	lancer une commande avec une priorité ∈ [-20(max), 19(min)]

renice	Changer la priorité d'un processus -n [p] <option-motif> <motif>
-p …	priorité du processus de PID …
-u …	priorité des processus appartenant à l'utilisateur …

* * → Résolution du TP6 :

Exercice 1 :

1- Créer dans votre répertoire de connexion un nouveau répertoire nommé exo1.
2- Lancer l'exécution du programme top et vérifier qu'il fonctionne. L'arrêter en tapant CTRL-C.
3- En utilisant les fonctionnalités du shell (&, fg, bg), lancer quatre instances du programme top en même temps. Mettre au premier plan la troisième, l'arrêter (CTRL-Z) puis la relancer en arrière-plan.
4- A l'aide des commandes jobs et kill %n, arrêter tous les compteurs.
5- Même question en utilisant les commandes ps et kill (avec un PID).
6- Ouvrez une autre session et lancer une commande dont l'exécution dure longtemps (par exemple cat sans paramètres).
7- A partir de la première session, déterminer le numéro du processus (PID) correspondant à la commande lancée. Arrêter ce processus avec la commande Kill -9 PID ps. Interprétez les résultats.
8- Refaire la question 5, en utilisant les commandes : kill -15 PID, puis kill -2 PID.
9- Ecrire le script suivant (bonjour) avec un éditeur, puis l'ajouter le droit d'exécution.

$ **cat > bonjour**
#!/bin/bash
while true
do
echo bonjour
sleep 30
done
<Ctrl-D>
$ **chmod u+x bonjour**

Lancer ce script en arrière-plan : $ ./bonjour&

10- Tuer ce processus en utilisant son PID ou son numéro de processus.
11- Déconnectez-vous, et connectez-vous de nouveau. Affichez vos processus en tapant : $ ps. Le processus bonjour n'apparaît pas, pourquoi ?
12- Quelle commande devez-vous exécuter pour afficher le processus qui exécute bonjour.
13- Tuez le processus bonjour.

→ Résolutions :

Q1/ mkdir exo1
Q2/ top
Ctrl-C
Q3/ top & top & top & top & fg %3
Crtrl-Z
bg %3
Q4/ kill %1 %2 %3 %4
Q5/ ps (liste des processus avec leurs PID)
kill PID1 PID2 PID3 PID4
(avec PIDi les PID des processus top lancés)
Q6/ ctrl-shift-N
cat
Q7/ ps -a
Q8/ kill -15 PID(cat)
Q9/

$ **cat > bonjour**
#!/bin/bash
while true
do
echo bonjour
sleep 30
done
<Ctrl-D>
$ **chmod u+x bonjour**
$ ./bonjour&

Q10/ ps -a
kill -9 PID(bonjour)
Q11/ logout puis login
ps
Q12/ ps -f (chercher le PID du processus)
pstree -p PID(du processus)
Q13/ kill -9 bonjour

Exercice 2 :

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. Lancez une commande longue en arrière-plan. Quel est le comportement du processus associé lorsqu'il reçoit les signaux suivants :
    - Sigkill (9)
    - Sigstop(19)
    - Sigcont(18)
3. Utilisez la commande nice pour lancer des commandes ayant une faible priorité.
4. Interprétez la hiérarchie des processus qui vous appartiennent.
5. La commande ps | wc compte deux processus en plus de ceux qui existent réellement lorsqu'on lance la commande. Pourquoi ?
6. Donner deux commandes pour reprendre l'exécution d'une instruction interrompue par un ctrl-Z.

→ Résolutions :

Q1/ ps ou ps -t
ps -u fedora
ps -l
ps -f
[PID : Processus et PPID : Processus Parent du PID]

Q2/ cat&
Sigkill (9) : kill -9 PID(cat) => Processus cat tué
Sigstop(19) : kill -19 PID(cat) => Processus cat stoppé
Sigcont(18) : kill -18 PID(cat) => Reprise Processus cat

Q3/ nice -n +19 top

Q4/ pstree -u fedora

Q5/ ps | wc : compte le nombre de processus attachés au terminal + l'entête ;

Q6/ kill -18 PID
Fg %job

Partie 5 :

CHAP 7 : Programmation Shell

→ Part 5 – Programmation Shell :

Exécuter un script (le droit x est obligatoire)	
bash nomFichier	exécution directe du fichier du script
#!/bin/bash	ajouter à l'entête du fichier et l'exécuter par ./nomFichier

Variables prédéfinies	
HOME	répertoire de login
PATH	répertoire à inspecter pour trouver les commandes tapées par l'utilisateur
TERM	type du terminal
PWD	répertoire courant
USER	utilisateur courant
UID	ID de l'utilisateur courant
GID	ID du groupe de l'utilisateur courant

Variables de substitution	
$0	nom du script
$1 à $9	arguments passés au script
$#	nombre d'arguments passés au script
$?	résultat de la commande précédente
$@	liste des arguments passés au script
NomDeLaVariable=valeur	Déclaration initiation de la variable ou assignation de la variable
NomDeLaVariable=$variable Ou NomDeLaVariable=$(command) Ou NomDeLaVariable=`command`	Assigner la valeur d'une variable ou l'interprétation d'une commande ; (avec ` est [alt gr + 7])

echo	Afficher du texte sur la sortie standard
Ø	avec saut de ligne à la fin
-n	sans saut de ligne à la fin
$NomDeLaVariable	Afficher la variable
$(commande)	Interpréter la commende
`commande ou variable`	Interpréter la commende
expr	Evaluer une expression ⇔ $((…))

read	Lire à partir de l'entrée standard <nomVariable> (sans $)

if	case	while	foreach	for
if […] ; then … elif […] ; then … else … fi	case $var in valeur1) … ;; valeur2) … ;; *) … ;; esac	while […] do … done	for i in tableau do … done tableau : `seq 0 10` ou $@ ou chemin/*	for((i=0 ;i<$var ;i++)) do … done

testes sur les fichiers	
[-e chemin]	si le fichier existe
[-d chemin]	si le fichier est un répertoire
[-f chemin]	si le fichier est un fichier ordinaire
[-L chemin]	si le fichier est un lien symbolique
[-r chemin]	si le fichier est lisible (readable)
[-w chemin]	si le fichier est modifiable (writeable)
[-x chemin]	si le fichier est exécutable (executable)
[chemin1 -nt chemin2]	si le fichier1 est plus récent que le fichier2
[chemin1 -ot chemin2]	si le fichier1 est plus ancien que le fichier2

Testes sur les chaines de caractères	
[-z chaine]	si la chaine est vide
[-n chaine]	si la chaine est non vide
[chaine1 = chaine2]	si les 2 chaines sont égales
[chaine1 != chaine2]	si les 2 chaines sont différentes

testes sur les nombres	
[nb1 -eq nb2]	nb1 == nb2
[nb1 -ne nb2]	nb1 != nb2
[nb1 -lt nb2]	nb1 < nb2
[nb1 -le nb2]	nb1 <= nb2
[nb1 -gt nb2]	nb1 > nb2
[nb1 -ge nb2]	nb1 >= nb2

** → Résolution du TP7 :

Exercice 1 :

1. Ecrire un script shell qui donne le nombre de sous répertoire en partant du niveau courant.
2. Ecrire un script shell qui donne le nombre de sous répertoire contenus dans une arborescence dont le chemin sera donné en paramètre.
3. Ecrire un script shell qui donne le nombre de sous répertoire contenus dans une arborescence dont le chemin sera demandé à l'utilisateur.
4. Ecrire un script shell qui donne pour une arborescence donnée le nombre de fichiers exécutables, de fichiers accessibles en lecture, et de fichiers accessibles en écriture.
5. Faire un script qui crée un certain nombre de fichiers, ce nombre étant passé en paramètre (fich00, fich01, fich02, …).
6. Faire un script shell qui prend en paramètre le login d'un utilisateur local et qui en donne les informations suivantes :
    - Nom de l'utilisateur
    - Son groupe
    - Son programme initial (son shell)
    - Catalogue de travail
    - En utilisera le fichier /etc/passwd.

→ Résolutions :

Q1/

Méthode1 :

```
#!/bin/bash
cmp=0
echo "liste des sous-rep de `pwd` :"
for i in *
do
        if [ -d $i ]
        then
          cmp=`expr $cmp + 1`
        fi
        done
echo "$cmp "
```

Méthode2 :

```
#!/bin/bash

nbrRep=`ls -ld */ | wc -l`
echo "Le Nombre de sous Repertoire du dossier courant est $nbrRep"
```

Q2/
Méthode1 :

```
#!/bin/bash
cmp=0
echo "liste des sous-rep de $1 :"
for i in $1/*
do
        if [ -d $i ]
        then
          cmp=`expr $cmp + 1`
        fi
done
echo "$cmp "
```

Méthode2 :

```
#!/bin/bash

nbrRep=`ls -ld $1/*/ | wc -l`
echo "Le Nombre de sous Repertoire du dossier courant est $nbrRep"
```

Q3/
Méthode1 :

```
#!/bin/bash
cmp=0
read ch
echo "liste des rep sous $ch"
for i in $ch/*
do
        if [ -d $i ]
        then
          cmp=`expr $cmp + 1`
        fi
done
echo "$cmp "
```

Méthode2 :

```
#!/bin/bash

echo "Entrer le chemain :"
read ch
nbrRep=`ls -ld $ch/*/ | wc -l`
echo "Le Nombre de sous Repertoire du dossier courant est $nbrRep"
```

Q4/

```
#!/bin/bash
echo "Entrer le chemain :"
read ch
nbrLecture=`ls -lR $ch/ | cut -d ' ' -f 1 | grep -c r`
nbrExecutable=`ls -lR $ch/ | cut -d ' ' -f 1 | grep -c x`
nbrEcriture=`ls -lR $ch/ | cut -d ' ' -f 1 | grep -c w`
echo "Le Nombre de fichier en mode Lecture est $nbrLecture"
echo "Le Nombre de fichier en mode Ecriture est $nbrEcriture"
echo "Le Nombre de fichier en mode Executable est $nbrExecutable"
```

Q5/

```
#!/bin/bash

for i in `seq 0 $1`
do
        cpt=$i
        if [ $i -lt 10 ]
        then
             cpt="0$i"
        fi
        echo `touch fich$cpt`
done
```

Q6/

```
#!/bin/bash

ligne=`grep -w ^$1 /etc/passwd`

if [ -z $ligne ]
then
     echo "l'utilisateur $1 n'existe pas"
else

     echo "Nom de l'utilisateur : $1"
     echo "Nom du Groupe: `grep -w ^$1 /etc/passwd | cut -d : -f 4`"
     echo "Catalogue travail : `grep -w ^$1 /etc/passwd | cut -d : -f 6`"
     echo "Son PRG SHELL : `grep -w ^$1 /etc/passwd | cut -d : -f 7`"
fi
```

Exercice 2 :

1. Faites un script qui vous disent bonjour en affichant votre login («Bonjour, toto») :

    - Quand vous tapez `saluer [votre-login]`
    - Quand vous tapez juste `saluer`

2. Créez une commande qui, lorsqu'elle est appelée, renvoie le nombre d'arguments qui lui ont été fournis, ainsi que le premier de ces arguments.
3. Vous voulez écrire un script qui vous dit combien de personnes sont loguées sur une machine donnée;
4. Un ensemble de noms de fichiers sont en minuscules. On veut tout basculer en majuscules. (les noms des fichiers seront donnés en paramètre)
5. Écrivez un script qui prend en argument un nom de répertoire et qui détruit tous les fichiers :

    - Finissant par Tilda (~) ;
    - Commençant et finissant par un dièse (#) ;
    - S'appelant `core` ;
    - S'appelant `a.out` ;

→ Résolutions :

Q1/

```
#!/bin/bash



if [ -z $1 ]
then
            user=$(whoami) # ou bien directement $USER
            echo "Bonjour, $user"

else

            echo "Bonjour, $1"
fi
```

Q2/

```
#!/bin/bash


echo "Le nombre d'argument : $#"
if [ -z $1 ]
then
            echo "rien n'est passé en argument"

else
            echo "la valeur du premier paramètre est : $1"
fi
```

Q3/

```
#!/bin/bash
echo "Entrer le nom de la machine"
read machine
host_now=$(hostname)
if test $machine = $host_now
then
         cpt=`who | wc -l`

         if [ -z $cpt ]
         then
            echo "Personne n'est connectée"

         else
            echo "Le nombre de personnes connectées : $cpt"
         fi
else
         echo "La machine n'est pas connectee"

fi
```

Q4/

```
#!/bin/bash

for i in $@
do
         if [ -e $i ]
         then
            mv $i ` echo $i | tr '[:lower:]' '[:upper:]' `
         else
            echo "Le fichier $i n'existe pas"
         fi
done
```

Partie 6 :

Préparation : Tentation de Résolution Examen 2016

** → Résolution de l'Examen 2016 :

Exercice :

1. Écrire un script Shell qui affiche le login d'un utilisateur dont on fournit l'UID.
2. Ecrire un script qui lit une ligne de caractères sur son entrée standard et l'écrit sur sa sortie, en passant tous les caractères en majuscules.
3. Écrire une ligne de commande qui compte le nombre de processus actifs dont vous êtes propriétaire.
4. Ecrire un script qui permet de créer un processus qui affiche le message « Vous êtes à l'EMSI RABAT» à chaque deux minute.
5. Lancer ce script en arrière plan.
6. Repérer le numéro du processus.
7. Essayez d'interrompre l'exécution de ce script mais dans une autre session. Que faut-il changer?
8. Arrêter ce processus.
9. Écrire un script Shell dont le nom est **process** permettant de copier la liste des processus de l'utilisateur exécutant ce script, puis afficher le nom de chaque processus sans doublons.
10. Ecrire un script qui recherche dans toute mon arborescence tous les fichiers qui n'ont pas été accédés depuis un temps T et dont la taille est supérieure à MIN. (T et MIN sont des arguments).
11. Ecrire un script Shell qui ajoute à l'intérieur du fichier un commentaire indiquant sa date de dernière modification, dont le chemin du fichier sera donné en paramètre.
12. Écrire un script « hello » qui obéisse à la spécification suivante :
    - Sans argument, il affiche « Hello user ! » en remplaçant « user » par le login de l'utilisateur courant.
    - Avec un argument « A », il affiche « Hello A ! ».
    - Avec plusieurs arguments « A B C D », il affiche « Hello A, B, C and D ! »

    De plus, on demande que si le nom du script commence par «bonjour», le texte soit en français et pas en anglais. C'est-à-dire que le comportement doit être le suivant :

    Dans ce fichier **'le nom du fichier '**, vous avez :
    $./hello Ghita
    Hello Ghita !
    $./hello Ghita Sara Rania
    Hello Ghita, Sara, Rania!
    $ln –s hello bonjour
    $./bonjour Sara Rania
    Bonjour Sara, Rania !

13. Faire un script qui crée un certain nombre de répertoires et chaque répertoire possède un fichier, ce nombre étant passé en paramètre

14. Écrivez le shell script **killprog** qui permet d'envoyer le signal SIGKILL à un processus désigné non pas par son PID mais par son nom.
    Par exemple : killprog xterm

Bonus :

1. Quelle est la différence entre un « Shell » et un « Terminal »? Ou est-ce la même chose?
2. Comment consulter le manuel d'aide en ligne ?
3. Qu'est-ce qu'une variable prédéfinie?
4. Qu'est-ce qu'un Shell script ?
5. Qu'est-ce qu'une redirection ? Au lieu d'avoir ce que l'on veut sur l'écran, on l'a dans un fichier ?

→ Résolutions :

Q1/

```
user=`cat /etc/passwd |  grep -w $1 | cut -d : -f 1`

if [ -z $user ]
then
     echo "Cette UID ne figure pas dans la liste"
else
     echo $user
fi
```

Q2/

```
#!/bin/bash

echo "Entrer votre string : "
read ch

echo `echo $ch | tr [:lower:] [:upper:]`
```

Q3/

```
#!/bin/bash
cpt=`ps -f -u $USER | tail -n +2 |wc -l`
echo "Le nombre de vos processus actif  : $cpt"
```

Q4/

```
$ cat >> emsi_rabat
 #!/bin/bash

 while true
 do
      echo "Vous êtes a l'EMSI RABAT"
      sleep 120
 done
ctrl-D
$ chmod u+x emsi_rabat
```

Q5/ **$ ./emsi_rabat&**

Q6/ **ps**

Q7/ **ps -f**

kill -2 PID

Q8/ **kill -19 PID**

Q9/

$ cat >> process

```
#!/bin/bash
echo `ps -f -u $USER | tr -s " " | cut -d " " -f 8 |tail -n +2 | sort | uniq`
```

Q10/

```
#!/bin/bash
echo `find / -size +$1 -atime +$2`
```

Q11/

```
#!/bin/bash
if [ -e $1 ]
then
        datelm=`ls -l $1 | tr -s " " | cut -d " " -f 6,7,8`
        datelm="# $datelm"
        `echo $datelm >> $1`
else
        echo "Le fichier n'existe pas"
fi
```

Q12/

```
#!/bin/bash
cpt=1
if [ -z $1 ]
then
      var="Hello $USER"
else
      var="Hello "
      for i in $@
      do
            if [ $cpt -eq 1 ]
            then
                  var="$var $i"
            elif [ $cpt -lt $# ]
            then
                  var="$var, $i"
            elif [ $# -ge 3 ]
            then
                  var="$var and $i"
            else
                  var="$var ,$i"
            fi
            cpt=`expr $cpt + 1`
      done
fi
var="$var !"
echo $var
```