Centre de Tele-enseignement Universitaire
UNIVERSITÉ DE FRANCHE-COMTÉ

Correction Examen

Expliquez et détaillez les réponses aux questions de l'examen. Si cela est nécessaire, précisez les hypothèses que vous avez posées pour résoudre les exercices : il n'est pas possible de poser des questions pendant l'épreuve et il peut parfois exister plusieurs interprétations différentes pour un exercice. Le barème est donné à titre indicatif.

Exercice 1 : Cours *(3 points)*

Question 1 :

Expliquer comment est créé un processus dans Linux.

cours p 130

Question 2 :

Comment est-il possible de changer le shell de connexion d'un utilisateur ? L'utilisateur a-t-il accès à ce réglage ?

- *Le shell de connexion d'un utilisateur est défini dans /etc/passwd en dernière position de la ligne correspondant à l'utilisateur. Pour changer ce shell, il faut avoir les droits de root et éditer ce fichier*
- *Le shell de connexion d'un utilisateur peut aussi être modifié avec la commande :*
  `usermod -s userLogin`*. il faut avoir les droits de root pour exécuter cette commande.*

Question 3 :

Quel est le rôle du fichier `/etc/profile` ?

Le fichier /etc/profile est exécuté systématiquement à chaque connexion utilisateur. Il permet de définir l'environnement de l'utilisateur comme le positionnement de variables d'environnement (ex : PATH), le prompt, etc. , les commandes à lancer à la connexion. Il est commun à tous les utilisateurs.

Exercice 2 : Ecriture de commandes *(3 points)*

Question 1 :

Donner une commande qui copie tous les fichiers du répertoire d'accueil de l'utilisateur `Durand` dont le nom commence par "facture" dans le sous répertoire `Charge` de cet utilisateur. On suppose que le sous-répertoire `Charge` existe.

```
cp ~Durand/facture*.* ~Durand/Charge
```

Question 2 :

Donner une commande qui attribue les droits d'exécution pour les autres à tous les fichiers de l'utilisateur `Dupond` qui ont pour groupe "master".

```
find ~Dupond -group "master" -exec chmod o+x {}\;
```

ou

```
find / -group ''master'' -user ''Dupond'' -exec chmod o+x {}\;
```

Question 3 : Donner une commande qui affiche les utilisateurs dont le groupe principal a pour gid : 1022.

```
grep ''*.*.*.1022.*''  /etc/passwd | cut -f1 -d'':''
```

Question 4 : Donner une commande qui compte le nombre d'utilisateurs dont le groupe principal a pour gid : 1022.

```
grep ''*.*.*.1022.*'' /etc/passwd | wc -l
```

Question 5 : Donner une commande qui trie les lignes d'un fichier de type carnet d'adresses (de nom `carnet`) sur le champs ""nom" et affiche les 15 premières lignes triées. Nous supposons que les différents champs de chaque ligne du fichier `carnet` sont séparés par des espaces, que le premier champs de chaque ligne contient le nom et que le fichier `carnet` est stocké dans le répertoire courant.

```
sort -k1 carnet | head -n15
```

Remarque : l'option -k1 de la commande sort n'est pas obligatoire, car par défaut le tri s'effectue sur le premier champs.

Exercice 3 : Interprétation de script shell *(3 points)*

Commenter ligne par ligne le script bash ci-dessous (en utilisant les numéros de lignes), puis résumer sa fonction en quelques lignes. Pour vous aider dans votre démarche, vous proposerez un nom parlant pour chacune des variables utilisées.

```
1  function func2() {
2    if [ -d $1 ]; then
3      TD=$(($TD + 1))
4
5      for f in $1/*; do
6        if [ -e $f ]; then
7          func2 $f
8        fi
9      done
10   elif [ -x $1 ]; then
11     TE=$(($TE + 1))
12   else
13     TF=$(($TF + 1))
14   fi
15 }
16
17 D=$1; TD=0; TF=0; TE=0
18
19 func2 $D
20
21 echo "TD : $TD"
22 echo "TE : $TE"
23 echo "TF : $TF"
```

Correction interprétation du script :

```
1 function func2() {
2 #Si le parametre de la fonction est un repertoire
3   if [ -d $1 ]; then
4
5     TD=$((TD + 1))
6    # TD est incremente de 1
7    # Pour chaque fichier contenu dans le repertoire passe en
8     #parametre, appel de la fonction fonc2 avec ce fichier comme parametre
9     for f in $1/*; do
10      if [ -e $f ]; then
11        func2 $f
12      fi
13    done
14  #sinon si le fichier est un executable, incrementation de la
15  #variable TE
16   elif [ -x $f ]; then
17     TE=$((TE + 1))
18 # sinon incrementation de la variable TF
19   else
20     TF=$((TF + 1))
21   fi
22 }
23
24 #initialisation des variables
25 D=$1; TD=0; TF=0; TE=0
26
27 #appel de la fonction func2 avec comme parametre le premier parametre
28 #du script
29 func2 $D
30
31 #affichage des valeurs des variables: nombre de répertoires,
32 #d'exécutables, de fichiers
33 echo "TD : $TD"
34 echo "TE : $TE"
35 echo "TF : $TF"
```

Exercice 4 : Ecriture de Script shell *(6 points)*

Dans cet exercice, nous vous demandons d'écrire des scripts shell permettant de gérer une "corbeille" depuis un terminal.

Question 1 : Nettoyage de la corbeille

Écrire le shell script `~/Corbeille/nettoyage.sh` qui va effacer de la corbeille tous les fichiers qui ont plus de 30 jours.
Ce script doit :

1. créer le répertoire `~/Corbeille/fichiers/` si il n'existe pas encore,
2. effacer tout les fichiers du répertoire `~/Corbeille/fichiers/` qui ont plus de 30 jours.

```
1  #!/bin/bash
2
3  # possibilité avec test si non connaissance de l'option -p
4  # création du répertoire s'il n'existe pas (option -p) de mkdir
5  mkdir -p ~/Corbeille/fichiers/
6
7  # recherche des fichiers de plus de 30 jours (-mtime +30)
8  # et effacement de chacun des fichiers trouvé (-exec rm {} \;)
9  find ~/Corbeille/fichiers -mtime +30 -exec rm {} \;
```

Question 2 : Déplacement d'un fichier dans la corbeille

Écrire le script shell `~/Corbeille/corbeille.sh` qui prend un fichier comme paramètre et déplace ce fichier dans la corbeille en modifiant son nom (préfixé de la date de déplacement).

Ce script prend en paramètre un nom de fichier (avec son chemin d'accès) et réalise les actions suivantes :

1. si le nombre de paramètres est différent de 1 : affichage d'une erreur et `exit 1`
2. si le paramètre n'est pas un fichier : affichage d'une erreur et `exit 2`
3. si le fichier passé en paramètre est présent dans le répertoire `~/Corbeille/fichiers/` : affichage d'une erreur et `exit 3`
4. appel du script `~/Corbeille/nettoyage.sh`
5. création du nom du fichier dans la corbeille. Le nouveau nom de fichier est composé de la manière suivante : `date-nomFichier` où

- `date` est remplacé par la date courante au format : année-mois-jour-heure-minute-seconde-nanoseconde
- `nomFichier` est remplacé par le nom du fichier sans son chemin.

6. déplacement du fichier passé en paramètre dans la corbeille en lui attribuant le nom créé précédemment.

NB : nous considérons qu'il est impossible que deux appels au script `corbeille.sh` aient lieu dans la même nanoseconde avec le même nom de fichier.

```
1  #!/bin/bash
2
3  # Si le nombre de paramètres est différent de 1
4  # sortie du script : erreur 1
5  if [ "$#" -ne 1 ]; then
6      echo -e "Usage:\n\t$0 FICHIER_A_EFFACER"
7      exit 1
8  fi
9
10 # initialisation de la variable SRC avec le paramètre du script
11 SRC=$1
12
13 # Si SRC n'est pas un fichier
14 # sortie du script : erreur 2
15 if [ ! -f "$SRC" ]; then
16     echo "\"$SRC\" n'est pas un fichier"
17     exit 2
18 fi
19
20 # SRC_DIR = chemin du fichier SRC (sans le nom)
21 SRC_DIR=$(dirname $SRC)
22
23 # teste si le fichier est déjà présent dans la corbeille
24 # si le répertoire passé en paramètre SRC_DIR est le même que
25 # le répertoire corbeille
26 if [[ $SRC_DIR -ef ~/Corbeille/fichiers/ ]]; then
27     echo "Il n'est pas possible d'utiliser la corbeille pour les \
28     fichiers dans \"~/Corbeille/fichiers/\""
29     exit 3
30 fi
31
32 # exécution du script nettoyage.sh :
33 # effacement des fichiers de plus de 30 jours
34 /bin/bash ~/Corbeille/nettoyage.sh
35
36 # variable DATE initialisée avec la date courante format long
37 DATE=$(date +"%Y-%m-%d-%H-%M-%S-%N")
38
39 # variable SRC_NAME initialisée avec le nom du fichier SRC sans le chemin
40 SRC_NAME=$basename($SRC)
41
42 # initialisation du nom de fichier contenant la date courante
43 DST="$HOME/Corbeille/fichiers/$DATE-$SRC_NAME"
44
45 # déplacement du fichier passé en paramètre dans la corbeille
46 mv "$SRC" "$DST"
```

Question 3 : Liste des fichiers placés dans la corbeille

Écrire un script shell `~/Corbeille/liste.sh` qui affiche les fichiers de la corbeille avec un format défini.
Le script shell réalise les actions suivantes :

1. appel du script `~/Corbeille/nettoyage.sh`
2. affichage de la liste des fichiers contenus dans la corbeille triés par ordre décroissant du moment de la sauvegarde.
   Les fichiers sont affichés de la façon suivante : jour/mois/année nomFichier, ces informations faisant partie du nom complet du fichier stocké dans la corbeille.

```
1   #!/bin/bash
2
3 # exécution du script nettoyage.sh :
4 # effacement des fichiers de plus de 30 jours
5   /bin/bash ~/Corbeille/nettoyage.sh
6
7 # ls ~/Corbeille/fichiers/ | sort -r :
8 # liste les fichiers contenus dans le répertoire
9 # ~/Corbeille/fichiers/ triés par ordre décroissant
10 # de date (donc de sauvegarde)
11 # pour chaque ligne (fichier : f) résultat de cette commande
12 # le traitement suivant est réalisé
13 # extraction des informations nécessaires
14 # contenues dans le nom du fichier et affichage
15   for f in $(ls ~/Corbeille/fichiers/ | sort -r); do
16       jour=$(echo $f | cut -f3 -d''-'')
17       mois=$(echo $f | cut -f2 -d''-'')
18       annee=$(echo $f | cut -f1 -d''-'')
19       nom=$(echo $f | cut -f8 -d''-'')
20
21      # affichage de chaque fichier au format demandé
22       echo "$jour/$mois/$annee $nom"
23 done
```

Exercice 5 : Mise en place du réseau d'un institut de recherche *(5 points)*

Un institut de recherche de pointe constitué de plusieurs laboratoires souhaite mettre en place un réseau reliant tout ses personnels aux services informatiques communs :

- Un centre de calcul informatique, dédié à la simulation des modèles des chercheurs
- Les outils administratifs : Un serveur de messagerie, un serveur de sortie sur Internet, un serveur de gestion des recherches.
- Trois serveurs de stockage en réseau, pour les documents et les fichiers de données.

Cet institut de recherche est constitué de trois départements, comprenant les nombres de chercheurs suivants (On supposera qu'il existe une machine connectée au réseau par chercheur) :

- Département robotique : 31 chercheurs.
- Département aérospatial : 240 chercheurs.
- Département informatique : 63 chercheurs.

La solution proposée doit respecter les critères suivants :

- Les services informatiques communs devront être situés sur un même réseau sécurisé
- Les autres réseaux devront être configurés pour utiliser le serveur de sortie Internet pour communiquer avec l'extérieur.
- La connexion Internet sera simplement désignée comme `AccesInternet1`. Sa configuration IP relève du fournisseur d'accès et sort du cadre de cet exercice.
- L'ensemble des réseaux entre les départements de l'institut de recherche devront passer par un unique équipement, de manière à pouvoir surveiller les échanges de données sensibles.
- On n'oubliera pas de prendre en compte les réseaux nécessaires à la communication entre les sous-réseaux.
- Dans un souci de clareté, on désignera dans les réponses les sous-réseaux proposés sous les noms de `SR1`, `SR2`, `SR3` ...
- De manière à réserver un nombre minimal d'adresses, on effectuera un découpage au plus juste des adresses réseaux.

Questions

Comme base de découpage, on propose l'une des trois plages réseaux suivantes : `12.0.0.0`, `164.20.0.0` et `194.42.10.0`.

1. Qu'est-ce que ces trois plages réseaux ont en commun ? Que cela implique-t-il en terme d'utilisation ?

   Ces trois plages réseaux sont toutes publiques, c'est à dire adressable sur Internet. Pour pouvoir les utiliser, il est nécessaire d'obtenir une autorisation de l'APNIC. L'unicité de ces adresses est garantie, ce qui permet de les utiliser directement en sortie du réseau si nécessaire.

2. À quelle classe chacune de ces plages appartient-elle ?

   Selon le découpage en classes, 12.0.0.0 correspond à un réseau de classe A. En y ajoutant le masque associé à cette classe, on obtient le réseau suivant en notation CIDR : 12.0.0.0/8.

   164.20.0.0 correspond à une adresse réseau de classe B, ce qui donne le notation CIDR suivante : 164.20.0.0/16.

   Enfin, le réseau 194.42.10.0 appartient à la classe C, soit en notation CIDR : 194.42.10.0/24.

3. Quelle plage sera-t-il préférable de retenir pour notre solution ? Justifier.

   Pour sélectionner une plage, il est nécessaire de déterminer le nombre d'adresses nécessaires pour réaliser le découpage prévu dans l'exercice.

   A partir des données de l'énoncé, il est nécessaire de pouvoir adresser au moins 31 (département robotique) + 240 (département aérospatial) + 63 (département informatique) = 334 machines clientes.

   Cela exclut le réseau 194.42.10.0/24, de classe C, qui ne permet d'adresser que 254 machines. En prévoyant trois entrées pour le centre de calcul, de manière conservatrice, les services communs informatiques y ajoutent 3 (centre de calcul) + 3 (outils administratifs) + 3 (serveurs de stockage réseau) soit 9 machines à adresser.

   Ce chiffre de 343 machines reste très inférieur à celui permis par le réseau 164.20.0.0/16, de classe B (plus de 60000) et ne nécessite pas l'utilisation du réseau 12.0.0.0, de classe A, dans l'optique d'un découpage au plus juste.

   On retient donc pour plage de découpage 164.20.0.0/16, de classe B.

4. Combien de sous-réseaux seront-ils nécessaires, en comptant les interconnexions entre bâtiments ? Justifier.

Pour ce découpage, un sous-réseau par bâtiment est nécessaire (performance, limite la quantité d'interconnexion à réaliser et permet le passage par un seul équipement), soit 4 sous-réseaux.

En plaçant un routeur par bâtiment, il faut y ajouter 3 sous-réseaux d'interconnexion. Au total, il nous faudra donc 7 sous-réseaux.

5. Quelle sera la taille de chaque sous-réseau ? Préciser votre méthode de calcul.

   Services communs informatiques

   Pour déterminer la taille minimale (découpage au plus juste) de sous-réseau à utiliser pour les 9 machines clientes prévues, il convient de prendre la puissance de deux immédiatement supérieure, sans oublier de prévoir deux adresses réservées supplémentaires (réseau et diffusion), ainsi qu'une adresse pour la passerelle.

   On retient donc un réseau de taille 16 (2^4).

   Département robotique

   En suivant le même raisonnement, 31 + 2 + 1 = 34 adresses sont nécessaires. La puissance de 2 immédiatement supérieure est 2^6, ce qui donne un sous-réseau de taille 64.

   Département aérospatial

   240 + 2 + 1 = 243 adresses nécessaires. On retient donc un sous-réseau de taille 256 (2^8).

   Département informatique

   63 + 2 + 1 = 66 adresses nécessaires. On retient donc un sous-réseau de taille 128 (2^7).

   Réseaux d'interconnexion

   Ces réseaux n'ayant pas de passerelle séparée (réseau routeur-routeur), on ne prévoit que les deux adresses réservées en supplément.

   2 + 2 = 4 adresses. On retient donc un sous-réseau de taille 4 (2^2).

6. Effectuer le découpage du réseau privé retenu à la troisième question de l'exercice :
   — Détailler l'ensemble de la procédure utilisée.
   — Donner la liste des sous-réseaux obtenus et leurs paramètres : adresse de réseau, masque en décimal et en notation CIDR, nombre d'adresses, nombre total de machines, adresse de la passerelle, adresse de broadcast, dernière adresse utilisée.

De manière à éviter des problèmes d'alignement de sous-réseaux, on choisit de découper la plage d'adresse en sous-réseaux par ordre de taille décroissante.

Remarque : pour les nombres d'adresse et de machine, se reporter à la question précédente.

SR1 (Aérospatial, 240 postes)

```
Adresse réseau : 164.20.0.0/24 (32 - 8 = 24)
Masque réseau  : 255.255.255.0
Passerelle     : 164.20.0.254
Broadcast      : 164.20.0.255
Dernier poste  : 164.20.0.240
```

SR2 (Informatique, 63 postes)

```
Adresse réseau : 164.20.1.0/25 (32 - 7 = 25)
Masque réseau  : 255.255.255.128
Passerelle     : 164.20.1.126
Broadcast      : 164.20.1.127
Dernier poste  : 164.20.1.63
```

SR3 (Robotique, 31 postes)

```
Adresse réseau : 164.20.1.128/26 (32 - 6 = 26)
Masque réseau  : 255.255.255.192
Passerelle     : 164.20.1.190
Broadcast      : 164.20.1.191
Dernier poste  : 164.20.1.159
```

SR4 (Services Informatiques Communs, 9 machines)

```
Adresse réseau : 164.20.1.192/28 (32 - 4 = 28)
Masque réseau  : 255.255.255.240
Passerelle     : 164.20.1.206
Broadcast      : 164.20.1.207
Dernier poste  : 164.20.1.201
```

SR5 (Réseau d'interconnexion, 2 machines)

```
Adresse réseau : 164.20.1.208/30 (32 - 2 = 30)
Masque réseau  : 255.255.255.252
Passerelle     : Aucune (interconnexion)
```

```
Broadcast      : 164.20.1.211
Dernier poste  : 164.20.1.210
```

SR6 (Réseau d'interconnexion, 2 machines)

```
Adresse réseau : 164.20.1.212/30 (32 - 2 = 30)
Masque réseau  : 255.255.255.252
Passerelle     : Aucune (interconnexion)
Broadcast      : 164.20.1.215
Dernier poste  : 164.20.1.214
```

SR7 (Réseau d'interconnexion, 2 machines)

```
Adresse réseau : 164.20.1.216/30 (32 - 2 = 30)
Masque réseau  : 255.255.255.252
Passerelle     : Aucune (interconnexion)
Broadcast      : 164.20.1.219
Dernier poste  : 164.20.1.218
```

Annexes SR

Options de la commande test :

option	paramètres	description
-e	fichier	vrai si fichier existe
-w	fichier	vrai si fichier existe et est autorisé en écriture
-r	fichier	vrai si fichier existe et est autorisé en lecture
-x	fichier	vrai si fichier existe et exécutable
-d	fichier	vrai si fichier existe et est un répertoire
-f	fichier	vrai si fichier existe et n'est pas un répertoire
-s	fichier	vrai si fichier existe et a une taille non nulle
-t	descripteur	vrai si descripteur est associé à un terminal

TABLE 1 – Test sur fichiers et les répertoires

option	paramètres	description
-z	chaine	vrai si chaine est vide
s1 = s2	s1, s2	vrai si s1 et s2 sont identiques
s1 != s2	s1, s2	vrai si s1 et s2 sont différentes

TABLE 2 – Test sur les chaînes de caractères

option	paramètres	description
n1 -eq n2	n1, n2	vrai n1 = n2
n1 -ne n2	n1, n2	vrai n1 != n2
n1 -gt n2	n1, n2	vrai n1 > n2
n1 -lt n2	n1, n2	vrai n1 < n2
n1 -ge n2	n1, n2	vrai n1 >= n2
n1 -le n2	n1, n2	vrai n1 <= n2

TABLE 3 – Test sur les nombres

Exemple de contenu d'un fichier `/etc/passwd`

```
alain::110:500:Alain Terrieur:/home/alain:/bin/bash
paul::120:500:Paul Hissont:/home/paul:/bin/zsh
```

Exemple de contenu d'un fichier `/etc/group`

```
etu::500:alain,paul
pro::510:alain,paul
exp::520:alain
```

Exemple de résultat de la commande `who`

```
user    :0       Dec 5  18:41
alain   pst/2    Nov 10 20:52
```

Commandes	**Description**	**Exemples**
at	exécution d'une commande à l'heure donnée	at 1023
basename	affiche le nom d'un fichier sans son chemin	basename fich
cal	donne le calendrier	cal 2019
cat	affiche le contenu d'un fichier -n : précède chaque ligne par son numéro	cat fich cat -n fich
cd	changement de répertoire	cd rep
chmod	change les droits d'accès d'un fichier -R : changement récursifs aux sous-rép.	chmod u+x fich chmod -R a+x rep
comm	affiche les lignes communes à deux fichiers	comm fich1 fich2
cmp	compare deux fichiers et retourne le code 0 s'ils sont identiques	cmp fich1 fich2
cp	recopie un fichier -r : recopie de répertoire -i : demande avant d'effectuer une recopie	cp path1 path2 cp -r rep1 rep2 cp -i fich1 fich2
cut	affiche une partie de chaque ligne d'un fichier -c : par numéros de colonnes (1 à 3 et 8) -f : par champs séparés par des délimiteurs -d*c* : où *c* donne le caractère délimiteur	 cut -c1-3,8 fich cut -f1,2 fich cut -f1 -d. fich
date	affiche la date courante (format long) Ex : format année-mois-jour-heure-mn-s-ns est donné par : `date +''%Y-%m-%d-%H-%M-%S-%N"`	date
df	donne l'espace libre dans les systèmes de fichiers	df
diff	donne les différences entre deux fichiers -i : majuscules = minuscules	diff fich1 fich2
dirname	affiche le chemin d'accès d'un fichier	dirname fich
du	donne le nombre de blocs disques utilisés par un fichier ou un répertoire	 du rep
echo	affiche une chaîne -n : affiche la chaîne sans nouvelle ligne	echo blabla echo -n blabla
find	recherche d'un fichier dans l'arborescence à partir du chemin *path* *-print* : affiche le résultat *-user* : nom du propriétaire *-group* : nom du groupe *-size* : taille en blocs *-exec* : commande à exécuter pour chaque fichier trouvé *-atime n* : nb jours depuis dernière consultation *-ctime n* : nb jours depuis création *-mtime n* : nb jours depuis dernière modification *Ex :* Suppression des fichiers de nom commençant par test ou de nom essai, créés aujourd'hui `find . \(-name "test*" -o -name essai\) -ctime 0 -exec rm {}\;`	find path -name fich1 find path -mtime 7
grep	recherche une chaîne dans un fichier -i : ignore les différences entre maj. et min. -n : numérote les lignes -v : sélectionne les lignes ne contenant pas chaîne -w : sélectionne les lignes où chaîne est un mot	grep chaine fich grep -i chaine fich
gzip gunzip	compresse un fichier décompresse un fichier compressé par gzip	gzip fich1 gunzip fich1.gz
head	affiche les premières lignes d'un fichier	head -n5 fich
hostname	donne le nom de la machine courante	
kill	tue un processus	kill pid
ls	liste le contenu du répertoire -a : affiche aussi les fichiers commençants par . -R : affiche aussi le contenu des sous-répertoires -i : donne le numéro d'inode	ls rep ls -a ls -R ls -i fich

mkdir	création d'un sous-répertoire	mkdir rep
more	affiche un texte page par page	more fich
mv	renomme un fichier ou répertoire	mv fich1 fich2
Commandes	**Description**	**Exemples**
nl	numérote les lignes d'un fichier	nl fich
nice	modifie la priorité d'un processus	nice +1 prgm
passwd	permet de changer le mot de passe (yppasswd)	passwd user
ps	liste les processus -a ou -e : donne la liste de tous les processus -u : donne le nom des propriétaires -o col1,col2,... : affiche uniquement les colonnes spécifiées	
pwd	affiche le nom du répertoire courant	
rm	efface un fichier -r : efface un répertoire (ou rmdir si répertoire vide) -f : force en cas de droits d'accès limités -i : demande confirmation	rm fich rm -r rep rm -rf rep rm -ri rep
seq	affiche la séquence de nombres demandée, séparés par un espace	seq 4 10
set	sans option ni argument : liste les variables du shell permet de positionner les paramètres positionnels Ex : *$1* prend la valeur a, *$2* la valeur b et *$3* la valeur c	 set a b c
sleep	attente d'un délais	sleep 10
sort	tri d'un fichier, basé sur les codes ASCII -r : tri inversé -d : tri uniquement sur les caractères, chiffres et espaces -i : ne trie que les codes ASCII entre 32 et 126 -f : sans différence entre min. et maj.	sort fich
tail	donne les dernières lignes d'un fichier	tail -n 10 fich
tar	permet de combiner plusieurs fichiers en une archive, sans compression -c crée une archive -A concatène à une archive existante	tar -c -f archive.tar fich1 fich2
tee	écrit sur un fichier et sur l'écran dans une suite de pipe	cat fich1 |tee fich2
time	donne les temps d'exécution d'une commande	time du -s
tr	transformation sur l'entrée standard -d : efface les caractères donnés	tr ch1 ch2 tr -d xyw
tty	donne l'identificateur du terminal	tty
wc	donne le nombre d'octets, mots et lignes d'un fichier -l : uniquement le nombre de lignes -w : uniquement le nombre de mots -c : uniquement le nombre d'octets	wc fich wc -lc fich
who	donne les utilisateurs connectés login terminal date de début de connexion	
whoami	donne le nom de l'utilisateur courant	
;	séparateur de commandes	date ; who
(commande)	séquence de commandes qui n'affectent pas le niveau courant	(cd /etc ; ls)
<	redirection de l'entrée depuis un fichier	read a < fich
>	redirection de la sortie sur un fichier	date > sortie
>>	ajoute la sortie à la fin du fichier	
&	exécute une commande en arrière plan	

Synthèse des différences d'interprétation des caractères spéciaux :

	Génération de noms de fichiers	**Expressions régulières**
?	un caractère quelconque, sauf <new line>	remplace 0 ou 1 fois le caractère qui précède
.	le caractère point	un caractère quelconque sauf <new line>
*	0 ou un nombre quelconque de caractères	remplace 0 fois ou n fois le caractère qui précède
`[a-i]`	un caractère entre a et i	un caractère entre a et i
`[!a-i]`	un caractère qui n'est pas entre a et i	un caractère entre a et i, ou **!**
`[^a-i]`	un caractère qui n'est pas entre a et i	un caractère qui n'est pas entre a et i
\	banalise le caractère qui suit	banalise le caractère qui suit
^	le caractère ^	ce qui suit est en début de ligne
$	référence une variable	ce qui précède est en fin de ligne