Formation

Notre expertise est votre avenir

# Travaux pratiques

# Unix / Linux - Programmation Shell

# Avertissement

Les énoncés de TP qui suivent permettent aux stagiaires de s'approprier les informations vues dans le chapitre correspondant. Ces TP peuvent être personnalisés et/ou complétés par le formateur.
La syntaxe des commandes à programmer est à considérer comme le cahier des charges qu'il est demandé de suivre.
Certains exercices peuvent nécessiter d'utiliser des fichiers particuliers, ces fichiers seront fournis par le formateur.

# SOMMAIRE

# Rappels du shell

1. Listez votre répertoire de connexions en format long, en affichant les fichiers cachés, le fichier le plus récent en fin de liste.

```
ls –al $HOME
```

2. Affichez le type de contenu du fichier /usr/share/doc/vim-common-7.4.629/LICENSE.
   Pouvez-vous afficher son contenu ?
   Si possible, affichez son contenu page à page.

```
File /usr/share/doc/vim-common-7.4.629/LICENSE
Less /usr/share/doc/vim-common-7.4.629/LICENSE
```

3. Listez, en format long, toutes les commandes de /bin commençant par les lettre "ls".

```
ls –l /bin/ls*
```

4. Dans votre répertoire d'accueil, créez un répertoire en lui positionnant le sticky-bit.

```
mkdir –m +t ~/rep1
```

5. Saisissez un fichier nommé prog1 dont le contenu est le suivant :

```
echo "Bonjour"
echo "Comment ça va ?"
```

Rendez ce fichier exécutable et lancez ce shell script.

```
Cat >>prog1
echo "Bonjour"
echo "Comment ça va?"
Ctl-D
chmod +x prog1
```

6. Recherchez dans le répertoire /etc , un fichier nommé vimrc. Débarrassez-vous des messages d'erreur et récupérez le résultat dans un fichier de votre répertoire personnel.

```
find /etc –name vimrc >> ~/result 2>/dev/null
```

7. Affichez le contenu de la variable LANG de votre session.

```
echo $LANG
```

# La ligne de commandes

1. En une seule ligne de commandes, dans votre répertoire d'accueil, listez le répertoire rep1, si ce répertoire n'existe pas, créez-le, puis créez trois fichiers dans ce répertoire, et enfin affichez son contenu en format long.
   `ls –l rep1 || mkdir rep1; touch /rep1/fic1 rep1/fic2 rep1/fic3; ls –l rep1`

2. En une seule ligne de commandes, dans votre répertoire d'accueil, créez le répertoire rep2 et dedans trois fichiers nommés fic1.txt, fic2.txt et fic3.txt.
   Changez les droits d'accès afin que seul le propriétaire ait uniquement le droit de lecture sur ces fichiers.
   `mkdir rep2; touch rep2/fic1.txt rep2/fic2.txt fic3.txt`

3. Écrivez un script appelé : copier, selon les consignes suivantes :
   COMMANDE : `$ copier fichier répertoire suffixe`

ENONCE :
Ecrivez un shell script nommé : copier.
Ce script doit permettre de copier un fichier ordinaire du répertoire courant vers un répertoire de sauvegarde en ajoutant au nom un suffixe (ex. : .save).
Les droits sur le fichier nouvellement créé sont modifiés afin que seul le propriétaire ait le droit de lecture (et rien d'autre !).
REMARQUE : utilisation de cp, chmod et des paramètres.
#copier.sh
Cp $1 $2/$1.$3
Chmod 400 $2/$1.$3

4. Écrivez un script appelé dupliquer selon les consignes suivantes :
   COMMANDE : `$ dupliquer`

ENONCE :
Ecrivez un shell script nommé : dupliquer.
Ce script aura les mêmes fonctions que le script précédent mais le nom du fichier, celui du répertoire et le suffixe seront donnés de manière interactive.
REMARQUE : utilisation de cp, chmod, read et des variables.

```
#dupliquer.sh

echo –e "veuillez saisir le nom du fichier :\c"
read vfichier
echo –e "veuillez saisir le nom du répertoire :\c"
read vrep
echo –e "veuillez saisir le suffixe :\c"
read vsuffixe

cp $vfichier $vrep/$vfichier.$vsuffixe
chmod 400 $vrep/$vfichier.$vsuffixe
```

5. Modifiez copier / dupliquer afin qu'ils fonctionnent correctement si on entre des chemins d'accès absolus pour le nom du fichier et le nom du répertoire.
   REMARQUE : utilisation des formes avancées des variables ${var...}.

```
#copier.sh
fic=${1#$HOME/}   #On recupère le nom du fichier sans le chemin d'accès
cp $fic $2/$fic.$3
chmod 400 $2/$fic.$3
```

```
#dupliquer - Modifie

echo –e "veuillez saisir le nom du fichier :\c"
read vfichier
echo –e "veuillez saisir le nom du répertoire :\c"
read vrep
echo –e "veuillez saisir le suffixe :\c"
read vsuffixe
vfichier=${vfichier#$HOME/}
cp $vfichier $vrep/$vfichier.$vsuffixe
chmod 400 $vrep/$vfichier.$vsuffixe
```

6. Affichez le contenu de la variable ville, si cette variable n'existe pas, l'affichage sera "Paris". Créez la variable ville avec le contenu de votre choix, et relancez la commande précédente.

```
echo ${ville-Paris}
```

7. Configurez votre shell pour que la casse des caractères soit ignorée lors du développement des noms de fichiers. Vérifiez en créant quelques fichiers de même nom mais avec minuscules et majuscules.
   A la fin de l'essai, revalidez la prise en compte de la casse des caractères.

```
shopt –s nocaseglob
```

8. Écrivez une ligne de commande permettant d'afficher le message suivant :

```
******************
*  Bonjour user1 *
*  il est 13h19  *
******************
```

```
echo "Bonjour $(whoami)"
echo "$(date '+ il est %Hh%M')"
```

Le nom correspond à votre nom d'utilisateur, et l'heure réelle s'affiche.

Si besoin, utilisez le man pour vous familiariser avec les syntaxes de la commande date.

# Gestion des fichiers

1. Dans votre répertoire d'accueil, créez un lien symbolique rep_etc vers /tmp .
   Exécutez les commandes suivantes et observez les résultats :

```
$ cd rep_tmp
$ pwd
$ pwd -P
$ cd
$ cd -P rep_tmp
$ pwd
$ pwd -P
```

2. Sous / quels sont les fichiers dont le numéro d'inode est égal à 2 (ext4) ? si ça ne marche pas essayez avec 128 (xfs).

```
find / -name inum 2
```

```
/home
/boot
find: `/run/user/1000/gvfs': Permission non accordée
/dev/pts/ptmx
/proc/sys/fs/binfmt_misc/status
/sys/fs
```

3. Créez un répertoire dans votre répertoire d'accueil ; puis, en mode conversationnel, recherchez et copiez une partie des fichiers du répertoire /etc se terminant par .conf .

```
mkdir rep
find /etc –name '*.conf' –ok cp {} rep \;
```

4. Écrivez une commande find, en mode conversationnel, qui affichera soit les caractéristiques (ls -l) soit le contenu (cat) de chaque fichier ordinaire à partir de votre répertoire d'accueil, ceci en fonction de la réponse utilisateur.

```
find . –type f \( –ok ls -l {} rep –o exec cat {}\; \)
```

5. Quelles sont les commandes de /usr/bin dont l'indicateur SUID est positionné ?

```
$ find /usr/bin -perm +4000 -exec ls -l {} \;
```

# Fonctions avancées de l'éditeur vi

Les TP de ce chapitre sont laissés à la discrétion du formateur.
Selon les besoins et demandes des stagiaires il peut s'avérer utile de consacrer quelques exercices à tel ou tel aspect de l'éditeur vi.

Exemples de manipulations :

- Les substitutions de chaînes de caractères.
- Adressage des lignes par chaînes de caractères.
- Copier ou déplacer un paragraphe d'un texte à l'autre en utilisant un tampon nommé.

# Les conditions

1. Écriture du script ficlec

2. COMMANDE : ficlec paramètre
   ENONCE : testez si le paramètre communiqué est un fichier ordinaire. Dans l'affirmative, vérifiez si ce fichier est accessible en lecture.
   Faites afficher les messages correspondants.
   REMARQUE : faire cet exercice avec les deux formes de "et" ( -a de test et && de commandes ).

```
[ –f "$1" –a –r "$1" ] && echo " $1 fichier ordinaire lisible"
```

3. COMMANDE : tri fichier1 fichier2

   ENONCE :
   a) Testez le nombre de paramètres de la commande :
      Si le nombre de paramètres est différent de 2, alors affichez le message d'information rappelant la syntaxe correcte à l'opérateur, p.ex. : "commande : tri fichier1 fichier2".
   b) Ensuite, la procédure lit une ligne sur l'entrée standard et range cette ligne dans un fichier d'après les règles suivantes :
      - La ligne est introduite en fin de fichier1 si elle contient au moins une lettre.
      - Elle est introduite en fin de fichier2 si elle contient au moins un chiffre et pas de lettres.
      - Le reste est redirigé vers le fichier poubelle.
      REMARQUE : utilisation de case, read et des caractères génériques.

```
#tri.sh

if test $# -ne 2
then
echo " tri fichier1 fichier2"
else
fic_lettre=$1
fic_chiffres=$2
echo "Votre ligne :"
read ligne
case $ligne in
*[a-Z]*) echo $ligne >> $fic_lettres;;
*[0-9]*) echo $ligne >> $fic_chiffres;;
*) echo $ligne >> fic_poubelle;;
esac
fi
```

4 Copier / dupliquer 2.0

Modifiez le script copier écrit précédemment en ajoutant les fonctionnalités suivantes :

- Si le nombre de paramètres n'est pas correct, le programme basculera en interactif c'est-à-dire sur

dupliquer ;

- Vérifiez si le fichier est un fichier ordinaire et s'il existe, dans le cas contraire sortir du programme avec un code de retour 1.
- Vérifiez si le répertoire est un répertoire et si on a le droit d'écriture, si pas de droit d'écriture, sortir du programme avec un code de retour 2, si le répertorie n'existe pas demandez à l'opérateur s'il souhaite le créer et continuer ou bien sortir du programme.#

```
#dupliquer - Modifie

if test $# -ne 3
then
echo –e "veuillez saisir le nom du fichier :\c"
read vfichier
if test ! –f "$vfichier"
then
echo "le fichier n'existe pas ou n'est pas un fichier ordinaire"
exit 1
fi
echo –e "veuillez saisir le nom du répertoire :\c"
read vrep
if test –d "$vrep"
then
      if [ ! -w $vrep ]
      then
            echo "le répertoire n'est pas accessible en écriture"
            exit 2
      fi
else
      echo –e "Le repertoire n'existe pas Voulez-vous le créer:\c"
      read vreponse
      if $vreponse = "o"
      then
            mkdir $vrep
      else
            exit
      fi
fi
echo –e "veuillez saisir le suffixe :\c"
read vfsuffixe
vfichier=${vfichier#$HOME/}
cp $vfichier $vrep/$vfichier.$vsuffixe
chmod 400 $vrep/$vfichier.$vsuffixe
```

# Programmation d'une boucle

1. Écriture du script testdir
   COMMANDE : testdir
   ENONCE : affichez la liste, au format long, des répertoires du répertoire courant.
   REMARQUE : utilisation de for et test.

```
for fic in $(ls)
do
if test –d $fic
then
ls –ld $fic
fi
done
```

2. Écriture du script mkfichier
   COMMANDE : mkfichier prefixe n
   ENONCE :
   a) Créez n fichiers vides (par défaut 5) ayant pour nom prefixe.n
      (exemple : fic.1 fic.2 efic.3 lorsque prefixe=fic et n=3)
   b) Ajoutez dans la boucle une question permettant de valider ou d'invalider la création du fichier "prefixe.n".
      REMARQUE : utilisation de if, while, test, read, let

```
#mkfichier
typeset –i nbfic
nbfic=${2-5}
prefixe=$1
while (( $nbfic >0 ))
do
echo -e " voulez-vous creer le fichier $prefixe.$nbfic ? :\c"
read reponse
if [ $reponse = "o" ]
then
touch $prefixe.$nbfic
fi
let nbfic=nbfic-1
done
```

1. Écriture d'un menu
   Créez le shell script nommé menu1 suivant :

```
$ vi menu1
echo Choisissez votre commande
PS3="Votre choix :"
```

```
select commande in pwd ls date ps quitter
do
      case $commande in
            pwd)  echo Votre répertoire est :
                  pwd;;
            ls)   echo Voici le contenu de votre répertoire :
                  ls -CF;;
            date) echo Vous avez demandé la date :
                  date;;
            ps)   echo Voici les processus en cours :
                  ps -aef | more;;
            quitter)   echo o-o-o-o-o Au revoir ! O-o-o-o-o
                       break;;
      esac
exec $0
done
```

Copiez menu1 en menu2 puis en menu3.
Dans menu2 supprimez exec dans la ligne exec $0.
Dans menu3 supprimez la ligne exec $0.
Exécutez les différents shells scripts et déduisez-en l'utilité de la ligne exec $0.

# Environnement utilisateur

Modifiez le shell script ~/.bashrc de la manière suivante :

1. Déclarez d'un "timeout" (variable TMOUT) de 5 minutes (voir la définition dans le man).

```
export TMOUT=300
```

2. Redéfinissez l'invite primaire PS1 pour qu'elle donne les renseignements suivants :
   "Nom de la machine@Répertoire de travail $ "

```
export PS1='\h@\w $'
```

3. Redéfinissez l'invite secondaire PS2 de la manière suivante :
   ("Numéro de commande"-suite) >

```
export PS2='\# -suite'
```

4. Créez un alias lu équivalent à ls -rtl (voir la définition dans le man).

```
alias lu="ls –rtl"
```

5. Définissez le masque pour que les fichiers créés aient les droits sous la forme `rw-r-----`

```
umask 0137
```

6.

Définissez et exportez une fonction permettant d'obtenir l'heure courante (exemple : "Il est : 15 heure 21").

```
heure()
{
      Date '+Il est :%H heure %M'
      }
      Typeset –fx heure
```

7. Définissez et exportez une fonction permettant d'obtenir la liste des sous-répertoires d'un répertoire passé en paramètre.

```
ss_rep()
{
ls –l $1 |grep '^d'
}
Typeset –Fx ss_rep
```

# Expressions régulières, grep

Reprenez les exemples du chapitre afin de vérifier les syntaxes et de vous les approprier.

1. A partir du fichier /etc/passwd, affichez la ligne correspondant à votre utilisateur.
   `$grep ^$LOGIN: /etc/passwd`
2. Utilisation de grep dans les scripts systèmes
   Afficher le nom de tous les fichiers du répertoire /etc, qui utilisent la commande grep.
   ls /etc |grep grep

3. Dans le fichier "edition"
   Afficher les lignes avec le prénom Jean (tout court).

```
grep '^.* Jean .*' edition
```

4. Dans le fichier "edition"
   Afficher les lignes se terminant par 3 ou 5.

```
grep '[35]$' edition
```

5. Dans le fichier "edition"
   Afficher les lignes se terminant par deux 1 ou deux 2.

```
grep –E '11|22$' edition
```

6. Dans le fichier "edition"
   Afficher les lignes ne commençant pas par D.

```
grep '^[^D]' edition
```

7. Dans le fichier "edition"
   Afficher et numéroter les lignes contenant un nombre commençant par « 35 ».

```
grep -n '\b35' edition
```

8. Dans le fichier "edition"
   Extraire les lignes qui correspondent aux utilisateurs ayant dans le deuxième champ un caractère "*" (pas de connexion possible, extrait d'un système Unix).

```
grep '^.*:\*:.*' edition
```

9. Ecrivez le script lsrep répondant au cahier des charges suivant :
   COMMANDE : lsrep
   ENONCE : affichez la liste au format long des répertoires du répertoire courant
   REMARQUE : utilisation de ls et grep.

```
#lsrep
ls –l |grep '^d'
```

10. Affichez tous les processus bash du système.

```
$ps –ef |grep bash$
```

11. Arrêt « courtois » du système.
    Utilisation typique de la commande grep par un administrateur qui veut vérifier s'il peut arrêter le système. Il ne s'autorisera à arrêter que si personne d'autre que lui n'est connecté.
    Connectez-vous à une session graphique.
    Connectez-vous sur une console virtuelle <Ctrl><Alt><F3> (p.ex.)
    Revenez au graphique <Ctrl><Alt><F1>
    Écrivez un script nommé "arret" qui va vérifier le nombre de shells ouverts et ne rebootera le système que s'il n'y a plus qu'un shell.
    Demandez confirmation avant le reboot.
    Même opération, sans la demande de confirmation, en une seule ligne de commande.

# Éditeur de texte sed

1. A partir de /etc/passwd construisez une liste des utilisateurs présentée de la manière suivante :

NOM = ...  UID = ... REP = ...

```
$cat /etc/passwd | sed 's/\([^:]*\):.*:\([^:]*\):.*:.*:\([^:]*\):.*/
NOM=\1      UID=\2 REP=\3/'
```

2. Sauvegardez le résultat dans un fichier pass1 en utilisant une redirection.

```
$cat /etc/passwd | sed 's/\([^:]*\):.*:\([^:]*\):.*:.*:\([^:]*\):.*/
NOM=\1   UID=\2 REP=\3/'>pass1
```

3. Copiez le fichier /etc/passwd dans votre répertoire d'accueil en l'appelant pass2. Recommencez l'opération en donnant comme nom pass3 au fichier de destination. A partir de pass2 et de pass3, utilisez le programme de la question 1 et les options -i et --in-place de sed pour sauvegarder ou écraser le fichier d'origine.

```
cp /etc/passwd pass2
cp /etc/passwd pass3
$cat passw2 | sed –i .sav 's/\([^:]*\):.*:\([^:]*\):.*:.*:\([^:]*\):.*/
NOM=\1      UID=\2 REP=\3/
```

```
$cat passw3 | sed –i 's/\([^:]*\):.*:\([^:]*\):.*:.*:\([^:]*\):.*/
NOM=\1   UID=\2 REP=\3/
```

# Présentation de awk

1. Mettez en forme le résultat de la commande "mount" afin que le résultat affiché soit sous la forme :

```
Le système de fichiers  aaaaa  est monté sur xxxxx
Le système de fichiers  bbbbb  est monté sur yyyyy
...
```

2.

```
$mount |awk  '{ print "Le système de fichiers " $1 "  est monte sur "
$3}'
```

3. Affichez sous forme de tableau (de même forme que la question précédente) avec en plus : une première ligne de titre et en dessous le tableau, mais ce tableau ne montre que les systèmes de fichiers sur disque (premier caractère /).

```
mount |awk 'BEGIN { print "\tFS  \t\tMonte  \t\tTYPE  \t\tOptions" }$1
~"^/" {print $1"\t\t"$3"\t\t"$5"\t\t"$6}'
```

4. Fabriquez un fichier "plat" (fichier sans retour à la ligne) à partir de passwd.
   Dans le fichier p3 obtenu, le séparateur d'articles sera % au lieu de new-line. Autrement dit, on impose ORS.

```
awk 'BEGIN {ORS="%"}  {print $0}' /etc/passwd>p3
```

5. A partir de ce fichier p3, retrouvez le fichier d'origine en passant de RS=% à ORS='new-line'.

```
awk 'BEGIN {RS="%"; ORS="\n"}  {print $0}' p3
```

6. Ecrivez un shell script qui vous permettra de tuer un programme en précisant uniquement son nom, sans avoir besoin d'aller relever son PID.

```
command=$1
MYPID=$(ps | grep $command |awk –v cmd=$command ' $4 == cmd { print $2 }'
kill -9 $MYPID
```

Si on souhaite prévoir la possibilité qu'il y ait plusieurs processus exécutant la même commande on pourrait passer par les paramètres de position et une boucle for pour tuer tous les processus identifiés.
.

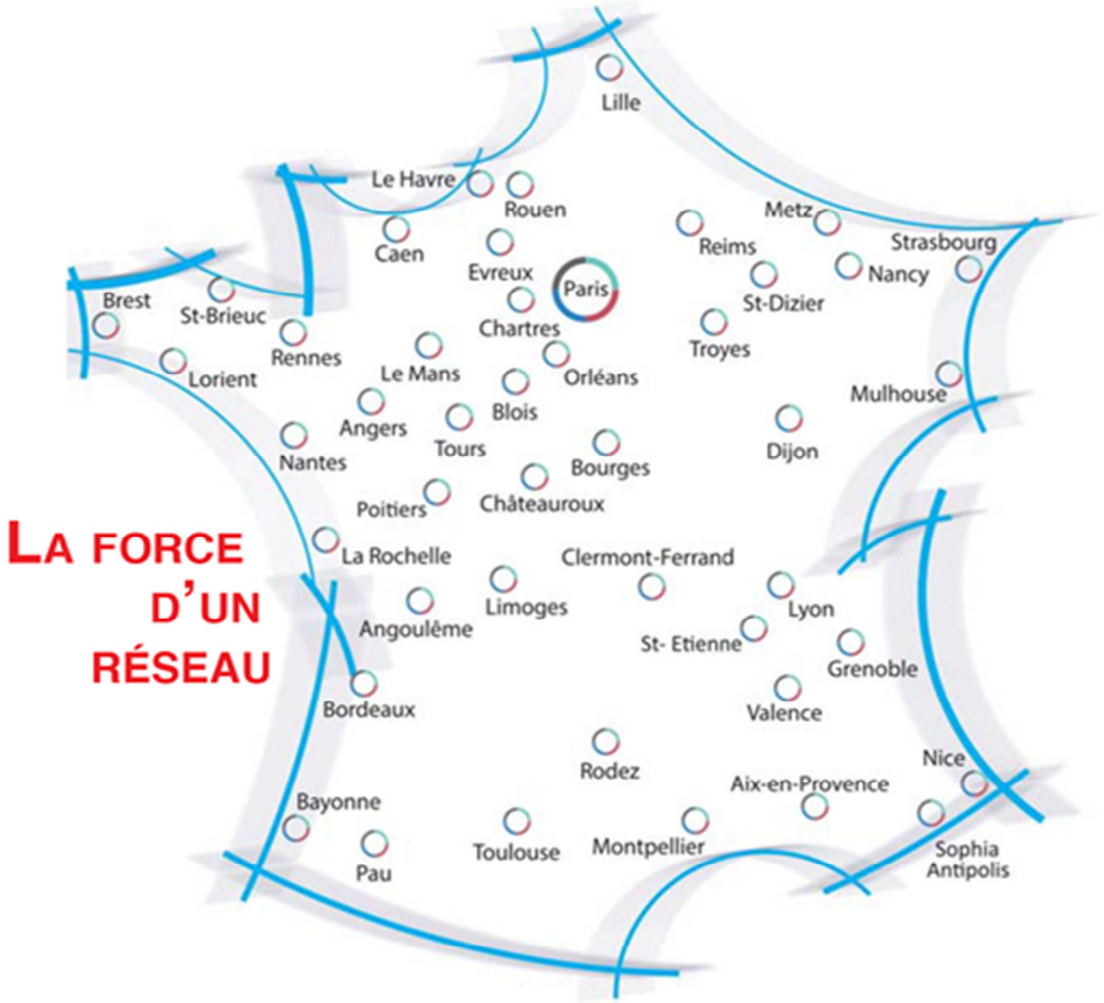
```
command=$1
MYPID=$(ps | grep $command |awk –v cmd=$command ' $4 == cmd { print $2 }'
kill -9 $MYPID
set $(echo $MYPID)
for proc
do
kill -9 $1
shift
done
```

7. Ecrivez une ligne de commande permettant d'extraire l'adresse IP du résultat de la commande ip a, en n'affichant que l'adresse de la carte réseau principale.

```
ip -o a |awk  ' $1 =="1:" && $3 == "inet" { print $4}
```

Formation

Notre expertise est votre avenir




Découvrez également l'ensemble des stages à votre disposition sur notre site

**http://www.m2iformation.fr**

N°Azur 0 810 007 689

PRIX D'UN APPEL LOCAL DEPUIS UN POSTE FIXE