IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

FIP SIT151

TP 2 - Unix/Linux

Automne 2023

Résumé

Cette seconde série d'exercices est à faire en autonomie. Des salles de TP Linux sont accessibles en libre service. La grosse majorité des questions est également faisable sur un PC Linux quelconque ; vous pouvez préférer utiliser votre propre machine si vous le souhaitez.

Comptez environ 3h pour faire ces exercices correctement. Certaines personnes déjà accoutumées à l'environnement Linux prendrons moins de temps (il ne s'agit pas de bâcler non plus). D'autres personnes moins accoutumées à l'outil informatique prendrons plus de temps. Avancez à votre rythme.

Il est impératif d'avoir fait ces exercices **avant** la seconde séance de TP programmée à votre emploie du temps. Cette courte séance sera l'occasion de débriefer ces exercices tous ensemble, avec un enseignant.

1 Travail sur les répertoires et fichiers

Vous êtes connecté(e). Lancez une fenêtre émulant un terminal. Dans cette fenêtre une chaîne de caractères (le prompt) vous indique que vous pouvez saisir une ligne de commande. Dans le tableau qui suit, vous trouverez en colonne de gauche sur chaque ligne un travail à réaliser. En colonne de droite, vous indiquerez la ou les commandes utilisées ainsi que leurs options et arguments, vous indiquerez aussi leur résultat.

	Travail	Aide	Réponse
1.1	Affichez le nom de votre répertoire de travail (c'est aussi votre répertoire d'accueil si vous venez de vous connecter).		-
1.2	Listez le contenu de votre répertoire d'accueil (listing long : option `-l`). Listez les fichiers cachés (idem).	`ls`	- -
1.3	Déplacez-vous dans le répertoire `/etc` Vérifiez que vous y êtes bien.	`cd`	- -
1.4	Revenez dans votre répertoire de travail en utilisant la commande la plus simple possible (sans aucun argument). Vérifiez que le changement est correct.		- -
1.5	Créez le répertoire Unix et déplacez-vous dedans. Vérifiez le déplacement.	`mkdir`	- - -
1.6	Copiez dans ce répertoire le fichier `/etc/passwd`. Vérifiez que la copie est correcte (avec arguments donnant au moins la taille des fichiers).	`cp`	- -
1.7	Renommez cette copie en lui donnant le nom `test`. Vérifiez...	`mv`	- -
1.8	Déplacez-vous dans le répertoire `/usr`. Vérifiez que vous y êtes.		- -
1.9	Depuis cet endroit, listez le contenu de votre répertoire Unix en utilisant le raccourci donné par le caractère `~` (sorte de constante contenant le nom du répertoire d'accueil [1])	`ls`	-

1. Le caractère `~` désigne mon répertoire à moi, et `~dupont` celui de l'utilisateur *dupont*.

1.10	Vous êtes toujours dans `/usr`, tentez d'y copier le fichier `test` que vous avez créé précédemment (réutilisez le raccourci `~` pour nommer votre fichier). Vous devez normalement rencontrer un problème... Lequel et pourquoi ? Pouvons-nous remédier à ce problème ?	`cp, ls`, *droits*	- Commentaires :
1.11	Retournez dans votre répertoire Unix. Faites une copie du fichier `test` que vous nommerez `test1`. Vérifiez le résultat.		- - -
1.12	Supprimez le fichier `test`. Vérifiez...	`rm, ls`	- -
1.13	Retirez le droit de lecture à tout autre que vous sur le fichier `test1` en utilisant la notation symbolique. Vérifiez...	`chmod`	- -
1.14	Rajoutez le droit d'écriture pour le groupe et ceci en notation octale (les droits devront donc être : `rw-rw--`). Vérifiez	*idem*	- -

1.15(2)	Quelle est la valeur de votre `umask` ? Quels droits auront vos fichiers par défaut (les fichiers non exécutables d'abord, les fichiers issus d'une compilation/édition de liens ensuite) ? Créant des fichiers, des répertoires et vérifiez.	`umask`
		`umask -S`
		`man 2 umask`
		`ls -l`
		`stat`	

1.16(2)	Quels sont vos quotas ?	`quota -vm` [3]	
1.17(2)	Quel est l'espace disque accessible depuis votre machine ?	`df`	
1.18(2)	Quel est l'espace disque occupé par tous vos fichiers ? (Attention, n'oubliez pas les fichiers cachés).	`du`	(Attention à l'unité d'affichage : option `-k`)
1.19(2)	Si vous avez déjà au moins une fois lancé `mozilla` ou `firefox` vous devez avoir un répertoire `.cache/mozilla`. Sinon lancez-le, ce répertoire sera créé. Déplacez-vous dedans et affichez la taille du répertoire cache qui s'y trouve. À l'aide du menu préférences d'un de ces navigateurs, videz votre cache	`du`	(Attention à l'unité d'affichage)

Les commandes `chmod` et `umask` manipulent traditionnellement les permissions Unix sous leur représentation en octal. L'octal, c'est la base 8, lorsque l'on compte de 0 à 7. Les nombres de 0 à 7 se codent sur 3 bits. Les permissions Unix `rwx` sont des drapeaux sur 3 bits. Voilà pourquoi on représente souvent les permissions Unix en octal. Cependant, pour plus de confort, les commandes `chmod` et `umask` modernes comprennent également la notation symbolique. Par exemple, pour ajouter les droits d'écriture au groupe pour un fichier, on peut faire `chmod g+w` *`fichier`*.

3. Vous pouvez également appeler le script école `/opt/bin/campux-quota.sh`, ou via le menu `Applications > Accessoires > Synthèse des quotas`

2. Question optionnelle : sachez que cette commande existe, mais on ne vous demande pas de la connaître par coeur. Vous ne serez pas évalué dessus.

Notation symbolique	Notation octale
`-x -w- r-`	124
`rwx r-x r-x`	755
`rw- -- --`	600

Notez que l'on positionne ici les 3 triplets `rwx`. On a donc une notation à trois nombres en octal. Il est également possible de positionner en même temps les drapeaux *suid bit*, *sgid bit*, et *sticky bit*. On a alors un quatrième nombre en octal qui est placé en tête des trois autres.

Autre chose : comprenez bien que le `umask` décrit un masque, c'est-à-dire les droits que l'on retire. C'est la négation binaire des permissions que les fichiers obtiendront par défaut.

1.1 Question bonus : un espace de partage

Voici un petit exercice qui devrait vous permettre de vérifier que vous avez bien assimilé le principe des permissions Unix.

L'idée est de mettre en place un espace vous permettant de partager facilement des fichiers avec vos collègues. (En effet, puisque les comptes Unix de l'école sont déjà en réseau, ce n'est pas la peine d'envoyer ses fichiers dans les nuages pour que le voisin puisse les récupérer...)

1.1.1 Mise en place

Voici la marche à suivre pour mettre en place cet espace de partage :

— Dans votre répertoire de travail (votre *home directory*), créez un sous-répertoire appelé (par exemple) : `partage`
— Ajouter les permissions de *lecture* et de *traversée* à ce sous-répertoire `partage`, pour le *groupe* et les *autres*. (Quelle commande tapez-vous ?)
— Vérifiez que votre répertoire de travail dispose bien également de la permission de *traversée* pour tout le monde.
— Dans ce sous-répertoire `partage`, créez deux sous-sous-répertoire appelés (par exemple) : `public` et `privé`
— Ajouter les permissions de *lecture* et de *traversée* au répertoire `public`, pour tout le monde.
— Ajoutez uniquement la permission de *traversée* au répertoire `privé`, pour tout le monde.

Vous disposez maintenant d'un espace `~/partage/public` accessible en lecture à toute personne disposant d'un compte Unix école, ainsi que d'un espace `~/partage/privé` accessible à toute personne disposant d'un compte Unix école mais de manière restreinte/cachée...

1.1.2 Utilisation

— Espace public

  Déposez un fichier dans votre espace `~/partage/public`. Assurez-vous que vous avez positionné les droits en lecture. Demandez à votre voisin de parcourir votre espace `public` et de vérifier qu'il peut bien accéder à votre fichier.

  Note : il faut que votre voisin connaisse au moins le nom de votre répertoire de travail, qui est typiquement votre nom de login. Par exemple pour visiter l'espace de partage public de Christophe Lohr, on peut faire `cd ~clohr/partage/public/`

— Espace privé

  Déposez un fichier dans votre espace `~/partage/privé`. Assurez-vous que vous avez positionné des droits en lecture. Demandez à votre voisin de parcourir votre espace `privé`. Normalement, il ne peut pas voir le contenu de ce répertoire (c.à.d. qu'il ne peut pas faire `ls`).

  Cependant, s'il connaît le nom de votre fichier, il peut y accéder : le nom du fichier devient en quelque sorte le mot de passe secret. À vous de choisir un nom de fichier que ne puisse pas être deviné facilement. (Pour choisir un nom un peu aléatoire, vous pouvez vous aider de commandes telles que `echo $RANDOM`, `mcookie`, `uuidgen`, etc.) Concrètement, seules les personnes à qui vous aurez communiqué le nom de votre fichier privé pourront y accéder. Faites l'essai.

2 Exercices sur l'environnement

	Travail	Aide	Réponse
2.1	Quel est le nom de votre Shell	Déjà vu plus haut...	

2.2	Affichez les variables d'environnement de votre Shell.	`env`	-
2.3	À l'aide de la commande `echo`, affichez les valeurs des variables suivantes : `TERM, MAIL, PATH, USER, LOGNAME, SHELL`. Rappel : pour accéder à la valeur d'une variable, faire précéder son nom par le caractère `$`. Exemple : `echo $var`		`echo $NOM_VAR` - - - - -
2.4	Créez une variable *utilisateur* de nom `VAR1` et de valeur `abc` et une variable d'*environnement* de nom `VAR2` et de valeur `xyz`. Vérifiez la portée des variables `VAR1` et `VAR2` avec la commande `env` (vous pouvez affiner l'affichage en utilisant un tube de communication : `env | grep VAR`)	`=` `export`	- -
2.5	Ouvrez deux autres fenêtres terminal, l'une depuis votre shell actuel avec la commande `xterm` ou `gnome-terminal`, l'autre depuis le menu graphique. Vérifiez si les shells associés à ces fenêtres connaissent vos variables déclarées dans la première fenêtre. Pour plus de rigueur, et si vous êtes courageux, vous pouvez repérer la filiation des processus shell concernés avec la commande `pstree -h`[2]. Expliquez...		-
2.6(2)	La commande `mount` existe sur le système. Elle est utilisable par tout le monde pour lister les montages de systèmes de fichiers (il faut normalement être `root` pour monter un système). Avec la commande `which`, vérifiez que cette commande est accessible Sinon, cherchez avec `locate`. Indiquez quel est son chemin.	`which`	-
2.7(2)	Avec la commande `whereis` cherchez si `mount` existe dans un répertoire standard. Indiquez le résultat. Avec la commande `ls -l`, listez chaque élément du résultat précédent. Que constatez-vous ?	`whereis` `whatis`	- -
2.8(2)	Quels sont les systèmes de fichiers montés sur votre machine ? Indiquez les systèmes locaux ainsi que les systèmes montés en réseau. Pour ces derniers vous indiquerez le nom des serveurs. Sur quel disque se trouve le système de fichiers racine (le `/`) ?	`mount`	-
2.9(2)	Quelle est l'imprimante par défaut de la machine sur laquelle vous êtes connectés actuellement ? À quelles imprimantes avez-vous accès ?	`lpstat`	-
2.10(2)	Quel est votre numéro d'utilisateur ?	`id` `$UID`	-

3 Exercices sur les processus

	Travail	Aide	Réponse
3.1	Testez la commande `ps`, d'abord sans argument puis avec les options `-u $LOGNAME`, et enfin en rajoutant l'option `-f`		Quelles différences dans les affichages ?
3.2(2)	Testez la commande `ps -ef`		Que voit-on ?
3.3	Lancez une commande qui ne se termine pas tout de suite, par exemple `xeyes`. Ouvrez une autre fenêtre terminal et dans celle-ci recherchez le numéro du processus correspondant à la commande `xeyes`. En restant dans cette seconde fenêtre, utilisez le résultat de `ps` pour supprimer le processus `xeyes`.	`ps -ax`, `kill`	fenêtre 1 - fenêtre 2 - -
3.4	Relancez la commande `xeyes`. Votre terminal ne répond plus, les commandes que vous y entrez ne sont pas prises en compte. Stoppez alors le processus `xeyes` (faites `Ctrl-Z`) et passez-le en background (`bg`). Tapez la commande `jobs`. Que vous répond t-elle ? [4]		
3.5(2)	Lancez un nouveau `xeyes` suivi cette fois du caractère `&`. Que se passe-t-il ? Faites `jobs`. Que voyez-vous ? À l'aide de la commande `kill` et du résultat de `jobs`, supprimez le second `xeyes`.[4]	`jobs`, `kill`	- - -
3.6	Repassez le premier `xeyes` en premier plan (`fg`) et supprimez-le (`Ctrl-C`).		- -
3.7	À l'aide de la commande `top` ou `htop` relevez les caractéristiques de votre machine : taille mémoire, taille du swap, charge moyenne, par CPU...	`top`, `htop`	-
3.8	Avec la commande `tty` relevez le nom du terminal virtuel associé à votre fenêtre terminal. Avec `ps -aux` retrouvez le numéro de processus associé à votre fenêtre. Trouvez le numéro du shell associé à la fenêtre.	`tty, ps`	- -
3.9	En utilisant le mécanisme des tubes de communication affichez la 23ème personne à s'être connectée dernièrement sur votre machine. L'historique des dernières connexions est accessible avec la commande : `last` Vous utiliserez les commandes `head` et `tail` pour filtrer la sortie standard de `last` et obtenir le résultat souhaité. [5]		`last | ..........`
3.10	Rediriger la sortie standard de la commande `ls` dans un fichier. Vous listerez le contenu de `/usr`.		

4. Le contrôle des tâches : `help jobs bg fg disown`

5. Si personne ne s'est connecté dernièrement, ça ne sera pas rigolo. Dans ce cas faites l'exercice en regardant l'historique des dernières commandes que vous avez tapées (commande `history`).

4 Exercice sur les commandes en réseau

Cette série de questions est également optionnelle : vous ne serez pas évalués dessus, et elles ne sont pas requises pour mener à bien vos enseignements. Cependant, l'expérience montre qu'il est bon d'avoir quelques compétences sur le sujet (si vous souhaitez travailler sur les PC de l'école depuis chez vous, transférer vos fichiers, etc.).

Remarque : traditionnellement on utilisait sous Unix les commandes `telnet` et `rlogin` pour les connexions à distance. Ces commandes sont maintenant à éviter car les mots de passe sont véhiculés en clair sur le réseau. Sous Linux, le serveur pour `rlogin` n'est d'ailleurs plus installé de manière standard sur les distributions courantes. Il faut préférer `ssh` que nous allons voir maintenant.

	Travail	Réponse
4.1	Repérez le nom d'une autre machine que la vôtre dans la salle. Connectez-vous dessus par `ssh`. Vérifiez que vous êtes bien sur la machine en question grâce à la commande `uname -n`.	- -
4.2	Dans quel répertoire êtes-vous arrivé avec `ssh` ? Listez son contenu. Est-il semblable au contenu de votre répertoire d'accueil sur la machine physique sur laquelle vous travaillez ? Pourquoi ?	- -
4.3	Déconnectez-vous en fermant votre session `ssh` par `exit` (ou `logout`). Un répertoire caché `.ssh` est créé ; que contient-il ?	-
4.4	Après avoir regardé sa page de `man`, lancez la commande `ssh-keygen`, et contentez-vous de faire `Enter` aux diverses questions posées (vous avez confiance ou vous avez lu le `man` ?). Que s'est-il passé, et que contient le répertoire `.ssh` ?	- -
4.5	Ajoutez votre clé publique ainsi générée dans votre liste de clés autorisées (`cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys`)[6]. Reconnectez-vous à la machine voisine. Que se passe-t-il ? Pourquoi ?	-
4.6	Exécutez la commande `ssh -X` *`machine_voisine`*. Dans la fenêtre de connexion sur la machine distante lancez la commande `xterm`. Que constatez-vous sur la machine locale ? (Ce résultat est obtenu grâce à l'option `-X`.[7])	-
4.7	Si vous disposez d'un compte différent sur un autre machine Unix (p.ex. au RésEl ?), copiez chez vous un fichier depuis cette autre machine avec la commande `scp`. Si non, copiez le fichier `/etc/hostname` de la machine voisine (cela a moins d'intérêt vu que vous avez des comptes partagés en réseau, c'est juste pédagogique...).	-

Remarques (à prendre en compte en dehors des séances de TP)

— Sous environnement MS-Windows il existe également des implémentations des commandes `ssh` et `scp` avec le logiciel *putty*. Ce logiciel est disponible librement à l'adresse suivante : `http://www.chiark.greenend.org.uk/~sgtatham/putty/`.

6. Le script `ssh-copy-id` permet de copier sa clef publique sur un compte distant.
7. Cela peut être une bonne idée d'utiliser l'option `-C` en complément.

— Si vous voulez obtenir directement l'affichage des fenêtres graphiques Unix sur votre écran MS-Windows c'est un peu plus compliqué... Pour cela il y a plusieurs niveaux de solutions suivant que le serveur graphique et/ou les applications graphiques se trouvent en local sur votre PC MS-Windows ou bien sur une machine Unix distante. Voici donc trois cas de figure distincts :
  — VNC : un système open-source de bureau à distance. Cette solution est disponible sur les machines MS-Windows de l'école. Un client sur votre machine MS-Windows se connecte à une machine Unix sur laquelle se trouvent des applications graphiques et un serveur graphique qui envoie au client VNC les graphismes des applications par le protocole VNC (un genre de flux jpeg). En retour le client VNC envoie au serveur graphique les événements clavier et souris (excepté parfois le caractère `~`...).
  — Une implémentation d'un serveur X-Window sous MS-Windows. Par exemple `Xming` `http://www.straightrunning.com/XmingNotes/`. Dans cette solution, seul le serveur X est présent sur le PC MS-Windows ; et il faut se connecter sur une machine Unix pour y exécuter des applications graphiques. Celles-ci, grâce au protocole X (un peu gourmand en bande passante mais qui se compresse très bien) dialogueront avec le serveur X sur la machine MS-Windows. Les communications graphiques pouvant se faire également au travers d'un tunnel ssh (un petit tutoriel au RéSel `http://resel.fr/configuration/xming/`).
  — *Cygwin* `http://www.cygwin.com/`. C'est une implémentation sous MS-Windows d'un environnement POSIX qui peut ainsi exécuter toutes les commandes et applications classiques Unix ; et notamment un serveur X-Window, des applications graphiques, un shell, etc. Il faut télécharger un programme de «setup», le lancer et chercher l'item *X11* puis les items *xorg*. Avec Cygwin vous pouvez aussi obtenir les mêmes commandes ssh et scp que sous Unix.

Ceci décrit comment faire une connexion graphique à distance, reste la question de *«comment passer le firewall ?»*

— Tapez la commande suivante depuis votre machine locale (Linux ou Windows, équipée des commandes `ssh`), et dites ce qu'elle fait :

```
ssh -L 5901:srv-disi-vnc-04.priv.enst-bretagne.fr:5950 \
                                srv-disi-vnc-04.priv.telecom-bretagne.eu
```

— Toujours sur votre machine locale (équipé d'un client VNC), et sans fermer la connexion précédente, tapez la commande suivante et dites ce qu'elle fait : `vncviewer :1`

5 Le Shell de Bourne, initiation à la programmation

Le shell est nous offre un langage de programmation avec des structures de contrôle classiques (*if*, *while*, *for*, etc.). Il permet l'utilisation de variables, il possède des commandes qui lui sont propres (commandes internes). Les commandes Unix deviennent de véritables fonctions du shell. Ainsi nous pouvons écrire des fichiers exécutables qui enchaînent des commandes ou des programmes, ces fichiers sont en quelque sorte des commandes de plus haut niveau.

Dans un certain nombre d'établissements, pour des raisons historiques (typiquement un héritage de Sun Solaris), le shell par défaut proposé aux utilisateurs est le shell `tcsh` (c'est encore un peu le cas à Télécom Bretagne). Pour connaître votre shell faites par exemple `echo $0`.)

Pour différentes raisons [8] nous ne nous attarderons pas sur l'apprentissage de ce shell. Aussi, si vous êtes sous un shell `tcsh`, il serait préférable que vous travailliez directement en Bourne Shell, sous `bash` par exemple. Vous disposez de deux façons d'obtenir `bash` :

— De manière ponctuelle. Vous pouvez vous contenter de lancer la commande `bash` dans votre shell courant. Le prompt de `bash` a été paramétré par les administrateurs système pour ressembler à celui de `tcsh`. C'est un peu troublant (vous pourrez le changer en modifiant la variable d'environnement `PS1`), mais vous avez bien un shell `bash`.

— De manière permanente (mais non nécessairement définitive, vous pourrez rechanger pour `tcsh` si le cœur vous en dit). La commande `ypchsh` (lisez le `man`) vous permet de changer votre shell par défaut par l'un de ceux listés dans le fichier `/etc/shell` (du serveur NIS, mais c'est globalement le même que celui de la machine locale). Notez qu'il y aura un délai le temps que se propagent les pages NIS depuis le serveur et que votre cache local se mette à jours (si vous êtes pressés, connectez-vous sur une autre machine qui ne vous a pas vu depuis longtemps).

5.1 Les variables

— Créez la variable `var` et donnez-lui la valeur `10`. Vérifiez avec `echo $var` que la variable existe et possède la valeur indiquée.

8. `http://www.grymoire.com/Unix/CshTop10.txt`

— Créez la variable `VAR` et fixez-lui la valeur `VAL1`.
— Ajoutez au contenu de `VAR` le contenu `VAL2` en séparant les deux valeurs par le caractère « `:` ». Cela revient à concaténer `VAL1` avec `:VAL2`. On doit obtenir le résultat `VAL1:VAL2` dans `VAR`.

5.2 Les paramètres positionnels

— À l'aide d'un éditeur, créez le fichier `exo52` et écrivez dedans les lignes suivantes :
```
#!/bin/sh
echo $1 $2 $4 $3 $0
echo "Nombre de paramètres: $x"
echo "Numéro du processus: $y"
```
Vous remplacerez les *$x* et *$y* ci-dessus par les expressions convenables.
— Sauvegardez et rendez le fichier exécutable avec la commande `chmod`.
— Exécutez ce script de la manière suivante : `./exo52 un deux trois quatre cinq`
— Que constatez-vous ? Expliquez.
— En fin de script, ajoutez les lignes qui suivent, ré-exécutez et expliquez la commande interne `shift` :
```
shift
echo $1 $2 $4 $3 $0
```
— En fin de script, ajoutez les lignes qui suivent, ré-exécutez et expliquez la commande interne `set` :
```
set assez dit la baleine
echo $1 $2 $4 $3 $0
```

5.3 Les paramètres positionnels (suite)

— Au prompt de votre Shell (`bash`!), tapez les commandes suivantes :
```
date
d=`date`
```
(attention, ce sont des accents graves, des apostrophes inversées ou backquotes)
```
set $d
echo $1 $3
echo $d
```
— Attendez quelques instants (au moins une minute) et refaites `echo $d`.
— En déduire le rôle des accents graves et le rôle de la commande `set`.

5.4 Les commandes UNIX sont des fonctions du Shell

Ce sont des fonctions au sens où elles rendent une valeur qui peut être utilisée et testée grâce aux structures de contrôle du Shell comme `if`. La valeur rendue est stockée dans la variable « `?` ». Elle est donc accessible par l'expression `$?`.

Essayez les commandes suivantes, et dites quelle est la valeur rendue :
— `grep root /etc/passwd`
`echo $?`
— `grep truc /etc/passwd`
`echo $?`
— `grep root /etc/pwd`
`echo $?`
— `touch truc` *(création d'un fichier vide de nom* `truc`*)*
`rm truc`
`echo $?`
— `rm truc`
`echo $?`
— `rm  clohr/.cshrc`
`echo $?`

Vérifiez pour chacune de ces commandes si les valeurs rendues sont conformes à ce qu'annonce le manuel de référence dans les sections `EXIT STATUS` (faites `man grep`, `man rm`).

5.5 Les commandes Unix sont des fonctions du Shell (suite)

— Écrivez ceci dans un fichier `exo55` :
```
#!/bin/sh
grep $1 /etc/passwd > /dev/null
if test $? -eq 0
```

```
then
        echo "l'utilisateur $1 existe sur cette machine"
else
        echo "$1 n'est pas un utilisateur enregistré sur cette machine"
fi
```

— Rendez ce fichier exécutable (`chmod`). Et exécutez-le avec divers arguments (`root`, `truc`,...)

Attention, les utilisateurs des machines du réseau de l'école sont gérés par un service centralisé appelé NIS (Network Information Service) ou encore *Yellow Pages*. Vous trouverez donc curieux que la commande ci-dessus vous indique que vous n'êtes pas enregistrés (si vous cherchez votre nom de login). Remplacez alors la première ligne utile (`grep`...) par `ypmatch $1 passwd`).

5.6 Structures de contrôle `if` et `case`

La commande `date` vous donne normalement la date en français. Écrivez un script (`exo56`) qui vous l'affiche en anglais. (Sans jouer avec la `locale ;-)`) Vous indiquerez en particulier l'heure au format suivant : `it's 5 past 15` ou `it's 20 to 16` selon que les minutes seront inférieures ou supérieures à 30.

Si votre environnement est tel que la commande date affiche son résultat en anglais alors faites un programme de traduction anglais vers français.

Les commandes à utiliser (éventuellement) sont : `date` pour obtenir la date, `set` pour fractionner la réponse de date en mots (récupérés dans `$1`, `$2`, etc.) [Note : on peut également utiliser `read` pour ça.] La variable `IFS` contient les séparateurs permettant à `set` de fractionner les chaînes de caractères qu'on lui passe en paramètre. Pour fractionner l'heure il faudra rajouter le caractère « `:` » dans `IFS` (ou positionner `IFS` à la valeur `":"`).

La principale structure de contrôle du Shell à utiliser est `case`.

5.7 Scripts interactifs

Créez un fichier `exo57` qui intègre une séquence interactive qui pourrait être la suivante :

```
$ exo57
 Veuillez entrer votre nom et votre prénom: nom prenom
 Bienvenue prenom nom
$ _
```

Ce qui est souligné ci-dessus est entré par l'utilisateur. Ce qui est en italique est le résultat émis par votre programme. Veuillez noter l'inversion de l'affichage, le nom est affiché après le prénom alors qu'ils ont été entrés dans l'ordre inverse. Cette inversion est bien entendu le fait du programme, ce n'est pas *magique*.

Commandes à utiliser : `read`, `set`, `echo`.

Un plus serait de vérifier que deux mots ont bien été rentrés par l'utilisateur et dans le cas contraire reprendre la séquence en son début (utilisation de `test` et de la structure de contrôle `while`).

Nota : si on veut supprimer l'écho de chaque caractère renvoyé par le terminal (par exemple lorsqu'il y a un mot de passe à entrer) il faut utiliser la commande `stty -echo`. La commande `stty echo` rétablira l'écho.

5.8 Programmez en shell au standard Unix

Nous avons vu aux 5.4 et 5.5 que les commandes Unix rendaient une valeur lors de leur terminaison et que cette valeur était stockée dans la variable « `?` » du Shell. Dans cet exercice nous allons voir comment écrire un script qui se comporte de la même façon.

Reprenez le fichier de la question 5.5 et complétez-le avec la commande `exit` à laquelle vous fournirez en paramètre une valeur correspondant au résultat, à savoir : *utilisateur trouvé* → résultat VRAI, valeur rendue `0`, *utilisateur non trouvé* → résulta FAUX, valeur rendue différente de `0`.

Testez différentes valeurs de retour. Quelle est la valeur maximum (autour de 256...) ?

5.9 Comprendre les mécanismes des processus et ce qu'est une commande interne du Shell

À l'aide d'un éditeur créez le fichier `exo59` et écrivez dedans ceci : `cd /usr; pwd`

— Rendez le fichier exécutable et exécutez-le.

— Faites alors "à la main" `pwd` et notez ce que cette commande affiche, vous constatez que vous n'êtes pas dans `/usr` alors que votre commande `exo59` semble vous y avoir conduit. Expliquez pourquoi vous n'êtes pas restés dans `/usr`.

— Exécutez ce fichier à la manière d'une commande interne du Shell (en `sh` ou `bash` vous faites `.<`*espace*`>nom_fichier`, en C-Shell vous faites `source nom_fichier`) et vérifiez son résultat avec `pwd`. Constat ?
— Qu'en déduisez-vous sur la nature de cd en tant que commande ?

6 Extra bonus

Bien vivre son shell au quotidien... Parcequ'on le vaut bien !

6.1 `ls`

1. Comment afficher tous les fichiers du répertoire `/etc` sachant que l'affichage doit se faire avec les contraintes suivantes :

   (a) on doit afficher des informations sur chaque fichier

   (b) les fichiers les plus récemment modifiés doivent être affichés à la fin

   (Pour cela, utilisez la commande Unix `man` qui vous permet d'obtenir la documentation sur n'importe quelle commande du système.)

2. Allez sous le répertoire racine du système. Exécutez la commande `ls -F`. Que fait-elle selon vous ? Observez l'affichage de `/lib`. Qu'est-ce que cela signifie ? Comment peut-on en savoir plus ?

3. Affichez tous les fichiers du répertoire `/usr/bin` ayant les lettres `l` ou `b` dans leur nom de même que les fichiers dont le nom se compose de 4 lettres, et tout ça en une seule commande.

6.2 `cat more less head tail`

1. Affichez le contenu du fichier `/var/log/dmesg` en une seule fois puis page-à-page. Affichez seulement les 5 dernières lignes. Affichez seulement les 5 premières lignes.

2. Lorsque vous êtes en affichage page-à-page, rechercher la chaîne de caractère `PCI`

6.3 `cp rm mv`

1. Créez les répertoires `JAVA, BIN, DIVERS`. À l'intérieur du répertoire `DIVERS`, créez les fichiers `youpla`, `youplala` et `on_est_les_champions`. En une seule commande, faites une copie du répertoire `DIVERS` et son contenu sous votre home-directory. Créez le fichier `ca_assure` dans le répertoire `JAVA`. Changez ensuite son nom en `ca_assure.un_max`. Effacez les répertoires `JAVA`, `BIN` et `DIVERS`.

2. Créez un fichier dont le nom est le caractère `-`. Essayez maintenant de le détruire. (Ça vous arrivera !!)

6.4 `find which locate whereis`

1. Donnez le chemin absolu du fichier `cat`. Donnez son chemin de façon relative à votre home-directory.

2. Recherchez dans l'arborescence des fichiers, à partir du répertoire `/etc` l'ensemble des fichiers qui ont été modifiés il y a 4 jours ; recherchez dans l'arborescence des fichiers, à partir du répertoire `/tmp` l'ensemble des fichiers dont les droits sont `rw-r-r-`.

6.5 `chmod`

1. Allez dans le répertoire `/usr/bin` et créez le fichier `beurk`. Quelle est la réaction du système ? Allez dans le répertoire `/tmp` et effectuez la même opération. Quelle est la réaction du système. Commentez.

2. Créez un répertoire `C` dans votre home-directory et interdisez à tous les utilisateurs (à part vous) d'entrer dans ce répertoire. Vérifiez en demandant à un autre utilisateur de tenter d'entrer dans votre répertoire. Enlevez le droit de lecture sur ce répertoire pour vous-même. Tentez à présent de pénétrer dans ce répertoire et d'en lire le contenu (vous effectuerez toutes ces opérations en positionnant les droits un-à-un, mais aussi en les positionnant de façon globale à l'aide de la notation octale).

3. Le mécanisme du *sticky bit*. Créez un répertoire `oula` dans votre compte. Donnez tous les droits à tout le monde sur ce répertoire. Créez ensuite dans ce répertoire un fichier `perso` sur lequel vous ne donnerez

aucun droit aux utilisateurs (à part vous bien sûr). Demandez à un autre utilisateur de détruire ce fichier. Y parvient-il ? Pourquoi ? Faites maintenant en sorte qu'il ne puisse plus détruire le fichier sans lui enlever le droit d'écrire dans le répertoire.

4. Question subsidiaire : le mécanisme du *suid bit*. La commande `passwd` vous permet de changer votre mot de passe (local). Le fichier où sont stockés les mots de passe en local est le fichier `/etc/shadow`. Quels sont les droits d'accès de ce fichier ? Comment se fait-il qu'un utilisateur puisse changer son mot de passe ?

6.6 *pipe* `grep wc tr cut`

1. Comptez le nombre de lignes (puis de mots, puis de caractères) du fichier `/etc/X11/xorg.conf`.
2. La commande `ypcat passwd` affiche les informations utilisateur provenant des pages jaunes en réseau NIS. Affichez, puis comptez, les d'utilisateurs dont le shell de login est `/bin/bash`.
3. N'affichez que le login de ces utilisateurs
4. Affichez leur login en majuscules

6.7 `&` ; <Ctrl>-C <Ctrl>-Z `jobs bg fg kill ps`

1. Quelle est la différence entre ces deux lignes de commande (et faites un <Ctrl>-C pour voir) : `xclock & xeyes` puis `xclock ; xeyes`

2. Réessayez `xclock ; xeyes` et faites un <`Ctrl`>-`Z`. Que se passe-t-il ?

3. Quels sont les `jobs` qui sont en activité et quels sont ceux qui sont stoppés ?

4. Réactivez le ou les jobs stoppés de manière à ce qu'ils s'exécutent en tâche de fond.

5. Vérifiez bien que tous vos jobs sont actifs. Choisissez-en un et placez-le en avant-plan. Puis tuez-le avec un <`Ctrl`>-`C`.

6. Repérez le numéro de processus de chacun des jobs encore actifs (attention, un numéro de job et un numéro de processus ce n'est pas la même chose).

7. Tuez les jobs encore actifs en indiquant soit le numéro de job, soit le numéro de processus.

6.8 `alias unalias`

1. Consulter la liste des alias qui sont d'ores et déjà définis pour votre compte. Détruisez l'alias `cp` et créez les alias suivants :
   (a) `lf` est un alias pour `ls -F`
   (b) `truc` est un alias qui permet d'afficher la date et la chaîne de caractères `"pffff"`
   (c) `motsdepasse` est un alias vers le fichier `/etc/passwd`

   Vérifiez que ces alias fonctionnent.

2. Ouvrez un autre terminal. Les alias définis précédemment sont-ils définis dans ce terminal ?
Faites en sorte que l'alias `lf` soit défini dans tout nouveau shell créé.

6.9 Les variables d'environnement

1. Que fait la commande interne `env` ?
2. Quelle est la commande qui vous permet de connaître le chemin du programme `cat`. Quelle variable d'environnement utilise-t-elle ?
3. Positionnez la variable d'environnement `PATH` à `/usr/bin`. Essayez d'exécuter la commande `useradd`. Quelle est la réponse du système ? Cherchez où se trouve cette commande et modifiez la variable d'environnement `PATH` de telle façon que vous puissiez exécuter la commande `useradd`.
4. Quelle est la variable d'environnement nécessaire au bon fonctionnement de la commande `man` ?
5. Créez une variable `i` en lui affectant une valeur quelconque Affichez la valeur de cette variable.
6. À partir du shell dans lequel vous avez créé cette variable `i`, créez un autre terminal. La variable `i` est-elle connue dans ce terminal ? Comment faire pour qu'elle soit connue ?
7. Définissez une variable dont la valeur est une chaîne de caractères contenant le caractère `$`. Affichez ensuite sa valeur. Que constatez-vous ? Comment faire pour résoudre ce problème ?

8. Définissez 2 variables `i` et `j` en leur affectant des chaînes de caractères quelconques. Définissez une variable `k` dont la valeur est la concaténation de `i` et `j`.

9. Définissez les variables `m` et `n` de valeur `2` et `3`. Exécutez une commande qui affiche la somme de ces 2 variables (pensez à utiliser la commande Unix `expr` ou aux fonctionnalités internes du shell).