Initiation à l'informatique

Enseignants : S. Mavromatis – P. Banton. Support originellement rédigé par N. Baudru

TP Linux

I) Connexion et déconnexion sous Linux

Pour démarrer une session de travail à partir de l'invite de connexion, vous devez vous connecter au système en précisant votre nom d'utilisateur (**login**), puis votre mot de passe associé. Le bouton *option* en bas à gauche vous permet de sélectionner un environnement graphique pour votre session. Nous vous proposons dans un premier temps de vous familiariser avec l'environnement graphique KDE. Vous aurez par la suite tout le loisir de découvrir par vous-même les autres environnements disponibles si vous le souhaitez. Choisissez donc KDE, puis connectez-vous.

II) Découverte du système de fichiers Linux via KDE

Ouvrez le gestionnaire de fichiers nommé **Dolphin** puis familiarisez-vous avec la souris :

1) À quoi sert le bouton gauche, le bouton droit et le bouton du centre lorsque vous cliquez sur le bureau, sur un dossier ou sur un fichier ?

Explorez l'arborescence Linux :

2) En utilisant l'onglet *aller* dans la barre de menu, déterminez dans quel répertoire (aussi appelé dossier) est contenu votre répertoire principal de travail $*HOME*, où $*HOME* est votre login ?

3) Quels répertoires et fichiers sont contenus dans votre répertoire $*HOME* ? (n'oubliez pas d'afficher les fichiers "cachés", i.e. les fichiers commençant par un *"."*)

4) Retournez dans votre répertoire $*HOME*. À l'aide du menu contextuel, créez puis supprimez un répertoire et un fichier dans $*HOME* (Attention de ne pas effacer n'importe quoi !)

5) Ensuite créez à partir de votre répertoire *$HOME* l'arborescence suivante :

- Initiation_Informatique
  - Bureautique
  - Linux
    - TP1
- Introduction_Programmation

Pensez toujours lors de vos différents TPs à bien gérer et classer vos fichiers pour pouvoir facilement les retrouver par la suite. Le nombre de fichiers que vous allez créer va croître très rapidement au cours des différents travaux que vous devrez réaliser à l'école lors de ces trois prochaines années. Vous vous apercevrez vite qu'une bonne organisation est indispensable.

III) Observation des processus via KDE

Un aspect important d'un système d'exploitation multi-utilisateur et multitâches tel que Linux est la notion de processus. En simplifiant un peu, un processus est un programme qui s'exécute dans une zone mémoire qui lui est réservée. Chaque processus est indépendant et exécute une tâche bien précise.

1) À partir du menu "K" (bouton en bas à gauche), ouvrez une application permettant de surveiller le système et d'afficher les processus actifs en cours.

Une fois l'application lancée, une liste de processus apparaît. Plusieurs colonnes sont intéressantes à analyser.

2) Quelle colonne décrit

- à quel utilisateur appartient un processus ?
- à quelle application un processus correspond ?
- l'identifiant PID d'un processus dans la machine ? (il faut faire apparaître cette colonne).

Sous Linux, les processus sont toujours créés à partir d'un autre processus. Les deux seuls processus créés initialement, i.e. au démarrage du système d'exploitation, sont les processus *idle* (PID 0, non visible) et *init* (PID 1). Le premier sert à gérer les temps d'inactivité du processeur et le second sert à créer les premiers processus nécessaires au système. Ces derniers vont eux-aussi créer des processus. Nous obtenons donc une structure arborescente où chaque processus à un père (le processus qui la créé) et un ou plusieurs fils (les processus qu'il a créés).

3) Dans le gestionnaire de processus, trouvez comment afficher :

- l'arborescence des processus,
- uniquement vos processus.

4) Lancez plusieurs éditeurs de texte et navigateurs internet. À l'aide du gestionnaire de processus, observez les processus créés correspondant aux diverses applications que vous venez de lancer. Une actualisation de l'affichage est peut-être nécessaire.

5) Dans le gestionnaire de processus, cliquez avec le bouton droit de la souris sur un des processus que vous venez de créer, puis envoyer le signal "terminer" (signal SIGTERM). Que s'est-il passé ? Essayez ensuite le signal "tuer" (signal SIGKILL). Comparez les deux options.

IV) Premier contact avec l'interpréteur de commande

Les interpréteurs de lignes de commande (ou shells) tels que bash, font partie des programmes les plus stables qui puissent tourner sur votre machine. Leur avantage principal est qu'ils permettent de travailler plus rapidement qu'à partir d'un environnement graphique. Nous présentons ici quelques commandes simples.

1) Trouvez comment lancer un shell à partir de l'environnement graphique (indice : un shell s'exécute généralement dans un "terminal" ou "console").

Au lancement du shell, une ligne terminant par "$" apparaît. C'est l'invite de commande (ou prompt). Celle-ci indique à l'utilisateur que le shell est en attente d'une commande. Si l'invite de commande n'apparaît pas, c'est que votre shell est déjà occupé à exécuter une autre commande. Il est donc inutilisable jusqu'à ce que le prompt réapparaisse.

Une commande est formée d'un ou plusieurs mots séparés par des espaces, et terminée par la touche ENTREE (qui demande au prompt l'exécution de la commande). Voici quelques commandes simples :

2) La commande *pwd* vous permet de déterminer dans quel répertoire vous êtes. Essayez cette commande. Analysez le résultat obtenu en le comparant à celui de votre voisin(e).

3) Déterminez l'utilité des commandes *whoami* et *ps -f -U mettre_ici_votre_login*.

Afin de mieux comprendre les différents résultats obtenus, vous pouvez consulter la documentation fournie par le système. L'accès aux informations concernant une commande s'effectue à l'aide de la commande *man* suivie du nom de la commande en question. Notez que pour sortir des pages de documentation, vous devez appuyer sur la touche "q".

4) Consultez les informations concernant les commandes *pwd, whoami* et *ps*.

Nous allons maintenant nous fixer un petit objectif : créer un alias, c'est-à-dire créer une nouvelle commande "courte" qui se comporte comme une commande plus longue. Utiliser des alias à comme principal avantage de gagner du temps (il y a moins de lettres à taper pour lancer une commande).

5) Que fait la commande *ls* ? Vérifiez que son résultat est correcte en utilisant le gestionnaire de fichier de KDE.

Nous allons maintenant rajouter des options à la commande *ls* pour en modifier son comportement. Les options sont le plus souvent des lettres précédées du signe "-".

6) Essayez les commandes *ls -a*, *ls -al* puis *ls -al –color=auto*. Comparez le résultat de ces trois commandes. Puis consultez les informations fournies sur la commande *ls* et vérifiez que vos déductions précédentes sur le résultat des différentes options de *ls* étaient correctes.

Nous allons maintenant créer un alias à la commande *ls -al –color=auto*. Pour cela il faut utiliser la commande *alias*.

7) Chercher sur internet comment créer un alias sous linux.

8) Créez un alias appelé *llc* se comportant comme la commande *ls -la –color=auto*.

9) Vérifiez que votre commande *llc* produit bien le même résultat que *ls -la –color=auto*.

10) Tapez dans le shell la commande *exit*. Que s'est-il passé ?

11) Ouvrez un nouveau shell et essayez de nouveau votre commande *llc.*

Normalement, votre commande *llc* précédemment définie ne devrait plus marcher. En effet, toutes les configurations faites dans un shell ne sont valables que dans ce shell. Lorsque vous fermez le shell puis en ouvrez un autre, le nouveau shell lancé est différent du précédent. Pour que les configurations de votre shell soient communes à tous les shells (et donc permanentes), il faut modifier le fichier de configuration des shells appelé *.bashrc* (si vous utilisez un shell bash). Nous verrons cela dans un prochain TP.

V) Se déplacer dans une arborescence

1) Créez un répertoire TP2 à l'intérieur du répertoire Linux. Vous devriez obtenir l'arborescence suivante :

- *Introduction_Informatique*
  - *Bureautique*
  - *Linux*
    - *TP1*
    - *TP2*
- *Introduction_Programmation*

Notez que les répertoires *Introduction_Informatique* et *Introduction_Programmation* se situent dans votre répertoire principal *$HOME* (ou *$HOME* est votre nom de login).

2) Ouvrez ensuite un terminal (shell). Vérifiez à l'aide de la commande *pwd* que vous vous situez bien dans votre répertoire *$HOME*. Puis vérifiez à l'aide de la commande *ls* que ce répertoire contient les répertoires *Introduction_Informatique* et *Introduction_Programmation.*

La commande *cd* permet de se déplacer dans une arborescence, c'est-à-dire entrer dans un répertoire ou sortir d'un répertoire. Pour entrer dans un répertoire, il suffit de taper la commande *cd* suivie du nom du répertoire cible. Pour sortir d'un répertoire, il suffit de tapez la commande *cd* suivie de "..".

3) Entrez dans le répertoire *Introduction_Programmation.* Vérifiez que votre commande a bien fonctionné à l'aide de la commande *pwd*.

4) Sortez du répertoire *Introduction_Programmation*. Vérifiez que votre commande a bien fonctionné à l'aide de la commande *pwd*.

5) Placez-vous maintenant dans votre répertoire TP2 en passant successivement par les répertoires *Introduction_Informatique* et *Linux*. Vérifiez que votre commande a bien fonctionné à l'aide de la commande *pwd*.

VI) Création de fichiers

1) Placez-vous dans le sous-répertoire TP2 puis créez un fichier vide *bonjour* à l'aide de la commande *touch*(n'hésitez pas à consulter le man si nécessaire).

2) Vérifiez que le fichier créé est bien vide à l'aide de la commande *cat*.

3) Écrivez la phrase "Hello World" dans le fichier *bonjour*. Pour cela tapez la commande *cat* > *bonjour* puis écrivez la phrase. Terminez en tapant simultanément sur les touches Ctrl et D .

4) Affichez le contenu du fichier *bonjour* à l'aide de *cat*.

5) Ajoutez la phrase "How do you do ?" à la fin du fichier *bonjour*. Pour cela tapez la commande *cat >> bonjour* puis écrivez votre texte. Terminez en tapant simultanément sur les touches Ctrl et D .

6) Vérifiez que tout c'est bien passé en affichant de nouveau le contenu du fichier *bonjour*.

7) À l'aide de la commande *cp*, copiez le fichier *bonjour* dans un nouveau fichier *salut* (toujours dans le répertoire TP2). Vérifiez que le contenu du fichier *salut* et identique à celui du fichier *bonjour* en utilisant la commande *diff.*

8) Tapez la commande *cat > salut*, puis une nouvelle phrase. Terminez l'exécution de cette commande en utilisant de nouveau la combinaison des touches Ctrl et D. Affichez le contenu de *salut*. Que s'est-il passé ? Quelle est donc la différence entre ">" et ">>" ?

Les symboles ">" et ">>" sont appelés des symboles de redirection. nous en reparlerons un peu plus loin.

VII) Notion de filtre de fichiers

Un filtre permet de définir un ensemble de fichiers à manipuler. Il est constitué de caractères "normaux" et de caractères "spéciaux" appelés *caractères de filtrage*. Les caractères de filtrage sont ?, [,], ^, et *. Dans une suite de caractères :

- ? représente 1 caractère quelconque
- [a,d,f,g] représente 1 caractère au choix entre a, d, f ou g
- [^a,d,f,g] représente 1 caractère au choix qui n'est pas a, d, f ou g
- [a-f] représente 1 caractère au choix compris entre a et f
- [^a-f] représente 1 caractère au choix non compris entre a et f
- * représente n'importe quelle chaîne de caractères.

Par exemple, pour lister tous les fichiers du répertoire courant (celui où vous êtes) dont le nom commence par *fic* suivi d'un caractère quelconque, il faut utiliser la commande *ls fic?*. Lorsque cette commande est interprétée, le caractère ? est remplacé par n'importe quel caractère.

1) Comment afficher tous les fichiers dont le nom commence par une chaîne de caractères quelconque, suivi d'un a ou d'un m, suivi de nouveau d'un a ou d'un m, et termine par une suite de caractères quelconque ?

2) Créez dans le répertoire TP2 un ensemble de fichiers vous permettant de vérifier votre réponse.

3) Placez-vous maintenant dans le répertoire */bin* en tapant la commande *cd /bin*. Affichez tout le contenu de ce répertoire à l'aide de *ls.* Ensuite listez seulement les fichiers :

- dont le nom commence par a
- dont le nom commence par une voyelle
- dont le nom contient sh
- de 3 lettres qui se terminent par sh

4) Placez-vous maintenant dans le répertoire */sbin* (tapez *cd /sbin)* puis affichez les noms des fichiers dont la 4ième lettre est a, b, c ou d.

5) Revenez dans votre répertoire courant en tapant simplement *cd*. Puis placez-vous de nouveau dans votre répertoire TP2.

VIII) Nom de fichiers et caractères spéciaux

Certains caractères sont dits "spéciaux" car ils sont utilisés dans le processus de filtrage précédent. Cependant, rien n'empêche d'utiliser ces caractères dans un nom de fichier. Mais il faut alors faire attention… Les questions suivantes illustrent certains problèmes pouvant se produire.

1) Dans *TP2*, créez trois fichiers de nom *a*b*, *acb* et *addb*. Exécutez la commande *ls a*b*. Comprenez le résultat.

2) Exécutez maintenant les commandes *ls a*b* et *ls a'*'b*. Expliquez.

3) La commande *mkdir* permet la création de répertoires. En utilisant cette commande, donnez trois façons de créer un répertoire dont le nom est *cou\cou*.

4) Créez un fichier dont le nom est *mon fichier.*

IX) Compression/Décompression de fichiers

1) Créez un répertoire TP3 à l'intérieur du répertoire Linux. Vous devriez obtenir l'arborescence suivante :

- *Introduction_Informatique*
  - *Bureautique*
  - *Linux*
    - *TP1*
    - *TP2*
    - *TP3*
- *Introduction_Programmation*

2) Placez-vous dans le répertoire *TP3.*

De nombreux fichiers sont disponibles sur le web dans des formats compressés. Ils prennent ainsi moins de place et sont donc plus faciles à télécharger. Deux formats très répandus sont les fichiers compressés en zip ou gzip. Ces fichiers sont reconnaissables par leur nom terminant par l'extension .zip ou .gzip. Les commandes pour décompresser ces types de fichiers sont respectivement *unzip* et *gunzip*. Inversement, les commandes permettant de compresser des fichiers aux formats zip ou gzip sont respectivement *zip* et *gzip*.

3) À l'aide du *man*, déterminez comment utiliser ces commandes.

4) Ouvrez à partir d'un terminal un navigateur web. Il suffit pour cela de taper le nom de l'application que vous voulez lancer. Par exemple il faut taper dans le terminal la commande *firefox* pour lancer le navigateur portant le même nom. Ensuite téléchargez dans votre répertoire *TP3* le mémo unix/linux

disponible sur la page web de cet enseignement.

5) Déterminez le poids du fichier téléchargé. Puis compressez le fichier téléchargé (i.e. le mémo unix) au format zip. Quel est le poids du fichier ainsi obtenu ? Décompressez ce fichier puis compressez-le au format gzip. Quel est le poids du fichier ainsi obtenu ?

Une autre extension souvent rencontrée sur le web est l'extension *.tar*. Un fichier possédant une extension .tar est souvent appelé une archive *tar*. Une archiver tar est simplement une concaténation "intelligente" d'un ensemble de fichiers ordinaires contenus par exemple dans un répertoire. Les fichiers tar sont ensuite généralement compressé.

6) Examinez à l'aide du *man* la commande *tar*.

7) Archivez votre répertoire TP3 au format .tar. Puis, listez le contenu de votre archive tar. Enfin restituez votre répertoire TP3.

X) Les droits d'accès

Avant de répondre aux questions suivantes, lisez les transparents 115 à 123 des diapositives (version complète) disponibles sur la page web de cet enseignement.

1) Dans votre répertoire *TP3,* créez l'arborescence suivante en utilisant uniquement des commandes shell :

- RepertoireTestDroit (avec les droits rwx rwx rwx)
  - *monFichierPerso* (avec les droits rw- rw- rw-) contenant le texte "Mon Secret"
  - Sous Repertoire 1 (avec les droits rw– rw– rw–)
    - fic (avec les droits rw– rw– rw–) contenant le texte "Hello Word"
  - Sous Repertoire 2 (avec les droits –x –x –x)
    - fic (avec les droits rw- rw- rw-) contenant le texte "Salut l'ami"

2) Parmi les commandes suivantes, quelles sont celles peuvant être utilisées à partir du répertoire courant *RepertoireTestDroit* ?

- ls ./Sous Repertoire 1
- ls -l ./Sous Repertoire 1
- cd ./Sous Repertoire 1
- cat ./Sous Repertoire 1/fic
- touch ./Sous Repertoire 1/fic2
- rm ./Sous Repertoire 1/fic
- ls ./Sous repertoire 2
- cd ./Sous repertoire 2
- cat ./Sous Repertoire 2/fic
- touch ./Sous Repertoire 2/fic2
- rm ./Sous Repertoire 2/fic
- echo "Comment vas tu" >> ./Sous Repertoire 2/fic

Qu'en déduisez-vous ?

3) Un élève décide de protéger son fichier *monFichierPerso* (contenant des informations importantes et confidentielles) des autres utilisateurs en lui affectant les droits — — —. Qu'en pensez-vous ?

XI) Les alias permanents

Dans cette dernière partie, nous allons voir comment créer des alias permanents. Pour ce faire, il faut utiliser les fichiers de configuration du shell bash (le shell normalement utilisé par défaut sous Linux). Ces fichiers de configuration sont simplement des fichiers textes contenant des commandes shell. Ils sont au nombre de trois : .bash_profile, .bashrc et .bash_logout. Le fichier qui nous intéresse ici est .bashrc.

1) Vérifiez que les fichiers .bash_profile et .bashrc sont bien présents dans votre répertoire $HOME. S'ils sont présents, ne les modifiez pas et passez directement à la question suivante. Sinon vous devez les créer, les éditez, puis les compléter de la manière suivante :

Le fichier *.bash_profile* doit contenir au minimum les quelques lignes suivantes :

```
#$HOME/.bash\_profile
if [ -f ~/.bashrc ]; then
        source ~/.bashrc
fi
# End $HOME/.bash\_profile
```

Et le fichier *.bashrc* les lignes suivantes :

```
#$HOME/.bashrc
if [ -f /etc/bashrc ]; then
        source /etc/bashrc
fi

# c'est ici que vous devez définir vos alias

# End $HOME/.bashrc
```

L'explication de ces lignes est hors de portée de ce TP. Nous y reviendrons lors du TP sur les scripts shell.

2) Pour créer des alias permanents il vous suffit de les définir à la fin du fichier *.bashrc.* Définissez dans le fichier *.bashrc* l'alias *llc* vu à la question 19. Vérifiez que l'alias est bien permanent.

XII) Gestion des processus

De manière assez simplifiée, un *processus* peut être vu comme un programme en cours d'exécution. Il faut bien faire la différence entre un programme et un processus. Un programme est une entité passive enregistrée quelque part (sur votre disque dur par exemple) qui ressemble en tout point à un fichier "normal". À ce stade un programme n'est ni plus ni moins qu'une suite de bits. Lorsque vous lancez un programme (ou une commande), il est stocké dans la mémoire (la RAM) de l'ordinateur. L'UC va alors lire la première instruction du programme, la décoder et effectuer l'action correspondante. Puis elle passe à

l'instruction suivante et recommence… Lors de l'exécution du programme, les cases mémoires où sont stockées votre programme peuvent changer de valeur. Ainsi, un processus évolue dans l'ordinateur. Ce n'est pas le cas d'un programme.

À partir d'un terminal, un processus peut être lancé de deux façons différentes :
- au premier plan (fg). Pour cela, il suffit de taper le nom du programme dans un terminal. Par exemple : *Emacs*
- en arrière-plan (bg) (on dit aussi en tâche de fond). Il faut alors taper le nom du programme suivi du symbole &. Par exemple : *Emacs &*

Lorsqu'un processus s'exécute au premier plan, le terminal devient inactif puisqu'il est occupé par ce processus. Au contraire, si le processus est lancé en arrière-plan, le terminal reste disponible (le prompt s'affiche de nouveau). Cette seconde solution permet de lancer plusieurs processus à partir du même terminal.

Il est possible de faire passer un processus du premier plan en arrière-plan, et vice-versa :

- Si le processus tourne au premier plan :
  - CTRL-Z le suspend : le processus est "en pause" ;
  - CTRL-C l'interrompt : le processus est arrêté brutalement.
- Lorsque le processus est suspendu (i.e. après CTRL-Z), un prompt s'affiche à l'écran. Alors :
  - la commande *fg* passe ce processus au premier plan ;
  - la commande *bg* passe ce processus en arrière-plan.
- Si le processus tourne en arrière-plan :
  - si c'est le seul processus lancé à partir du terminal, alors *fg* le passe au premier plan ;
  - sinon, il faut d'abord afficher la liste des processus lancés à partir de votre terminal à l'aide de la commande *jobs*, puis taper la commande *fg numJob*, où *numJob* est le numéro correspondant à votre processus donné par *jobs*.

1) Lancez *Emacs* au premier plan à partir d'un nouveau terminal. Faîtes-le passer en arrière-plan puis affichez la liste des processus liés à ce terminal (commande *ps*).

2) Déterminez ce que fait l'option -l de la commande *ps*. Quel est le processus père du processus correspondant à *Emacs* ? Repassez finalement *Emacs* au premier plan.

3) Lancez un nouveau terminal, puis la commande *ps -Af* (voir le man pour comprendre les options A et f). Identifiez les processus correspondant à ce nouveau terminal et au terminal précédent. Tuez le processus correspondant au premier terminal à l'aide de la commande *kill –9 PID*, où *PID* est l'identifiant du terminal à tuer. Que se passe-t-il?

4) Relancez un nouveau terminal et lancez *Emacs* en arrière-plan dans ce nouveau terminal. Vérifiez que son processus père est bien celui du terminal à partir duquel vous l'avez lancé. Tuez ce nouveau terminal (celui à partir duquel *Emacs* a été lancé). Peut-on toujours accéder à *Emacs* ? Qu'en déduisez vous ? Quel est le nouveau processus père d' *Emacs* ?

XIII) Redirections

Certains caractères spéciaux sont appelés *caractères de redirection*. Ils permettent de rediriger l'entrée standard (généralement le clavier) et la sortie standard (généralement l'écran) d'une commande vers des fichiers. Ainsi, la commande utilisera les données du fichier plutôt que celles provenant du clavier, et enregistrera le résultat dans un fichier plutôt que de l'afficher à l'écran. Les principaux caractères de redirections sont les suivants :

- *commande* > *fic* : permet d'enregistrer le résultat de *commande* dans le fichier *fic*. Le contenu de *fic* est écrasé.
- *commande* >> *fic* : rajoute à la fin du fichier *fic* le résultat de *commande*.
- *commade* < *fic* : les données contenues dans *fic* sont utilisées par *commande*
- *commande1* | *commande2* : le résultat obtenu par *commande1* est utilisé comme données par *commande2*. Le symbole "|" s'appelle un *pipe* ou *tube de redirection*.

1) Lancez la commande *ls -l*. À l'aide d'une redirection, stockez le résultat de cette commande dans un fichier *toto*.

2) À l'aide d'une redirection non écrasante écrivez le résultat de la commande *ps -A* à la suite du fichier *toto*. Vérifiez que tout s'est bien passé en affichant le fichier *toto*.

3) À l'aide d'une redirection, créez un fichier *tata* contenant la liste des processus tournant sur le réseau (utilisez *ps -Af*).

4) À l'aide du man, déterminez ce que font les commandes *wc*, *sort*, *head* et *tail*. Testez-les ensuite avec leurs différentes options. Par exemple :

- comptez le nombre de mots contenus dans votre fichier *bonjour* ;
- comptez le nombre de caractères contenus dans votre fichier *bonjour* ;
- comptez le nombre de processus dans *tata*
- triez les lignes du fichier *tata* dans l'ordre alphabétique par rapport au premier champ
- extrayez les 10 premières lignes du fichier *toto*
- extrayez les 5 dernières lignes du fichier *toto*

5) En utilisant les commandes *tail, head* et *sort* ainsi que deux pipes, enregistrez dans un fichier *titi* les lignes 4 à 23 du fichier *tata* triées suivant le nom des propriétaires des processus.

XIV) Recherche d'une chaîne de caractères dans des fichiers textes

La commande *grep* permet de rechercher une chaîne de caractères (souvent appelée motif) dans les lignes d'un fichier texte. La commande *grep* est indispensable pour rechercher des informations, notamment dans les fichiers de configuration.

1) À l'aide de la commande *man*, déterminez quelles options il faut utiliser pour que *grep* :

- donne seulement le nombre de lignes trouvées contenant le motif
- donne seulement le nom des fichiers dans lesquels le motif a été trouvé
- donne les lignes où le motif n'a pas été trouvé
- ne tienne pas compte de la casse (ne différencie pas les majuscules des minuscules)
- impose que le motif corresponde à un mot entier sur une ligne du fichier

2) Lisez dans le *man* la partie correspondant aux expressions régulières (aussi appelées modèles de chaînes). Le concept d'expression régulière est différent du concept de filtrage.
Quelles sont ces différences ? Regardez en particulier les symboles "." et "?" et "*".

3) Déterminez ensuite ce que produisent les commandes suivantes :
- grep "^.o" tata
- grep -ni bash tata
- grep -ni "td$" tata

4) Comment sélectionner les lignes du fichier *tata* commençant par un "r" et terminant par un chiffre ?

5) En combinant les commandes *ps* et *grep* avec un pipe, affichez uniquement les processus lancés par vous.

6) En combinant les commandes *ps*, *grep* et *wc* avec des pipes, déterminez le nombre de processus lancés par vous.

XV) Recherche de fichiers dans une arborescence

1) Recherchez à l'aide du man et du web quelles sont les différences entre les deux commandes de recherche de fichiers *locate* et *find*. Entraînez-vous à les utiliser. Par exemple recherchez la commande *ls* et le fichier *titi*.

2) Simulez (grossièrement) la commande *find* à l'aide des commandes *ls* et *grep*.

XVI) Les variables d'environnement

Tout cet exercice doit être réalisé à partir du même terminal.

Une variable est définie par son nom et sa valeur. Le nom d'une variable est une suite de caractères. Par exemple Ma_Variable. La valeur d'une variable est obtenue en faisant précéder le nom de la variable par le symbole $. Par exemple $Ma_Variable.

1) À l'aide de la commande *echo* et du symbole *$*, affichez la valeur des variables d'environnement suivantes *HOME, USER, DISPLAY, TERM, HOSTNAME, SHELL, PATH* (Exemple : echo $HOME). A quoi correspondent-elles ?

2) Faites la distinction entre les commandes suivantes:

- *touch fichier $USER*
- *touch "fichier $USER"*
- *touch 'fichier $USER*'

Déduisez-en l'utilité de ' et de ".

Il est possible de créer de nouvelles variables d'environnement ou de modifier celles existantes. Pour cela il suffit de définir leur nouvelle valeur. Par exemple, si on écrit dans un terminal *TEMP0=bonjour*, on crée une nouvelle variable *TEMP0* qui a comme valeur la chaîne de caractères *bonjour*.

Attention : il ne doit pas y avoir d'espace autour du signe =.

3) On va maintenant manipuler quelques variables.

- Créez une variable TEMP2 ayant comme valeur la chaîne de caractères : *ma première variable*.
- Créez une variable TEMP3 qui prend comme valeur la valeur de la variable PATH.
- Vérifiez que TEMP3 et PATH ont bien la même valeur à l'aide de la commande *echo*.
- Donnez à PATH une nouvelle valeur, par exemple "salut". Vérifiez que tout s'est bien passé.
- Essayez maintenant de lancer la commande *ls*. Que se passe-t-il ?
- Redonnez à PATH sa valeur initiale. Celle-ci est enregistrée dans TEMP3.
- Essayez de nouveau de lancer la commande ls. Qu'en déduisez vous ?

XVII) Un premier script

Un script shell est un fichier texte qui peut être interprété par le shell. Un script étant un fichier en mode texte, il doit être créé avec un éditeur de texte. Un script est une suite d'instructions élémentaires qui sont éxécutées de façon séquentielle (les unes après les autres) par l'interpréteur de script (ici c'est le shell lui-même). La première ligne réfère toujours à un shell (dans les exemples qui suivent le shell choisi est le bash).

1) Placez-vous dans votre répertoire *TP3*. Ouvrez un éditeur de texte en arrière-plan puis écrivez les lignes suivantes dans l'éditeur :

```
\#!/bin/bash
echo "Bonjour $USER"
echo "Ceci est mon premier script"
```

2) Sauvegardez ce fichier sous le nom de *bonjour*, puis ajoutez lui le droit en exécution (x). Vous venez d'écrire un premier script nommé *bonjour*.

3) Voyons maintenant comment l'exécuter. Il suffit pour exécuter un script de taper son nom (comprenant le chemin d'accès) dans le terminal. Par exemple, vous pouvez taper :
- ./bonjour si vous êtes dans le répertoire *TP3,*
- ./TP3/bonjour si vous êtes dans le répertoire *Linux*,
- ~/Linux/TP3/bonjour de n'importe où, …

4) Comprenez ce qui s'est passé lors de l'exécution.

XVIII) Les arguments des scripts

Lorsqu'on demande à un shell d'exécuter une commande (ou un script), il sépare le nom de la commande (ou du script) des arguments (i.e. tout ce qui suit le nom de la commande, séparés par un ou plusieurs espaces). En bash, les arguments de la ligne de commande sont automatiquement stockés dans des variables spéciales :

- $0 représente le nom de la commande ou du script
- $1, $2, … représentent les arguments
- $# représente le nombre d'arguments

- $* représente tous les arguments

1) Créez un script qui affiche son nom, ces quatres premiers arguments et son nombre d'arguments.

2) Créez un script qui prend comme argument le chemin d'accès vers un répertoire. Ce script doit simplement afficher le contenu du répertoire passé en argument. Testez votre script.

3) Créez un script qui prend comme arguments le nom d'un nouveau répertoire et le nom d'un fichier. Ce script crée le nouveau répertoire donné en argument dans votre répertoire courant, puis crée dans ce nouveau répertoire le fichier donné en argument. De plus, le script interdit à tous les utilisateurs l'accès à ce fichier en écriture et en exécution. Testez votre script.

XIX) Les conditionnelles dans les scripts

L'instruction *if* permet d'exécuter des instructions si une condition est vraie. Sa syntaxe est la suivante :

```
if [ condition ]
then
    action
fi
```

action est une suite de commandes quelconques. On peut aussi utiliser la forme complète :

```
if [ condition ]
then
    action1
else
    action2
fi
```

Attention : les espaces autour des crochets sont obligatoires !

Le shell étant souvent utilisé pour manipuler des fichiers, il offre plusieurs opérateurs permettant de vérifier diverses conditions sur ceux-ci : existence, dates, droits. D'autres opérateurs permettent de tester des valeurs, chaînes ou numériques. Quelques opérateurs :

- -e filename : vrai si filename existe. Exemple : [-e ./toto]
- -d filename : vrai si filename est un répertoire. Exemple : [-d /bin]
- -f filename : vrai si filename est un fichier ordinaire. Exemple : [-f ./tata]
- -r filename : vrai si filename est accessible en lecture (r). Exemple : [-r ./bonjour]
- num1 -eq num2 : vrai si num1 = num2. Exemple : [$nb -eq 12]

Exemple :

```
\#!/bin/bash
if [ $1 -eq 0 ]
then
 echo $2
else
 echo $3
fi
```


Ce script utilise trois arguments. Si le premier argument est égal à 0 alors on affiche le second argument, sinon on affiche le troisième argument.

1) Testez ce script puis modifiez-le afin de bien comprendre le fonctionnement de l'instruction *if*.

2) Écrivez un script qui prend en argument le nom d'un fichier. Si ce fichier existe, il affiche son contenu, sinon il renvoie un message d'erreur "le fichier n'existe pas". Testez votre script.

XX) Les boucles dans les scripts

La boucle *for* permet d'exécuter une suite d'instructions avec une variable parcourant une suite de valeurs. Par exemple :

```
\#!/bin/bash
for x in un deux trois quatre
do
    echo x= $x
done
```

Ici, l'instruction echo *x=$x* est exécuté une fois pour chaque valeur possible de x. Ce script affiche donc à l'écran les lignes suivantes :

x= un
x= deux
x= trois
x= quatre

Le nom de la variable doit être écrit entre le *for* et le *in* alors que la suite de valeurs que prend successivement la variable doit être placée entre le *in* et le *do*. Les instructions à répéter sont placées entre le *do* et le *done*.

On utilise fréquemment *for* pour énumérer des noms de fichiers contenus dans un répertoire. Par exemple le script suivant affiche tous les fichiers de votre répertoire courant :

```
\#!/bin/bash
for fichier in ./*
do
       echo "$fichier"
done
```

On peut aussi utiliser une boucle pour traiter les arguments passés sur la ligne de commande :

```
\#!/bin/bash
for arg in $*
do
  echo $arg
*done*
```

Nous rappelons que $* représente tous les arguments passés sur la ligne de commande.

1) Testez tous ces programmes puis modifiez-les afin de bien comprendre leur fonctionnement.

2) Écrivez un script qui, pour chaque fichier du répertoire courant, indique si c'est un répertoire ou non.

3) Écrivez un script qui reproduit le comportement de *ls -R* en utilisant uniquement les commandes *ls, cd* et *echo*.

Remarque : `*ls*` entre ` permet de récupérer le résultat de l'exécution de la commande *ls.* Par exemple *echo ls* affiche à l'écran "ls" alors que *echo* `*ls*`affiche à l'écran le résultat de *ls*.