Stage Unix
2008-2009
—Licence informatique 3ème année—

Unix

Le support de cours se trouve a l'adresse suivante : `http://www-igm.univ-mlv.fr/~beal/Enseignement/Unix/toutunix.ps.gz`.
Il est conseillé d'écrire les réponses aux exercices dans un fichier séparé.

Exercice 1. *(Au commencement, il y avait who, man ...)*

1. La commande *man* permet, entre autres, d'obtenir des informations sur une commande unix (astuce : cette commande documente également les fonctions du langage C). Vous pouvez quitter une page man en tapant "q".
   Indiquez ce que font les commandes *who*, *date*, *echo*, *ls*.
2. La longueur d'une page man peut facilement décourager l'étudiant innocent. Cependant, une commande utile peut sauver du temps : taper "/" suivi d'un mot effectue une recherche de ce mot dans cette page. Utilisez n (resp. p) pour obtenir les résultats suivants (resp. précedents).
   Trouvez dans la page de man de *ls* l'option permettant d'obtenir l'affichage long ligne par ligne, et celle permettant de trier selon la taille des fichiers.
3. La touche du clavier ↑ permet dans un terminal de remonter dans les commandes précédements tapés. La touche tabulation fait de l'auto-complétion. Nous n'avons pas trouvé d'exercice étant donné la difficulté de ces commandes.

Exercice 2. *(Naviguer dans l'arborescence)*

1. Utilisez les commandes *pwd*, *cd*, *ls*. A quoi servent-elles ? Trouvez à quoi correspondent "/", ".", "..", "~".
2. Essayez de dessiner sur une feuille l'arborescence des fichiers, en débutant à la racine.
3. Sous Unix, les fichiers et répertoires cachés débutent par un ".". Placez-vous à la racine de votre répertoire. Trouvez l'option de *ls* permettant d'afficher les fichiers et répertoires cachés. La commande *clear* (ou son raccourci Ctrl+l) permet de "nettoyer" votre terminal.

Exercice 3. *(Gérer ses fichiers)*

1. Par respect pour l'arborescence de votre compte étudiant, créez un répertoire "Stage" et placez vous y.
2. Sans paramètre, la commande *touch* positionne la date de dernière modification à l'heure courante. Si le fichier n'existe pas, un fichier vide est créé. Créez un fichier vide du nom qu'il vous plaira. Maintenant, supprimez-le.
3. Créez un répertoire du nom qu'il vous plaira, puis supprimez-le.
4. Créez un fichier "fichier" avec un peu de texte dedans, $echo\ "votre\ texte" > fichier$ fera l'affaire. Renommez ce fichier en "fichier1" (Renommer un fichier, c'est comme déplacer ce fichier avec un autre nom).
5. Copiez "fichier1" en "fichier2". La commande *cat* affiche le contenu d'un fichier, que constatez-vous ?
6. Faites un lien de "fichier1" vers "fichier3". Que constatez-vous avec *cat*, avec $ls - li$? Ajoutez encore du texte dans "fichier1", $echo\ "qsdf" >> fichier1$. Quand est-il du contenu de "fichier2" ? de "fichier3" ?
7. Faites un lien symbolique "lien1" vers "fichier1". Quel est le contenu de "lien1" ? Supprimez "fichier1". Quel est maintenant le contenu de "lien1" ?

Exercice 4. *(Les éditeurs de texte)*

1. Ouvrez votre programme favori parmis $emacs, gedit, vi, kate, autre$ (astuce, un non-programme s'est glissé dans cette liste).
2. Créez un fichier que vous nommerez "votrelogin.txt". Vous y écrirez les réponses aux questions.

   Trouvez les raccourcis clavier suivants :

   (a) Sauvegarder un fichier

   (b) Charger un fichier

   (c) Rechercher un mot dans le texte

   (d) Copier/Couper/Coller

Exercice 5. *(Vous avez le droit de ...)* Sous Unix, tout est fichier (fichiers standards, répertoires, périphériques ...). Le système affecte des droits à ces fichiers pour trois catégories d'utilisateurs : vous, votre groupe, le reste du monde. Pour chaque catégorie, il existe trois types de droits : lecture 4, écriture 2 et exécution 1.

1. Selon vous, qui fait partie de votre groupe ? Comment pouvez-vous trouver cette information ?
2. Quels sont les droits sur le fichier "fichier2" (si vous ne l'avez pas effacé, sinon, n'importe quel autre fera l'affaire) ?
3. Quels sont les droits sur le fichier "/etc/passwd" ? Pouvez-vous écrire dedans ? Pourquoi ?

4. Créez un répertoire et modifiez ses droits pour que les gens de votre groupe puisse y écrire des données. (Faites des tests avec vos voisins)
5. Retirez le droit d'exécution pour les gens de votre groupe. Qu'arrive-t-il à votre voisin lorsqu'il essaye d'écrire dans ce répertoire.
6. Allez regarder les droits associés à la racine de votre compte ? Est-ce bien sécurisé ?
7. Utilisez la commande *umask* pour que par défaut, les utilisateurs du "reste du monde" n'aient pas le droit en lecture, ni en exécution sur vos nouveaux fichiers.

Exercice 6. *(Les processus)*

1. Ecrivez, compilez et lancez un petit programme C effectuant une boucle infinie.
2. Vous pouvez garder la main sur votre terminal en ajoutant "&" à la fin de la commande. Si vous n'y aviez pas pensé (personne n'est parfait), tapez "Ctrl+C".
3. La commande *ps* indique les processus courants. Repérez votre programme. Ouvrez un nouveau terminal et tapez encore *ps*. Pourquoi votre programme n'apparait-il pas ? Trouvez comment le faire apparaitre.
4. Terminez votre programme à l'aide de *kill*.
5. Relancez votre programme et quittez le terminal. Votre programme s'execute t-il toujours ?
6. Relancez votre programme en ajoutant *nohup* avant (et "&" après). Quittez le terminal. Votre programme s'execute t-il toujours ? Tuez votre programme et soulagez votre processeur.
7. A quoi sert la commande *top* ?

Exercice 7. *(Pipe et les commandes pour faire des combos ...)* Le "pipe" (ou "tube") permet de rediriger la sortie standard d'une commande vers l'entrée standard d'une autre commande. On l'utilise avec le caractère |. Par exemple, la commande *more* affiche ce qu'elle lit sur son entrée standard, avec arrêt du défilement en bas de page. La commande *ls* affiche ses résultats sur la sortie standard.

1. Tapez la commande $ls - lR|more$. Expliquez ce qu'il s'est passé.
2. A l'aide de la commande *wc*, indiquez le nombre d'etudiants dans votre filiere
3. Plaçons-nous dans un cas concret ! Vous êtes informaticien pour la sociologie des usages (sisi, ça existe..). Vous avez "aspiré" les commentaires du site web flickr.com. Une partie de ces commentaires se trouve dans le fichier `http://tinyurl.com/flcomments`. Téléchargez-le mais ne faites pas l'extraction. Ceci est un échantillon, il s'agit de 10.000 commentaires effecutés sur les premières photos ce site. Chaque ligne du fichier correspond à un commentaire. Les colonnes correspondent aux champs suivants :

– date d'aspiration
– identifiant photographe
– identifiant photo
– identifiant commentateur
– date du commentaire
– texte du commentaire

Le séparateur entre les colonnes est TAB.

Les lignes sont pour l'instant triées par identifiant photo. Du coup, collector, vous avez tous les commentaires effecutés sur la première photo publique publiée sur ce site, la photo "74", `http://www.flickr.com/photos/12037949754@N01/74`. Par la même occasion, vous voyez comment voir une photo si ça vous amuse :
`http://www.flickr.com/photos/`$<$ *idphotographe* $>/<$ *idphoto* $>$.

Pour les questions suivantes, vous allez avoir besoin des commandes *zcat*, *cut*, *wc*, *grep*, *head*, *tail*, *uniq*, *sort*, *tr*.

En aucun cas, vous ne devez extraire le contenu du fichier.

(a) Affichez le contenu du fichier. Quels est l'avantage d'effectuer cet affichage sans extraction préalable ?

(b) N'affichez que les 5 dernières lignes, en évitant de lire la dernière.

(c) Trouvez et affichez les commentaires contenant le mot Flickr, ou le mot Paris (sans la majuscule, on a une surprise), ou le mot face, ou tout autre selon votre inspiration.

(d) Affichez uniquement le texte des commentaires.

(e) Affichez uniquement les lignes "destination - source". En passant, ceci suffirait pour contruire le graphe des commentaires de Flickr. Si vous avez fait un peu de travail sur les graphes, pensez aux nombreuses applications possibles. Sinon, contentez vous d'acquiescer d'un mouvement de tête convaincu et convainquant. Partie bonus, avec *awk*, affichez le graphe "source - destination", bien plus joli.

(f) Affichez ce même graphe, mais trié selon l'identifiant du photographe.

(g) La même chose, mais sans doublon.

(h) Comptez le nombre de photos

(i) Affichez les lignes complètes, mais triés selon la date du commentaire.

(j) Question ultra-bonus qui vous donne la licence directement (ou pas) : faire la liste des mots qui apparaissent dans les commentaires avec leurs fréquences (trié par fréquences décroissantes of course).

Exercice 8. *(Redirections)* Les commandes lisent les données sur l'entrée standard (le clavier par défaut), affichent les résultats sur la sortie standard (le terminal par défaut). Les erreurs sont envoyés sur la sortie d'erreur (le terminal

également par défaut). On peut toutefois rediriger ces entrées sorties à l'aide de ces opérateurs :

- > Redirige la sortie standard
- >> Redirige la sortie standard en fin de fichier (n'écrase pas si existe déjà)
- < Redirige l'entrée standard
- 2 > Redirige la sortie d'erreur

1. La commande *cat* écrit sur la sortie standard ce qu'elle lit sur l'entrée standard. Que fait la commande $cat < /etc/passwd > \$HOME/monpwd$?
2. Executer une nouvelle fois cette commande. Le fichier à t-il changé ? Réessayer en utilisant >> au lieu de >. Expliquer.
3. Executer $cat < \$HOME/monpwd > \$HOME/monpwd$. Expliquer.
4. Lancer *ls* avec l'option de listage récursif pour lister l'ensemble des fichiers du répertoire /etc. Rediriger cette liste vers la poubelle (/dev/null). Qu'est-ce qui est affiché ?

Exercice 9. *(Compresser des données à la Cesar)*

1. Faites une archive de votre répertoire "Stage" au format *.tar.gz*
2. Faites une archive de votre répertoire "Stage" au format *.tar.bz2*

Exercice 10. *(Et path le ch'min)*

1. Modifiez la variable d'environnement *PATH* pour que votre super programme de boucle infinie soit exécutable de n'importe où ?
2. Pourquoi si vous ouvrez un autre terminal, cela ne fonctionne plus ? Faites les modificitations nécessaires pour que cela marche.
3. Définissez un alias "lla" qui définisse un "ls" qui affiche les fichiers cachés au format long.
4. Modifiez (ou creez si nécessaire) le fichier *.bashrc* se trouvant à la racine de votre compte pour que cela fonctionne aux prochains démarrages de votre session.
5. Que contient le fichier *.bash_history* également à la racine de votre compte ?
6. Affichez toutes les variables d'environnement. Affichez uniquement celles dont les valeurs contiennent "fr".
7. Changez la langue des pages man (regardez la variable d'environnement "LC_ALL").

Exercice 11. *(L'art de la commande find)*

1. Comment chercher tous les fichiers commençant par un "a" majuscule ou minuscule, suivi d'éventuellement quelques lettres ou chiffres, et se terminant par un chiffre entre 3 et 6 ?

2. Comment fait-on pour indiquer que le fichier recherché a été modifié il y a plus de 30 jours ? Il y a 30 jours ? Il y a moins de 30 jours ?
3. Un jour viendra où vous ne pourrez plus ouvrir votre session car vous aurez dépassé le quota d'espace disque. Pour prévenir ce jour malheureux, ayez le bon réflexe. Recherchez sur votre compte la liste des fichiers ayant une taille supérieure à 1Mo (ou plus, ou moins, suivant la notion que vous vous faites de "gros fichier"
4. C'est le nettoyage de printemps, utilisez la commande *find* pour retrouver tous les fichiers de nom a.out, tous les fichiers dont le nom se termine par ~ et tous ceux dont le nom commence par #. Supprimez-les, sans utiliser le symbole | (uniquement les paramètres de la commande *find*)

Exercice 12. *(Quelques notions de réseau)*

1. Vérifiez que la machine "étudiant" est en vie. Rassurez-vous ou affolez-vous selon la réponse. (mais en silence svp)
2. Étudiez le chemin pour aller de la l'université jusqu'à *www.google.fr* (pensez à une commande que l'on vient de présenter). Envoyez la sortie de cette commande dans un fichier. Relancez la commande en envoyant la sortie dans un autre fichier. Comparez les deux fichiers avec la commande *diff*. Déduisez. Trouvez comment afficher ces deux fichiers en deux colonnes pour mieux comparer.

Exercice 13. *(Internet sans X)*

1. Téléchargez l'archive du dernier noyau (`http://www.eu.kernel.org/pub/linux/kernel/v2.6/linux-2.6.26.tar.bz2`) sans ouvrir de navigateur. Plusieurs méthodes sont valides.
2. Imaginez que l'histoire du quota dépassé se soit produit (à cause de la grosse archive). Avant d'aller voir les gens du CRI (de 12h à 13h30), vous pouvez essayer de résoudre le problème vous-même. Pour vous mettre en condition, vous pouvez utiliser un terminal sur la machine de votre voisin. Connectez vous à votre compte en ftp pour supprimer l'archive récalcitrante que vous venez de télécharger.

Exercice 14. *(Programmation shell)* Si vous connaissez déjà un langage de programmation, vous verrez, c'est facile. Sinon, vous verrez, c'est facile.

Le shell (sur vos machine, le bash) est un interpréteur de commande. C'est-à-dire qu'il peut lancer des processus (les commandes que vous utilisez depuis le début du stage), mais aussi exécuter des instructions propre à son langage.

Vous pouvez écrire la suite de commandes dans un fichier, d'extension *sh*. Le fichier devra débuter par la ligne $\#!/bin/sh$, pour lui indiquer avec quel interpréteur interpréter la suite.

1. Une varible est un conteneur nommé pour une donnée. Définissez une variable et affichez ensuite son contenu, la syntaxe pour la définition est la suivante : $var = data$ (sans espaces). Utilisez le symbole "$" pour l'accès à la donnée.
2. Vous pouvez tester l'existence d'un fichier par exemple avec
   $if\ [\ -e\ mon\_fichier\ ]\ ;\ then\ fais\_quelquechose\ ;\ fi.$
   Écrivez l'instruction permettant de tester l'existance du répertoire "tp_l3" à la racine de votre compte et de le créer s'il n'existe pas. (*man test*) Voir sur cette page pour plus d'informations sur les tests : `http://fr.wikibooks.org/wiki/Programmation_Bash_Tests`
3. La boucle for :
   $for\ name\ in\ word\ ;\ do\ fais\_quelquechose\ ;\ done.$
   À l'aide d'une boucle for, compressez tous les fichiers commençant par "fi" dans le répertoire courant. Voir plus d'informations sur les boucles ici : `http://fr.wikibooks.org/wiki/Programmation_Bash_Boucles`
4. Petit jeu. Ecrivez un script qui permet de dire à l'utilisateur s'il est proche (brulant), éloigné (gelé), etc. d'un nombre mystère. Vous pouvez mettre en place un système de *cheat code*, i.e. quand le code est saisie l'utilisateur gagne automatiquement ! Astuces : la commande *read truc* attends que l'utilisateur entre une chaine de caractères sur l'entrée standard de manière bloquante, et la place dans la variable truc. Ecrire $\$((variable))$ interprète la valeur numérique contenue dans *variable* (et $\$((variable - 5))$ effectue la soustraction).
5. Ecrivez un script sh pouvant prendre en paramètre 1 ou 2 fichiers (disons a et b). S'il n'y a qu'un argument, il faudra utiliser pour b un fichier fixé en "dur" dans le script. Si le nombre d'argument est incorrect, il faudra afficher un texte précisant l'usage du script. En sh, on accède aux arguments par la variable $ suivit de son numéro (le premier argument est $1). Leur nombre est déterminé par $#. . Le programme devra tester l'existance de chacun de ces fichiers (pensez à utiliser une fonction). La valeur de retour d'une fonction se teste avec $?.

Exercice 15. *(Utile pour sa survie universitaire)* Le contenu de cet exercice est réutilisable pour la plupart des logiciels sous linux

1. Télécharger et installer xtris (`http://www.iagora.com/~espel/xtris/xtris.html`). Quelques principes généraux pour installer un logiciel sous linux.
   – Lire le README, il y a souvent toutes les étapes qui y sont décrites.
   – Si un script "configure" existe, le lancer. S'il y a trop d'informations, utilisez un pipe et *less*. En général, ce script prépare le programme pour être installé comme si vous étiez administrateur de la machine. Pour l'installer en local sur votre répertoire, l'option "- - prefix repertoire" est bien utile.

– En général, il faudra ensuite lancer *make* et *make install*.

2. Pour jouer avec votre voisin, vous aller avoir besoin de votre adresse IP. La commande $ifconfig$ permet de faire cela. Utilisez la commande $find$ pour savoir où est stocké l'executable $ifconfig$. Qu'en déduire ?
3. Télécharger et installer pidgin (`http://www.pidgin.im/`).

Exercice 16. *(S'il vous en reste..)*

– Rappels de C
– Bibliothèques dynamiques
– makefile
– Initiation à LaTeX
– Une idée ?