Unix : concepts et outils avancés
TD 1

Olivier Aumage
LIP, bureau 302
email : Olivier.Aumage@ens-lyon.fr

27 septembre 2001

1 Introduction

Le but de la série de Travaux Dirigés accompagnant le cours sur les *Concepts et outils avancés d'Unix* est de faire découvrir (et apprécier !) — en encourageant la curiosité à travers l'observation et l'expérimentation — la complexité, bien sûr, mais surtout l'ingéniosité et l'esthétique des fonctionnalités et mécanismes constituant les systèmes UNIX.

Le TD d'aujourd'hui a plusieurs objectifs, principalement destinés à donner une base solide aux TD ultérieurs :
- compléter la prise en main du système informatique, des terminaux et des shells ;
- introduire quelques notions de base sur le fonctionnement et l'exploitation des systèmes de fichiers sous d'UNIX ;
- traiter la gestion des entrées-sorties en **C**.

Le manuel Beaucoup de commandes et fonctions seront utilisées au cours des différentes séances de TD. Une description — parfois un peu synthétique — de ces commandes et de ces fonctions est quasiment toujours disponible par l'intermédiaire de la commande `man`. N'hésitez pas à y faire largement appel (ex : `man man`, `man ls`, `man -s3S printf`, etc.) Il est parfois nécessaire de spécifier une section pour `man` (exemple de `printf` qui est à la fois une commande et une fonction). Les sections usuelles pour les fonctions **C** sont : `2`, `3C`, `3S` et `3N`.

2 Terminal : prise de contact, rappels

La première partie de ce TD présente (ou rappelle) les commandes usuelles disponibles à partir des terminaux.

Répertoires

- Tapez les commandes `ls`, `ls -l`, `ls -a`, et `ls -i`. Combinez les options en tapant `ls -lai`. Expliquez les différentes sorties obtenues.
- Créez un sous-répertoire `Unix` dans votre répertoire `travail` avec les commandes `cd travail` et `mkdir Unix`.
- Tapez `cd Unix ; ls -la ; cd .. ; ls -l`. Que représentent les répertoires '.' et '..' ? Utilisez la commande `pwd` pour connaitre votre répertoire courant.
- Explorez le système. Regardez par exemple ce qui se trouve à la racine (répertoire `/`). Revenez à *votre* répertoire racine en tapant `cd`.
- Comparez `ls -l travail/Unix` et `ls -ld travail/Unix`. Quel est l'effet du `-d` ?

Les fichiers

- Allez dans votre répertoire `travail/Unix` et créez un fichier `test.txt` avec votre éditeur de texte favori (`emacs`, `xemacs`, `vi`, etc.) Utilisez votre imagination pour écrire quelques lignes dans votre fichier et enregistrez-le (`CTRL-x` puis `CTRL-s` avec `emacs`).
- Visualisez le contenu du fichier avec les commandes `cat`, `tac`, `more` et `less` (`man` si vous voulez en savoir plus sur ces commandes).
- Renommez votre fichier avec la commande `mv test.txt test_2.txt`. Observez le résultat avant et après avec `ls` et `ls -i`.
- Créez un répertoire `Test` ainsi qu'un sous-répertoire `Test/Tst` et déplacez-y le fichier avec la commande `mv test_2.txt Test/Tst/`. Tapez `ls -l ; ls -l Test/Tst/`.
- Tapez `ls -Rlai`. Observez les numéros des *inodes* dans la colonne de gauche. Chaque fichier est associé à un inode (nœud d'information).
- Déplacez/renommez le répertoire `Tst` avec la commande `mv Test/Tst Nouveau_Tst`. Utilisez `ls -Rlai` pour observer que les *inodes* n'ont pas changé.
- Dupliquez le fichier avec la commande `cp Nouveau_Tst/test_2.txt test_3.txt`. Utilisez `ls -Rlai` pour vérifier que le nouveau fichier a bien un nouveau numéro.
- Créez un lien physique sur le fichier `test_3.txt` avec la commande `ln test_3.txt phy_ln.txt`. Comparez les *inodes* de ces deux fichiers. Observez également la seconde colonne du résultat de `ls -l`.
- Créez un lien symbolique sur le fichier `test_3.txt` avec la commande `ln -s test_3.txt sym_ln.txt`. Comparez les *inodes* de ces deux fichiers. Observez aussi comment ce nouveau fichier apparaît au travers des commandes `ls` et `ls -l` et les modifications éventuelles au niveau de `test_3.txt`.

Les droits d'accès aux fichiers

Commande `chmod`

- Tapez `ls -l` dans le répertoire `travail/Unix/Test`. Observez les droits des fichiers et des répertoires. Ils sont classés dans l'ordre {utilisateur, groupe, autres}, `r` étant le droit de lecture, `w` le droit d'écriture et `x` celui d'exécution.
- Tapez `chmod -r test_3.txt ; ls -l test_3.txt ; more test_3.txt`. Restaurez l'ancien état avec `chmod +r test_3.txt`.
- Tapez `chmod -w test_3.txt ; ls -l test_3.txt`. Essayez de modifier le fichier via l'éditeur de texte. Restaurez l'ancien état par `chmod +w test_3.txt`.
- Dupliquez le fichier `test_3.txt`, supprimez les droits d'écriture et supprimez le fichier avec la commande `rm`. Les droits d'écriture n'empêchent pas la suppression.
- Tapez `chmod -w . ; ls -l`. Essayez de créez un nouveau fichier dans le répertoire `Test`. Essayez de créez un nouveau fichier dans le répertoire `Nouveau_Tst`. Restaurez l'ancien état par `chmod +w .`
- Tapez `chmod -x Nouveau_Tst ; cd Nouveau_Tst`.
- Tapez `less Nouveau_Tst/test_2.txt`. Restaurez l'ancien état par `chmod +x Nouveau_Tst`.
- Comment sont gérés les droits sur les liens physiques ?
- Créez à votre racine un répertoire `incoming` où n'importe quel membre de votre groupe puisse y déposer des fichiers mais sans pouvoir lire le contenu ni du répertoire ni des fichiers déposés. Vous devez disposer de tous les droits sur ce répertoire et les utilisateurs qui ne font pas partie de votre groupe ne doivent disposer d'aucun droit.

Commande `umask`

- La commande `umask` contrôle les droits par défaut des fichiers crées. Sa syntaxe est `umask 0ooo` avec `0ooo` un nombre octal commençant par un 0 et comprenant les droits *ugo* avec `read=4`, `write=2`, `execute=1`.
- Tapez `umask` pour voir votre masque courant.
- Tapez `umask 124 ; mkdir T ; ls -l`.

- Créez et sauvegardez un nouveau fichier texte avec un éditeur de texte lancé depuis le `shell`. Observez ses droits.
- Créez et sauvegardez un nouveau fichier texte mais avec un éditeur de texte lancé depuis un autre `shell`.
- Restaurez votre ancien masque (probablement par `umask 022`).

Listes de contrôle d'accès (SOLARIS)

- SOLARIS dispose de commandes permettant de régler plus finement les droits d'accès. Utilisez la commande `man` pour connaître le fonctionnement de `setfacl` et `getfacl`. Ces commandes prennent tous leur intérêt lorsqu'on utilise des outils de travail coopératif tels que CVS (*Concurrent Versions System*).
  Le premier usage de `setfacl` (avec l'option `-s` pour `set`) sur un fichier donné doit spécifier tous les champs de la liste de contrôle d'accès. Les modifications ultérieures (avec l'option `-m` pour `modify`) peuvent ne concerner qu'une partie des champs.
  Créez un répertoire où seulement une autre personne du cours a les droits en lecture, les autres personnes n'ayant que les droits de lecture et d'exécution (bien entendu, vous ne devez pas supprimer vos propres droits !).

Les processus

- Tapez `ps`, puis `ps -ef`, `ps -elf`.
- Lancez le programme `/home/oaumage/boucle`. Ce programme est une boucle infinie : `for(;;);`.
- Interrompez le programme avec `CTRL-c`.
- Relancez le programme et interrompez-le par `CTRL-z`. Tapez `ps`. Relancez le programme par `fg` (*foreground*) puis interrompez-le à nouveau par `CTRL-z`. Tapez `bg` (*background*). Maintenant, le programme s'exécute en *tâche de fond*.
- Tuez le programme par la commande `kill -KILL` *numéro du processus*. Le numéro de processus est obtenu par la commande `ps`. Appuyez sur `ENTER`.
- Lancez le programme en tâche de fond par la commande `/home/oaumage/boucle &`.
- Lancez la commande `top`. Appuyer sur la touche `h` pour accéder à l'aide. Tuez le processus en tâche de fond à l'aide de `top`. Quittez `top` (touche `q`).

3 Entrées-sorties en C

La seconde partie du TD est consacrée à une introduction aux fonctionnalités d'*entrées-sorties* en **C**.

Principes

Entrées-sorties standards

- Comparez les résultats de `printf("tic");`, de `fprintf(stdout, "tic");` et de `fprintf(stderr, "tic");`.
- Comparez les résultats de `{fprintf(stdout, "tic"); while(1);}` et de `{fprintf(stderr, "tic"); while(1);}`.
- Comparez les résultats de `{fprintf(stdout, "tic"); fflush(stdout); while(1);}` et de `{fprintf(stderr, "tic"); while(1);}`.
- Comparez les résultats de `{fprintf(stdout, "tic\n"); while(1);}` et de `{fprintf(stderr, "tic\n"); while(1);}`.
- Comparez `scanf("%d", &x);` et `fscanf(stdin, "%d", &x);`.

Les objets `stdin`, `stdout` et `stderr` sont respectivement appelés *entrée standard*, *sortie standard* et *sortie standard d'erreur*.

Fichiers Les objets *Fichiers* (le terme exact est *stream*) ont pour type `FILE *`. Quel est le type de `stdin`, `stdout` et `stderr` ? (cf. /usr/include/stdio.h)

- Créez un fichier à partir d'un programme à l'aide de `f = fopen("fichier", "w");`. Cette fonction retourne un objet de type `FILE *`. Ecrivez quelques lignes dans le programme en utilisant `fprintf` avec comme premier paramètre `f`. N'oubliez pas de fermer le fichier avec `fclose(f);`.
- Que se passe-t-il si vous exécutez plusieurs fois votre programme ? Remplacez `w` par `a` dans la commande `fopen` et réitérez l'experience.
- Le *mode* `w` correspond au mode de création de fichier (*write*) et le mode `a` est le mode d'ajout (*append*). Il est egalement possible d'ouvrir un fichier en mode lecture avec la lettre de mode *r* (*read*). Créez deux programmes `p1.c` et `p2.c`. Le programme `p1.c` demande un entier `n` à l'utilisateur puis écrit les entiers de `0` à `n - 1` dans un fichier `fichier_int` (avec éventuellement des informations supplémentaires si nécessaire). `p2.c` doit être capable de lire la suite d'entiers écrite dans `fichier_int` et de la stocker dans un tableau de `n` entiers alloué dynamiquement avec `malloc`, puis d'afficher le contenu du tableau.

Quelques exemples supplémentaires

- Écrivez un programme qui affiche le nombre de lignes d'un fichier. Comparez le résultat de votre programme avec la commande `wc -l fichier` accessible par le shell.
- Les arguments passés en ligne de commande à votre programme sont accessibles par l'intermédiaire de la fonction `main` lorsqu'elle est déclarée de la manière suivante : `int main(int argc, char *argv[])`. Le paramètre `argc` correspond au nombre d'arguments présents sur la ligne de commande (plus 1 pour le nom du programme lui-même). Le paramètre `argv` contient les arguments sous la forme de chaînes de caractères : `argv[0]` est le nom du programme, `argv[1]` est le premier argument... Modifiez votre programme pour que le nom du fichier à analyser puisse être passé en paramètre.
- Modifiez votre programme pour afficher également le nombre de mots et comparez avec `wc fichier`