ASR2 Utilisation des systèmes Informatiques TP6

Département Informatique – IUT Bordeaux 1 – 2005-2006

Objectifs : le but de cette séance est de vous familiariser avec le langage de commandes (shell) et les processus sous UNIX.

1 Les alias

Tapez les commandes **alias**, **man alias** et **help alias** et essayez de comprendre à quoi sert la commande **alias**.

1. Essayez la commande **myls**
2. Tapez **alias myls="ls -l"** puis essayez à nouveau la commande **myls**.
3. Trouvez à l'aide de la commande **man** les options permettant la confirmation (interactive) des actions des commandes **rm** et **mv.**
4. Testez les commandes **rm** et **mv** avec ces options en paramètre. Evitez de supprimer des fichiers importants !
5. Définissez des alias pour les commandes **rm** et **mv** de sorte que leurs actions soient confirmées par l'utilisateur.
6. Remarquez que les ''vraies'' commandes rm et mv restent accessible par **\rm** et **\mv**.
7. Vous remarquerez que ces alias ne sont pas définis dans les autres consoles (xterm) et pour de nouvelles sessions. Comment faire pour définir pour vous ces alias partout et tout le temps ?
8. Indication : voir le fichier script de configuration personnalisé **~/.bashrc** du langage de commande shell **bash**. Pour l'exécuter, tapez **source ~/.bashrc**
9. Après avoir résolu la question précédente, terminez la session et redémarrez-en une nouvelle, pour voir si vos alias sont maintenant pris en compte.

Important

Il arrive parfois de détruire par erreur des fichiers (ça n'arrive pas qu'aux autres :)) Vous pouvez les retrouver dans le répertoire **/net/miroir** qui contient les sauvegardes quotidiennes des 3 dernières semaines. Allez-donc y faire un petit tour ...

2 La gestion des processus

Tout programme UNIX en train de s'exécuter s'appelle un ***processus***. Votre shell est donc un processus. Un processus (B) lancé à partir d'un processus (A) est un fils de A. On dit aussi que B est un sous-processus de A, père de B.

Lorsque vous lancez une commande **ls** à partir de votre shell, vous lancez un processus fils du shell qui exécute la commande demandée. Tout processus (père) communique à sa descendance (ses fils) un héritage constitué notamment des variables d'environnement (voir exercice suivant). Un processus est identifié par un numéro (PID=Process IDentifier) qu'il peut connaître ; il peut également prendre connaissance du numéro de son père (PPID=Parent PID).

Empilement de shells

1. Empilez plusieurs shells en tapant successivement **bash, bash, tcsh, csh, sh, ...** et observez la parenté des différents processus grâce aux commandes **ps -l** et **pstree -p -u <user>**
2. Dépilez les shells, en utilisant plusieurs fois et successivement les commandes **exit** et **ps -l** afin de vérifier que vous êtes revenu dans votre premier shell dans les liens de parenté.

Processus en tâche de fond (background)

Avec la syntaxe habituelle, lorsque vous exécutez une commande à partir du shell, le shell est bloqué en attendant que la commande (processus fils du shell) ait fini son exécution. Il est possible d'exécuter une commande en *tâche de fond* ou arrière-plan à partir du shell sans pour autant que ce dernier soit bloqué. Il suffit de le lui signaler en ajoutant le symbole **&** à la fin de votre commande.

Par exemple, comparez **xterm** et **xterm &**

Remarque :

C'est pourquoi il est pratique d'utiliser **emacs ... &** au lieu de **emacs ....** L'éditeur de textes reste ainsi disponible en permanence en même temps que la fenêtre xterm (console) à partir de laquelle on a lancé emacs.

3 Les variables du shell (bash)

Sous votre shell, il est possible de définir des variables. Par exemple, dans votre environnement est définie la variable **PWD** contenant le nom complet de votre répertoire de travail en cours. La liste de toutes les variables définies s'obtient par la commande **set**. Toutes les variables ne sont pas exportables vers un processus fils. Celles qui le sont sont appelées aussi **variables d'environnement**.

Valeur des variables

1. Exécutez les commandes **echo PWD** et **echo $PWD**. Qu'en déduisez-vous ? Comment accède-t-on au contenu d'une variable ?

2. Que font les commandes

- **MYVAR=PWD**
- **MYVAR=$PWD** ?

3. Affichez le contenu de la variable **PATH**. Sauvegardez **SPATH** (S, pour Sauvegarde) et vérifiez la valeur sauvegardée. Mettre ensuite la chaîne vide dans **PATH** et essayez les commandes **emacs**, **cd**, **pwd**, **ls**. Que peut-on en conclure ? [1]

Comment lancer (avec succès) la commande **ls** sans redonner à **PATH** sa valeur d'origine ? Modifier la valeur de la variable **HOME** en lui affectant le nom absolu de votre répertoire de travail en cours et exécutez la commande **cd** sans arguments. Résultat produit ? Restaurez les valeurs de **PATH** et de **HOME** et vérifiez que tout est « rentré dans l'ordre ».

4. Il est pratique d'avoir un répertoire **~/bin/** contenant des commandes personnelles.

- Créez ce répertoire, placez-y un fichier **hello** contenant la commande **echo Bienvenue $USER** et rendez ce fichier exécutable.

- Comment modifier la variable **$PATH** pour que la commande **hello** fonctionne depuis n'importe quel répertoire ?

- En pratique, quelle différence avec l'utilisation d'alias ?

Visibilité des variables

Les commandes **env** et **printenv** permettent toutes deux de visualiser l'ensemble des variables d'environnement. La commande **set** permet en plus d'afficher les variables non-exportables. Dans une fenêtre **xterm**, définissez deux variables **UN** et **DEUX** prenant respectivement les valeurs 1 et 2. Exécutez la commande **export UN** et

1. affichez le contenu des deux variables d'environnement que vous venez de définir

2. lancez la commande **xterm &** et affichez à nouveau le contenu des deux variables dans la nouvelle fenêtre. Quel est le rôle de la commande **export** ?

1 Les commandes cd et **pwd** sont des commandes internes du programme **bash** qui ne nécessitent pas de consulter la variable **PATH**. La commande (interne) **help** du **bash** vous informe sur les commandes internes.

4 Le cycle de développement

Copiez chez vous le répertoire

/net/exemples/ASR2/TP6/TABLES-1.0

Remise en forme de texte

1. Regardez l'état du source du fichier **tables.cc**. Estimez le temps qu'il vous faudrait, à la main, pour rectifier la présentation afin d'essayer de comprendre le fonctionnement.
2. Vous pouvez le remettre en forme automatiquement par la commande **indent tables.cc**
3. Comparez le style « GNU » produit par la commande précédente avec le style « Kernighan & Ritchie » (option **-kr**).
   - Lequel préférez-vous ?
   - Formulez vos critères de préférence : encombrement, aération, conformité au style enseigné ici, etc.

Makefile

Pour recompiler ce programme, vous pouvez taper

g++ -Wall -o tables tables.cc

ou

make

Que préférez-vous, et pourquoi (répondez franchement) ?

Les messages suggèrent une légère modification. Éditez le texte avec **emacs tables.cc** et sans sortir d'emacs, appuyez sur [F1] (puis retour). Dans la fenêtre **compilation**, un clic sur un message d'erreur vous amène sur la ligne concernée.

La commande **make** est pilotée par le fichier **Makefile** qui décrit des actions à effectuer.

Essayez **make clean**, **make archive**, etc. Essayez de comprendre le fichier **Makefile** et d'y ajouter des **«** cibles » utiles. Par exemple « envoi »' pour expédier l'archive du projet à votre binôme (par courrier en pièce attachée).