Introduction aux systèmes d'exploitation (IS1)

TP 6– Enchaînements de commandes

Exercice 1

Le shell `bash` est un véritable langage de programmation (ou plus précisément un langage de scripts

Valeurs de retour

Exercice 2

Dans l'architecture Unix, un processus qui se termine communique en principe à son environnement, et en particulier à son processus père, une information sur les conditions de son arrêt, que l'on appelle valeur de retour. L'exercice précédent est en particulier un exemple où le shell se comporte différemment suivant la valeur de retour des processus qu'il exécute.

Les valeurs de retour possibles pour les différentes commandes sont en général documentées dans la page de manuel correspondante.

Par convention, une valeur de retour nulle (égale à 0) signifie que l'exécution et la terminaison du processus se sont déroulées normalement. Une valeur strictement positive correspond à un cas particulier ou à une erreur.

1. Dans un terminal, la commande :

   ```
   echo $?
   ```

   permet d'afficher le code de retour de la dernière commande exécutée[1]. Quelle est la valeur de retour de `sleep 2` après avoir attendu de récupérer la main ?
2. Exécutez `sleep 600` et interrompez l'attente avec Ctrl-C. Que dit `echo $?` ? (Tapez à nouveau `echo $?`. Quelle est la nouvelle valeur ? Pourquoi ?)
3. En remplaçant `commande1` et `commande2` au choix par `sleep 2` et/ou `sleep 600` dans ce qui suit, et en utilisant au besoin Ctrl-C pour faire échouer un processus, donnez un tableau décrivant le succès (déroulement normal) ou l'échec pour chacune des commandes suivantes, en fonction du succès et de l'échec de `commande1` et `commande2` :

   ```
   commande1; commande2
   commande1 && commande2
   commande1 || commande2
   ```

   Peut-on comparer ces opérateurs à la logique booléenne ?
4. En déduire une description en quelques mots du comportement de chacune des listes de commandes suivantes :
   commande$_1$; commande$_2$; ... ; commande$_n$
   commande$_1$ && commande$_2$ && ...&& commande$_n$
   commande$_1$ || commande$_2$ || ...|| commande$_n$

Exercice 3 – Enchaînements imposés.

1. La commande `cmp` permet de tester si deux fichiers sont identiques. Créez deux fichiers identiques `pareil` et `meme` et un fichier différent `differe`. Comparez le comportement et la valeur de retour de la commande `cmp` selon qu'elle a été appelée avec deux fichiers identiques ou avec deux fichiers différents.
2. Le message affiché par la commande `cmp` lorsque les deux fichiers sont différents correspond-il à la sortie standard ou à la sortie erreur standard ?
   (Indice : essayez de rediriger la sortie standard ou à la sortie erreur standard pour le voir.)
3. Ecrivez une séquence d'instructions qui compare deux fichiers de votre répertoire courant et affiche "les deux fichiers sont identiques" quand c'est le cas, et n'affiche rien sinon.
4. Réciproquement, écrivez une commande qui affiche "les deux fichiers sont différents" quand c'est le cas et n'affiche rien sinon.

[1] Cette notation sera expliquée en détail dans le TP sur les variables du shell.

5. Combinez ces deux séquences pour afficher la phrase correcte en fonction du résultat de la commande (indice : vous pouvez délimiter une séquence d'instructions à l'aide de parenthèses). Expliquez pourquoi votre solution fonctionne.
6. Même question en faisant en sorte que seule la phrase demandée s'affiche, et jamais le message associé à la commande `cmp`.

Redirections et tubes

Exercice 4 – Enchaînement de commandes et redirections

Comme on a déjà eu l'occasion de le voir, les commandes peuvent être enchaînées grâce aux opérateurs `;`, `&&` et `||`. En combinant ces derniers et des redirections, écrivez une ligne de commande pour exécuter chacune des opérations suivantes :

- Écrire le contenu de votre répertoire de login dans le fichier `replog` et votre identifiant (`id`) dans le fichier `identifiant`.
- Si le fichier `~/toto` existe, afficher son contenu à l'écran, sinon ne rien afficher.
- Écrire dans un fichier `fic` la liste des fichiers (au sens large) de votre répertoire de login et la liste des processus lancés sur votre machine (une seule redirection).

Exercice 5 – Lancer des commandes dans un autre terminal.

Le but de cet exercice est d'exécuter des commandes depuis un terminal dans un autre terminal.

1. Lancez la commande `ps -l`. La commande elle-même est exécutée dans un processus, qui apparaît dans la liste. Identifiez son processus parent. La commande qui est à l'origine de ce processus parent est le shell dans lequel vous tapez les commandes. Cette commande s'appelle `bash` (sur d'autres systèmes, elle pourrait s'appeler `sh`, `ksh`, `tcsh`, etc.)
2. Ouvrez un autre terminal et vérifiez qu'un nouveau processus `bash` a été lancé.
3. Lancez la commande `bash` depuis un terminal et vérifiez qu'elle a généré un nouveau processus, dont le père est le précédent.
4. Le shell lit les commandes que vous tapez sur son entrée standard. Il est donc possible de rediriger cette entrée. Créez un fichier de nom `commande`, contenant uniquement la ligne suivante :
   `ps -l`
   et lancez la commande `bash` en redirigeant son entrée pour qu'elle lise le fichier `commande`.
   Sur le résultat produit, vous pourrez vérifier l'identité du processus ainsi exécuté, et l'identité de son père. Vérifiez que le processus (fils) n'existe plus.
5. À partir d'un terminal, essayez d'exécuter des commandes dans un autre terminal en utilisant un tube (rappel : les tubes ont été vus au TP5).
6. Essayez de voir le résultat des commandes dans le premier terminal. Faites en sorte de pouvoir voir aussi les messages d'erreur.