TP - ASR5 Système d'Exploitation

Shell

2018/02/06

I Programmation d'un shell

Le shell est un programme qui lit une commande sur l'entrée standard et l'exécute. L'une des difficultés est d'analyser la ligne de commande. Le code : cli_skel .c effectue une analyse basique, décompose la ligne de commande et appelle la fonction executevp(). Cette fonction ne fait qu'afficher la tâche à effectuer. Vous devez remplacer cet appel par celui de la commande système execvp().

Q.I.1) - Faire un shell qui permet d'exécuter une commande simple en premier plan par exemple :
- ls −l /tmp/
- emacs

Q.I.2) - Assurez-vous que :

2(a) - Le shell continue d'exister après la fin de la tâche (il faut faire un fork() en plus du execvp()).

2(b) - Si la commande lancée est erronnée, le shell le signale et retourne dans un état convenable.

Q.I.3) - Faire un shell qui permet d'exécuter des commandes en tâche de fond ou au premier plan.
- emacs &

Q.I.4) - Assurez-vous que :

4(a) - Lorsqu'une tâche quelconque se termine, le shell signale la chose imédiatement. Ce signalement doit se faire même s'il y a déjà une tâche en premier plan.

4(b) - La fin d'une tâche ne crée pas de processus *zombie*.

4(c) - La fin d'une tâche lancée en tâche de fond ne débloque pas le shell.

Q.I.5) - Faire un shell qui permet d'exécuter des commandes avec un tube par exemple :
- ls −l | less
- ls | xargs emacs &

Q.I.6) - *(question subsidiaire)* Ajouter la possibilité d'exécuter des commandes avec plusieurs tubes, ou des redirections vers des fichiers.

II Configuration personnelle : retour sur les variables d'environnement

Un utilisateur peut mettre en place des variables d'environnement à chaque invocation de shell : shell de connexion (en mode console par exemple, `Alt+Shift+F1` pour la première console `tty1`), ou de shell graphique (`konsole` sous KDE, `eterm` sous Enlightenment par exemple).

- Qu'est que `$HOME` et `~` ?
- À quoi servent les fichiers `$HOME/.bash_profile` et `$HOME/.bashrc` ?
  Le fichier `$HOME/.profile` ?

> **Solution:** Sourcé lors de l'ouverture d'un shell bash en mode console. Contient en conséquence souvent un `source $HOME/.bashrc`.

Sourcé lors de l'ouverture d'un shell bash graphique (qui n'a pas vocation de connexion).

Si le shell utilisateur est sh, sourcé lors de l'ouverture d'un shell `sh` en mode console.
Pour savoir quel est le shell courant : `$> echo $SHELL`
Pour changer de shell : `$> exec chsh`

Un admin peut changer ce(s) fichier(s) copié(s) par défaut lors de la création d'un user en modifiant les fichiers de `/etc/skel/`.

— Pourquoi est-il conseillé de mettre en tout début de `HOME/.bashrc` :

```
if [[ $- != *i* ]] ; then
        # Shell is non-interactive.  Be done now!
        return
fi
```

Solution:

```
# Test for an interactive shell.  There is no need to set anything
# past this point for scp and rcp, and it's important to refrain from
# outputting anything in those cases.
if [[ $- != *i* ]] ; then
  # Shell is non-interactive.  Be done now!
  return
fi
```

C'est utile quand on invoque des commandes à distance avec `ssh` !

— `PATH`
  — À quoi sert la variable `PATH` ?
  — Comment est-elle utilisée ?

Solution: Rappelons que les chemins traversés le sont dans l'ordre donné, et que c'est différent de `+=$PATH:$HOME/bin` (au moins en temps, voire en effet car pas de "surcharge" d'un nom d'exécutable possible dans ce cas).

  — Créez un répertoire `~/bin` que vous ajouterez en précédent à la variable `PATH`.
  À quoi cela peut servir ?

Solution: PATH+=$HOME/bin:$PATH
- Surcharge de nom d'exécutable par des applicatifs utilisateur.
- Mise en place de programmes utilisateur

  — Si un programme/bibliothèque est disponible en version 3.14 et 3.33.
  Comment l'/les installer de façon générique ?

Solution: On crée un lien symbolique :
$> ln −s applicatif−3.14/ applicatif et le cmake ou autotools exécuté dans `applicatif/`. Il suffira de changer le lien symbolique pour changer de version. C'est ainsi que sont gérées les alternatives pour beaucoup de distribution : `/etc/alternatives/` sous Debian

  — Pourquoi ajouter `.` et `..` à `PATH` ?
  Comment ?

Solution: C'est pratique, mais peut consommer du temps.
Retour sur la récursion.

— On rencontre souvent des fichiers correspondant à `nom_fichier~`, surtout chez les utilisateurs d'`emacs`.
  — Quel est le problème à taper `rm *~` pour supprimer tous ceux qui sont dans le répertoire courant ?
    alias rmc="rm * " pour éviter de purger des répertoires complets, car '*' est très très près de "Entrée". Sauf que si définie ainsi, donne des erreurs si le répertoire courant ne contient pas les fichiers en question. Voir `http://graal.ens-lyon.fr/~ycaniou/Teaching/1718/L3/bashrc`
    Par contre, il faut leur faire comprendre qu'**un alias ne peut être appelé dans un script !** Donc il faut qu'ils comprennent ce qu'ils doivent mettre dans un alias, et définir comme une fonction ou mettre dans un autre script (à sourcer ou pas ; et la différence entre les 2).
  — Si on définit `alias rmc="rm * "`, peut-on écrire un script utilisant `rmc` ?
    Non, car alias. Voir remarque précédente.
— `PS1`
  — À quoi sert `PS1` ?
  — Quelle est la valeur actuelle sur votre terminal ?
  — Est-ce la même dans chaque terminal ?

    Solution: C'est une question ambigûe : elle peut l'être, mais pas forcément.
    Il suffit de changer une affectation dans l'un des terminaux pour que ça ne soit plus le cas.

  — Si on la change sur un terminal, quelle est l'incidence dans un autre ?
  — Configurez selon la ligne suivante :

```
export PS1="\[\033]0;\u@\h:\w\007\]\[\033[01;32m\]\u@\h\[\033[01;34m\] \w \$\[\033[00m\]"
```

    Qu'est-ce que cela fait ?
    Est-ce la seule possibilité de configuration ?
    Comment rendre ce changement permanent ?

    Solution: L'idée de mettre du rouge si root (sur leur machine perso)
    On peut définir les variables suivantes :

```
BLACK='\E[30m'
RED='\E[31m'
GREEN='\E[32m'
MARRON='\E[33m'
BLUE='\E[34m'
MAGENTA='\E[35m'
CYAN='\E[36m'
WHITE='\E[37m'
  NOCOLOR="\E[0m"
```

    Mais le mieux est de leur faire voir qu'on peut même faire de l'inversion vidéo.

— Complétion
  — Qu'est-ce que la complétion et la complétion "intelligente" ?

    Solution: Complétion : permet de compléter les noms de commandes, voire de fichiers, selon l'existant. Cela permet donc de taper sans faire de typos, avec recherche par suffixe.

    Complétion intelligente : permet de compléter les options des commandes et les noms de fichier en relation avec l'application (pdf pour un viewer de pdf par exemple).

    Il faut bien comprendre la différence entre la complétion, et la complétion intelligente

(sur les options, voire les fichiers utilisés ; et montrer les limites de cette complétion intelligente : par exemple, ça peut ne pas fonctionner avec `libreoffice` si les fichiers contiennent des espaces dans les noms.

— On trouve généralement dans le `~/.bashrc` les lignes suivantes. Expliquez.

```
if [ -e /etc/bash_completion ]; then
  source /etc/bash_completion
fi
```

Solution: Il faut s'assurer que le package bash-completion est installé, ou que le fichier `/etc/bash_completion` est installé pour ajouter l'application sans risquer de générer une erreur.

Ici, il faut comprendre la différence entre exécuter un fichier, et le sourcer. (re-, car déjà posé plus haut...)

— Vous savez que votre appareil photo ou votre téléphone inscrit des méta-données dans les photos (comme le type de l'appareil, la date, l'endroit, et les coordonnées GPS). Ces informations peuvent être très pratiques pour des albums personnels, mais certains les ont déjà utilisées pour pister des utilisateurs de facebook, voir où les enfants sont mis en garderie, *etc.* Commentez :

```
function YCremoveExifInfo {
    exiftool -All="" .
}

function YCfb {
    local list1=`ls *jpg 2>/dev/null`
    local list2=`ls *JPG 2>/dev/null`
    for i in ${list1} ${list2}; do
        echo ". Converting $i -> r$i"
        convert ${i} -resize 800x600 r${i}
    done
    echo ". Removing Exif info"
    YCremoveExifInfo
}
```

Solution: Il est important de comprendre les problèmes de sécurité/vie privée générés par les appareils qui ajoutent des informations `exif` dans les photos prises (GPS, nom du matériel utilisé, etc.). Vous pouvez scripter un resize des images pour vos fils d'actualité ou mails, car envoyer des trucs de 4Mo c'est long et inutile. Ces scripts fournissent un exemple qui nécessite `exiftool` et `convert` d'installés.