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

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

- `PATH`
  - À quoi sert la variable `PATH` ?
  - Comment est-elle utilisée ?
  - Créez un répertoire `~/bin` que vous ajouterez en précédent à la variable `PATH`.
    À quoi cela peut servir ?
  - Si un programme/bibliothèque est disponible en version 3.14 et 3.33.
    Comment l'/les installer de façon générique ?
  - Pourquoi ajouter `.` et `..` à `PATH` ?
    Comment ?
- 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 ?
  - Si on définit `alias` rmc="rm * ", peut-on écrire un script utilisant `rmc` ?
- `PS1`
  - À quoi sert `PS1` ?
  - Quelle est la valeur actuelle sur votre terminal ?
  - Est-ce la même sur chaque terminaux ?
  - 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 ?
- Complétion
  - Qu'est-ce que la complétion et la complétion "intelligente" ?
  - On trouve généralement dans le /.bashrc les lignes suivantes. Expliquez.

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

- 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
}
```