Portabilité des scripts

Portabilité des scripts

- Une norme de l'IEEE : Posix
- Un standard de l'Open Group : SUS (Single Unix Specification)
- La version 3 de ce standard (abrégé habituellement en SUSv3) a intégré le contenu de la norme POSIX.
- pour plus d'info :
  - http://www.opengroup.org/standards/unix

Portabilité des scripts

- Un grand nombre de shells et de langages de scripts semble converger vers le standard POSIX.
- On peut appeler Bash avec l'option --posix ou insérer un set -o posix au début d'un script pour que Bash se conforme très étroitement à ce standard.
- Une autre alternative consiste à utiliser, dans le script, l'en-tête #!/bin/sh plutôt que #!/bin/bash pour désactiver les fonctionnalités étendues de Bash.

Portabilité des scripts

- La plupart des scripts Bash fonctionnent directement avec ksh, et vice-versa, car Chet Ramey a beaucoup travaillé sur le portage des fonctionnalités du Kourn aux dernières versions de Bash.
- Sur un UNIX commercial, les scripts utilisant les fonctionnalités spécifiques aux commandes standards GNU peuvent ne pas fonctionner. Ceci devient de moins en moins un problème de nos jours car les outils GNU ont petit à petit remplacé les versions propriétaires même sur les UNIX dits “solides” (commerciaux).

Mécanismes complémentaires

Options du shell

- La commande set
  - set -x : Active le mode “trace”
  - set - : désactive le mode “trace”
  - set -e : quitter immédiatement si le retour d’une commande est différent de 0
  - set -C : interdire l’écrasement de fichiers existants par redirection
  - set -o option : activer certaines options du shell
    - exemple: $ set -o posix
- Désactiver l’option :
  - set +<l’option>
  - ex: $ set +x

- Options du shell :
  - $ bash -x monscript # exécution du script en mode trace
    - affiche les étapes de substitution, signalées par la ligne +
  - $ bash -v monscript # exécution en mode “bavard”
    - affiche la ligne telle qu’elle est écrite dans le script, avant que le shell ne l’interprète
  - $ bash -n monscript
    - vérification de la syntaxe
  - $ bash -e monscript
    - provoque la fin d’exécution d’un script si une commande se termine avec un code de retour différent de zéro

Les options d'un script

- On peut lire les options d'un script en tant que liste d'arguments avec les variables de substitution, mais il existe un moyen plus approprié pour supporter tous les formats des options:
- Commande *getopts* :
  - getopts ":opt1:opt2" variable
  - à chaque appel de cette commande on consomme une option dans $variable
  - $OPTARG contient la valeur de l'option

- Exemple :

```
#!/bin/bash
usage() {
  echo "Usage: `basename $0` -n <5-10> -m <chaîne de caractères>" 1>&2;
  exit 1;
}

while getopts ":n:m:" option; do
    case "${option}" in
        n)
            n=${OPTARG}
            ((n >= 5 && n <= 10)) || usage
            ;;
        m)
            m=${OPTARG}
            ;;
        -)
            [ ${OPTARG%%=*} = “help” ] && usage #option longue
            ;;
        :)
            echo "Error: -${OPTARG} nécessite un argument." 1>&2 && usage
            ;;
        *)
            usage
            ;;
    esac
done
shift $((OPTIND-1))
if [ -z "${n}" ] || [ -z "${m}" ]; then
    usage
fi
echo "n = ${n}"
echo "m = ${m}"
```

tee

- Cet utilitaire peut être imaginé comme un « T » de plomberie, d'où son nom, c'est-à-dire un accessoire que l'on insère dans un pipeline pour disposer d'une sortie annexe. tee lit les données sur son entrée et les copie sur sa sortie standard, tout en envoyant une copie supplémentaire dans tous les fichiers dont le nom lui est fourni en argument.
  - `$ ls | grep 'sh' | tee fichier1 | wc -l`
- Il est également possible d'utiliser le nom de fichier spécial /dev/tty pour obtenir un affichage à l'écran des données qui transitent par tee .

xargs

- Elle prend des lignes de texte sur son entrée standard, les regroupe pour les transmettre en argument à une autre commande. C’est le moyen le plus efficace de transformer des données arrivant dans un flux (de type pipeline ou autre) en arguments sur une ligne de commande.
- Exemple :
  - $ ls | xargs -n 1 grep ’bash’
- Exemple :
  - $ xargs -t -n 2 diff <<fin
    fic1 fic2 fic3
    fic4 fic5 fic6
    fin
    diff fic1 fic2
    diff fic3 fic4
    diff fic5 fic6
- Options principales :
  - -t pour avoir l'écho des commandes
  - -n pour découper le flux d'entrée en paquets

TD/TP 5

1. Écrire un script qui nécessite la saisie d’un nom de fichier complet en option -f (avec son chemin d’accès) et affiche successivement le nom seul et le chemin seul.
2. En utilisant xargs dans un script, cherchez le texte donné en premier argument du script dans tous les fichiers du répertoire donné en deuxième argument.

TD/TP 5

(suite)

3. Écrire un script bash qui prend des options -h ou --host et -p ou --port pour télécharger une page web en utilisant curl ou wget

Devoir: Écrire un script qui prend en options l'url d'un site Web et un port pour télécharger une copie du site entier avec pages HTML, CSS, JS, images …