Travaux pratiques: programmation en Shell

Les exercices sont tous indépendants et peuvent donc être traités dans un ordre quelconque. Pour tous ces exercices, il est important:

- de tester les arguments (nombre, accessibilité, type ...);
- et de rediriger les sorties «inutiles» pour n'offrir à l'utilisateur que les messages émis explicitement par les commandes.

Liste des exercices:

1. Trouver les scripts shell

Écrire un script qui liste les noms de fichiers de type "Shell script" dans une arborescence donnée (par exemple `/usr`). Affichez le nombre de lignes de chaque fichier.

2. Commande which

Écrire un script imitant la commande `which`:

- parcourir les répertoires précisés par la variable PATH et afficher le premier chemin contenant la commande donnée en argument,
- Prendre en compte l'option `a` qui affiche tous les chemins possibles; utiliser pour cela la commande interne `getopts`.
- Analyser le script de la commande `which`.

3. Affichage de la date

La commande `date` affiche la date en anglais ou de manière abrégée. Écrivez une fonction shell `datefr`qui l'affiche complètement.

Exemple:
`Lundi 4 décembre 2006 à 14h`

4. Fonctions liées au système de fichiers

Créer un fichier contenant la déclaration de fonctions offrant une "syntaxe" à la MS_DOS telles que

- `dir [arg...]`: affichage du contenu d'un répertoire (0 ou plusieurs arguments)
- `cwd [arg]`: déplacement dans l'arborescence avec un prompt indiquant le répertoire courant (0 ou 1 argument)
- `ren arg1 arg2`: renommage (2 arguments)
- `del arg [...]`: suppression (au moins 1 argument)
- `typ arg [...]`: affichage de fichiers (au moins 1 argument)

Le Shell exécute t-il en premier ces fonctions ou ses propres fonctions internes? ici on a utilisé les noms `cwd` et `typ` au lieu de `cd` et `type` qui sont des fonctions internes pour éviter toute ambiguïté.

5. Affichage de fichiers avec une bannière

La commande `cat f1 f2 f3 ...` affiche les fichiers `f1`, `f2`, `f3`, etc sans séparer la fin de `fi` du début de `fi+1`. Écrivez un script shell qui ajoute au début et à la fin de chaque fichier une bannière comportant par exemple, la date (en français), le nom de l'utilisateur et le nom du fichier (nom absolu ou relatif).

Indication: utilisez la fonction `datefr` définie précédemment.

6. Renommage interactif de fichiers

Ecrire une commande qui renomme les entrées d'un répertoire. Pour chacune des entrées du répertoire, l'utilisateur doit saisir un nouveau nom s'il souhaite renommer l'entrée; l'entrée est alors renommée. Cette commande doit:

- afficher chaque entrée du répertoire courant (si la commande est appelée sans argument) ou d'un répertoire passé en argument (en vérifiant que celui-ci existe);
- lire la réponse de l'utilisateur : `RETURN` signifiera «pas de renommage du fichier correspondant» et toute autre réponse sera utilisée comme nouveau nom du fichier;
- tester l'existence d'une entrée ayant le même nom que la réponse.

La commande `mv f1 f2` écrase le fichier `f2` si celui-ci existe. Dans ce cas, soit affichez un message et passez au fichier suivant, soit demandez un nouveau nom, soit demandez confirmation.

7. Envoi d'un message

Écrire une procédure `envoi nom fich` qui envoie le contenu du fichier `fich` à l'utilisateur `nom` soit par `write` s'il est connecté et accepte les messages soit par mail.

La commande devra tester la syntaxe d'appel (2 arguments), l'existence du fichier `fich`, la déclaration de l'utilisateur `nom` dans le fichier `/etc/passwd` (une entrée commençant par `nom:`).

8. Recopie d'une arborescence

Écrire une procédure `svgdir r1 r2` qui effectue la copie du répertoire `r1` (qui doit exister) sur un répertoire `r2` (qui ne doit pas exister) en reconstruisant la même arborescence.

Cet exercice fait intervenir la récursivité. Deux méthodes sont possibles, soit se déplacer dans l'arborescence à copier, soit ne pas se déplacer.

9. Jour de la semaine

Écrire une procédure qui affiche à quel jour de la semaine correspond une date donnée.

Exemple : le 14 Juillet 1789 est un mardi.

La date peut être passée comme argument à la commande ou lue au clavier sous une forme à déterminer (14/7/1789 ou 14 7 1789 ou 14 Juillet 1789 ou ...). Vérifiez la validité de la date.

Indication : la commande `cal x y` affiche le calendrier du mois `x` de l'année `y`.

10. Gestion d'un annuaire

On se propose d'écrire un script shell pour gérer un annuaire. Cet annuaire comporte une série de lignes, chacune composée de 5 champs : nom, prénom, numéro de téléphone, bureau et service. Les champs sont séparés par le caractère deux-points comme le montre l'exemple ci-après :

- annuaire.txt

Ce script shell est appelé avec une seule option pour réaliser chacune des fonctions ci-après :

- afficher l'annuaire trié par service et par nom (-t)
- afficher l'annuaire trié sur le numéro de téléphone (-T)
- afficher la liste des noms des inscrits (-M)
- afficher le dernier inscrit (-d)
- afficher la liste des inscrits sous la forme Nom.Prénom (-m)
- rechercher un inscrit à partir de son nom (-g bac) ou d'une partie seulement (-g bou) et sans distinction des majuscules/minuscules
- ajouter un nouvel inscrit (-a)

- créer un fichier par service (-e)
- supprimer un inscrit (-s Bac)
- lister le personnel d'un bâtiment (-b X). Le bâtiment est indiqué par la première lettre du bureau.
- etc (selon votre imagination ou vos besoins).

En outre, le programme doit respecter les consignes ci-après :

- tester la syntaxe d'appel : il faut au moins un argument : le nom de l'annuaire et au maximum une option
- tester si l'annuaire existe et est accessible
- sans option, le script shell affiche simplement l'annuaire trié par nom
- pour l'ajout d'un inscrit, boucler en lecture clavier pour saisir successivement chacun des champs et demander confirmation avant de l'enregistrer
- pour la suppression, l'argument doit représenter exactement le premier champ
- être protégé contre les signaux TERM (N° 15) et INTR (N° 2 et touche CTRL C).

11. Gestion d'un calepin

On se propose d'écrire un script shell pour gérer un calepin de notes. Une note est définie comme étant une seule ligne de texte ASCII et les notes sont numérotées de 1 à n.

Ce script shell offre les fonctions suivantes :

- aide en ligne : liste des commandes avec leur syntaxe (h)
- affichage de la note courante (l) ou de la note x (lx)
- affichage de la note suivante (+ ou Return)
- affichage de la note précédente (-)
- destruction de la note courante (d) ou de la note x (dx)
- ajout d'une note en fin de calepin (a)
- affichage du nombre de notes du calepin (n)
- sortie du programme (q ou Ctrl D)
- affichage du calepin complet (p)
- traitement des notes par contenu (l/exp/ ou d/exp/)
- traitement des notes par intervalles (lx-y ou dx-y)
- etc (selon vos désirs)

Conventions :

- les codes ci-dessus (entre parenthèses) sont des exemples possibles
- les valeurs x et y correspondent à des nombres entiers
- la valeur /exp/ correspond à une expression régulière

Ecrire le programme qui doit :

- tester la syntaxe d'appel : 0 paramètre pour utiliser le nom d'un calepin par défaut ou 1 paramètre : le nom du calepin
- créer un calepin vide ou tester si le calepin est accessible
- afficher soit le nom du calepin créé, soit la dernière note

- afficher un prompt incluant le numéro de note courante
- boucler sur la lecture du clavier pour lire les requêtes de l'utilisateur
- tester la réponse et effectuer le traitement demandé
- être protégé contre les codes intr et quit

Indications :

- définir une note courante : la dernière note affichée
- lorsqu'elle est détruite, la précédente devient la note courante
- écrire chaque traitement demandé dans une fonction
- utiliser des variables communes : nom du calepin, nombre de notes, note courante, réponse, etc...
- pour afficher la note 6, on peut utiliser sed -n 6p calepin
- tester les valeurs limites (début/fin de calepin pour -, +, etc...)
- écrire une fonction vide qui teste si le calepin est vide et répond vrai ou faux Utiliser son résultat pour conditionner l'appel d'une autre fonction
- écrire les fonctions progressivement : commencer par nombre, puis ajouter. Pour liste et detruire, dans un premier temps, se limiter à la note courante.
