Licence d'informatique
Module de C/Unix

`apply` - Application multiple d'une commande

Examen – Partie Unix

Philippe MARQUET

Juin 1996

Les documents de cours et TD sont autorisés.
On rendra deux copies séparées pour la partie Unix et pour la partie langage C. La partie Unix est à rendre sur une *copie de couleur.*

1 Questions de cours

Question 1 (`rehash`)
Chapitre ¡¡ Interpréteur de commandes ¿¿
Donner un extrait de session C-shell nécessitant l'utilisation de la commande `rehash`.

Question 2 (Lien symbolique/lien physique)
Chapitre ¡¡ Système de fichiers — Commandes de base ¿¿
Caractériser la différence entre un lien symbolique et un lien physique.

Question 3 (Substitution des expressions régulières)
Chapitre ¡¡ Interpréteur de commandes ¿¿
Expliquer en quoi il est intéressant que les substitutions d'expressions régulières pour les fichiers soient réalisées par le shell lui-même.

2 Application multiple d'une commande

2.1 Commande `apply`

Nous allons écrire la commande `apply` qui exécute une commande pour chacun des arguments passés à `apply`. La syntaxe de base de `apply` est la suivante :

`apply` *command* *argument*...

L'exemple suivant est équivalent à `ls` :

```
apply echo *
```

L'exemple

```
apply 'cp toto' titi tata
```

est équivalent aux commandes

```
cp toto titi
cp toto tata
```

Question 4 (Algorithme `apply`)
Donner l'algorithme de la commande `apply`.

Question 5 (Commande `apply`)
Donner l'implémentation en Bourne Shell de l'algorithme proposé à l'exercice précédent.

2.2 Option −*number*

L'option −*number* de la commande `apply` permet de spécifier le nombre d'arguments qui doivent être passés à la commande *command* à chaque appel. La syntaxe de la commande `apply` est donc :

`apply` [−*number*] *command* *argument*...

Ainsi, l'exemple suivant compare successivement les fichiers `a1` et `b1`, `a2` et `b2`, etc. :

```
apply -2 cmp a1 b1 a2 b2 ...
```

Si *number* est nul, la commande *command* est exécutée sans argument, une fois pour chacun des arguments. L'exemple suivant exécute 5 fois la commande `who` :

```
apply -0 who 1 2 3 4 5
```

Notons qu'il n'est pas valide de préciser une valeur de l'option −*number* telle que le nombre d'arguments de *argument* ne soit pas un multiple de *number*.

Question 6 (Algorithme de l'option −*number*)
Donner l'algorithme de l'option −*number* de la commande `apply`.

Question 7 (Commande `apply` −*number*)
Donner l'implémentation en Bourne Shell de l'algorithme proposé à l'exercice précédent.

2.3 Séquences %*n*

On étend maintenant la commande `apply` de la manière suivante (tout en gardant l'option −*number*). La commande *command* peut comporter des séquences de caractères de la forme %*n*, *n* étant un chiffre de 1 à 9. Cette séquence est alors remplacée pour chaque exécution de la commande *command* par le *n*-ième argument passé à cette commande. L'exemple

```
apply 'ln %1 /usr/copies' *
```

crée un lien dans le répertoire `/usr/copies` pour chacun des fichiers du répertoire courant. L'exemple suivant

```
apply 'mv %1 %1.old' *.c
```

renomme l'ensemble des fichiers sources C du répertoire courant. L'exemple

```
apply -2 'find %2 -name %1 -print' file1 dir1 file2 dir2 ...
```

recherche le fichier `file1` dans l'arborescence de racine `dir1`, le fichier `file2` dans l'arborescence de racine `dir2`, etc.

Lorsque une séquence %*n* est présente dans la commande *command*, les arguments non référencés par un % ne sont pas passés à la commande *command*. Par exemple,

```
apply -2 'ls %1' tata titi toto tutu
```

est équivalent à la séquence de commandes

```
ls tata
ls toto
```

Notons qu'il n'est pas valide de spécifier un argument %*n* avec une valeur plus grande que la valeur spécifiée pour l'argument −*number* (par défaut 1).

Notons aussi qu'il n'est pas possible d'utiliser le caractère % dans la commande sans qu'il soit interprété par la commande `apply`.

Question 8 (Algorithme de la commande `apply` étendue)
Donner l'algorithme de la commande `apply` étendue par l'emploi de séquences %*n*.

Question 9 (Commande `apply` étendue)
Donner l'implémentation en Bourne Shell de l'algorithme proposé à l'exercice précédent.

2.4 Option `-f`

Nous ajoutons maintenant l'option `-f` (−*file*) qui autorise la lecture des paramètres de la commande à exécuter dans un fichier. La syntaxe de `apply` est alors la suivante :

`apply` [−*number*] *command* *argument*...
`apply` [−*number*] `-f` *file* *command*

La puissance de cette option vient du fait que le fichier dans lequel lire les arguments peut être spécifié comme étant l'entrée standard par l'utilisation du pseudo nom de fichier `-`. Par exemple,

```
find . -name '*.c' -print | apply -f - "grep $string %1 /dev/null"
```

recherche la chaîne de caractères contenue dans la variable `$string` dans l'ensemble des fichiers sources C du répertoire courant et de ses sous-répertoires.

Question 10 (Algorithme de l'option `-f`)
Donner l'algorithme de la commande `apply` étendue par l'option `-f`.

Question 11 (Option `-f` de la commande `apply`)
Donner l'implémentation en Bourne Shell de l'algorithme proposé à l'exercice précédent.