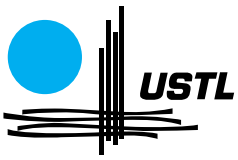

Licence d'informatique
Module de C/Unix

Travaux Pratiques

`apply` — Appliquer une commande

Philippe Marquet

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
```

Exercice 1 (Algorithme `apply`)
Donner l'algorithme de la commande `apply`.

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`*.

Exercice 2 (Algorithme de l'option *`-number`*)
Donner l'algorithme de l'option *`-number`* de la commande `apply`.

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
```

Id : apply.tex,v 1.3 2000/11/28 18 :34 :55 marquet Exp

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.

Il est à noter que les arguments non référencés par une séquence `%`*n* sont passés à la commande comme habituellement :

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

est équivalent à la séquence de commandes

```
ls tata titi
ls toto tutu
```

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 `%` suivi d'un chiffre dans la commande sans qu'il soit interprété par la commande `apply`.

Exercice 3 (Algorithme de la commande `apply` étendue)
Donner l'algorithme de la commande `apply` étendue par l'emploi de séquences `%`*n*.

L'option `-f` (`-`*file*) 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.

Exercice 4 (Algorithme de l'option `-f`)
Donner l'algorithme de la commande `apply` étendue par l'option `-f`.