Erick Gallesio, Stéphane Lavirotte, Jean-Paul Rigault
Polytech'Nice – Sophia
SI3
2017-2018

TD n°5
Primitives de base sur les fichiers/répertoires

Ces exercices utilisent les primitives Posix de manipulation de fichiers et de répertoires vues en cours, plus éventuellement quelques autres dûment signalées. N'hésitez pas à invoquer la commande `man` pour obtenir des informations d'utilisation des fonctions et commandes Shell citées. En cas d'ambiguïté, souvenez-vous que les primitives Posix sont dans les sections 2 et 3 du manuel (`man 2` … ou bien `man 3` …).

Pour réaliser ce TD, nous vous fournissons une partie du code pour vous permettre de vous concentrer sur les parties importantes. Vous pouvez récupérer l'archive à l'adresse suivante :

```
http://trolen.polytech.unice.fr/cours/progsys/td05/td05_distrib.zip
```

1 Lister dossiers et fichiers : la commande `ls`

Exercice n°1:

Écrire le programme `lsrec` qui affiche les informations des fichiers ou répertoires qui lui sont passés en paramètres. Plus précisément, votre programme devra produire (à peu près) les mêmes sorties que la commande `ls` avec les options `-aR`). Si aucun argument n'est passé à la commande `lsrec`, celle-ci listera le contenu du répertoire courant.
Suggestion : Pour écrire ce programme, vous aurez besoin de plusieurs fonctions de bibliothèque, comme par exemple :

- les manipulations de chaînes de caractères en C (`man string`) ;
- les primitives Posix de manipulation de répertoires (`opendir()`, `readdir()`,`rewinddir()`,`closedir()`) ;
- la primitive Posix permettant d'obtenir les attributs d'un fichier (`stat()`).

2 Copie de Fichiers

Exercice n°2:

Réalisez le programme `mycp1` qui réalise la copie d'un fichier dans un autre fichier, ou d'un ensemble de fichiers dans un répertoire. La syntaxe d'invocation de ce programme doit être :

```
mycp1 fic1 fic2
```

ou bien

```
mycp1 fic1 fic2 ... dir
```

Dans le premier cas le fichier `fic1`, qui doit exister, est copié dans le fichier `fic2` ; si ce dernier existe, il est silencieusement écrasé, sinon il est créé. Dans le second cas, les fichiers `fic1`, `fic2`, … sont copiés dans le répertoire `dir` ; les fichiers `fic1`, `fic2`, … doivent être des fichiers ordinaires et doivent déjà exister ; `dir` doit aussi exister avant l'exécution de la commande ; si `dir` contient déjà des fichiers de mêmes noms que ceux qui sont copiés, les premiers sont silencieusement écrasés.

> **Suggestion** : Pour écrire ce programme, vous aurez besoin finalement d'assez peu de fonctions de bibliothèque, principalement :
> - les fonctions Posix de gestion élémentaire des E/S (`open()`, `close()`, `read()`, `write()`)
> - la primitive Posix permettant d'obtenir les attributs d'un fichier (`stat()`).

3 Héritage des descripteurs de fichiers entre père et fils

Exercice n°3:

Écrire le programme `fd_herit` permettant de montrer l'héritage des descripteur de fichiers et répertoires ouverts par un processus après un appel-système à `fork()`, en particulier :

- un processus hérite du descripteur des fichiers ouverts par son père, et partage le pointeur d'E/S ;
- un processus fils hérite aussi des répertoires ouverts par son père ;

Erick Gallesio, Stéphane Lavirotte, Jean-Paul Rigault
Polytech'Nice – Sophia
SI3
2017-2018

TD n°5
Primitives de base sur les fichiers/répertoires

Pour aller plus loin

Exercice n°4:

Etendez votre programme `lsrec` pour qu'il se comporte comme la commande `ls` d'Unix avec les options `-laR`.

> **Suggestion** : Pour écrire ce programme, vous aurez besoin de plusieurs fonctions de bibliothèque, comme par exemple :
> - les fonctions Posix de formatage des dates (`man strftime`)
> - la fonction Posix permettant d'obtenir des informations sur un utilisateur à partir de son uid (`getpwuid()`) ;

Exercice n°5:

Complétez le programme `mycp1`, pour obtenir un nouvel exécutable, `mycp2`, qui prenne en compte les options `-v` (verbose), `-r` (recursive) et `-i` (interactive) de la même manière que la commande `cp` (lisez les pages de man de cette commande et faites vos propres essais pour identifier son comportement avec les options considérées).

Suggestion : Pour écrire ce programme, vous aurez besoin des fonctions indiquées ci-dessus, plus quelques autres :

- les manipulations de chaînes de caractères en C (`man string`) ;
- les primitives Posix de manipulation de répertoires (`opendir()`, `readdir()`, `closedir()`).