TD4 – Manipulation de fichiers

1 Objectif

Ces exercices utilisent les primitives de manipulation de fichiers et de répertoires de POSIX. Ils consistent à réécrire certaines commandes standard d'UNIX : `printenv`, `ls`, `cp`.

2 Affichage de l'environnement

La commande `printenv` affiche la valeur des *variables d'environnement* (comme `DISPLAY`, `TERM`, `USER`, `SHELL`, etc.). Elle se comporte différemment selon qu'on lui fournit ou non des noms de variables en paramètres.

Identifiez ce comportement (utilisez `man` et faites vos propres essais d'invocation).

Identifiez le comportement de `printenv` et écrivez un programme `printenv.exe` qui le reproduise exactement (en laissant de côté le traitement des options `--help` et `--version`).

Indication Les variables d'environnement sont rangées dans une variable externe nommée `environ` (`man 5 environ`). Cette variable désigne en fait un tableau de chaînes de caractères (techniquement un tableau de pointeurs sur caractères, `char *environ[]`, ou encore un double pointeur sur caractères, `char **environ`). Donc `environ` est du même type que `argv`, le second argument de `main()`. Le dernier élément du tableau `environ` est le pointeur nul et chaque chaîne de caractères contenue dans `environ` a la forme suivante :

```
<nom de la variable>=<valeur de la variable>
```

Par exemple :

```
CLASSPATH=./home/catherine/prog101/classes
```

3 Un ls récursif

3.1 Un ls récursif simple

Écrivez un programme `lsrec.exe` qui affiche les informations relatives aux 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`.

Indication Vous aurez besoin d'un certain nombre de fonctions de bibliothèques, notamment :

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

3.2 Un ls récursif en format long

Étendez votre programme pour qu'il se comporte comme la commande `ls` avec les options `-laR`.

4 Copie de fichiers

4.1 Une version simplifiée de cp

Réalisez le programme `mycp1.exe` 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.exe fic1 fic2
```

ou bien :

```
mycp1.exe 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.

Indication Vous aurez besoin 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()`).

4.2 Une version un peu plus complète de cp

Complétez le programme précédent, pour obtenir un nouvel exécutable, `mycp2.exe`, 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).

Indication Vous aurez notamment besoin des fonctions de manipulation de chaînes de caractères en C (`man string`).