Programmation système

TP 1 – Mise en route

Exercice 1: `echo`

Le but de cet exercice est simplement de vous remettre dans le bain de la programmation C.

1. Implémentez en C une version simplifiée du programme `echo`. Plus précisément, votre programme `myecho` va recevoir un nombre arbitraire d'arguments passés en ligne de commande, et il devra les afficher sur sa sortie standard, séparés par des espaces (`" "`) et suivis d'un caractère de fin de ligne (`"\n"`).
2. Vérifiez que votre programme se comporte comme la commande `echo` du système (sans options). Par exemple, `./myecho *` devrait lister les fichiers du répertoire courant, et `./myecho abc | wc -c` devrait afficher *exactement* la même chose que `echo abc | wc -c`.

Exercice 2: Bascule utilisateur/noyau

Le but principal de cet exercice est de voir dans quelle mesure le basculement entre le mode utilisateur et le mode noyau peut avoir un impact sur les performances du système. Un autre but est de prendre l'habitude de trouver des informations dans les pages `man`.

1. En utilisant les fonctions de bibliothèque `stdio`, écrivez un petit programme C qui lit ce qu'on lui donne sur son entrée standard (`stdin`) et le recopie sur sa sortie standard (`stdout`). Le programme doit lire et écrire *caractère par caractère* (cf. `man 3 fputc` et `man 3 fgetc`).
2. Testez votre programme à l'aide d'un fichier volumineux (disons au moins 1 Mo). Par exemple, vous pouvez rediriger la sortie de `man gcc` vers l'entrée de votre programme. Mesurez le temps d'exécution à l'aide du programme `time` (cf. `man 1 time`).
3. Maintenant, créez une deuxième version de votre programme dans laquelle vous désactivez la bufferisation que la bibliothèque `stdio` utilise par défaut (aussi bien pour `stdin` que pour `stdout`). Comment faire *sans chercher sur le Web ?* Commencez par regarder la page `man 3 stdio`, qui introduit toutes les fonctions fournies par `stdio`. Quel serait un mot clé pertinent à chercher dans cette page ? Suivez les références pour trouver la bonne page. (Malheureusement il n'y a pas d'hyperliens dans les pages `man`.)
4. Refaites le même test que dans la question 2, et comparez les résultats. (Si la différence n'est pas assez impressionnante, augmentez la taille du fichier à copier.) Expliquez vos observations.

Exercice 3: `echo` avec `getopt()`

Le but de cet exercice est de vous familiariser avec la fonction `getopt()` qui sera réutilisée dans les TPs suivants.

1. Commencez par survoler la page `man 3 getopt` pour comprendre l'intérêt de `getopt()`. Il y a un exemple d'utilisation vers la fin de la page.

2. En vous servant de `getopt()`, reprenez votre programme `myecho` de l'exercice 1, et modifiez-le de façon à ce qu'il accepte les options `-s` et `-n` :
   `-s` indique que l'on ne veut pas séparer les arguments par des espaces.
   `-n` indique que l'on ne veut pas terminer la sortie par un caractère de fin de ligne.
3. Toujours à l'aide de `getopt()`, ajoutez l'option `-S` ("S" majuscule) qui prend en argument une chaîne de caractères à utiliser pour séparer les arguments de `myecho`. (Cette chaîne de caractères remplace donc l'espace utilisé par défaut.) Par exemple, `./myecho -S "//" 3 7` doit afficher `3//7`. Lorsque cette option est combinée avec l'option `-s` ("s" minuscule), l'option qui apparaît en dernier dans la ligne de commande prévaut.

Remarque : Beaucoup d'implémentations de `echo` reconnaissent l'option `-n`, mais `-s` et `-S` sont des extensions non-standard.

S'il vous reste du temps:

Exercice bonus 1: `stat()`

1. Prenez un peu d'avance et familiarisez-vous déjà avec l'appel système `stat()` qui est décrit dans `man 2 stat`. Vous l'utiliserez dans le TP 3 pour implémenter votre propre version de la commande `ls`.
2. En utilisant `stat()`, écrivez un programme qui affiche la taille, la date de modification, et le type (fichier, répertoire, ou autre) d'un fichier passé en argument. Inutile de mettre en forme la date de modification ; il suffit de l'afficher en l'état (c'est à dire en secondes depuis le premier janvier 1970). Quant au type, affichez juste "`f`", "`d`", ou "`?`" pour indiquer un fichier normal, un répertoire, ou autre chose. Pour connaître le type du fichier, utilisez les macros `S_ISREG` et autres sur le champ `st_mode` de la structure passée à `stat()`. La page `man` de `stat()` contient une référence vers une autre page `man` décrivant ces macros.