Programmation 3

L2 Informatique 2012-2013

Fiche de TP 1

Notions abordées : les entrées/sorties ; lecture de chaînes de caractères ; arguments d'un programme.

Un programme C peut être exécuté comme une commande habituelle et peut donc recevoir des arguments. Un argument est une chaîne de caractères. Pour profiter de cette fonctionnalité, la fonction `main` doit être écrite en respectant le prototype suivant :

```
1 int main(int argc, char *argv[]);
```

La variable `argc` contient le nombre d'arguments avec lesquels le programme vient d'être lancé et la variable `argv` est un tableau de chaînes de caractères, indicé de 0 à `argc - 1`, qui contient les arguments dans l'ordre de leur saisie. Il est a noter que le nom du programme est lui-même considéré comme un argument et occupe de ce fait invariablement la première position du tableau `argv`.

Par exemple, supposons que l'on souhaite lancer un programme nommé `prog` avec les arguments `arg1` et `autreArg`. Pour cela, nous saisirons la commande `./prog arg1 autreArg`, qui a pour effet d'affecter à la variable `argc` la valeur 3 et le tableau `argv` va vérifier `argv[0] = "prog"`, `argv[1] = "arg1"` et `argv[2] = "autreArg"`.

Exercice 1. (Programmation d'un système de cryptage simple)

Le but de cet exercice est de programmer un système de cryptage élémentaire basé sur le cryptage dit de *César*. Il consiste, étant donné un texte *en clair* (*i.e.*, non crypté), et une constante $K \in \mathbb{Z}$ appelée *clé de cryptage*, à décaler circulairement chacune des lettres du texte de K positions par rapport à sa place dans l'alphabet. Par exemple, pour le texte en clair :

`Les TP de programmation sont super !`

et la clé $K = 3$, on obtient le texte *crypté* suivant :

`Ohv WS gh surjudppdwlrq vrqw vxshu !`

Les caractères non alphabétiques ne sont pas modifiés. La formule qui permet de calculer, étant donné une lettre alphabétique en clair ℓ, la lettre cryptée c qui lui correspond est la suivante :

$$c := (\ell + K) \mod 26. \tag{1}$$

Réciproquement, pour décrypter un texte crypté, il suffit de décrypter chacune des lettres alphabétiques qu'il contient. Étant donné une lettre cryptée c, le calcul de la lettre en clair ℓ qui lui correspond s'effectue par la formule suivante :

$$\ell := (c - K) \mod 26. \tag{2}$$

1. Écrire un programme dont le nom d'exécutable est `crypt`, qui accepte comme arguments le nom d'un fichier source, le nom d'un fichier cible, et une valeur numérique. Ce programme doit écrire dans le fichier cible la version cryptée du texte en clair contenu dans le fichier source, en utilisant la clé de cryptage donnée par la valeur numérique en argument, en suivant la formule (1).
2. Écrire un autre programme, dont le nom d'exécutable est `decrypt` qui réalise le décryptage. Il prend comme arguments le nom d'un fichier source, le nom d'un fichier cible, et une valeur numérique. Il écrit dans le fichier cible la version en clair du texte crypté contenu dans le fichier source, en utilisant la clé de cryptage donnée par la valeur numérique en argument, en suivant la formule (2).
3. Tester les deux programmes précédents, en particulier en vérifiant que le texte obtenu en cryptant puis en décryptant successivement un texte est bien égal au texte de départ.

Exercice 2. (Programmation de `cmp`)

La commande `cmp` disponible sur les systèmes Unix permet de comparer deux fichiers quelconques (*i.e.*, pas forcement des fichiers contenant du texte), octet par octet.

1. Écrire un programme dont le nom d'exécutable est `newcmp`, qui accepte comme arguments le nom de deux fichiers quelconques. Le programme doit comparer les deux fichiers, octet par octet, et affiche la chaîne `Meme contenu.` si les deux fichiers sont égaux, ou `Contenu different.` sinon.
2. Modifier le programme précédent en lui ajoutant une option `-m`. Cette option doit avoir pour effet d'afficher, en cas de différence entre les deux fichiers en argument, le premier octet, converti en caractère qui cause l'inégalité, ainsi que sa position. Par exemple, si le contenu des deux fichiers coïncide pour les 247 premiers octets et qu'il existe une différence pour les caractères en position 247 (la numérotation des octets commence à 0), avec un `a` dans le premier fichier et un `+` dans le second, la chaîne affichée doit être de la forme `Contenu different : 247 -> a +.`. Le programme `newcmp`, appelé sans l'option `-m` doit garder le même comportement que dans la question précédente.

Exercice 3. (Programmation de `cat`)

La commande `cat` disponible sur les systèmes Unix permet d'afficher le contenu du fichier texte en argument sur la sortie standard.

1. Écrire un programme dont le nom d'exécutable est `newcat`, qui accepte comme argument le nom d'un fichier texte et affiche son contenu sur la sortie standard.
2. Modifier le programme précédent en lui ajoutant une option `-l`. Cette option doit avoir pour effet d'afficher un numéro devant chaque ligne de texte affiché. La numérotation doit commencer à 0. De plus, le programme `newcat`, appelé sans l'option `-l`, doit garder le même comportement que dans la question précédente.
3. Améliorer le programme de façon à ce que la place occupée par l'affichage des numéros de ligne soit la même pour chacune des lignes. Par exemple, si le fichier comporte 150 lignes, chaque numéro de ligne doit occuper une place de 3 caractères.

Exercice 4. (Programmation de `wc`)

La commande `wc` disponible sur les systèmes Unix permet de compter le nombre de caractères, de mots ou de lignes qui composent un fichier texte.

1. Écrire un programme dont le nom d'exécutable est `newwc`, qui accepte comme argument le nom d'un ficher texte et affiche sur la sortie standard le nombre de caractères dont il est composé.
2. Modifier le programme précédent en lui ajoutant une option `-a`. Cette option doit avoir pour effet d'afficher sur la sortie standard uniquement le nombre de caractères affichables du fichier en argument. Autrement dit, les espaces, les tabulations et les retours à la ligne ne sont pas comptés.
3. Modifier le programme précédent en lui ajoutant une option `-l` qui a pour effet de compter le nombre de lignes du fichier en paramètre. Les lignes vides (ou qui ne contiennent que des espaces ou des tabulations) ne doivent pas être comptées.
4. Modifier le programme précédent en lui ajoutant une option `-m` qui a pour effet de compter le nombre de mots du fichier en paramètre. Un mot est, au sens intuitif du terme, une chaîne de caractères qui ne contient ni espaces, ni tabulations, ni saut de ligne.