



TP6 : Arguments de la ligne de commande, E/S en C

CSH : Initiation au C et au shell
Première année



[TP6]

★ Exercice 1. Les arguments de la ligne de commande sont des chaînes de caractères

Le programme *copie.c* duplique un fichier *source* dans un fichier *destination*. Lisez-le attentivement.

▷ **Question 1.** Comment le programme connaît-il le nom du fichier *source*? Celui du fichier *destination*? Compilez et exécutez ce programme.

▷ **Question 2.** Comment vérifiez-vous que la copie s'est bien déroulée?

Modifiez le programme *copie.c* afin que les deux noms de fichiers soient passés en arguments de la ligne de commande. Vérifiez le bon fonctionnement du programme ainsi modifié pour des arguments corrects. Notez que si le fichier *destination* existe déjà, il est modifié par l'exécution de ce programme.

▷ **Question 3.** Que fait le programme dans les cas où les arguments ne sont pas conformes (en nombre, en nature) à ceux attendus?

★ Exercice 2. Les arguments de la ligne de commande sont des entiers

Le programme *max2.c* est celui de la séance 2. Compilez-le sous le nom *max2* et exécutez-le; Le programme *max2.sh* est celui de la séance 3. Exécutez-le.

On souhaite modifier *max2* pour lui fournir les données en arguments de la ligne de commande (comme cela se passe pour *max2.sh*).

Pour réaliser cela, un programmeur pressé a modifié *max2.c*, sous le nom *max2.v2.c*. Examinez ce programme et compilez-le. Vous obtenez un message d'erreur qui signifie que les parties gauches et droites d'affectations ne sont pas du même type : en effet, *a* et *b* sont des entiers, *argv[1]* et *argv[2]* sont des adresses (pointeurs). Exécutez le code exécutable (qui est tout de même généré, car ces erreurs sont des *warnings*).

Notre programmeur pressé a fait une première tentative de correction en forçant une conversion de type, dans le fichier *max2.v3.c*. Lisez-le, compilez-le et vérifiez que l'erreur de compilation ne se produit plus.

▷ **Question 1.** Exécutez le programme obtenu ... et concluez.

▷ **Question 2.** Corrigez le programme. Vous pourrez utiliser la fonction *atoi* (cf. la page de manuel correspondante).

★ Exercice 3. Mesurer la complexité d'un fichier C

On souhaite avoir une mesure de la complexité de divers programmes C. Pour cela, plusieurs métriques sont utilisables, que nous allons explorer au fil des questions.

La première idée est de mesurer la longueur du texte. Il est assez courant d'annoncer le nombre de lignes composant un programme donné (Noyau linux 2.6 : 4 millions; GCC : 2,5 millions; Windows XP : 40 millions, ...).

▷ **Question 1.** Faites un programme C *complexite* prenant le nom d'un fichier en argument et comptant le nombre de lignes le composant (il faut compter les occurrences du caractère '\n').

▷ **Question 2.** Modifiez votre programme pour ne pas tenir compte des lignes blanches (il faut utiliser un booléen réinitialisé à chaque ligne indiquant si on a vu un caractère autre que '\t' et ' ').

▷ **Question 3.** (facultative) Modifiez votre programme pour ne pas compter ce qui se trouve à l'intérieur de commentaires. On souhaite traiter à la fois le style C (*/* ... */*) et le style C++ (*// ... fin de ligne*). On utilisera un booléen indiquant si on se trouve actuellement dans un commentaire ou non.

Cette métrique de complexité est trompeuse car un long programme ne comportant qu'un enchaînement de commandes sans *if*, *for* ou autres est sans doute moins complexe qu'un programme plus court présentant un schéma d'exécution plus complexe.

▷ **Question 4.** Modifiez votre programme pour qu'il compte les occurrences du caractère '{', qui est présent dans la plupart des structures syntaxiques du C.

Voici deux extensions possibles pour votre programme, à réaliser si vous avez le temps :

- Comptez séparément les occurrences des différentes constructions syntaxiques du C.
- Faites en sorte que votre programme compte les lignes lorsqu'on lui passe l'argument supplémentaire *--long* tandis qu'il compte la complexité si on lui passe l'option *--complexe*.

Exercices à la maison

1. Ecrivez un programme C qui accepte un nombre variable d'arguments sur la ligne de commande et affiche pour chacun d'eux le nombre de caractères correspondant. Ainsi, si ce programme est compilé sous le nom *longueur_arg*, l'exécution de la commande

`longueur_arg 0 bonjour 2.56 adieu` produira la sortie :

```
l'argument no 0 contient 12 caractere(s)
l'argument no 1 contient 1 caractere(s)
l'argument no 2 contient 7 caractere(s)
l'argument no 3 contient 4 caractere(s)
l'argument no 4 contient 5 caractere(s)
```