TP5 : Arguments de la ligne de commande, lecture et écriture de fichiers

CSH : Initiation au C et au shell

Première année

[TP5]

★ **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* ?

▷ **Question 2.** Celui du fichier *destination* ? Compilez et exécutez ce programme.

▷ **Question 3.** 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 4.** 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 ***essai1max2.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 ***essai2max2.c***. Lisez-le, compilez-le et vérifiez que l'erreur de compilation ne se produit plus.

▷ **Question 5.** Exécutez le programme obtenu ... et concluez.

★ **Exercice 3. Transformer une chaîne de chiffres en entier**

Rappel. Un nombre entier s'écrit dans un système de numération à l'aide de caractères (ou symboles) appelés chiffres. En base 10, les chiffres sont les caractères '0', '1', '2', ..., '9'. Ainsi, la chaîne de caractères "42" représente le nombre que l'on appelle quarante-deux et que l'on peut écrire 42.

Écrivez dans un fichier ***horner.c*** la fonction `convertir` (telle qu'on l'a définie en séance de TD), ayant comme paramètre une chaîne de caractères et dont la valeur est l'entier correspondant (en base 10). Compilez ce texte C par la commande `gcc -o convertir horner.c`

▷ **Question 6.** Que signifie le message d'erreur obtenu ?

Lisez le contenu du fichier ***test_horner.c***. Écrivez un fichier ***Makefile*** pour compiler ***horner.c*** en ***horner.o***, et créer avec ***test_horner.c*** un fichier exécutable ***test_convertir*** afin de tester votre fonction `convertir`. Exécutez ***test_convertir*** et, en cas d'erreur, corrigez.

Modifiez le texte de ***horner.c*** en ajoutant les lignes :

```
int main() {
  int a ;
  a = convertir("11") ;
  printf("%d\n", a) ;
  return 0 ;
}
```

▷ **Question 7.** Que se passe-t-il lorsque vous exécutez la commande **make** ? Expliquez.

★ **Exercice 4. Quelques essais plus ou moins étonnants...**

▷ **Question 8.** Exécutez les commandes :

```
test_convertir *
```

```
test_convertir a
```

```
test_convertir b
```

```
test_convertir \*
```

```
test_convertir ' '
```

Expliquez ce que vous observez.

Modifiez le texte de ***horner.c*** pour que la conversion ne soit effectuée que si les caractères sont des chiffres entre '0' et '9' (à vous de choisir ce qui se passe si ce n'est pas le cas).

★ **Exercice 5. Récupérer des entiers sur la ligne de commande**

Modifiez ***max2.c*** afin que ses arguments soient lus correctement sur la ligne de commande. Compilez et exécutez pour tester.

★ **Exercice 6. Une fonction magique**

Lisez le texte du fichier ***atoi.c***. Modifiez le ***Makefile*** pour que ***test_convertir*** soit construit à partir de ***test_horner.c*** et ***atoi.o***. Exécutez pour tester le résultat obtenu.

Modifiez ***max2.c*** en utilisant la fonction `atoi` pour convertir ses arguments. Compilez (sans utiliser ***horner.o*** ni ***atoi.o***) et exécutez pour tester.

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)
   ```

2. On souhaite améliorer l'esthétique de l'horloge graphique utilisée au TP3 en affichant les entiers 3, 6, 9 et 12 à leurs positions respectives sur le cadran. On note (`X_C`, `Y_C`) les coordonnées du centre de l'horloge, `R_int` son rayon intérieur, et `R_ext` son rayon extérieur.

   La fonction graphique `write_gr(x, y, s)` permet d'afficher la chaîne de caractères `s` au point de coordonnées (`x`, `y`).

   Donnez une séquence d'instructions C permettant de réaliser cet affichage.