INF203 - Travaux pratiques, séance 4

Codage des caractères et des entiers
Chaîne de caractères
Compilation séparée et Makefile
Horner

[INF203]* Préambule** : utilisez ***installeTP.sh pour créer un répertoire ***TP4***. Vous ferez ceci systématiquement au début de chacune des prochaines séances, ce ne sera plus indiqué dans les énoncés.
Dans cet énoncé et tous ceux qui suivront, lorsqu'il est écrit "compilez le programme ***xxxx.c***", cela veut dire (sauf avis contraire), qu'il faut créer un fichier de code exécutable de nom ***xxxx***.

Formats de fichiers et codage

[TP4]

1. Lisez attentivement le contenu du fichier ***code_carac.c***, et vérifiez que vous le comprenez. Compilez-le et exécutez-le.
   [a] Quel est le code Ascii du caractère 'A' ? Et celui du caractère '5' ? À quel caractère correspond le code décimal 109 ? et 77 ? Pouvez-vous connaître le code du caractère *espace* ? ■

2. La commande **ascii** permet d'afficher la table des codes Ascii. Retrouvez les réponses aux questions du paragraphe précédent en utilisant cette commande. Affichez la table des codes Ascii sous forme décimale seulement. Quel est le code du caractère *espace* (noté usuellement `SP` ou `SPACE`) ? Comment est noté le caractère de code zéro ?

En vous aidant du programme ***code_carac*** et de la commande **ascii**, repérez les codes des caractères "imprimables" de la table Ascii.
[b] Quel est l'intervalle (en décimal) des codes des caractères imprimables ? ■
Ecrivez un programme C qui affiche à l'écran la liste des codes des caractères imprimables, ainsi que le caractère correspondant (un couple code-caractère par ligne).
[c] ... et joignez le texte de ce programme à votre compte rendu. ■

Chaînes de caractères : copie, comparaison, conversion en majuscules

[d] Par quel caractère se termine une chaîne de caractères en C ? ■

1. Lisez le fichier ***fcts_chaines.c***. Complétez-le en écrivant le corps des fonctions `mon_strcpy` et `mon_strcmp` de prototypes `void mon_strcpy(char dest[], char src[]) ;` et `int mon_strcmp(char s1[], char s2[]) ;`.
   — la fonction `mon_strcpy` duplique la chaîne `src` dans la chaine `dest`.
   — la fonction `mon_strcmp` compare les deux chaînes `s1` et `s2` et renvoie 0 si les deux chaînes sont identiques, 1 si elles sont différentes.
2. Pour tester ces fonctions, on a écrit un programme ***test_fcts_chaines.c***. L'ensemble des fichiers sources nécessaires à ce programme est donc ***fcts_chaines.c***, ***fcts_chaines.h*** et ***test_fcts_chaines.c***. En vous inspirant de celui du ***TP3***, écrivez un ***Makefile*** pour compiler ce programme en utilisant la compilation séparée.
   [e] Dessinez le graphe de dépendance des fichiers sources, des fichiers ***.o*** et de l'exécutable à produire ***test_fcts_chaines***. ■

3. Compilez et testez (et corrigez si nécessaire). Pensez à tester un premier mot plus court que le suivant (par exemple "mer" puis "borde"), et inversement.
4. Ajoutez une nouvelle fonction à ***fcts_ chaines.c*** de profil : `void min_2_maj(char dest[], char src[])` qui convertit une chaîne de lettres minuscules (`src`) en chaîne de lettres majuscules (`dest`).
5. Complétez (ou modifiez) ***test_ fcts_ chaines*** pour tester cette nouvelle fonction.
   [f] Le fait d'ajouter la fonction `min_2_maj` nécessite-t-il de modifier le ***Makefile*** ? Pourquoi ? ■

[g] Expliquez ce que vous comprenez du passage des paramètres des fonctions `min_2_maj` et `mon_strcpy`. ■

Taille des entiers, temps d'exécution, débordement

1. Lisez le contenu du fichier ***debordechar.c***, et prévoyez ce qui va se passer lors de l'exécution. Compilez-le et exécutez-le.
   [h] Quelle est la taille en octets et en bits d'une variable de type `unsigned char` ? Quel est le plus grand entier représentable avec le type `unsigned char` ? Quel est le plus petit ? Mêmes questions pour le type `char` ? ■

2. Créez une copie de ***debordechar.c*** sous le nom ***debordeshort.c***. Modifiez ce fichier afin qu'il serve à tester des variables de type `short int` (au lieu de `char`), compilez, exécutez.
   [i] Répondez aux mêmes questions que ci-dessus, pour le type `short int`. ■

3. Effectuez la même expérience pour le type `int` (donnez à l'exécutable le nom ***debordeint***).
   [j] Par combien (environ) est multiplié le temps d'exécution entre ***debordechar*** et ***debordeint*** ? Si l'on définit un type d'entiers représentés sur 8 octets, et qu'on effectue la même expérience, combien de temps (environ) faudra-t-il attendre pour voir s'afficher le résultat ? ■

Schéma de Horner

Complétez ***fcts_ chaines.c*** avec une fonction de profil `int mon_atoi(char s[])` qui calcule (et retourne) l'entier représenté par la chaîne `s` en base 10. Par exemple, si `s` est la chaîne `"203"`, le résultat de la fonction sera l'entier 203. Ajoutez ce qu'il faut à ***test_ fcts_ chaines.c*** pour tester cette fonction.

[k] joignez ***fcts_ chaines.c*** à votre compte rendu. ■

Un script pour générer des Makefile

Vous trouverez dans le répertoire du TP4 un script nommé `make_gen.sh`. Ce script est découpé en blocs, dont le dernier est incomplet.
[l] Il vous est demandé ici d'analyser chaque bloc, et d'expliquer ce que fait chaque ligne. Quel est le but du script ? Comment fonctionne-t-il ? ■

Le dernier bloc du script est incomplet. En vous aidant du bloc 4 et en vous appuyant sur la fonction `extract_headers`, complétez ce bloc.
[m] Ajouter le script complété à votre compte-rendu. ■