TP 1 - Variables, printf(), scanf(), compilation, exécution

Consignes générales

Afin de travailler proprement, vous allez créer un répertoire HAI202I à la racine de votre compte. Pour chaque séance de TP vous créerez un nouveau répertoire, appelé TP1 (TP2, etc), dans le répertoire initial HAI202I. Vous écrirez les programmes associés à chaque TP dans le répertoire correspondant. Pour faciliter votre travail, nous vous conseillons de nommer les fichiers des programmes en référence à l'exercice concerné. Par exemple, le programme demandé à l'exercice 3 du TP 1 sera nommé **tp1-exo3.c**. Une bonne façon de faire est également de prévoir un en-tête en début de fichier qui renseigne par exemple : auteur, date, numéro et but de l'exercice. Pour rappel, les commentaires en C se font par l'encadrement suivant :

`/*` *commentaires* `*/`

1 Création d'un programme en C à partir d'un éditeur de texte

Pour créer un programme en C avec un éditeur de texte (Gedit, Emacs, VScodium...), , il suffit de créer un nouveau fichier portant l'extension `.c`. Par exemple : `firstprog.c`. ATTENTION, pour nommer les fichiers (ainsi que les répertoires) en UNIX, veillez à ne pas utiliser de caractères spéciaux, de caractères accentués ou d'espaces, limitez vous aux lettres majuscules, minuscules et aux tirets.

Pour créer un nouveau fichier, vous avez deux solutions :

- Dans un terminal, positionnez vous dans le répertoire où vous voulez créer le fichier, par exemple tapez la commande `cd ~/HAI202I/TP1/`
  Puis, avec l'éditeur *gedit*, tapez la commande `gedit firstprog.c &` pour ouvrir gedit et créer le fichier `firstprog.c`.
  Ou, avec l'éditeur *emacs*, tapez la commande `emacs firstprog.c &` pour ouvrir emacs et créer le fichier `firstprog.c`.
- Directement dans un éditeur de texte (Gedit, Emacs, ...). Par exemple dans Emacs, sous le menu `File`, cliquez sur `"Open File ... "`. Tout en bas de la fenêtre Emacs vous trouverez le buffer interactif d'Emacs. Celui-ci affiche `"Find file: "` suivi du chemin du répertoire où vous avez ouvert l'éditeur ou bien du chemin du fichier courant. Il suffit de taper dans ce buffer le nom du fichier que vous souhaitez créer en le précédent du chemin complet dans l'arborescence des fichiers (rappel : le caractère `~` correspond au chemin associé au répertoire de votre compte). Ex : `"Find file: ~/HAI202I/TP1/firstprog.c "`. Valider en tapant sur la touche `Entrée`.

Si le fichier que vous essayez de créer existe déjà, alors ces deux solutions ne feront qu'ouvrir le fichier existant dans l'éditeur de texte. Attention, penser à sauvegarder le fichier pour valider sa création. Pour sauvegarder un fichier sous Emacs on clique dans le menu `File` puis `Save`. Sinon, en raccourci clavier on fait `<Ctrl-x><Ctrl-s>`.

2 Compilation d'un programme en C

Pour compiler un programme en C il faut faire appel à un compilateur. Pour nos TP, nous utiliserons le compilateur C de la suite GNU Compiler Collection (GCC). Le compilateur est invoqué en ligne de commande en exécutant

```
gcc -Wall firstprog.c -o firstprog
```

Pour compiler vous avez 2 solutions :

- Ouvrez un terminal et positionnez vous dans le répertoire contenant votre programme et exécutez la commande de compilation `gcc -Wall firstprog.c -o firstprog`. Le compilateur affichera des erreurs de compilation si le programme n'est pas correct ou bien il générera l'exécutable de ce programme (dans notre exemple, `firstprog`).
- Dans Emacs, sous le menu `Tools`, cliquez sur `"Compile ... "`. Tout en bas de la fenêtre Emacs, le buffer interactif indique : `"Compile command: make -k "`. Positionnez vous dans le buffer et remplacez `"make -k "` par la commande de compilation `"gcc -Wall firstprog.c -o firstprog"` et tapez sur la touche `Entrée` du clavier. Une sous-fenêtre apparaît dans Emacs pour donner le compte rendu de la compilation. Si aucune erreur n'est détectée, l'exécutable du programme sera généré (dans notre exemple, `firstprog`).

Attention, pensez bien à remplacer le fichier `firstprog.c` et le nom d'exécutable `firstprog` dans la commande de compilation par les noms correspondants à votre programme.

3 Exécution d'un programme en C

Dans un terminal, positionnez vous dans le répertoire où se trouve le programme puis lancez le en exécutant la ligne de commande suivante :

```
./firstprog
```

où `firstprog` correspond au nom que vous avez donné à votre programme. Attention, le `./` est obligatoire.

Remarque : vous pouvez également lancer le programme directement dans Emacs. Pour cela, dans le menu `Tools` cliquez sur `"Shell Command... "`. Tout en bas de la fenêtre Emacs, le buffer interactif indique : `"Shell command: "`. Tapez dans ce buffer la ligne de commande relative à l'exécution de votre programme (Ex : `"./firstprog "`) puis tapez sur `Entrée`. Une sous-fenêtre apparaît dans Emacs. Cette sous-fenêtre permettra de voir les affichages du programme. Attention, ce mode d'exécution ne permet pas d'interagir en entrée avec le programme (saisie de valeurs au clavier). Pour faire cela, vous devrez lancer le mode Shell interactif en tapant la combinaison de touche `<Alt-x>` puis entrer `shell` dans l'invite de commande d'Emacs.

4 Prise en main

4.1 Compilation et exécution

Dans un premier temps, nous allons essayer de compiler et exécuter un programme simple. Récupérez le programme `firstprog.c` disponible sur le Moodle du cours (`HAI202I > TD_TP > Fichiers TD_TP1`). Enregistrez ce fichier dans le répertoire `HAI202I/TD1` que vous aurez pris la peine de créer. Compilez ce programme et exécutez le. Votre terminal doit afficher un message.

4.2 Compilation et correction d'erreurs

Récupérez sur l'espace Moodle le programme `secondprog.c` et enregistrez le dans le même répertoire que précédemment. Compilez ce programme. Si vous avez des messages d'erreur c'est normal. Il vous faut ouvrir ce fichier avec un éditeur de texte (Gedit, Emacs, ...) et corriger une erreur. Compilez à nouveau ce programme et exécutez le.

4.3 Création d'un nouveau programme

Créer un nouveau fichier `thirdprog.c` avec un éditeur de texte. Écrire dans ce fichier le code d'un programme permettant d'afficher la valeur de l'expression suivante : `(1+2)*5+8/3-47`. Compilez et exécutez ce programme.

4.4 Les différentes étapes de la compilation

La compilation se décompose en plusieurs étapes :

1. La **pré-compilation** réalisée par le pré-processeur C : il s'agit de normaliser le code source (suppression des commentaires, espaces inutiles...) et de traiter quelques directives de pré-compilation (indépendantes de la syntaxe du C) introduites par le caractère # :
   — `#include <`*nom de fichier*`>` : permet d'inclure le contenu d'un autre fichier, généralement des fichiers de déclaration de fonctions de la bibliothèque du `C` : *stdio.h*, *math.h*, *limits.h*...
   — `#define <`*identifiant*`> <`*contenu de remplacement*`>` : permet la définition une macro ; le pré-processeur remplacera tous les identifiants du code source par le contenu de remplacement. Exemple : `#define vrai 1`
   — `#if <`*condition*`>` *portion de code* `#endif` : permet d'intégrer sous condition une portion de code (compilation conditionnelle).
   Le résultat de cette étape peut être observé par un appel à `gcc` avec l'option `-E` : `gcc -E` *prog.c*

2. La **compilation** : la traduction du code source en code assembleur. Un appel à `gcc` avec l'option `-S` permet d'enchaîner les étapes de pré-compilation et assemblage ; on obtient un fichier texte *prog.s* (en assembleur) à partir du fichier texte *prog.c* (en source C) : `gcc -S` *prog.c*.
3. L'**assemblage** : la transformation de l'assembleur en langage machine (fichier binaire). Un appel à `gcc` avec l'option `-c` permet d'enchaîner les étapes de pré-compilation, de compilation et d'assemblage ; on obtient un fichier binaire *prog.o* (en assembleur) à partir du fichier texte *prog.c* (en source C) : `gcc -c` *prog.c*.
4. L'**édition de liens** : construit un programme exécutable (fichier binaire par défaut *a.out*) à partir du code machine *prog.o* produit et du code machine des fonctions de la bibliothèque : `gcc` *prog.o*.

Reprenez le programme `firstprog.c`, produisez et observez (taille et contenu) la sortie du pré-processeur, le programme en assembleur, le programme en langage machine et le programme exécutable. Pour observer le contenu d'un fichier texte, un simple `cat nom_de_fichier` ou `more nom_de_fichier` suffit. Pour les fichiers binaires, vous pouvez utiliser la commande `od -b nom_de_fichier` qui vous affiche la valeur de chaque octet en octal ; l'option `-a` interprète comme des caractères les octets dont la valeur décimale est inférieure à 128.

5 Exercices

Exercice 1 Écrivez un programme qui demande de saisir 2 entiers au clavier et qui affiche leur somme et leur produit à l'écran.

Exercice 2 Écrivez un programme qui demande de saisir 2 entiers au clavier et qui les stocke dans 2 variables (a et b par exemple). Affichez à l'écran l'adresse des variables a et b ainsi que leur valeur. Ensuite, échangez les valeurs de a et de b et refaites le même affichage (adresse et valeur).

Exercice 3 Écrivez un programme qui définit dans une variable réelle une température exprimée en degré Fahrenheit et qui affiche à l'écran la conversion de cette température en degré Celsius : $C = \frac{5}{9}(F - 32)$
Écrivez ensuite le programme qui fait l'opération inverse. Il se peut que votre programme ne calcule pas le bon résultat (toujours 0). Si tel est le cas, pensez à vérifier l'expression représentant votre calcul en gardant en tête que le calcul avec des entiers renvoie un entier.

Exercice 4 Récupérez sur Moodle le programme *debordement.c* et compilez-le. Dans une premier temps, ignorez le message de warning généré.

1. Exécutez-le en saisissant `12` pour i et `A` pour c. Qu'observez-vous ? Est-ce en accord avec les valeurs attendues pour i, j, c et d.
2. Corrigez la saisie de la valeur de i en vous aidant du message du compilateur. Exécutez à nouveau avec les mêmes valeurs. Qu'observez-vous ? Essayez également de saisir pour c les caractères `AAA`, puis `AAAAAAAAAAAAAAAAAA` (une vingtaine de `A`).
3. Corrigez alors la saisie de la valeur de c. Exécutez à nouveau avec les valeurs `12` et `a`. Qu'observez-vous ? Essayez également de saisir pour i les caractères `12A`, `12AAA`, `12AAAAAAAAAAAAAAAAAAAA`.
4. Permutez les instructions relatives à la saisie de la valeur de i (lignes 9 et 10) avec celles de saisie de la valeur de c (lignes 12 et 13). Exécutez à nouveau avec `A` et `12`. Qu'observez-vous ? Essayez également avec `AAA`...
5. Que peut-on en déduire sur les emplacements mémoire des variables i, j, c et d ? Confirmez vos intuitions en affichant les adresses mémoires et nombre d'octets occupés par ces 4 variables. Vous dessinerez sur une feuille l'occupation de la mémoire afin d'illustrer les phénomènes observés lors des 4 questions précédentes.