IN3S02 : Sujet du TP de C

Groupe ESIEE, Denis BUREAU, octobre 2007.
Attention ! Le sujet peut être modifié jusqu'à la veille du TP.

1 Les objectifs

- Être capable de réaliser à partir d'un terminal sous linux des programmes en C.
- Réaliser sur machine les exercices du TD.
- Utiliser les tableaux, les pointeurs, l'allocation dynamique.
- Expérimenter la liaison Java ↔ C.

2 Prise en mains

2.1 L'éditeur `nedit`

Taper `nedit hello.c &` pour lancer l'éditeur de texte en arrière-plan, puis cliquer sur `New file` pour créer le fichier `hello.c` qui contiendra le programme ci-dessous
(il est également possible de lancer `nedit` par le menu "KDE" ou d'utiliser un autre éditeur de texte que vous maîtrisez).

2.2 Premier programme

```
#include <stdio.h>
int main( void )
{
  printf( "Bonjour !\n" );
} /* main() */
```

2.3 La compilation

Pour utiliser la bonne version du C, il faut préciser au compilateur `gcc` quelques options.
Pour éviter d'avoir à les retaper à chaque compilation, `linux` nous permet de créer un *alias*.
Pour éviter d'avoir à recréer l'alias à chaque login, il faut le stocker dans le fichier `~/.cshrc` (à la racine de votre compte), car ce fichier est automatiquement lu au début de chaque session.
Aide: taper `ls -a` pour voir TOUS les fichiers d'un répertoire.

Pour cela, lancer l'éditeur de texte sur ce fichier et insérer dans la partie Linux (si elle existe) la ligne suivante : `alias mycc gcc -ansi -pedantic -W -Wall`
Fermer la fenêtre Terminal et en ouvrir une nouvelle pour prendre en compte ce nouvel alias. Revenir dans le bon répertoire de travail. Taper `alias` pour vérifier que `mycc` est maintenant disponible.

Pour lancer la phase de compilation, taper `mycc -c hello.c` ; un message d'erreur s'affiche ; ajouter l'instruction manquante puis relancer la commande de compilation jusqu'à ce qu'il n'y ait plus de message d'erreur ; le fichier `hello.o` doit être créé ; le vérifier.

Remarques : - le pré-processeur est automatiquement lancé avant la compilation proprement dite et interprète toutes les lignes commençant par `#` .
- En cas de besoin, la liste des options du compilateur peut être obtenue par `man gcc` .

2.4 L'édition de liens

Pour lancer cette phase, taper `gcc -lm hello.o -o hello` ; si aucun message d'erreur ne s'affiche, le fichier `hello` doit être créé ; le vérifier.
Remarque : le lien avec la **l**ibrairie **m**athématique n'est utile que lorsque le programme utilise des fonctions déclarées dans `<math.h>` .

2.5 L'exécution

Pour lancer le programme, taper simplement `./hello` ; vérifier l'affichage.

2.6 Modification du programme

Modifier le programme pour qu'il pose la question `"numero ? "`, qu'il saisisse un entier (par exemple 6), puis qu'il affiche `"Bonjour numero 6 !"` . Sauvegarder le fichier et lire le paragraphe suivant.

2.7 Compilation & édition de liens enchaînées

Pour lancer les 2 phases automatiquement, taper `mycc -lm hello.c -o hello` ; si aucun message d'erreur ne s'affiche, le fichier `hello` doit avoir été modifié ; le vérifier.

Attention ! Ne pas lancer les 2 phases séparément est certainement une facilité lorsque tout se passe bien, mais peut être un handicap lorsqu'il y a des erreurs. Il faut alors déterminer si c'est une erreur de compilation ou bien une erreur d'édition de liens.

Par exemple, dans un programme appelant la fonction `sqrt`, oublier la ligne `#include <math.h>` provoquera une erreur de compilation car la fonction `sqrt` sera inconnue du compilateur, mais oublier l'option `-lm` provoquera une erreur de l'éditeur de liens car il ne trouvera pas le code compilé de la fonction `sqrt`.
En général, tout message d'erreur ne commençant pas par `fichier.c:LL:CC` provient de l'éditeur de liens (LL = n° de ligne, CC = n° de colonne).

3 Exercices du TD

1. Passage de paramètres
2. Chaînes de caractères
   Attention ! La version de compilateur que nous utilisons n'accepte pas pour spécifier des tailles de tableau des constantes définies par `const int MAX=5;` Il faut donc utiliser le pré-processeur et écrire `#define MAX 5`
3. Pointeurs et chaînes de caractères
4. Pointeurs et allocation dynamique

4 Appel Java $->$ C

1. Il s'agit dans cette partie de montrer comment un programme java peut appeler des fonctions et des procédures écrites en C. L'objectif n'est pas de tout retenir par cœur (ce n'est pas au programme du QCM !), mais de comprendre le fonctionnement et de pouvoir se reporter à ce tp ultérieurement (notamment pour le projet). Cela va vous paraître très long, d'une part car c'est la première fois que vous mettrez en œuvre ces outils, mais aussi car les instructions sont très détaillées.

2. En suivant précisément les instructions figurant sur cette page
(`http://www.esiee.fr/~bureaud/Unites/in103/Java2C.htm`),
faire fonctionner l'exemple qui y est décrit, en exécutant toutes les commandes (en jaune) des paragraphes 1.A à 1.E ; les fichiers Java et C sont fournis au 1.F. Cela donne-t-il bien les résultats attendus ?

3. Dans un autre répertoire et en suivant les 5 mêmes étapes qu'à l'exercice précédent, réaliser un programme démontrant la possibilité d'utiliser en Java les formatages fournis par C (voir exemple d'exécution au 4.7 ci-dessous). Commencer par compléter la classe Java ci-dessous :

```
// MyFormat.java
public class MyFormat
{ // declaration des 3 sous-programmes C appeles ci-dessous
  // (lengthInt, formatInt, et formatInteger)
  // qui seront a realiser en C dans le fichier MyFormat.c

  static { System.loadLibrary( "MyFormat" ); } // chargera libMyFormat.so
  public static void main( String [] args )
  { int len;   String s;
    len = lengthInt( "[%5d]", 12 ); // nb de caracteres de la String a afficher
    System.out.println( "aff1=" + len );
    len = lengthInt( "[%04d]", 12 );
    System.out.println( "aff2=" + len );
    s = formatInt( "[%5d]", 12 ); // String formatee d'un int
    System.out.println( "aff3=" + s );
    s = formatInt( "[%04d]", 12 );
    System.out.println( "aff4=" + s );
    s = formatInteger( "[%5d]", new Integer(12) ); // String formatee d'un Integer
    System.out.println( "aff5=" + s );
    s = formatInteger( "[%04d]", new Integer(12) );
    System.out.println( "aff6=" + s );
  } // main()
} // MyFormat
```

4. Compiler la classe (jusqu'à ce qu'il n'y ait plus d'erreur) et générer le `.h` .
Pour simplifier, il est possible de ne s'occuper que de `lengthInt()` dans un premier temps.

5. Reprendre le processus 1.A à 1.E ci-dessus, puis les parties 3.A et 3.B de ce même document.

6. Écrire le fichier `MyFormat.c` avec les contraintes suivantes :
   - Utiliser `sprintf()` pour les 3 fonctions ; taper `man sprintf` pour obtenir de l'aide ou consulter le poly papier ou en ligne.
   - Penser à convertir une `jstring` en `const char *` avant de l'utiliser, et à rendre la mémoire après.
   - Penser à convertir une `const char *` en `jstring` avant de la retourner vers Java.
   - Allouer dynamiquement toute chaîne de caractères C et libérer la mémoire après.
   - Accéder à la valeur de l'`Integer` à travers sa méthode `intValue()`.

7. Exécuter le programme. L'affichage devrait être :

```
aff1=7
aff2=6
aff3=[   12]
aff4=[0012]
aff5=[   12]
aff6=[0012]
```