Partie 1 : Installation, Configuration et compilation

Fichiers à récupérer :

- Compilateur : mingW
- Editeur : vs code
- Source code : fibo.c

Installer vs code

Installer MingW

- Tester la commande (en ligne de commande) : gcc
- Si vous avez fatal error, alors c'est bien installé
- Sinon faudra l'installer et configurer la variable d'environnement Path

Configuration de votre dossier de travail (obligatoirement)

Ouvrir la ligne de commande en mode administrateur

Ensuite, taper les commandes ci-dessous :

- Cd \ (vous ramène dans le disque local C)
- md TP_C (crée le dossier TP_C)
- cd TP_C (rentre dans le dossier TP_C)
- md Lab1 (On crée un Lab par séance de TPs, le numéro représente la séance)
- cd Lab1 (on rentre dans Lab1)
- code . (on ouvre le dossier Lab1 dans vs code)

Dans vs code

- Créer un fichier nommé fibo.c et copier le code dans fibo.c qui se trouve dans le package.
- Compiler puis exécuter le fichier en faisant les commandes ci-dessous (dans un terminal vs code) :
    - gcc fibo.c (Si la compilation se passe bien, aucun message d'erreur, faire la commande ci-dessous) :
    - .\a.exe (exécution)
- Une alternative peut être aussi :
    - gcc fibo.c -o fibo.exe (on donne un nom fichier exécutable)
    - .\fibo.exe (pour exécuter)

Une troisième alternative peut être l'utilisation de make comme suit :

- Créer un fichier nommé makefile contenant le texte ci-dessous :
  fibonacci: fibo.c
  gcc fibo.c -o fibo.exe
  .\fibo.exe
- Ensuite taper seulement la commande make qui va lancer les commandes dans le fichier makefile

Partie 2 : Préprocessing – compilation – édition de liens – création de librairies

Compilation :

Traduction d'un code source de haut niveau en un programme équivalent cible (code machine).

Préprocessing :

L'objectif étant de produire un code intermédiare

- Qui est une image du code source
- Et qui peut être exécuté avec plus d'efficacité

Etapes de la compilation :

Soit le programme tp1_dic.c

/* tp1_dic. */

```
#include <stdio.h>
int main()
{
  int x = 2, y = 3;
  int z = ++x - y++ == 4;
  printf("z = %d", z);
  return 0;
}
```

Production du fichier (extension .i) après le préprocessing :

```
gcc -E tp1_dic.c -o tp1_dic.i
```

Production du fichier (extension .s) après l'obtention de l'assembleur :

```
gcc -S tp1_dic.c -o tp1_dic.s
```

Production du fichier binaire (pas exécutable avec l'extension .o)

```
gcc -c tp1_dic.c -o tp1_dic
```

Production du fichier exécutable (en le renommant)

```
gcc tp1_dic.c -o tp1_dic
```

On peut avec une seule commande obtenir l'ensemble de ces fichier avel l'option -save-temps

```
gcc -save-temps tp1_dic.c
```

Création de librairies

Soient les fichiers hellofirst.c, hellosecond.c et twohello.c donnés ci-dessous :

```
/* hellofirst.c */
#include <stdio.h>
void hellofirst()
{
printf("The first hello\n");
}

/* hellosecond.c */
#include <stdio.h>
void hellosecond()
{
printf("The second hello\n");
}
/* twohello.s */
void hellofirst(void);
void hellosecond(void);
int main(int argc,char *argv[])
{
   hellofirst();
   hellosecond();
   return(0);
}
```

Librairie statique

Une librairie statique est une collection de fichiers .o par la compilateur.

Une librairie statique est également appelée une archive.

Création d'une librairie statique

```
gcc -c hellofirst.c hellosecond.c    /* production des fichiers .o */
ar -r libhello.a hellofirst.o hellosecond.o /* création de la librairie libhello.a */
gcc twohellos.c libhello.a -o twohellos  /* Création d'un exécutable twohellos avec l'archive libhello */
```

Librairie partagée

Toutes les adresses (variable references and function calls) dans chaque module sont relatives au lieu d'être absolus, ce qui permet de charger les modules partagés dynamiquement et d'être exécutés pendant que le programme s'exécute.

Création d'une librairie partagée

```
gcc -c -fpic shellofirst.c shellosecond.c
gcc -shared shellofirst.o shellosecond.o -o hello.so /* création de la librairie hello.so ou la commande ci-dessous */
gcc -fpic -shared shellofirst.c shellosecond.c -o hello.so
gcc stwohellos.c hello.so -o stwohellos /*création de l'exécutable stwohellos */
```