# Exemple 0 (Cours 1)

## Options de compilation

Dans un terminal, aller dans le répertoire qui contient main.c et compiler le fichier main.c avec des options différentes

en tapant les 4 lignes:

```
gcc src/main.c -o Exemple0 -lm
gcc -O2 src/main.c -o Exemple0_Release -lm
gcc -O0 -g src/main.c -o Exemple0_Debug -lm
gcc -O2 -g src/main.c -o Exemple0_RelWithDebInfo -lm
```

Comparer les tailles des fichiers (taper **ls -l**) et les temps d'exécution (taper **time ./Exemple0** pour l'exécuter puis faire de même pour les autres exécutables)

## Utilisation du debugger

Dans ce même répertoire, compiler le fichier main2.c en mode debug:

```
gcc -O0 -g src/main2.c -o Exemple0_2_Debug -lm
```

L'exécution de Exemple0_2_Debug devrait s'arrêter sur une erreur.

Pour voir ce qui se passe, on peut exécuter le code sous le contrôle du debugger gdb.

Tapez:

```
gdb Exemple0_2_Debug
```

Le debuggeur se lance, charge le code, affiche un message qui se termine par :

> For help, type "help". Type "apropos word" to search for commands related to "word"... Reading symbols from ./Exemple0b... (gdb)

et se met en attente.

Lancer l'exécution, taper (à la fin de la ligne qui commence par **(gdb)**) :

```
run
```

L'exécution s'arrête avec un message similaire à :

```
Program received signal SIGSEGV, Segmentation fault.
0x0000555555555223 in A () at main2.c:21
21            x[k] = B(k);
```

qui indique quelle instruction a provoqué l'arrêt.

Afficher les valeurs des variables (plus généralement,les expressions) au moment de l'arrêt:

```
print(k)
print(x)
print(x[k])
```

ce qui donne en général la cause de l'erreur.

Il existe beaucoup d'autres fonctionnalités: consulter le manuel de gdb.

**Taper CTRL-D pour sortir du debugger.**

Recompiler sans l'option -g:

```
gcc -O2 src/main2.c -o Exemple0_2_Release -lm
```

Exécuter Exemple0_2_Release dans le debugger.

Quelle sont les informations que l'on perd en n'ayant pas utilisé l'option -g à la compilation ?

## Utilisation du profiler gprof:

Compilation

```
gcc -pg src/main.c -o Exemple0_GProf -lm
```

Execution et mesure des temps

```
./Exemple0_GProf
```

L'exécution génère un fichier *gmon.out* qui contient les mesures de temps et le nombre d'exécutions de chaque fonction.

La commande

```
gprof ./Exemple0_GProf
```

utilise le fichier *gmon.out* pour afficher le rapport des mesures.

Cela permet de savoir quelles fonctions prennent le plus de temps et donc qu'il faut optimiser en priorité.