

Programmation Langage C

Introduction et Éléments de base

Youssef ALJ

17 février 2020

Pré-requis :

- Initiation au langage C (cf cours du premier semestre).
- Mathématiques de base (Terminale S : un peu d'arithmétique).

Organisation des notes :

- 15 % participation (compte rendu de TP).
- 15 % contrôle continu.
- 70 % examen

- 1 Concepts de base (écriture d'un programme C, types de base, lecture/écriture).
- 2 Structures de contrôle.
- 3 Les boucles.
- 4 Tableaux en C.
- 5 Les fonctions.
- 6 Les chaînes de caractères.
- 7 Les pointeurs.

- Le langage C a été développé aux laboratoires AT&T dans les années 1970 par Dennis Ritchie.

Le cycle de création d'un programme en langage C est le suivant :

- 1 Concevoir un algorithme.
- 2 Utiliser un éditeur pour écrire le code source.
- 3 Compilation à partir du code source.
- 4 Édition des liens.
- 5 Éventuellement corriger les erreurs de compilation.
- 6 Exécuter le programme et le tester.
- 7 Éventuellement corriger les bugs.
- 8 Éventuellement Recommencer depuis le début.

On n'a pas besoin de mémoriser ces étapes. Ceci viendra avec la pratique.

Exemple : on veut écrire un algorithme qui affiche "bonjour tout le monde" à l'écran.

Début

Variables :

Écrire("Bonjour tout le monde")

Fin

- Avec un éditeur (par exemple notepad++) on saisit le texte suivant :

```
#include <stdio.h>

int main()
{
    printf("bonjour tout le monde");
    return 0;
}
```

- On sauvegarde (souvent avec les touches CTRL-s).
- Durant la sauvegarde, on fait attention à ce que le nom du fichier finisse par '.c' pour qu'il soit reconnu comme étant un programme C.
- Y a-t-il des mots qui vous sont familiers ?

Le compilateur est un programme spécial qui :

- lit les instructions enregistrées dans le code source "bonjour.c"
- analyse chaque instruction.
- traduit ensuite l'information en langage machine compréhensible par le micro-processeur de l'ordinateur.

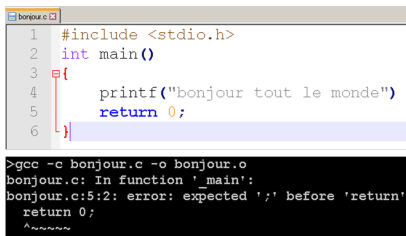
- Il existe plusieurs compilateurs :
 - gcc : GNU Compiler Collection.
GNU : acronyme récursif qui veut dire GNU's Not Unix.
 - Le compilateur : Microsoft Visual Studio.
 - ...
- On va utiliser le compilateur gcc. Voir TP.
- On compile en ligne de commandes windows (ou linux) via la commande suivante : `>gcc -c bonjour.c -o bonjour.o`
- Le résultat est la création d'un nouveau fichier appelé fichier objet qui a comme extension '.o' et qui a comme nom `bonjour.o`.

L'éditeur des liens (en anglais le linker) permet de :

- faire le liens entre les différents fichiers ".o"
- cette étape de link est aussi faite en utilisant `gcc`. `>gcc bonjour.o -o bonjour`
- on reviendra plus en détails sur le fonctionnement du linker dans les prochaines séances.

- Plusieurs compilateurs font la compilation et l'édition des liens en même temps.
- Cela se fait avec la commande suivante : `>gcc bonjour.c -o bonjour`

- Plusieurs erreurs de compilations peuvent apparaître.
- Elles sont dues par exemple à une mauvaise syntaxe (oubli du " ;", mot réservé mal écrit, etc.).
- Il faudra les corriger en éditant le fichier source, sauvegarder et recompiler puis ré-exécuter.



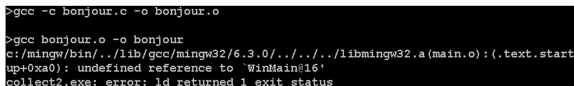
```
1  #include <stdio.h>
2  int main()
3  {
4      printf("bonjour tout le monde")
5      return 0;
6  }
```

```
>gcc -c bonjour.c -o bonjour.o
bonjour.c: In function '_main':
bonjour.c:5:2: error: expected ';' before 'return'
    return 0;
    ^~~~~~
```

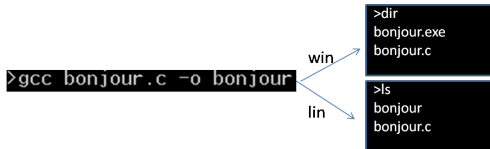
- Elles sont dues au fait que le linker n'arrive pas à faire le lien entre les fonctions définies par les programmes écrits.
- Une fonction principale doit toujours exister qui sert de point d'entrée aux autres fonctions.
- cette fonction principale est appelée la fonction `main`.



```
1 #include <stdio.h>
2 int _main()
3 {
4     printf("bonjour tout le monde");
5     return 0;
6 }
```



```
>gcc -c bonjour.c -o bonjour.o
>gcc bonjour.o -o bonjour
c:/mingw/bin/./lib/gcc/mingw32/6.3.0/../../../../libmingw32.a(main.o):(.text.startup+0xa0): undefined reference to `WinMain@16'
collect2.exe: error: ld returned 1 exit status
```



- Si la compilation s'est effectuée sans erreur on verra nouveau fichier qui apparait.
- Ce fichier est appelé un exécutable.
- Sous Windows, ce fichier exécutable porte l'extension '.exe' (voir figure ci-dessus).
- Pour lancer le fichier exécutable, dans notre exemple, on saisit en ligne de commande Windows : `>bonjour`.
- Dans un système d'exploitation de type Unix on saisit : `./bonjour`

- On utilise un programme appelé `gdb`.
- On reviendra sur la correction des bugs dans les prochaines séances.

Synthèse des principales étapes :

1

Edition

```
bonjour.c x
1  #include <stdio.h>
2  int main()
3  {
4      printf("bonjour tout le monde");
5      return 0;
6  }
```

2

Compilation
+
Edition des liens

```
>gcc bonjour.c -o bonjour
```

win

lin

```
>dir
  bonjour.exe
  bonjour.c
```

```
>ls
  bonjour
  bonjour.c
```

3

Exécution

win

```
>bonjour
```

```
bonjour tout le monde
```

lin

```
>./bonjour
```

```
bonjour tout le monde
```



```
#include <stdio.h>
// ceci est un commentaire

int main()
{
    // un autre commentaire

    /*
    Une autre facon
    d ecrire un commentaire sur plusieurs lignes
    */
    printf("bonjour tout le monde");
    return 0;
}
```

- Les commentaires sont un outil de base pour documenter son programme.
- Cette documentation n'est pas utile seulement pour les autres mais pour vous aussi.
- Les commentaires améliorent la lisibilité du code écrit.
- Sans commentaire la lecture est difficile.
- Il est donc recommandé d'ajouter des commentaires autant que possible.

Toujours par souci de lisibilité il est recommandé respecter les règles d'indentation :

- Décaler d'un cran (= trois espaces) vers la droite tout bloc inclus dans un précédent.
- Cela permet de repérer qui dépend de quoi et si tous les blocs ouverts sont fermés.

Exemple d'un programme C mal écrit

```
#include <stdio.h>

void main()
{
    int a = 0;
    if (a == 0)

{
    printf("a est egal a 0");
}
else
{
    printf ("a est different de 0");
}
}
```

Exemple d'un programme C mieux écrit

```
#include <stdio.h>
// fonction main indispensable pour chaque programme C
void main()
{
    int a = 0;
    // si a est nul on affiche : "a est nul"
    if (a == 0)
    {
        printf("a est egal a 0");
    }
    // sinon on affiche : "a n est pas nul"
    else
    {
        printf ("a est different de 0");
    }
}
```