Programmation C / C++

#1 Compilation

par David Albert

GEMA
GRANDES ÉCOLES DES MÉTIERS D'AVENIR

2024

Table des matières

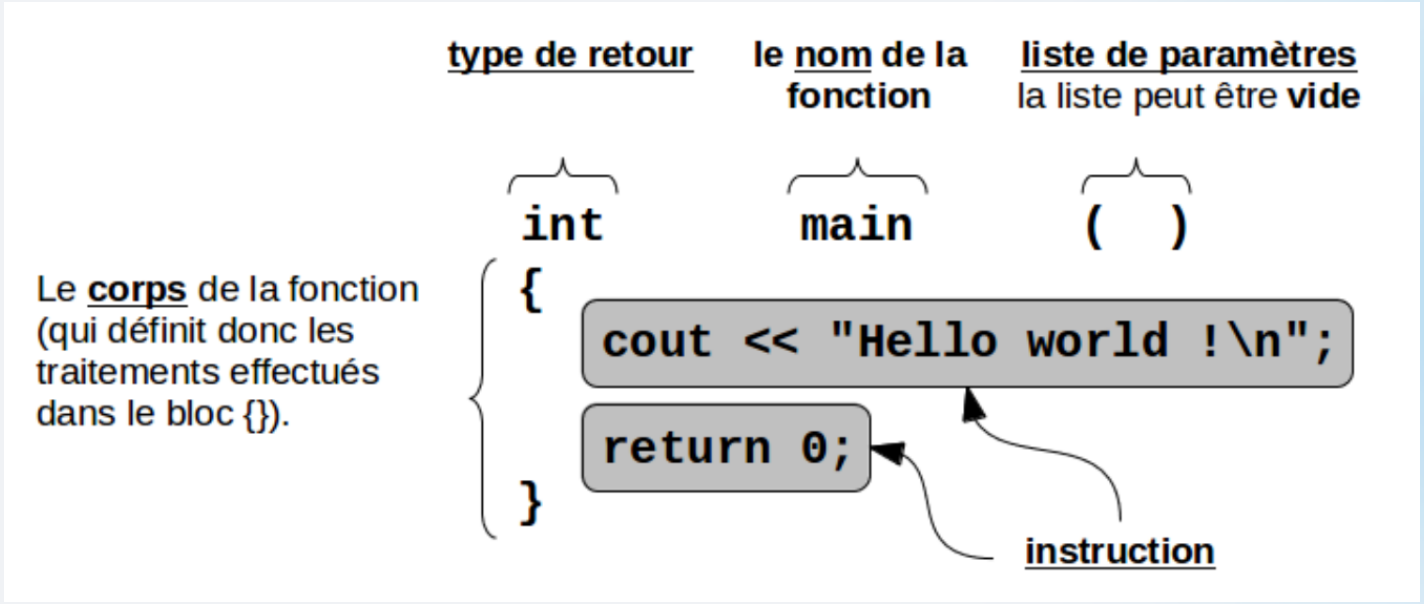
01

Premier programme

Premier programme C++

Voici un exemple de programme écrit en C++.

```cpp
// Ce programme affiche le message "Hello world !" à l'écran
#include <iostream>

int main()
{
    std::cout << "Hello world !\n"; // Affiche "Hello world !"
    return 0;
}
```



!

Tout programme C/C++ doit posséder une (et une seule) fonction nommée ***main*** (dite fonction principale) pour **indiquer où commencer l'exécution**.

Equivalent en C

Voici un exemple de programme écrit en C.

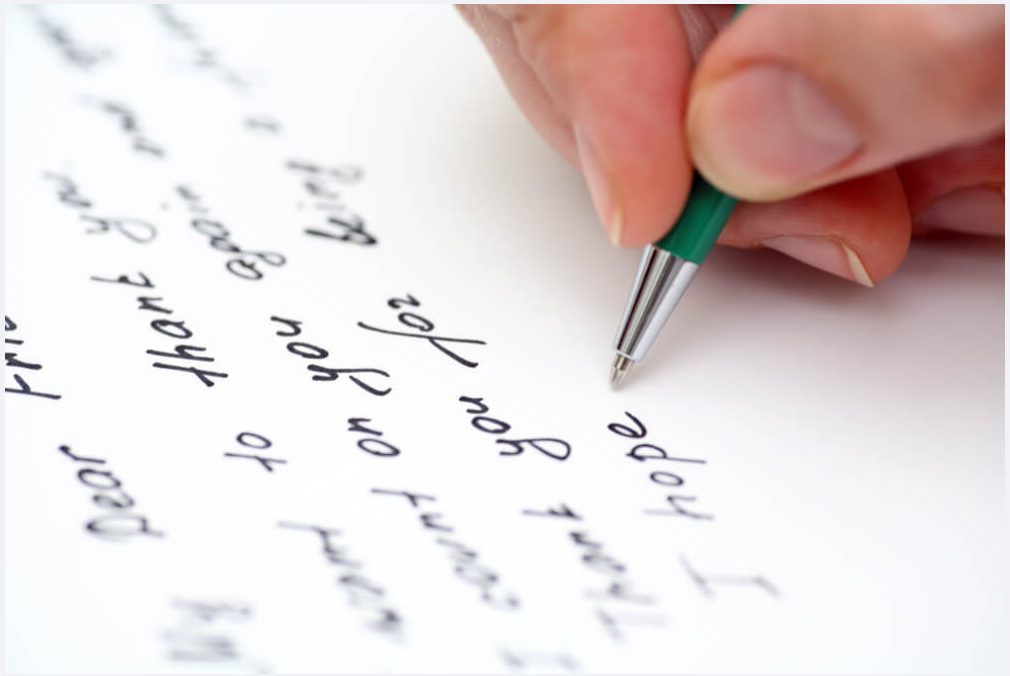
```c
// Ce programme affiche le message "Hello world !" à l'écran
#include <stdio.h> /* pour printf */

int main()
{
    printf("Hello world !\n"); // Affiche "Hello world !"
    return 0;
}
```

02

Chaîne de compilation

Rappel

Langage humain...



Langage machine...

!

Nous avons besoin d'un moyen de passer de l'un à l'autre

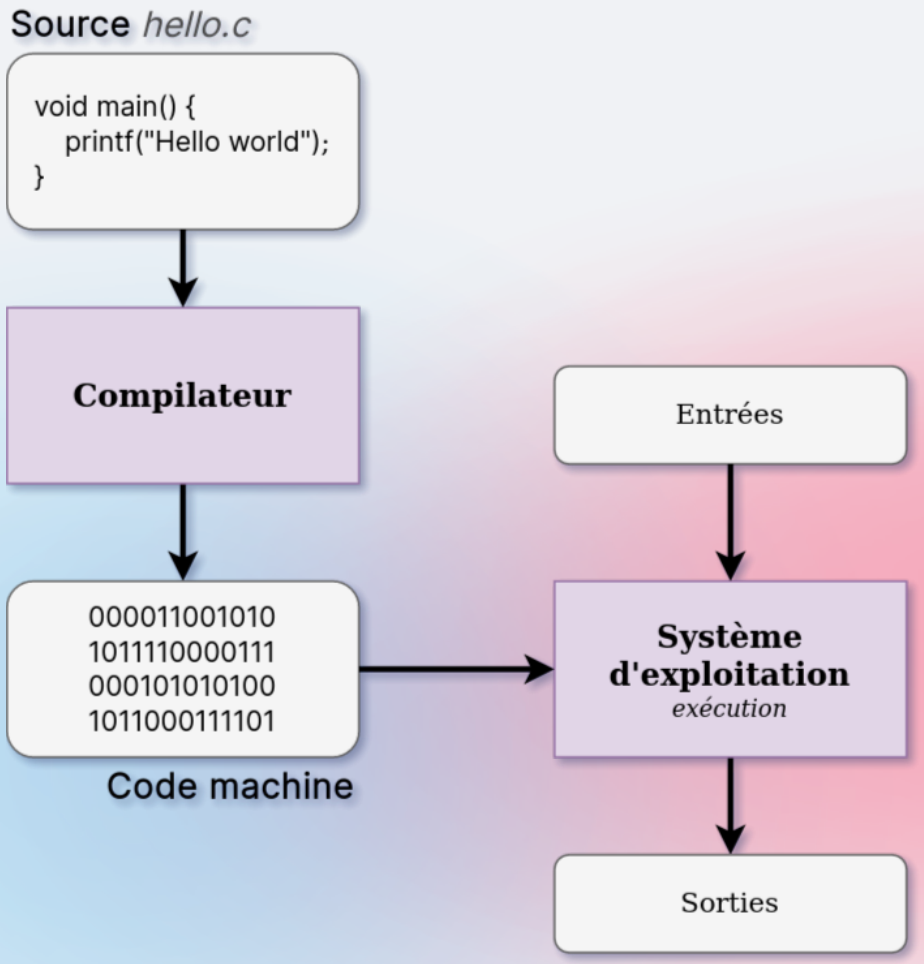
Les langages compilés...

Dans les langages compilés, le code source (le votre) est traduit en code binaire (celui compris par l'ordinateur) grâce à un logiciel (le **compilateur**).

Le résultat de la compilation est le programme exécutable.

```
./monProg # on peut l'exécuter directement
```

Exemples : C, C++, Pascal, Ocaml



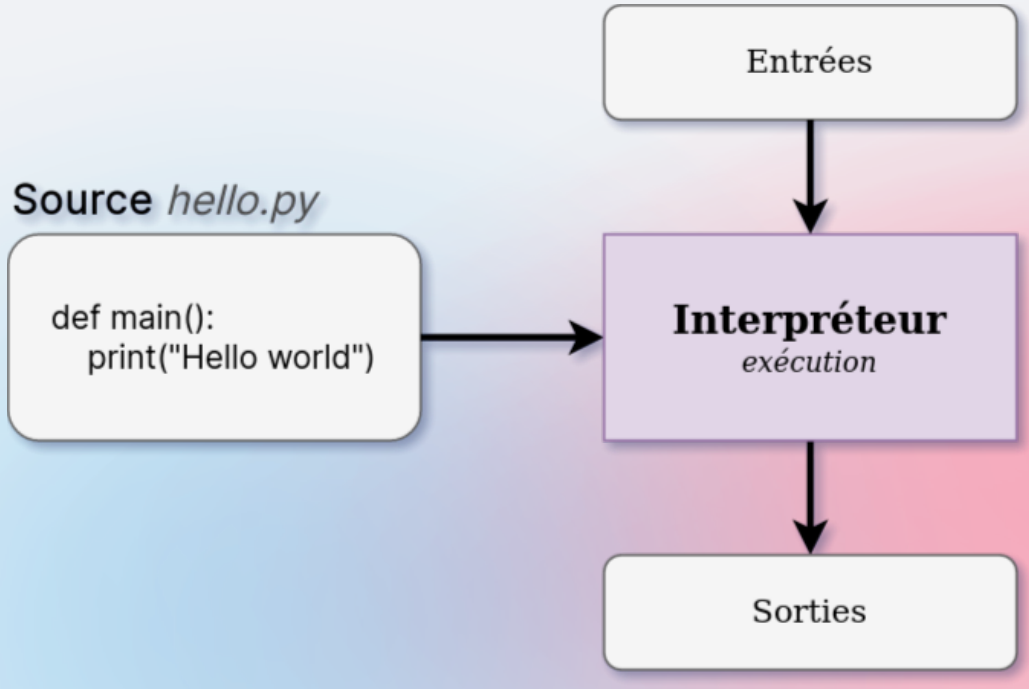
... et les langages interprétés...

Dans les langages interprétés, le code source (le votre) est interprété par un logiciel (l' **interpréteur**).

Pour exécuter notre programme, on appelle l'interpréteur.

```
python monProg.py # c'est l'interpréteur qui s'occupe
# de l'exécution du programme
```

Exemples: Java, **Python**, Bash



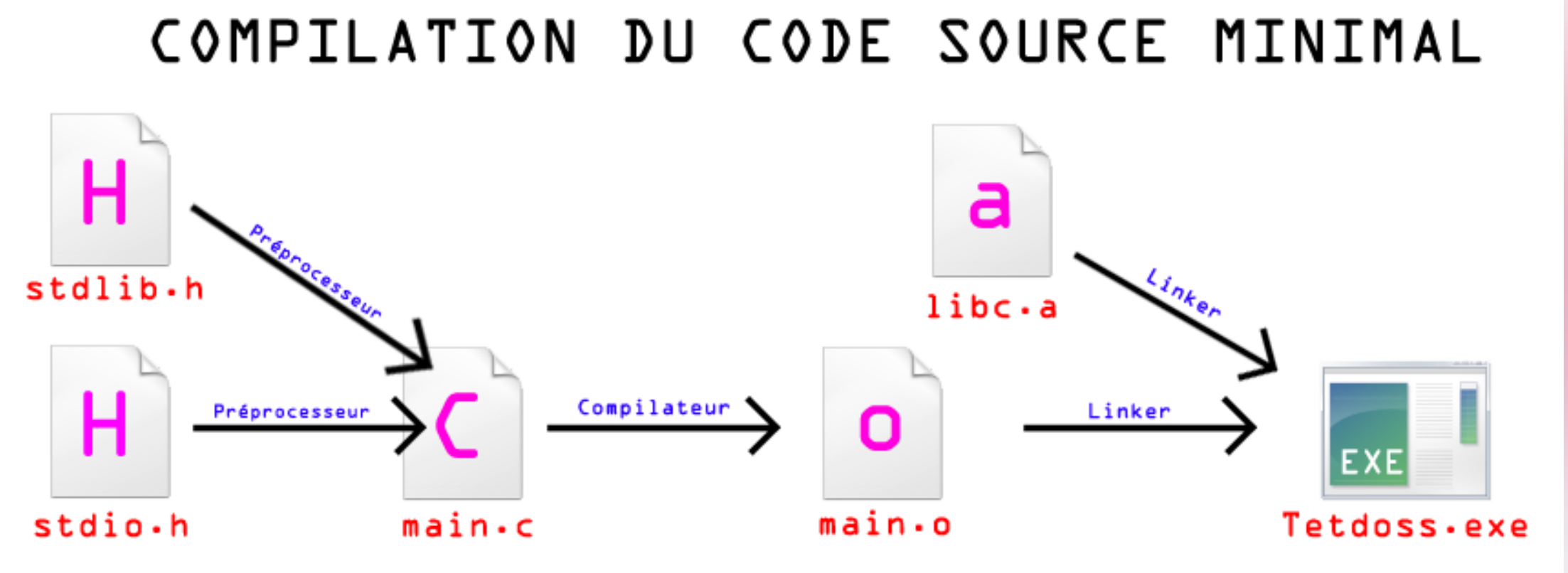
Chaîne de compilation

Les différentes étapes de fabrication d'un programme sont :

1. Le **préprocesseur** (pré-compilation)
    - Traitement des directives qui commencent toutes par le symbole dièse (**#**)
    - Inclusion de fichiers (.h) avec #include
    - Substitutions lexicales : les "macros" avec #define
2. La **compilation**
    - Vérification de la syntaxe
    - Traduction dans le langage d'assemblage de la machine cible
3. L'**assemblage**
    - Traduction finale en code machine (appelé ici code objet)
    - Production d'un fichier objet (.o ou .obj)
4. L'**édition de liens**
    - Unification des symboles internes
    - Étude et vérification des symboles externes (bibliothèques .so ou .DLL)
    - Production du programme exécutable

Chaîne de compilation

1. Le **préprocesseur** (pré-compilation)
2. La **compilation**
3. L'**assemblage**
4. L'**édition de liens**



Le compilateurs `g++`

1. Le **préprocesseur** (pré-compilation)
   - `g++ -E <fichier.cpp> -o <fichier_precompile.cpp>`
2. La **compilation**
   - `g++ -S <fichier_precompile.cpp> -o <fichier.s>`
     // ou pré-compilation, compilation et assemblage ensemble :_
   - `g++ -c <fichier.cpp> -o <fichier.o>`
3. L'**assemblage**
   - `as <fichier.s> -o <fichier.o>`
4. L'**édition de liens**
   // un seul fichier :
   `g++ <fichier.o> -o <executable>`
   // ou plusieurs fichiers :
   `g++ <fichier1.o> <fichier2.o> <fichiern.o> -o <executable>`

Le compilateurs `g++`

Fabrication

Nous pouvons combiner en une seule ligne de commande toutes les étapes (préprocesseur, compilation, assemblage et édition des liens) :

```
g++ <fichier.cpp> -o <executable>**
```

Note :
Le compilateur `gcc` permet de compiler des fichiers écrits en C.
A quelques instructions près, il s'utilise de la même façon que `g++`.

03

Préprocesseur

Preprocessor

Une particularité propre du compilateur `g++` est qu'il n'opère pas directement sur le code source fournit par le programmeur. Une phase spéciale de réécriture du programme précède toute compilation. L'utilitaire se chargeant de cette phase de pré-traitement se nomme le **préprocesseur**.

Macrogénération de code
Le langage C possède un moyen de réaliser de la macrogénération de code. Cette macrogénération ne peut s'effectuer que durant la phase du préprocesseur. Il apparaît donc que ce mécanisme ne fait pas partie intégrante du langage.

!

Toutes les instructions du préprocesseur commencent par un **#**.

Instructions du preprocessor

Instruction *#define*
L'instruction **#define** du préprocesseur permet de définir une macro constante ou macro paramétrée.

Instruction *#undef*
Cette instruction permet d'annuler une définition de macro.

Instruction *#include*
L'instruction #include permet d'inclure un fichier dans celui en cours de traitement.

Instructions *#ifdef ... #else ... #endif*
Permet d'introduire une portion de code selon un critère particulier.

i

Des exemples d'utilisation des instructions du preprocessor seront donnés en TP.