Mise à niveau

L.Pierre
Laurence.Pierre@univ-grenoble-alpes.fr

Mini-cours pour mise à niveau C

- Séances pour la mise à niveau
  - Cours lundi 9h45-11h15
  - TP mardi 15h30-18h45 + vendredi 13h30-16h45
    - Commencer à **préparer avant mardi** après-midi
    - **Entre mardi soir et vendredi, travailler les exercices non faits** :
      - la séance de vendredi doit vous permettre de **poser des questions** / **vous faire aider** sur les exercices que vous n'auriez pas su faire
    - **Tous** les exercices doivent être **finis** au plus tard vendredi soir
- Cours = explications sur les points les plus durs
  - Compréhension des phases de **compilation** (sections 1,2)
    - => écriture modulaire, utilisation bibliothèques, création makefile,...
  - Bases de la **gestion mémoire**, pointeurs, allocations,... (sections 6,7)

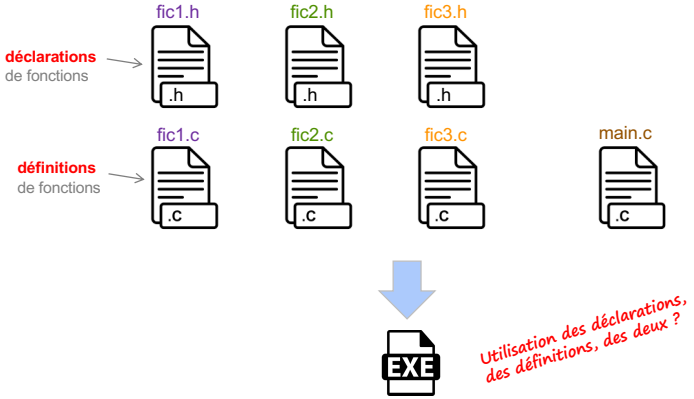
Compilation

Organisation modulaire

- L'organisation d'une application est généralement modulaire

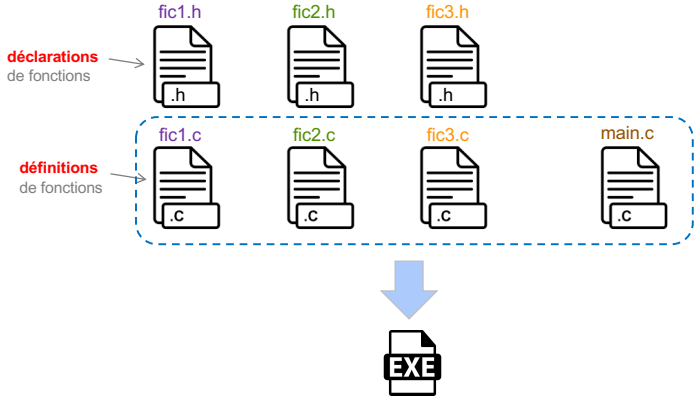
Organisation modulaire

- L'organisation d'une application est généralement modulaire



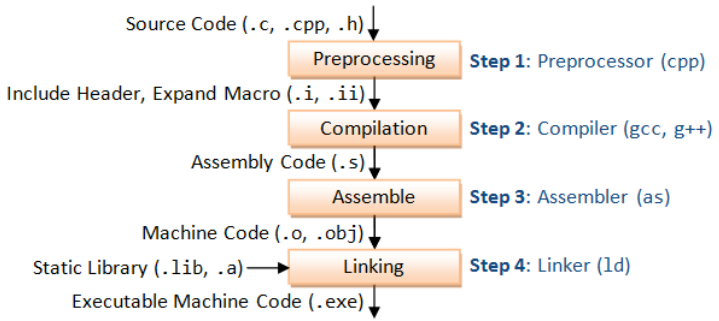
Organisation modulaire

- L'organisation d'une application est généralement modulaire



Étapes de la compilation

- Le compilateur C (e.g. gcc) réalise plusieurs étapes :



https://www3.ntu.edu.sg/home/ehchua/programming/cpp/gcc_make.html

Étapes de la compilation

main.c

```
#include <stdio.h>
#include "myfic.h"
#define M1 20

int main() {
  float x = 100.5;
  printf("The value of M1 is %d\n", M1);
  myfunction(x, M1);
  return 0;
}
```

myfic.h

```
#ifndef MYFIC_H
#define MYFIC_H

void myfunction(float, int);

#endif
```

myfic.c

```
void myfunction(float, int) {

  …

}
```

Étapes de la compilation

main.c

```
#include <stdio.h>
#include "myfic.h"
#define M1 20
int main() {
  float x = 100.5;
  printf("The value of M1 is %d\n", M1);
  myfunction(x, M1);
  return 0;
}
```

myfic.h

```
#ifndef MYFIC_H
#define MYFIC_H
void myfunction(float, int);
#endif
```

myfic.c

```
void myfunction(float, int) {
  …
}
```

(1) Preprocessing

```
gcc -E main.c
```

```
... Contenu de stdio.h ...
void myfunction(float, int);

int main() {
  float x = 100.5;
  printf("The value of M1 is %d\n", 20);
  myfunction(x, 20);
  return 0;
}
```


Étapes de la compilation

```
... Contenu de stdio.h ...
void myfunction(float, int);

int main() {
  float x = 100.5;
  printf("The value of M1 is %d\n", 20);
  myfunction(x, 20);
  return 0;
}
```

(2) Compilation

main.s

```
_main:                          ## @main
    ...
    movb    $0, %al
    callq   _printf
    movl    $20, %edi
    movss   -8(%rbp), %xmm0
    movl    %eax, -12(%rbp)
    callq   _myfunction
    ...
```


Étapes de la compilation

```
_main:                          ## @main
    ...
    movb    $0, %al
    callq   _printf
    movl    $20, %edi
    movss   -8(%rbp), %xmm0
    movl    %eax, -12(%rbp)
    callq   _myfunction
    ...
```

(3) Assembling

Étapes de la compilation

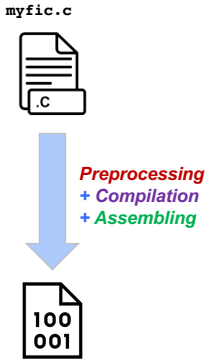
```
_main:                          ## @main
    ...
    movb    $0, %al
    callq   _printf
    movl    $20, %edi
    movss   -8(%rbp), %xmm0
    movl    %eax, -12(%rbp)
    callq   _myfunction
    ...
```

myfic.c

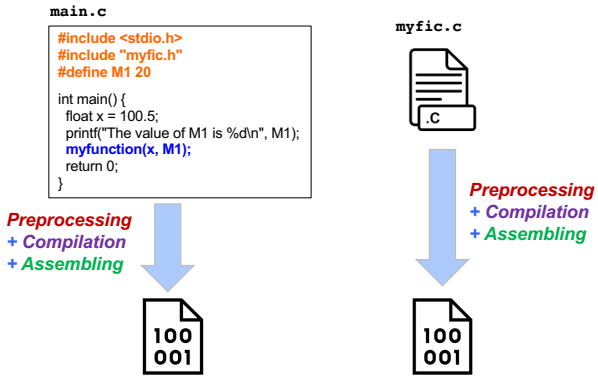
```
void myfunction(float, int) {
  …
}
```

(3) Assembling

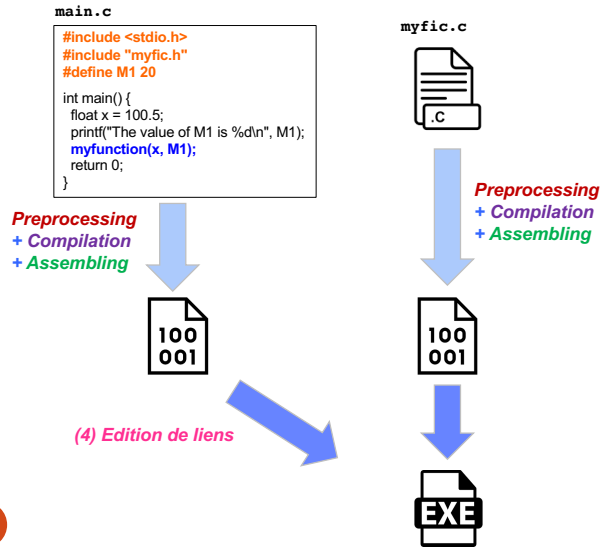
Étapes de la compilation



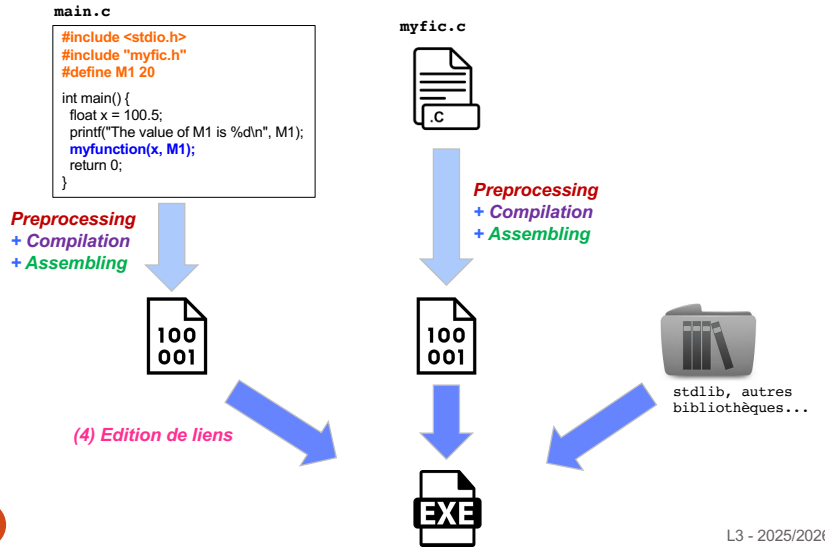
Étapes de la compilation



Étapes de la compilation

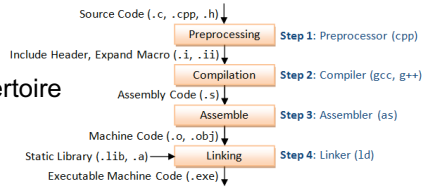


Étapes de la compilation



Étapes de la compilation

- Exemple simple :
  - 5 fichiers, dans le même répertoire
    - convertir.h, convertir.c
    - outils.h, outils.c
    - monpg.c
- Quelles phases ?




```
gcc -c convertir.c
```
préprocesseur, compilateur, et assembleur → convertir.o

```
gcc -c outils.c
```
préprocesseur, compilateur, et assembleur → outils.o

```
gcc -c monpg.c
```
préprocesseur, compilateur, et assembleur → monpg.o

```
gcc convertir.o outils.o monpg.o -o monpg
```
linker → monpg

Directives pour le préprocesseur

- Directive **#define**
  - Exemple simple

```
...
#define PUISSANCE 3

int main() {
  float nb, resultat;
  // Saisie au clavier
  printf("Saisir le nombre\n");
  scanf("%f", &nb);
  // Calcul
  resultat = pow(nb, PUISSANCE);
  // Affichage du resultat
  printf("Res. de l'elevation a la puissance = %f\n", resultat);
  return 0;
}
```

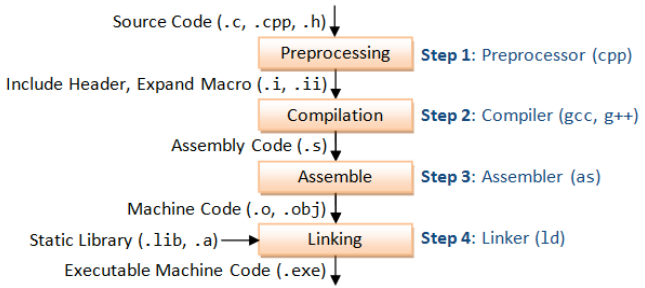

Directives pour le préprocesseur

- Directive **#define**
  - Exemple simple

```
...
#define PUISSANCE 3

int main() {
  float nb, resultat;
  // Saisie au clavier
  printf("Saisir le nombre\n");
  scanf("%f", &nb);
  // Calcul
  resultat = pow(nb, 3);
  // Affichage du resultat
  printf("Res. de l'elevation a la puissance = %f\n", resultat);
  return 0;
}
```


Directives pour le préprocesseur

- Directives
  - A partir de quelle étape le code traité ne contient-il plus aucune directive pour le préprocesseur (ni #define ni autre)



→ **Aucune** option de compilation à destination du **préprocesseur** n'aura de sens à partir de cette étape !

Directives pour le préprocesseur

- Directives **#if**, **#ifdef**, **#ifndef**
  - Les directives **#ifdef** et **#ifndef** permettent d'inclure ou non du code, suivant que la macro soit définie ou pas

```
#ifdef nom                  #ifndef nom
code                        code
#else                       #else
code                        code
#endif                      #endif
```

  - La directive **#if** permet d'inclure ou non du code, suivant la valeur d'une expression (valeur non nulle ≈ vrai)

```
#if expression
code
#else
code
#endif
```


Directives pour le préprocesseur

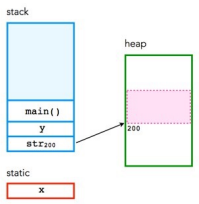
- Directives **#ifdef**, **#ifndef**
  - Exemple : **compilation "en mode debug"**

Commande de compilation :

```
gcc -DDEBUG fic.c
```

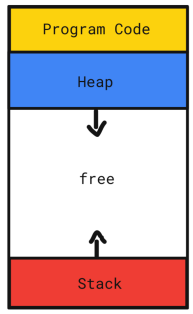
```
#include <stdio.h>

#ifndef MYINITVAL
#define MYINITVAL 100
#endif

void print_val(int i){
#ifdef DEBUG
  printf("  Integers are initialized with %d\n", MYINITVAL);
#endif
  printf("  The current value of this integer is %d\n", i);
}

int main() {
  int x = MYINITVAL;
  printf("Value of x?\n");
  print_val(x);
  ...
}
```


Gestion mémoire



Pile et tas

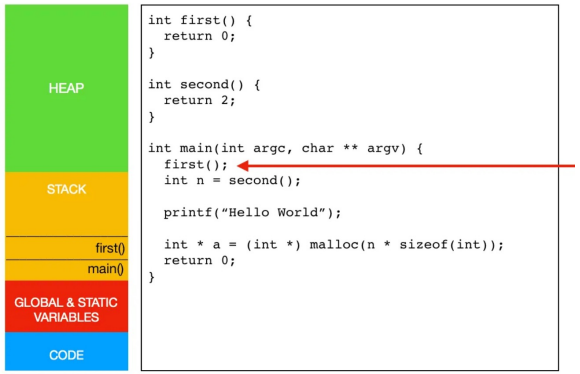
- Variables locales
  - Stockées sur la pile (stack)
  - Un cadre de pile (stack frame) par appel de fonction
- Blocs alloués dynamiquement
  - Dans le tas (heap)



https://www.linkedin.com/pulse/what-where-stack-heap-maxim-malisciuc

Pile et tas

- Exemple

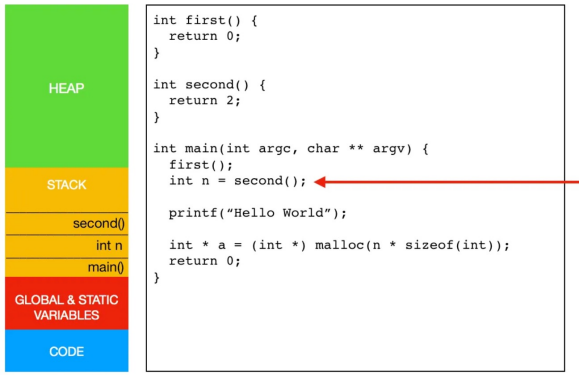



```
int first() {
  return 0;
}

int second() {
  return 2;
}

int main(int argc, char ** argv) {
  first();
  int n = second();

  printf("Hello World");

  int * a = (int *) malloc(n * sizeof(int));
  return 0;
}
```

https://medium.com/@wireless.patrick/a-tale-of-memories-stack-and-heap-7528f49aea6e

Pile et tas

- Exemple

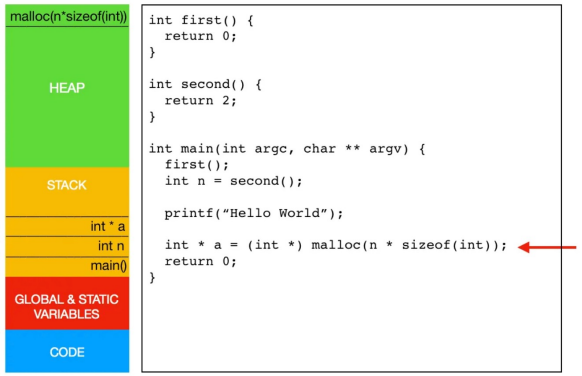



```
int first() {
  return 0;
}

int second() {
  return 2;
}

int main(int argc, char ** argv) {
  first();
  int n = second();

  printf("Hello World");

  int * a = (int *) malloc(n * sizeof(int));
  return 0;
}
```

https://medium.com/@wireless.patrick/a-tale-of-memories-stack-and-heap-7528f49aea6e

Pile et tas

- Exemple




```
int first() {
  return 0;
}

int second() {
  return 2;
}

int main(int argc, char ** argv) {
  first();
  int n = second();

  printf("Hello World");

  int * a = (int *) malloc(n * sizeof(int));
  return 0;
}
```

https://medium.com/@wireless.patrick/a-tale-of-memories-stack-and-heap-7528f49aea6e

Pointeurs, adresses

- Déclaration d'une variable de type pointeur
  - Exemples :

```
float *pf;
char *s;
```

- Opérateurs
  - Adresse d'une variable : `&`
  - Valeur pointée (repérée) par un pointeur : `*`
  - Exemples :

```
float *pf;
float a;
a = 5.92;
pf = &a;
*pf = 3.14;
```

Pointeurs, adresses

- Déclaration d'une variable de type pointeur
  - Exemples :

```
float *pf;
char *s;
```

- Opérateurs
  - Adresse d'une variable : `&`
  - Valeur pointée (repérée) par un pointeur : `*`
  - Exemples :

```
float *pf;
*pf = 3.14;
```

Pointeurs, adresses

- Déclaration d'une variable de type pointeur
  - Exemples :

```
float *pf;
char *s;
```

- Opérateurs
  - Adresse d'une variable : `&`
  - Valeur pointée (repérée) par un pointeur : `*`
  - Exemples :

```
float *pf;
pf = (float *)malloc(sizeof(float));
*pf = 3.14;
```

Où est `pf` ? où est la valeur repérée par `pf` ?