HE VD IG

Série 0x26

Solution

Compilation séparée et bibliothèques

INFO2-TIN — 3 mars 2026, 7efbe5

Exercice 1 : Généralités

(a) Parmi les propositions suivantes, lesquelles sont vraies pour un projet C en compilation séparée ?

A. Chaque fichier `.c` est compilé indépendamment en fichier objet `.o`
B. L'éditeur de liens (linker) assemble les fichiers objets
C. Les fichiers `.h` sont compilés en `.o`
D. Un symbole global peut être défini dans un seul fichier `.c`

(a) **A, B, D**

(b) Quel est le rôle d'un fichier d'en-tête `.h` ?

A. Déclarer les fonctions, types et constantes accessibles aux autres fichiers
B. Contenir des définitions de variables globales sans `extern`
C. Remplacer le besoin de compiler les `.c`
D. Contenir le code exécutable final

(b) **A**

(c) On dispose de `main.c` et `maths.c`, avec un en-tête `maths.h`. Quelle commande compile et lie correctement l'exécutable `app` ?

A. `cc -o app main.c maths.h`
B. `cc -c main.c maths.c -o app`
C. `cc -c main.c && cc -c maths.c && cc -o app main.o maths.o`
D. `cc -o app main.o maths.h`

(c) **C**

Exercice 2 : Analyse de code

(a) On considère les fichiers suivants :

```c
// maths.h
#ifndef MATHS_H
#define MATHS_H

double mean(const double *values, size_t n);

double sum(const double *values, size_t n);

#endif
```

```c
// maths.c
#include "maths.h"

double sum(const double *values, size_t n) {
    double s = 0.0;
    for (size_t i = 0; i < n; i++) s += values[i];
    return s;
}

double mean(const double *values, size_t n) {
    return sum(values, n) / n;
}
```

```c
// main.c
#include <stdio.h>
#include "maths.h"

int main(void) {
    double v[] = {1.0, 2.0, 3.0, 4.0};
    printf("%g\n", mean(v, 4));
}
```

1. Quel fichier doit contenir les **définitions** des fonctions ?
2. Quel fichier doit contenir les **déclarations** des fonctions ?
3. Pourquoi le `#ifndef` est-il utile ?

> 1. Les définitions doivent être dans `maths.c`.
> 2. Les déclarations doivent être dans `maths.h`.
> 3. Le garde d'inclusion évite les inclusions multiples et les redéfinitions.

Exercice 3 : Programmation

(a) Créez un module `stats` qui expose les fonctions suivantes :
— `double stats_min(const double *v, size_t n);`
— `double stats_max(const double *v, size_t n);`
Indiquez ce qui doit se trouver dans `stats.h` et dans `stats.c`.

```
// stats.h
#ifndef STATS_H
#define STATS_H
#include <stddef.h>
double stats_min(const double *v, size_t n);
double stats_max(const double *v, size_t n);
#endif
```

```
// stats.c
#include "stats.h"
double stats_min(const double *v, size_t n) {
    double m = v[0];
    for (size_t i = 1; i < n; i++) if (v[i] < m) m = v[i];
    return m;
}
double stats_max(const double *v, size_t n) {
    double m = v[0];
    for (size_t i = 1; i < n; i++) if (v[i] > m) m = v[i];
    return m;
}
```

(b) On souhaite créer une bibliothèque statique `libgeom.a` contenant `geom.c` et `geom.h`. Donnez les commandes nécessaires pour :

1. Compiler `geom.c` en fichier objet.
2. Créer la bibliothèque statique.
3. Lier un programme `main.c` avec cette bibliothèque.

```
cc -c geom.c -o geom.o
ar rcs libgeom.a geom.o
cc main.c -L. -lgeom -o app
```

(c) Complétez le `Makefile` minimal suivant pour construire `app` à partir de `main.c` et `maths.c`.

```
CC = cc
CFLAGS = -Wall -Wextra -std=c11

app: main.o maths.o
    # TODO

main.o: main.c maths.h
    # TODO

maths.o: maths.c maths.h
    # TODO
```

```
app: main.o maths.o
    $(CC) $(CFLAGS) -o app main.o maths.o

main.o: main.c maths.h
    $(CC) $(CFLAGS) -c main.c

maths.o: maths.c maths.h
    $(CC) $(CFLAGS) -c maths.c
```