

Info2 - Compilation Séparée

17. 8. 2021

Modèle: 1 fichier source C (main + fonctions)

↳ gestion des arguments

↳ gestion des erreurs

↳ gestion des fichiers

Maintenabilité: facile

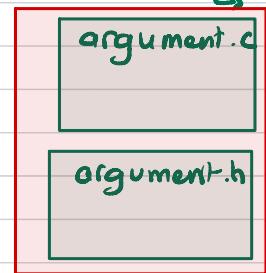
Partage des fonctions: compliqué

Décomposition du source C en plusieurs

MODULES logiciels

↳ .c

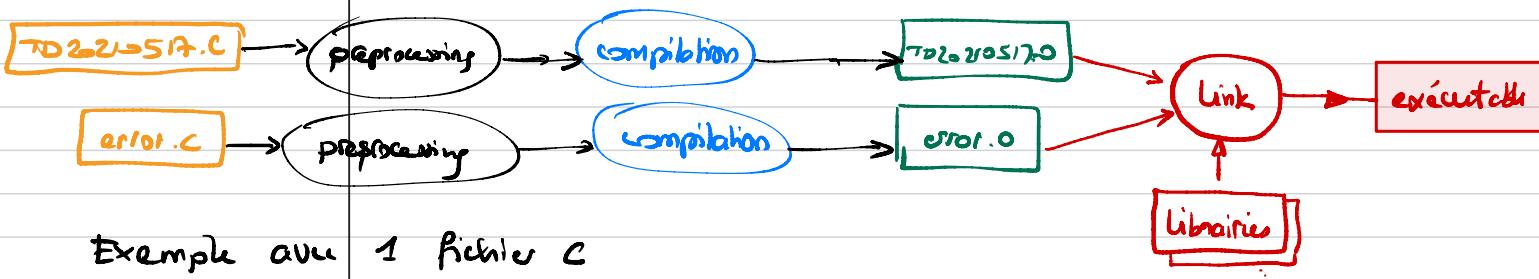
↳ .h



module "argument"

module "error"

Compilation: "make" → cherche tous les fichiers ".c" pour les compiler et effectuer le "link" pour générer l'exécutable.



Exemple avec 1 fichier C

```
→ TD20210517 git:(main) ✘ make
→ gcc -std=c99 -Wall -g -Iinclude -c TD20210517.c -o obj/TD20210517.o -MMD -MF obj/TD20210517.d
→ gcc -o app obj/TD20210517.o -lm
```

↓ ↓ ↓ ↓

exéc. objet source objet

```
→ TD20210517 git:(main) ✘ make
→ gcc -std=c99 -Wall -g -Iinclude -c error.c -o obj/error.o -MMD -MF obj/error.d
→ gcc -std=c99 -Wall -g -Iinclude -c TD20210517.c -o obj/TD20210517.o -MMD -MF obj/TD20210517.d
→ gcc -o app obj/error.o obj/TD20210517.o -lm
```

Les 2 fichiers .c sont compilés SÉPARÉMENT

TD 20210517

↳ TD 20210517.c

↳ main

↳ void displayError (eErrorCode e);

Créer le type enumératif eErrorCode avec
E-NONE = 0
E-ERROR-1
E-ERROR-2.

Main: récupérer le nbre d'argument : n

Si $n = 2 \rightarrow$ displayError (E-NONE)
 $< 2 \quad ..$
 $> 2 \quad E-ERROR-1$
 $E-ERROR-2$

TD 20210517

TD20210517.c

#include < stdio.h
#include < stdint.h

#include "error.h"

main

ERROR

error.c "implémentation du module"

#include "error.h"

code de la fonction displayError

error.h "présentation du module"

#include < stdio.h> //printf

typedef enum ... eErrorCode

prototype fonction displayError

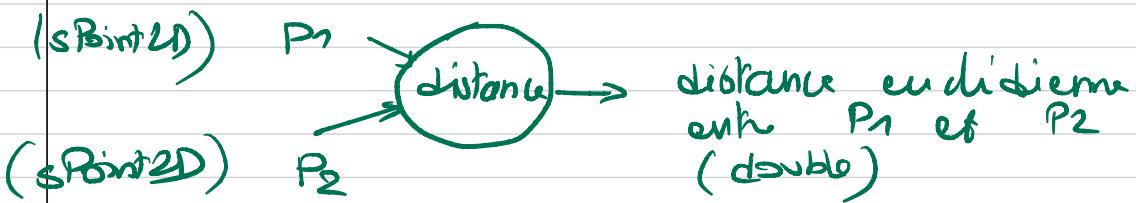
14/02

TD 2021-2022

création d'un nouveau module. computation

typedef structure "sPoint2D" (x et y double)

fonction distance



code de retour : `NO_ERROR` si $dist > 0$
`ERROR_3` si $dist == 0$.

Dans le main,

- le calcul de la distance entre P_1 et P_2 avec affichage de la distance
- idem entre P_1 et P_3

computation.c

computation.h

19h25

```

→ TD20210517 git:(main) ✘ make
mkdir obj
gcc -std=c99 -Wall -g -Iinclude -c error.c -o obj/error.o -MMD -MF obj/error.d
gcc -std=c99 -Wall -g -Iinclude -c TD20210517.c -o obj/TD20210517.o -MMD -MF obj/TD20210517.d
In file included from computation.h:4:0,
  from TD20210517.c:16:
error.h:7:5: error: redeclaration of enumerator 'E_NO_ERROR'
  E_NO_ERROR = 0,
  ^~~~~~

```

Compilation: 1) preprocessing TD20210517.c

```

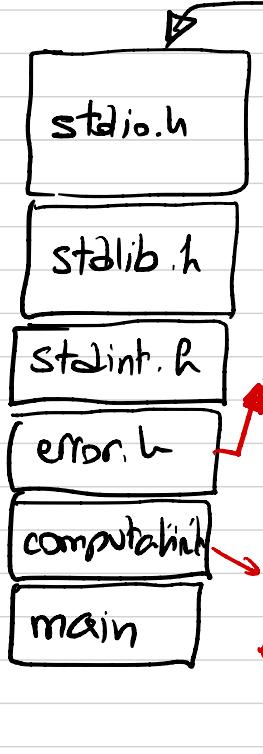
#include <stdio.h> // standard library
#include <stdlib.h>
#include <stdint.h>

#include "error.h"
#include "computation.h"

int main(int argc, char const *argv[])
{

```

→ preprocessing



```

#include <stdio.h> // standard library

typedef enum
{
    E_NO_ERROR = 0,
    E_ERROR_1,
    E_ERROR_2,
    E_ERROR_3
} eErrorCode;

void displayError(const eErrorCode e);

#include <math.h>
#include "error.h"

typedef struct {
    double x;
    double y;
} sPoint2D;

eErrorCode distance(c

```

stdio.h
stdlib.h
stdint.h
error.h
computation.h
main

error.h > en double

```

#include <stdio.h> // standard library

typedef enum
{
    E_NO_ERROR = 0,
    E_ERROR_1,
    E_ERROR_2,
    E_ERROR_3
} eErrorCode;

void displayError(const eErrorCode e);

```

le fichier error.h est inclus 2x lors du preprocessing !

```

// error.h

#pragma once

#include <stdio.h> // standard library

typedef enum
{
    E_NO_ERROR = 0,
    E_ERROR_1,
    E_ERROR_2,
    E_ERROR_3
} eErrorCode;

void displayError(const eErrorCode e);

```

#pragma once concerne le fichier dans lequel il est placé

Si donc être au début !

et alors TOUS vos .h