

## M2 CCI

*UE Programmation et Langages*

Cours « Programmation »

*Laurent Mounier*

# Exemple de programme C

```
#include <stdio.h>
#define M -1
int main() {
    int x ;
    int s=0 ;

    printf("entrez une valeur entiere \n") ;
    scanf ("%d", &x) ;

    while (x != M) {
        if (x > 0) {
            s = s + x ;
        } ;
        printf("entrez une valeur entiere \n") ;
        scanf ("%d", &x) ;
    } ;

    printf("le resultat est %d \n", s) ;
    return 0 ;
}
```

bibliothèque (entrée/sorties)

déclaration de constante

début du « programme principal »

lexique

instructions

fin du « programme principal »

# bibliothèques

## Accès à des **fonctions et données existantes**

- fournies par le  **système d'exploitation**  :
  - entrées/sorties, gestion des fichiers, réseaux, etc.
  - fonctions mathématiques (trigo, logarithme, etc.)
  - chaînes de caractères, etc.
- fournies par  **des tiers ou par le développeur**  (ré-utilisation de code) pour des besoins spécifiques
  - Affichage 3D
  - Gestion de données spécifiques
  - Interaction avec une base de données, des capteurs, etc.

**Remarque :** `<stdio.h>` désigne le fichier `/usr/include/stdio.h`

# Déclaration de constantes

```
#define M -1
```

→ remplace la chaîne de caractère M par -1 dans la suite du fichier

- pré-traitement du compilateur (pre-processor)
- pas une « vraie » définition de constante (cf. slide suivant)
- mécanisme assez largement utilisé en C

# Programme principal

- **point d'entrée** du programme = début de l'exécution
- le nom « main » est imposé ...
- défini une **fonction qui renvoie un entier**  
(main : la fonction  $\rightarrow$  entier)
- **valeur de retour** (convention pour l'interpréteur de commande)
  - 0 si pas d'erreur
  - $\neq 0$  sinon (code d'erreur)
- Il peut exister des **paramètres** (cf. plus tard ...)

**Remarque** : un programme C peut aussi définir et utiliser d'autres fonctions !

# Lexique

```
int x ; // x : un entier
```

déclare une variable de **nom** x et de **type** int (entier relatif)

```
int s=0 ; // s : un entier ← 0
```

déclare une variable de **nom** s, de **type** int, **initialisée** à la valeur 0

- Il existe d'autres **types** (cf. prochain cours!) : réels, caractères (mais pas de booléens ...)
- les **noms des variables** (*identificateurs*) sont des suites de lettre/chiffres/autres ne débutant pas par un chiffre ...
- autre manière de déclarer une constante :     const int s = 0

# Instructions

- Actions élémentaires :

- affectation :

- `s = s + x // s ← s + x`

- appels de fonctions ...

- `scanf ("%d", &x) ; // lire (x)`

- `printf ("le resultat est %d \n", s) ; // ecrire (« le résultat est », s)`

- Composition d'actions (structures de contrôles) :

- séquentielle : ;

- conditionnelle : `if (x > 0) { ...} // si x > 0 alors ...`

- Itérative : `while (x != M) {...} // tantque x ≠ M faire ...`

→ voir la fiche de traduction « notation algo » ↔ langage C ...

# Compilation

## Rappel :



→ sous Linux : utilisation du compilateur **gcc**

```
gcc -Wall exemple1.c -o exemple 1
```

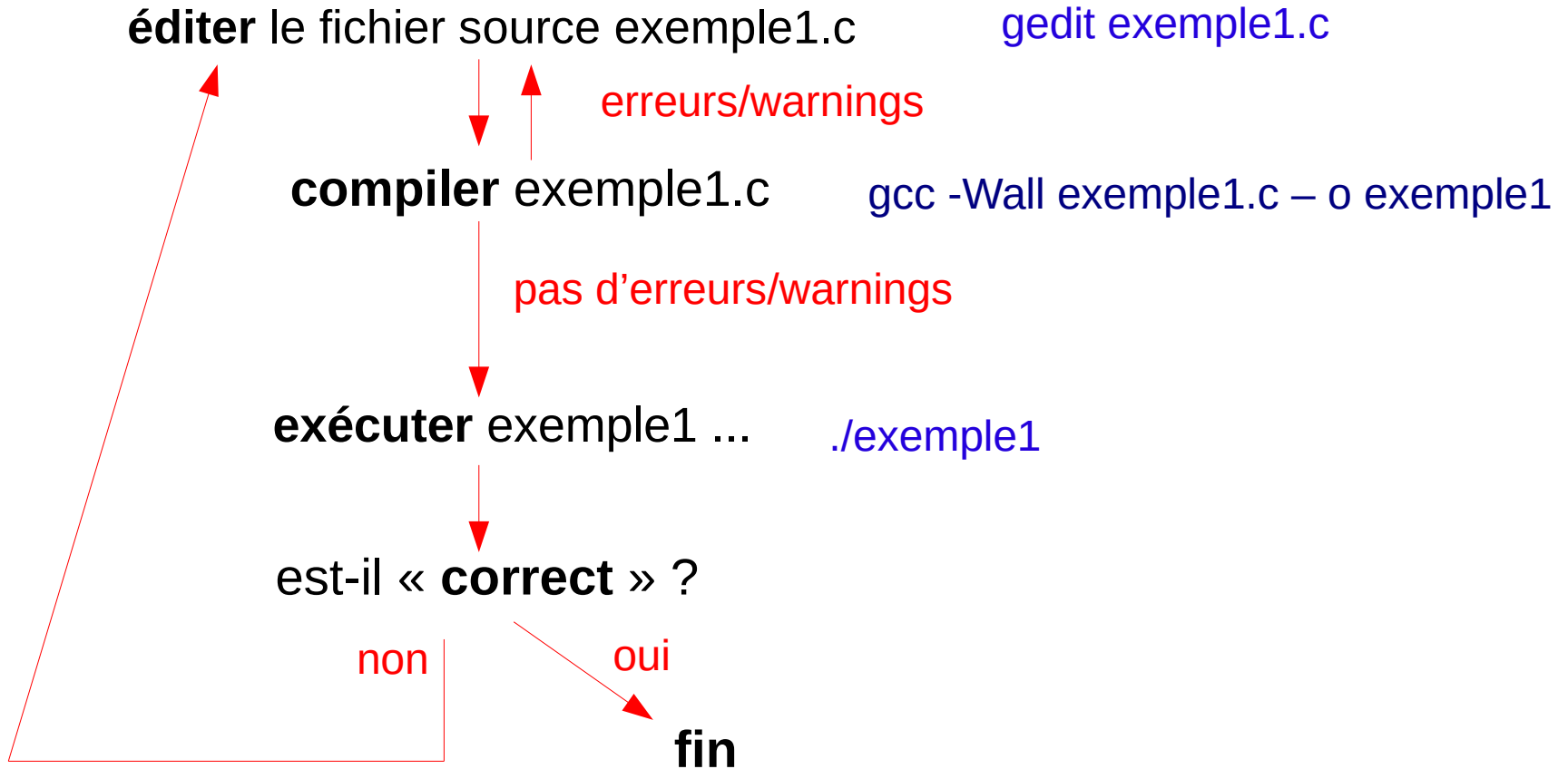
active tous les « warnings »

fichier source

fichier exécutable



# Développement d'un programme ...



**Remarque :** on pourra aussi utiliser VSCode ...

# Mon programme est-il « correct » ?

## correct ?

- les résultats obtenus sont (toujours ?) *les bons* ... (~ cahier des charges)
- les performances (temps, mémoire, consommation) sont « acceptables »
- Il est « robuste » (à des utilisations maladroites/malveillantes)

→ pas de techniques existantes pour garantir ces propriétés « par construction » ...

## Une approche possible en pratique : **test du programme**

- plusieurs méthodes possibles, certaines (semi-)automatiques ...
- dans un premier temps : tests manuels « simples »
  - essayer de « **couvrir** » l'**ensemble des entrées**
  - essayer d'**obtenir l'ensemble des résultats possibles**
  - essayer des **entrées « incorrectes »** (que le programme devrait rejeter)

# Exercices

- 1 Comment tester le programme donné en exemple ?
- 2 Ecrire un algorithme qui permet de lire un entier  $n$  au clavier et d'afficher la somme des entiers de 0 à  $n$  ()
- 3 Coder cet algorithme en C ...
- 4 Comment testez-vous votre programme ?