Plan

Caractéristiques de la fonction main

La fonction `main` est la fonction automatiquement appelée lors de l'exécution du programme. Un programme n'est **exécutable** que s'il contient une (unique) fonction `main`.

L'exécution commence à la première instruction de son bloc, et se termine dès que l'une des conditions suivante arrive :

- la fin d'exécution de la **dernière instruction** du bloc du `main` ;
- l'exécution d'une instruction `return` du (premier appel) du main ;
- un appel à la fonction `exit` (dans le `main` ou une fonction appelée).

Un même programme peut être exécuté plusieurs fois (séquentiellement ou simultanément) : on appelle **processus** une instance d'exécution d'un programme.

Valeur de retour du main

Un processus qui termine de lui-même (pas par un `kill` ou `^C`) renvoie une valeur (comprise entre 0 et 255) dénommée **statut de terminaison**.

- Cette valeur est récupérée par le système d'exploitation
- Par convention (détails dans documentation de chaque programme) :
  - 0 si c'est un cas normal d'arrêt du programme
  - une valeur positive en cas d'arrêt anormal (prévu !)

En cas d'erreur, une bonne pratique est d'**afficher un message** explicitant l'erreur en plus d'une valeur positive de terminaison. Exemple :

```
> ls
fic
> echo $?
0
> ls rep
ls: rep: No such file or directory
> echo $?
1
```

Sous linux, le statut de la dernière commande terminée est stockée dans la variable d'environnement $? et peut être consulté par un `echo $?`.

Codes d'erreur des fonctions de la bibliothèque

La bibliothèque `stdlib.h` définit une fonction permettant d'arrêter (proprement) un programme avec un code de retour depuis n'importe quelle fonction (et pas seulement le `main`) :

Définition

```
void exit(int status)
```

Deux constantes prédéfinies existent :

- EXIT_SUCCESS en cas d'arrêt normal
- EXIT_FAILURE pour signaler une erreur

Par ailleurs, de nombreuses fonctions des bibliothèques retournent des valeurs permettant de détecter des erreurs :

- `printf` retourne le nombre de caractères réellement affichés
- `scanf` retourne le nombre d'items réellement saisis.

Exemple

```c
#include <stdio.h>
#include <stdlib.h>

_Bool divisible(int a, int b) {
    if (b==0) {
        printf("Erreur divisible: second parametre a 0\n");
        exit(EXIT_FAILURE);
    }
    return (a%b == 0);
}

int main(void)
{
    int x, y;
    printf("Entrez deux nombres : ");
    if (scanf("%d%d", &x, &y) != 2) {
        printf("Erreur de saisie !\n");
        return EXIT_FAILURE;                    // ou      exit(EXIT_FAILURE)
    }
    if(divisible(x,y)) printf("%d est divisible par %d\n", x, y);
    else printf("%d n'est pas divisible par %d\n", x, y);
    return EXIT_SUCCESS;                        // ou      exit(EXIT_SUCCESS)
}
```

Paramètres du `main`

On peut passer des paramètres à un programme :

Définition

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

- `argc` est appelé le **compteur d'arguments** : il contient le nombre de mots de la ligne de commande (le séparateur de mots étant l'espace)
  - les arguments s'arrêtent au caractère `<Entrée>` ou au premier caractère séparateur de commande complexe : ; & | < > ...
    ↪ *Pour intégrer un espace ou un caractère spécial à un argument, on encadre l'argument avec des ' ou des " ou on dé-spécialise avec *
- `argv` (pour valeurs d'arguments) est un **tableau de `argc` string** :
  - `argv[0]` contient le nom de la commande, c'est-à-dire la référence au programme exécutable

Exemple : `ls -l "mon rep" > res` demande l'exécution d'un programme avec 3 arguments :

`argc=3` `argv[0]="ls"` `argv[1]="-l"` `argv[2]="mon rep"`

Exemple : affichage des paramètres

```c
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char* argv[])
{
    printf("Ce programme a %d arguments : ",argc);
    for(int i=0;i<argc;i++) {
        printf("argv[%d]=\"%s\" ",i,argv[i]);
    }
    printf("\n");
    exit(EXIT_SUCCESS);
}
```

Exemple

Cet appel `./prog para1 12 "mon 3eme para" 2\<3 < fic` produit :

Exemple : affichage des paramètres

```c
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char* argv[])
{
    printf("Ce programme a %d arguments : ",argc);
    for(int i=0;i<argc;i++) {
        printf("argv[%d]=\"%s\" ",i,argv[i]);
    }
    printf("\n");
    exit(EXIT_SUCCESS);
}
```

Exemple

Cet appel `./prog para1 12 "mon 3eme para" 2\<3 < fic` produit :

```
Ce programme a 5 arguments :
```

Exemple : affichage des paramètres

```c
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char* argv[])
{
    printf("Ce programme a %d arguments : ",argc);
    for(int i=0;i<argc;i++) {
        printf("argv[%d]=\"%s\" ",i,argv[i]);
    }
    printf("\n");
    exit(EXIT_SUCCESS);
}
```

Exemple

Cet appel `./prog para1 12 "mon 3eme para" 2\<3 < fic` produit :

```
Ce programme a 5 arguments : argv[0]="./testMain" argv[1]="para1"
argv[2]="12" argv[3]="mon 3eme para" argv[4]="2<3"
```

Plan

Plan