Hello World

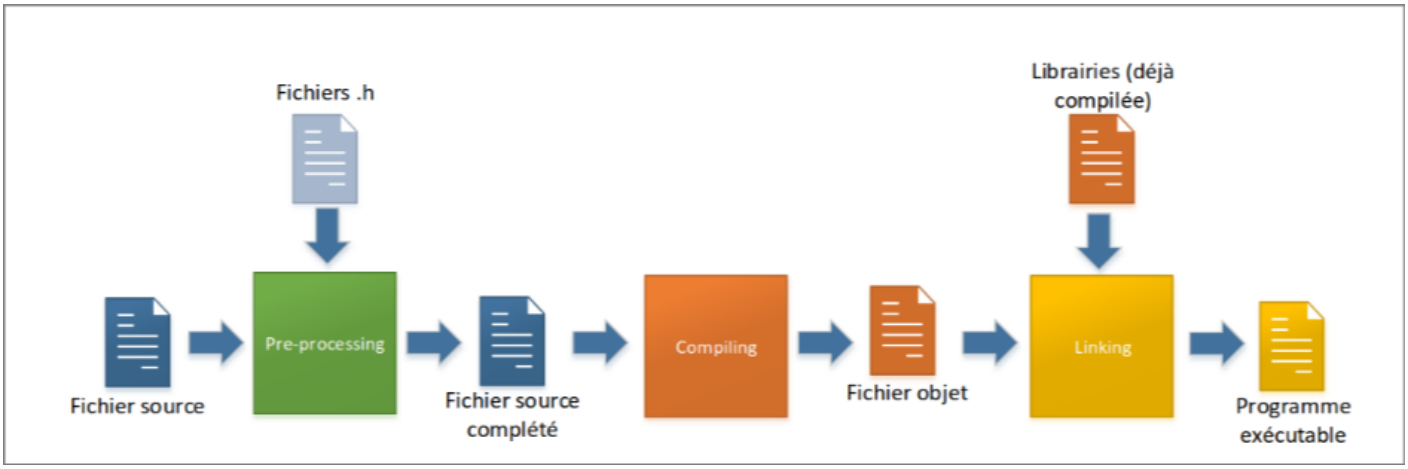
```
/* Les lignes commençant par # sont des directives au préprocesseur C
   Dans ce cas avec #include c'est une sorte d'import qui dit qu'il fait inclure une
librairie. Dans ce cas on importe les librairies standard stdio et stdlib */
#include<stdio.h>
#include<stdlib.h>

/* Le programme principale exécuté se trouve dans la fonction main */
int main(void) {
    /* Printf vient de la librarie stdio et permet d'afficher du texte dans la console */
    printf("Hello World !\n");

    /* A la fin de tous les programmes en C, il retourne un entier. 0 dans le cas d'un succès
(qui est déjà présent dans la constante de stdlib EXIT_SUCCESS); ou 1 dans le cas d'un échec
(constante EXIT_FAILURE de stdlib).
    Cela permet d'indiquer à des programmes qui utiliserait celui-ci, si l'exécution s'est
bien passée ou non */
    return EXIT_SUCCESS;
}
```

Libraries standard en C

Le langage C définit un certain nombre de librairies standard. Parmis celles ci, en voici 5 qui seront beaucoup utilisée dans ce cours d’introduction au C :

Librarie	Usage
stdio.h	Nécessaire pour les entrées sorties standard (gestion clavier et écran). **A inclure dans tous les programmes**
stdlib.h	Reprends les constantes et les fonctions importantes. **A inclure dans tous les programmes également**
string.h	Reprends les fonctions de manipulation de chaine de caractères (comparaison, copie, recherche, concaténation, etc)
math.h	Reprends les fonctions mathématiques (puissances, trigonométrie, etc)
time.h	Manipulation de la date et de l’heure

Processus de compilation



- Tout commence avec les fichiers source, c'est à dire les fichiers d'extension `.c`.
- Ensuite la phase de **pre-processing** va inclure les fichiers en-tête (sur lesquels on va revenir plus tard, mais ceux-ci contiennent les signatures des fonctions des libraries à importer). De cet étape de pre-processing, va résulter un fichier contenant tout le code source du projet + le contenu des fichiers en-têtes.
- Une fois la pre-processing finie, la **compilation** du fichier commence, ce qui résulte avec un fichier objet `.o`.
- Ensuite la partie suivante est le **linking** qui va lier les librairies au fichier `.o` pour donner le fichier exécutable final.

Déconstruire le processus de compilation en ligne de commande

Si vous voulez essayer (de le faire manuellement) par vous même, vous pouvez faire les commandes suivantes :

- Fichier .c : créer simplement un hello world comme montré plus haut dans un fichier `hello.c`
- Précompilation : `gcc -E hello.c`
- Compilation : `gcc -c hello.c`
- Linking : `gcc hello.o -o hello`

Compilation manuelle et console

Si vous voulez compiler le code par ligne de commande vous n'avez pas besoin de taper plein de commandes. Il suffit juste de faire `gcc \*.c` cependant il faut faire attention à plusieurs choses :

- Toujours inclure tous les fichiers .c dans la commande gcc
- Faire attention à la version de gcc

- Ne pas oublier d’ajouter les flags de compilations (qui donne des instructions suplémentaires au compilateur) à la commande gcc pour les projets de l’école

Notions fondamentales

```
/* Tout d'abord on doit inclure les librairies stdio et stdlib dans tous les projets. On a déjà parlé de ce que fait le #include dans le bloc de code Hello World */
#include<stdio.h>
#include<stdlib.h>

/* On donne les signatures des méthodes présentes dans le fichier dès le début car le compilateur C va lire le fichier de haut en bas et doit pouvoir directement savoir quelles fonctions existent dans le fichier */
float ajoute(float, float);
float soustrait(float, float);
float multiplie(float, float);
float divise(float, float);

/* La fonction main est le programme principale, ce qui va être exécuté lorsque l'on lance l'exécutable compilé du code */
int main(void) {
    /* Dès le début de la fonction on est obligé de déclarer nos variables */
    float n1, n2, resultat;
    char operation;

    /* Printf vient de stdio et permet d'afficher du code dans la console. Le caractère \n sert à retourner à la ligne */
    printf("Calculatrice simple\n");
    printf("Entrez l'opération à réaliser :");

    /* Scanf permet de récupérer un input d'un utilisateur dans la console. %f définissant un nombre flotant, %c un caractère et %*c servant à éliminer le denier caractère (le \n, soit le retour à la ligne) */
    /* Ces 3 valeurs (2 nombre flotatnts et un caractère) seront donc stoqué dans 3 variables (on passe donc les ADDRESSES de n1, opération et n2 en préfixant les variables d'un &) */
    scanf("%f %c %f%*c", &n1, &operation, &n2);

    /* Le  switch en C ne fonctionne qu'avec des valeurs entières. Par exemple ici '+' correspond à la valeur entière 43 dans la table ASCII. */
```

```
    switch(operation) {
        case '+':
            resultat = ajoute(n1, n2);
            break;
        case '-':
            resultat = soustrait(n1, n2);
            break;
        case '*':
            resultat = multiplie(n1, n2);
            break;
        case '/':
            resultat = divise(n1, n2);
            break;
    }

    /* Le printf ici fonctionne avec le même type de syntaxe que le scanf vu plus tot */
    printf("==> %f %c %f = %f\n", n1, operation, n2, resultat);

    /*  Enfin on retourne l'exit code du programme, ici un succès */
    return EXIT_SUCCESS;
}

/* Les méthodes annoncées dans l'en-tête plus haut sont définie ici */
float ajoute(float nombre1, float nombre2) {
    return nombre1 + nombre2;
}

float soustrait(float nombre1, float nombre2) {
    return nombre1 - nombre2;
}

float multiplie(float nombre1, float nombre2) {
    return nombre1 * nombre2;
}

float divise(float nombre1, float nombre2) {
    return nombre1 / nombre2;
}
```

Types en C

Les types de C sont très basiques contrairement à ceux d'autres langages (de plus haut-niveau) tel que Java.

Type	Explication	Codé sur	Représentation dans printf/scanf	Valeurs admissibles
`char`	Destiné à contenir un seul caractère. Il y a une conversion automatique char en type entier, ainsi 'c' en char deviendra 99 (sa valeur ASCII) en entier	8 bits	`%c`	Tous les caractères codés sur 8 bits
`short`	Destiné à contenir des valeurs entières petites	16 bits	`%hi`	De -2^{15} à $+2^{15} - 1$
`int`	Destiné à contenir des valeurs entières	32 bits	`%i` ou `%d`	De -2^{31} à $+2^{31} - 1$
`unsigned int`	Destiné à contenir des valeurs entières non signées (strictement positives)	32 bits (mais 16 bits minimum)	`%u`	De 0 à $+2^{32} - 1$
`long int`	Destiné à contenir de grandes valeurs entières (cependant sous Unix, il est la même que `int`)	32 bits minimum	`%li`	De -2^{31} à $+2^{31} - 1$
`long long int`	Destiné à contenir des plus grandes valeurs entières	64 bits	`%lli`	De -2^{63} à $+2^{63} - 1$
`float`	Destiné à contenir des valeurs avec fraction décimale (précision simple)	32 bits	`%f`	
`double`	Destiné à contenir des valeurs avec fraction décimale (plus précis)	64 bits	`%lf`	

C **ne dispose pas de type booléen**, cependant la valeur entière `0` est toujours considérée comme FAUX et tout autre valeur est considérée comme VRAI.

Plus de représentation printf et scanf

Caractères spéciaux

Symbole	Signification
`\n`	Caractère de controle `LF` qui fait un retour à la ligne sous Linux
`\r`	Caractère de controle `CR`. `\r\n` provoque un retour à la ligne sous Windows
`\t`	Tabluatino vers la droite
`\\`	Caractère `\`
`%%`	Caractère `%`

Autres types non élémentaires

Symbole	Signification
`%s`	Chaine de caractère
`x`	Donnée `unsigned int` au format hexadécimal

Précision de l'affichage

Symbole	Signification	Valeur	Affichage
`%3d`	Donnée formattée sur 3 chiffres, les absences de chiffres sont remplacées par des espaces	`9`	`  9`
`%03d`	Même chose mais les espaces sont remplacés par des 0	`9`	`009`
`%.2f`	Permet de préciser le nombre de chiffres derrnière la virgule d'un valeur fractionnelle	`9.191`	`9.19`

Fonctions et protoypes

Les signatures des fonctions comme mises au début du fichier de la calculatrice sont appellé des prototypes ou des signatures de fonction. Elles annoncent les fonctions qui vont être présentes. Sauf qu'en vérité, ces signatures sont dans des fichiers séparés appellée *en-têtes* dans des fichiers `.h`. Ces fichiers sont ensuite inclus dans le programme en utilisant `#include "file.h"`.

Lorsque l'on inclu un code on inclu toujours le fichier en-tête et jamais le fichier .c. A noter que si on veut importer un fichier en-tête bien précis in peut specifier le chemin d'accès entre guillemets (exemple `#include "file.h"`) mais lorsque l'on veut ajouter une librarie standard, on va la mettre entre chevrons (exemple `#include <stdio.h>`)

Lorsque l’on a plusieurs fichiers dans un projet C, il est important de bien garder la règle d’un seul dossier par projet, sinon ça risque fort de foutre la merde. Lorsuqe

---

Revision #3
Created 20 September 2023 06:25:45 by SnowCode
Updated 6 January 2024 18:14:47 by SnowCode