Le langage C est très utilisé depuis des décennies dans différents domaines du développement logiciel. Il s'agit d'un langage généraliste élaboré au début des années 1970 par Ken Thomson et Dennis Ritchie afin de répondre aux exigences d'écriture du premier système d'exploitation Unix. Le langage C compte parmi les langages plus utilisés aujourd'hui. Il comporte des instructions et des structures de données de haut niveau tout en permettant des opérations de bas niveau notamment sur la mémoire et les ressources matérielles. Le langage C présente l'avantage d'être portable et il très performant du point de vue de son exécution et de son empreinte mémoire. Sa syntaxe est standardisée et répond à une norme ANSI[1].

Écriture du code source

Au-delà de la phase de conception, la première étape du développement d'un programme consiste à écrire le code source destiné à répondre aux besoins du programme à générer. Ce code source doit rigoureusement respecter la syntaxe du langage C mais aussi, le plus souvent, des conventions de codage destinées à apporter une meilleure lisibilité au code. Il s'agit alors de se conformer à un format prédéfini et structuré pour l'organisation des lignes de code écrites. Le code source correspond à du texte écrit à l'aide d'un éditeur classique ou proposant une coloration syntaxique correspondant au langage C.

Des commentaires sont également à insérer dans le code source afin d'améliorer la compréhension lors de l'analyse ou de la relecture du code. Il est en effet important que le code source soit correctement commenté notamment lorsqu'un certain niveau de complexité des traitements codés est atteint. En langage C et selon la norme ANSI, les commentaires doivent être insérés entre les marques `/*` et `*/`. Ils n'ont strictement aucune influence sur l'exécution du programme.

Le code ci-dessous montre un exemple simplifié de programmation en langage C :

```c
#include <stdio.h>

/* Fonction principale */
int main(void)
{
  int nb = 4;

  printf("Valeur : %d\n", nb);
  nb++;  /* Incrémentation */
  printf("Valeur : %d\n", nb);

  return 0;
}
```

Le traitement effectué par le code présenté en exemple se limite à l'affichage de la valeur initiale d'une variable entière et de sa valeur après une opération d'incrémentation.

Selon la syntaxe du langage C, chaque instruction se termine par un point-virgule.

[1] American National Standards Institute

Il est important d'utiliser l'**indentation** pour améliorer la lisibilité et la forme du code. Il s'agit du décalage des lignes de code vers la droite notamment au sein d'un bloc de code placé entre les accolades `{`et `}`. Dans l'exemple présenté ci-dessus, les accolades délimitent le code associé à la fonction principale `*main()*`.

Les fichiers relatifs au code source d'un programme écrit en langage C portent toujours l'extension `.c` ou `.h`. Les traitements sont écrits dans un ou plusieurs fichiers `.c` alors que l'extension `.h` est réservée aux fichiers d'entête décrivant notamment le prototype de certaines fonctions utilisés dans le reste du code. Par exemple, le fichier d'entête `*stdio.h*` contient la description et le format de la fonction `*printf()*` appelée dans la fonction principale `main()`.

Génération d'un fichier binaire exécutable

Une fois que le code source est écrit et mis en forme, il doit être traduit en langage machine afin de pouvoir être exécuté par la machine pour lequel il a été écrit. Cette opération est appelée **compilation**. Le résultat de la compilation est un fichier au format binaire. Il est ainsi constitué d'une succession de `0` et de `1` que le processeur de la machine est en mesure de traiter. Il est nécessaire de disposer d'un logiciel spécifique afin de réaliser la compilation et donc la transformation du code source en code binaire.

Ce logiciel est désigné par le terme **compilateur**. L'un des compilateurs le plus utilisés pour traiter du code écrit en langage C est GCC[2]. Il est gratuit et très largement mis en œuvre pour le développement des logiciels libres.

L'appel au compilateur GCC se fait au moyen de la commande `***gcc***`. Cette commande est lancée avec un certain nombre d'arguments et d'options en fonction des besoins de la compilation. Le nom du fichier source à compiler est l'argument à passer au minimum à la ligne de commande pour l'appel du compilateur. Il est aussi conseillé d'activer les options d'affichage des avertissements liés à la correction du code source écrit. Par ailleurs, et sans précision du nom à donner au binaire généré, celui-ci est par défaut nommé `*a.out*`. Il est cependant possible d'utiliser l'option `-o` afin d'indiquer le nom à donner au fichier binaire à générer.

La ligne de commande présentée ci-dessous permet la compilation d'un code source contenu dans un unique fichier `*exemple.c*` et la génération d'un fichier binaire `*binExemple*` :

```
gcc -Wall exemple.c -o binExemple
```

L'option **`-Wall`** active l'affichage des messages d'avertissement. Elle est très utile lorsqu'il s'agit de produire un programme performant, de le tester et de s'assurer de sa bonne exécution. Elle permet notamment de mettre en évidence des variables déclarées mais qui ne sont jamais utilisées et qui occupent ainsi inutilement de la mémoire.

Préalablement à l'opération de compilation proprement dite, l'exécution de la commande `*gcc*` effectue automatiquement un appel à un préprocesseur qui assure des étapes préparatoires à la compilation. Il s'agit notamment de traiter les inclusions de fichiers d'entête réalisées par la directive `#include` en recopiant le contenu du fichier passé en paramètre dans le fichier courant. Il peut également s'agir de directives spécifiques destinées à gérer une compilation conditionnelle mais aussi à définir des macros ou des constantes au moyen de la directive `*#define*`.

Par exemple, la définition de la constante PI peut se faire au moyen de la ligne suivante :

```
#define PI 3.14159
```

Concernant les macros, le code ci-dessous montre un exemple de définition d'une macro :

[2] GNU C Compiler* à l'origine puis *GNU Compiler Collection* suite au support progressif d'autres langages de programmation

```
#define BONJOUR() printf("Bonjour\n")
```

Par convention, les constantes et les macros sont exprimées en lettres majuscules. Selon les exemples présentés ci-dessus, le préprocesseur remplacera systématiquement toutes les occurences de `PI` et de `BONJOUR()` rencontrées dans le code respectivement par `3.14159` et `printf("Bonjour\n")` avant l'exécution de la compilation.

Le code source ci-dessous présente un exemple de définition et d'utilisation d'une macro et d'une constante :

```
#include <stdio.h>
/* Définition de macro   et de constante     */
#define MSG() printf("Début...\n")
#define NB 9

/* Fonction principale */
int main(void)
{
  MSG();
  printf("Valeur : %d\n", NB);

  return 0;
}
```

Exécution du programme

Dès lors que la compilation du code s'est correctement déroulée et que le fichier binaire a été généré, l'exécution du programme se limite à une simple commande :

```
./binExemple
```

L'exécution du programme se lance donc en ligne de commande à partir d'un émulateur de terminal.

Méthode utilisée durant les TP de programmations

Bien que gcc existe sous Windows, notamment grâce à MinGW[3], celui-ci n'est pas installé sur les PC Windows du lycée, en vrai informaticiens nous coderons donc sous Linux ! Plusieurs solutions s'offrent alors à nous pour compiler les premiers programmes :

1. **Méthode 1 : Coder et compiler** sur une machine virtuelle linux disposant des outils de compilation

    a. Sur votre machine perso qui boote sous Linux
    b. Sur la Debian CIEL ou une autre machine virtuelle sous Linux

    Avantages :

    - Pas de transfert de fichier à effectuer

    Inconvénients :

    - La DEBIAN_CIEL en mode interface graphique peut être lente et instable sur les PC du lycée.
    - Si vous codez sur un Linux sans Interface Graphique, vous ne disposez pas d'un éditeur de texte avec fonctionnalité avancées

2. **Méthode 2 (recommandée) : Coder sous Windows (sous vscode par exemple) et compiler sous Linux** en transférant les fichiers .c à l'aide de WinSCP.

    a. Sur la debian_ciel ou une autre machine virtuelle Linux lancée en mode Headless (voir procédure ici)
    b. Sur le serveur de la section

    Avantages :

    - Vous pouvez coder sous VSCode qui offre de bonnes fonctionnalités (coloration syntaxique, indentation, soulignage des erreurs…)
    - Vos fichiers sources sont conservés sur votre session.
    - Cela vous permet d'utiliser les protocoles SFTP (avec WinSCP) et SSH (Avec Putty) et de gagner en aisance avec ces logiciels.

    Inconvénients :

    - Vous devez transférer les fichiers sources à chaque modification

Si gcc n'est pas installé sur linux, il suffit de taper :

```
sudo apt update
sudo apt install build-essential
```

[3] Minimalist GNU for Windows: https://sourceforge.net/projects/mingw/

Démarrer une machine Virtualbox en mode « Headless »

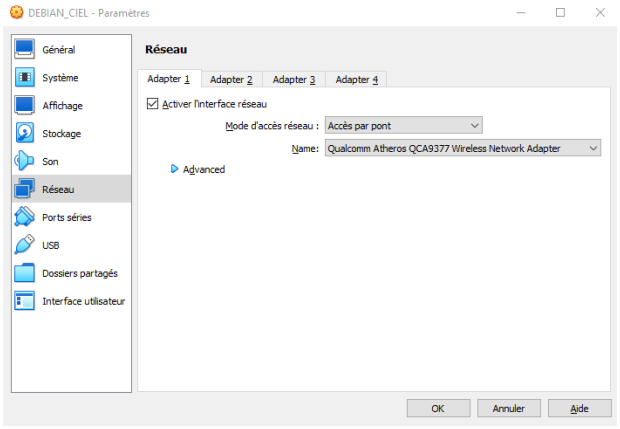
Pour qu'une machine virtuelle soit moins gourmande en ressource sur son PC Hôte, il est possible de la démarrer en mode « Headless » ou «Sans affichage » en français.

La machine continuera alors son exécution mais sans générer d'interface graphique.

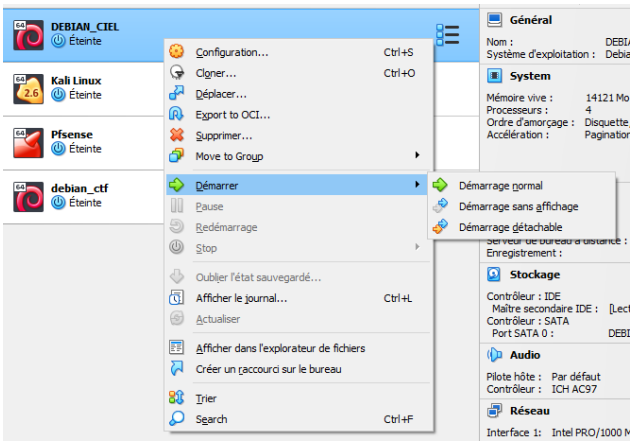
- Si le service ssh est activé sur la machine, il restera disponible
- Si un service Web ou autre tourne sur la machine il sera également disponible.

Configurer la carte réseau de la machine

Pour qu'il soit possible d'accéder à votre machine via le SSH, il est préférable que l'interface réseau de celle-ci soit configurée en « Accès par pont ». Votre machine virtuelle obtiendra alors une adresse IP dans le même réseau que votre machine hôte.

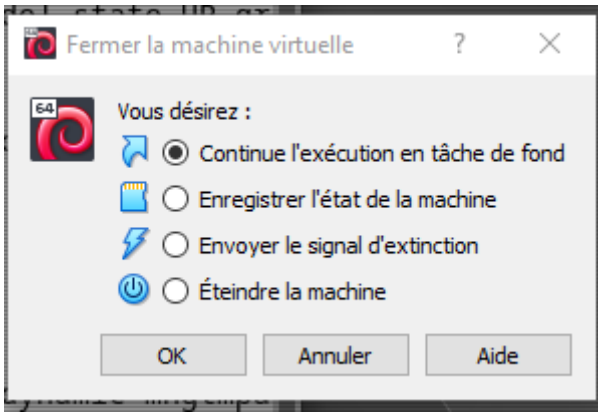


Procédure de démarrage



1. Dans l'interface VirtualBox cliquer sur « Démarrer-> Démarrage détachable »
2. Attendre que votre machine démarre
3. Ouvrir une session
4. Récupérer son adresse IP et la noter (avec ifconfig ou ip a)

5. Cliquer sur la « x » pour fermer la machine et sélectionner « Continuer l'exécution en tâche de fond » :



6. Ouvir vos sessions ssh (putty) et sftp (winscp) à l'aide de l'adresse IP récupérée

Exercices :

Exercice 1 :

Coder, compiler et exécuter le programme fourni en exemple dans la partie : « Ecriture du code source »

Donner comme nom d'exécutable « seq4Prog1 »

Exercice 2 :

En vous aidant du cours « saisie_clavier.pdf » réaliser un programme qui demande à l'utilisateur, son nom, son prénom et son année de naissance et restitue ensuite prénom, nom et âge :

```
ciel_user@debCIELVauban:~/progC$ ./seq4Prog2
Entrer votre nom:Guivarch
Entrer votre prénom:Sébastien
Entrer votre année de naissance au format yyyy:1985
Vous vous appelez Sébastien Guivarch et vous avez 39 ans
```