esiea
INGÉNIEUR·E·S D'UN NUMÉRIQUE UTILE

C-SYSTÈME

Fiche : Rappel compilation

Table des matières

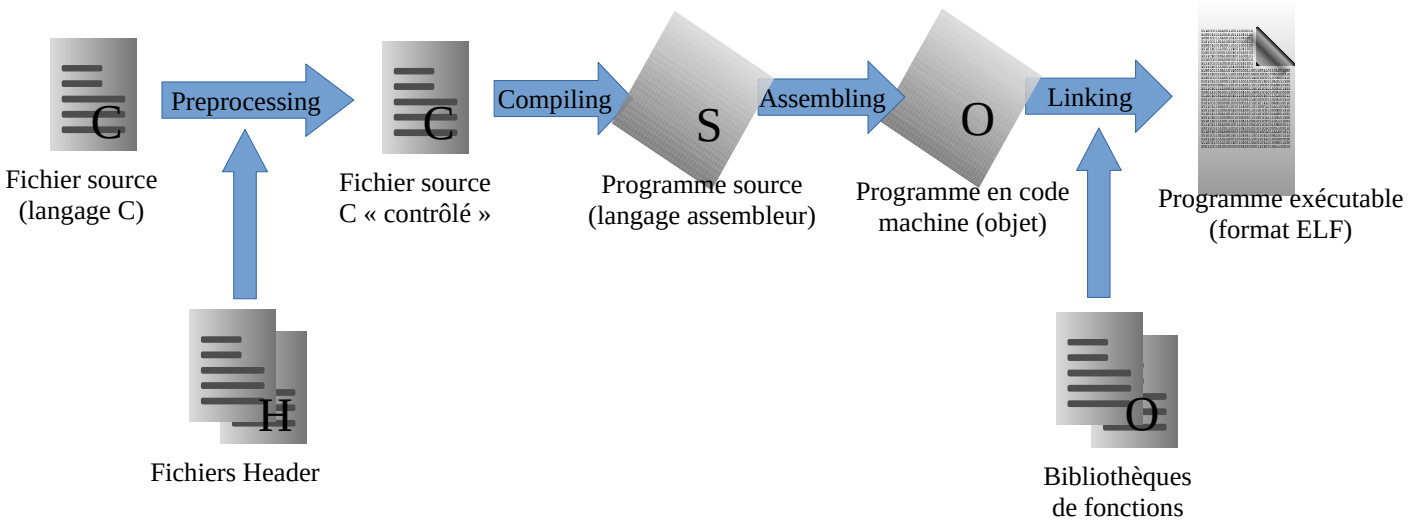
By Rexy

Les prérequis

Pour compiler un programme source afin d'obtenir un fichier binaire exécutable, votre système doit posséder un environnement de programmation fonctionnel. Cet environnement est constitué au minimum du compilateur, des bibliothèques standards, des fichiers d'en-tête (« Headers »), d'un analyseur de syntaxe, etc.
Certaines distributions Linux permettent d'installer rapidement l'ensemble de ces éléments via leur gestionnaire de paquetages. En effet les gestionnaires de paquetages proposent des « meta-paquetages » qui sont des listes de paquetages constituant un ensemble cohérents pour un objectif particulier (programmation C, mathématique, logiciels éducatifs, environnement graphique, etc.). Sous Linux-Mageia, ces « meta-paquetages » porte un nom commençant par « task- ». Le meta-paquetage permettant d'installer l'ensemble de l'environnement de développement C est nommé « task-c-devel ».

```
urpmi task-c-devel
```

Les étapes de la compilation



Exemple (welcome.c) :

```
#include <stdio.h>
int main (void)
{
/* display a welcome message */
        printf("Rexy is welcome you\n");
        return 0;
}
```

La génération de l'exécutable est réalisée par le compilateur GCC (GNU C Compiler) via la commande « gcc » (ou « cc »). Il est conseillé de définir le nom du fichier exécutable (welcome). Dans le cas contraire, ce fichier sera nommé « a.out ».

```
gcc -o welcome welcome.c
```

Pour raison de sécurité, sous Linux, ce fichier n'est pas généré avec les droits en exécution. Il faut donc les ajouter pour le lancer.

```
chmod  u+x welcome
```

GCC a effectué toutes les étapes de la compilation sans laisser de trace intermédiaire. Voici le détail du travail :

1. Le préprocesseur effectue les opérations suivantes sur le code source C :
   - supprime les commentaires (// et /* */) ;
   - contrôle la syntaxe du langage ;
   - traite les directives de compilation commençant par le caractère « # » (define, elif, else, endif, error, if, ifdef, ifndef, include, line, pragma, undef). Exemple : les « fichiers header (.h) », définissant le prototype des fonctions externes, sont intégrés au code source s'ils sont spécifiés par la directive « #include ».

   Pour créer un fichier issu du « préprocessing :

```
gcc -E welcome.c > welcome.i
```

   En analysant ce fichier, on distingue deux sections : « #1 welcome.c » contenant les fichiers « header » (.h) et « #2 welcome.c » contenant le source d'origine épuré de ses commentaires.
2. Le compilateur transforme ce fichier en code assembleur (appelé aussi « codes d'opérations » ou « opcode ») pour le µ-processeur de son choix (appelé aussi « architecture cible »). Ce µ-processeur est

par défaut celui de la machine exécutant « gcc ». L'option « -m » permet de changer d'architecture (man gcc). Le fichier généré prend automatiquement l'extension « .s ».

```
gcc -S welcome.i
gcc -dumpmachine (pour connaitre votre architecture --> définie par la variable d'environnement MACHTYPE)
```

3. À partir de ce fichier, l'assembleur va générer le code binaire d'instructions machine. S'il rencontre des appels à des fonctions de bibliothèques dynamiques externes, il va intégrer l'appel au chargeur de bibliothèques dynamiques (programme « ld-linux.so »). Dans notre cas, le seul appel est celui de la fonction externe (« printf ») qui fait partie de la bibliothèque standard du C (libc.so). Le fichier généré prend automatiquement l'extension « .o ».

```
gcc welcome.s
ldd welcome.o : liste les appels aux fonctions des bibliothèques dynamiques externes que le programme « welcome.o » aura besoin
```

4. L'éditeur de lien va maintenant lier les autres objets (.o) dans le cas d'un projet plus conséquent constitué de plusieurs fichiers (sources ou bibliothèques). Dans notre cas, rien de plus n'est à lier (le fichier « welcome » initial est le même que le fichier « welcome.o » généré avant l'édition de lien).

```
INFO : la commande « ldconfig -p » permet de lister les bibliothèques dynamiques disponibles pour le système. Par défaut, le système cherche dans les répertoires « /lib », « /usr/lib » et « /usr/local/lib ».
Le répertoire « /etc/ld.so.conf.d » contient des fichiers informant le chargeur de bibliothèque dynamique (ld-linux.so) de l'emplacement d'autres répertoires contenant des bibliothèques dynamiques. Si vous ajoutez un fichier dans ce répertoire, vous devez avertir le chargeur dynamique via la commande « ldconfig »
```