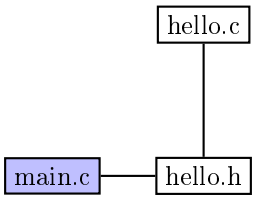
Guide de survie pour Makefile

Un Makefile c'est quoi ?

Les Makefiles sont des fichiers permettant d'executer un ensemble d'actions, comme la compilation d'un projet, la mise à jour d'un rapport, où l'archivage de données. Les Makefiles sont des fichiers shell particulier ils respectent cependant les conventions de codages du shell. L'idée de ce document est de présenté l'idée générale d'un Makefile via un exemple basique, puis d'aller vers un exemple plus complexe, le Makefile pour notre robot.

Un exemple basique

Considérons un exemple de projet très basique, constitué de 3 fichiers : hello.c, hello.h et main.c. Nous pouvons faire le graphe des dépendances suivants



De manière plus littérale nous avons définit des fonctions dans hello.c, ces fonctions sont referencées par leurs prototypes dans hello.h. Par ailleurs nous avons définies des fonctions dans main.c, nous sous entendons que les fonctions de main.c font appels à celles de hello.c (d'où l'inclusion de hello.h dans main.c).

Nous voulons donc compiler main.c pour créer un executable que nous appellerons *project*. Si nous essayons directement de faire "gcc -Wall -Werror -std=c99 main.c project" nous allons avoir des références indéfinies. En effet il faut d'abbord compiler hello.c en hello.o pour que la machine connaisse les fonctions définit dans le hello.c (c'est le rôle du hello.o). Ainsi pour créer project il faut faire deux lignes de commande "gcc -Wall -Werror -std=c99 hello.c -c" puis "gcc -Wall -Werror -std=c99 hello.o main.c -o project".

Deux nouvelles options se présentent "-c" qui permet de dire au compilateur de créer un fichier ".o" et "-o" qui permet d'appeler le linker (c'est lui qui dit au compilateur que certainnes fonctions appelées dans main.c sont définies dans hello.o).

Maintenant que nous avons compris comment nous pouvions compiler ce projet, nous allons regarder comment automatiser ces étapes. Pour cela regardons la synthaxe type d'un Makefile.

```
1    cible : dependances
2            commandes
```

Analysons ces deux lignes de commandes, le “cible” permet de donner un nom ainsi dans le shell nous pourrons taper “make cible” pour effectuer les actions correspondantes. Dans notre cas la cible serait “project”. Ensuite les “dépendances” sont les fichiers nécessaires à la création de la cible, dans notre cas cela serait “hello.o”. Un exemple de

```
1    cible : dependances
2            commandes
```