



FORMATION
EIRBOT

GNU Make
Mise en pratique

Sébastien DELPEUCH

21 Novembre 2020



Introduction

L'archive contenant tous les fichiers est disponible [ici](#)

L'objectif de cette mise en pratique est d'écrire un premier Makefile avancé permettant de compiler un projet typique de l'ENSEIRB-MATMECA. Attention l'objectif est de comprendre comment construire un Makefile copier le Makefile tout fait ne sert à rien.

1 Présentation de la structure

Dans un premier temps présentons la structure du projet :

```
Makefile  src  tst

./src :
auxiliaire.c  global.c  good.c  loop.c  queue.c  stockex.c
auxiliaire.h  global.h  good.h  loop.h  queue.h  stockex.h

./tst :
tst.c
```

À la racine du projet il y a le [Makefile](#) (pour l'instant vide), un dossier [src](#) avec les fichiers sources et un dossier [tst](#) avec un fichier de test. L'objectif est de faire deux exécutables, un exécutable [project](#) qui lancera le main contenu dans [loop.c](#) et un exécutable [test](#) qui lancera le main contenu dans [tst.c](#).

2 Création du Makefile

2.1 Questions d'introduction

1. Quels sont les fichiers qu'il faudra compiler pour l'exécutable `project` ?
2. Quels sont les fichiers qu'il faudra compiler pour l'exécutable `test` ?
3. Pourquoi ne doit t'on pas compiler `loop.c` pour l'exécutable `test` ?

2.2 Premier Makefile simple

1. Décommentez et complétez la règle `source` dans le Makefile, cette règle a pour but de compiler tous les fichiers sources (`auxiliaire.c` `global.c` `good.c` `queue.c` `stockex.c`) en des fichiers objets (`.o`) (indication : on utilisera l'option `-c` de `gcc`). Exécutez la commande `make source` dans un terminal, qu'est ce que cela produit ?
2. Décommentez et complétez la règle `project` pour compiler `loop.c` tout en liant avec les `.o`, faites en sorte que l'exécutable s'appelle « `project` » et non « `a.out` » (indication : on utilisera l'option `-o` de `gcc`). Exécutez la commande `make project` dans un terminal puis `./project`, le programme est-il correctement exécuté ?
3. Faites le nécessaire pour avoir un exécutable `test`. (l'exécution de `test` peut prendre un certain temps (100 000 fois le jeu, vous pouvez stopper avec `Ctrl-C`))

2.3 Optimisation

La règle `source` est pénible car lorsque nous rajoutons un fichier nous devons réécrire à chaque fois `gcc $(CFLAGS) -c src/<newfile>.c`. Nous allons vouloir factoriser ça en utilisant une règle générique utilisant des caractères spéciaux de Makefile, cette règle permet de compiler n'importe quel `.c` en `.o` :

```
%o: src/%c
    gcc $(CFLAGS) -c $<
```

Utilisez cette règle pour optimiser le Makefile.

- Attention, cette règle est spéciale et ne peut donc pas être utilisée en tant que dépendance, ie `project : %o` ne fonctionnera pas
- N'oubliez pas que c'est un fichier shell, il est donc possible de déclarer une variable contenant tous les noms des fichiers objets à obtenir par exemple