Rapport projet PCR

Benjamin Voisin, Romain De Beaucorps

14 octobre 2022

1 Introduction

L'objectif de ce projet est de construire un programme pour simplifier et optimiser la compilation de programmes, notamment lorsque ceux-ci comportent beaucoup de fichiers et de modules avec des inter-dépendances. Nous allons donc construire un programme `mymake` qui, grâce à la lecture d'un fichier `Makefile`, se contentera de compiler uniquement les programmes ayant été modifiés, ainsi que les programmes dépendant de ceux-ci, ce qui permettra de ne re-compiler que ce qui est nécessaire.

2 Réponses aux questions

Question 1 Pas trop de difficultés pour la création du `Makefile`, les dépendances des fichiers `a.o, b.o, c.o, d.o` sont simples. Pour la cible `main.o` il faut simplement inclure les headers `c.h` et `d.h`, et pour la cible main, il faut inclure tous les fichiers `.o` qu'on a construits. On a cependant perdu un peu de temps avant de comprendre que la commande `make` ne fonctionne pas avec un fichier intitulé `MakeFile` au lieu de `Makefile` (en revanche `makefile` fonctionne bien). La cible `clean` permet de supprimer les fichiers en `.o` si besoin. Voici le contenu du fichier `Makefile` :

```
main: a.o b.o c.o d.o main.o
	gcc main.o a.o b.o c.o d.o -o main

main.o: main.c c.h d.h
	gcc -c main.c

a.o: a.c a.h
	gcc -c a.c

b.o: b.c b.h
	gcc -c b.c

c.o: c.c c.h
	gcc -c c.c

d.o: d.c d.h
	gcc -c d.c

clean:
	rm *.o main
```

Question 2 lien vers notre code `regle.c`
patate

Question 3

Question 4 On construit ici le Makefile comme on a fait précédemment dans la question 1, et dans le cas de notre projet on obtient ce fichier :

```
mymake: main.c regle.o ens_regles.o lecture.o
    cc -Wall -Wextra -o mymake main.c regle.o ens_regles.o lecture.o

debug: main.c regledbg.o ens_reglesdbg.o lecture_dbg.o
    cc -Wall -Wextra -g -o debug/dbg_make -Og main.c debug/regledbg.o debug/ens_reglesdbg.o debug/lectu

regle.o: regle.c regle.h
    cc -c regle.c

regledbg.o: regle.c regle.h
    cc -g -c regle.c -o debug/regledbg.o

ens_regles.o: ens_regles.c ens_regles.h
    cc -c ens_regles.c

ens_reglesdbg.o: ens_regles.c ens_regles.h
    cc -g -c ens_regles.c -o debug/ens_reglesdbg.o

lecture.o: lecture.c lecture.h
    cc -c lecture.c

lecture_dbg.o: lecture.c lecture.h
    cc -g -c lecture.c -o debug/lecture_dbg.o

clean:
    rm *.o
```

Question 5 lien vers le code : `lecture.c`

L'objectif ici est de lire le fichier `Makefile` pour obtenir un ensemble de règles (appelé `ens`), que l'on pourra ensuite exécuter suivant leur dépendances. Pour l'initialiser, il nous faut d'abord connaître le nombres de règles. On fait donc un premier parcours du fichier (à l'aide de la fonction `nombre_regles`, qui compte le nombre de `:` dans le fichier). On suppose ici que le fichier `Makefile` est bien formé, et qu'il y a donc égalité entre le nombre de `:` et de règles.

Ensuite, on va ajouter les règles une à une dans l'ensemble de règles. Encore une fois, avant de créer notre règle avec ses prérequis et ses commandes, il nous faut connaître le nombre de prérequis, et de commandes. On re-parcourt ensuite la section du fichier définissant la règle, en ajoutant d'abord les prérequis, puis les commandes.

Question 6

Question 7

Question 8

3 Synthèse

Au final, ce projet fonctionne surprennament bien. Sur différents programmes de test, ainsi que sur le projet `mymake` lui même, tout fonctionne parfaitement bien sans qu'il n'y ait de problèmes apparents (à condition que le fichier `Makefile` soit correctement écrit). La performance de ce programme est difficile à évaluer, mais le programme semble fonctionner de manière similaire à la commande `make` en terme de temps d'exécution.

Il est cependant clair que certaines améliorations peuvent être apporté à notre programme. Nottament, lors de la lecture du fichier `Makefile`, on parcours au total 3 fois le fichier pour pouvoir construire notre ensemble de règle. Ce fonctionnement permet d'éviter l'utilisation de `realloc`, mais peut sembler ineffiace lorsque le fichier `Makefile` deviens vraiment grand.

Nous n'avons également pas eu le temps d'implémenter un système de gestion d'erreurs en cas d'un fichier `Makefile` incomplet ou mal-formé, ce qui serait une grande amélioration.

Un autre projet intéressant à regarder, en complément de celui-ci, serait de concevoir un programme permettant de construire le fameux fichier `Makefile`, ce qui garantirait une absence d'erreurs humaines dans la conception du fichier, et simplifierait aussi grandement la vie des programmeurs, car la conception de tels fichiers peut vite devenir fastidieuse.

4 Bibliographie

- Pour le fonctionnement de la fonction `getline`, nous avons utilisé les instructions du site `man7.org`, à cette addresse getline(3) — Linux manual page.
- Pour la lecture des fichiers du dossier (qu'on fait dans la fonction "appliquer_ens_regles"), on utilise le code trouvé à l'addresse : https://www.sanfoundry.com/c-program-list-files-directory/