UMEÅ UNIVERSITET
Institutionen för datavetenskap

October 9, 2022

5DV088

Systemnära programmering

mmake

Namn Vincent Johansson
CS-användare dv14vjn

Handledare
Abdulsalam Aldahir, Oscar Kamf, William Sandström

Innehållsförteckning

1 Kompilering och körning

För att kompilera programmet behöver alla nödvändiga filer ligga i en och samma map. Dessa filer inkluderar mmake.c, parser.c, parser.h samt Makefile.

Som exempel utgår vi ifrån att filerna ligger i /home/mmake.
Steg 1. Öppna ett terminal fönster.
Steg 2. Gå till mappen /mmake, detta görs med commandot 'cd mmake'.
Steg 3. Kontrollera att filerna finns i mappen, för att se vilka filer som finns i mappen används kommandot 'ls'. Om filerna finns i mappen, kommer en utskrift komma på ny rad 'Makefile mmake.c parser.c parser.h'.
Steg 4. Skriv 'make' i terminalen. Om alla filer finns och allt fungerar, kommer utskriften se ut som nedan fast med alla flaggor till gcc som angivits i Makefile.

'gcc -c -o mmake.o mmake.c -g -std=gnu11 ...'
'gcc -o mmake mmake.o parser.o -g -std=gnu11 ...'

Programmet kan nu köras genom att skriva './mmake' i terminalen, givet att användaren är i rätt map med terminalen (/home/mmake i detta exemplet). Programmet imiterar kommandot make, fast i en mycket enklare form. Programmet skriver ut de kommandon det utför, och skriver ut error meddelande om något går fel. Några exempel nedan.

touch A.test

touch D.test
touch B.test
touch ABC.test

mmake: No rule to make target 'nosuchfile.txt'

mmakefile: Could not parse makefile

2 Användarhandledning

Programmet är en egen implementation av make kommandot i bash och synopsis för programmet är följande:

mmake [-f MAKEFILE] [-B] [-s] [TARGET...]

För att köra programmet skrivs kommadot *'./mmake'* i den katalog där programmet är sparat. Om användaren vill köra programmet med specifika argument skrivs de efter mmake, exempelvis *'./mmake -f exmake -B -s extarget'*. I exemplet kommer makefilen 'exmake' användas och enbart 'extarget' kommer att byggas. Programmet tvingas till ombyggnad av 'extarget'genom flaggan -B och ingen utskrift kommer att ges eftersom -s flaggan används. Mer detaljerad information om hur varje flagga används går att läsa i sektion 2.1.

2.1 Definition av flaggor

Nedan följer detaljerad definition av alla flaggor som kan användas.

2.1.1 -f MAKEFILE

Flaggan används för att specificera namn på vilken makefil som ska användas av programmet. Om ingen fil specificeras används makefil med namn 'mmakefile' i den aktiva katalogen.

Makefilen måste bestå av en eller flera regler och ska ha följande format:

```
target : [PREREQUISITE...]
        program [ARGUMENT...]
```

En regel består alltid av två rader. Den övre raden får inte börja med blanktecken och först anges namn på den fil som ska skapas även kallat target. Efter target och ett kolon anges prerequisites vilket är filer som behövs för att kunna skapa target filen. Den andra raden ska alltid börja med ett *tab* tecken följt av vilket program som ska användas för att kompilera target filen och vilka argument programmet ska köras med. Nedan följer ett exempel på hur en makefil skulle kunna se ut.

```
mexec: mexec.o parser.o
	gcc -o mexec mexec.o parser.o

mexec.o: mexec.c
	gcc -c mexec.c

parser.o: parser.c
	gcc -c parser.c
```

2.1.2 -B

Denna flaggan används för att tvinga programmet att bygga om alla filer. När flaggan inte används kommer programmet att kontrollera om de target filer som det finns regler för i makefilen redan finns i katalogen och om de finns kontrollera om prerequisite filerna har modifierats efter att target filen skapades.

2.1.3 -s

När programmet körs kommer information skrivas ut till standard output för varje kommando som körs, men om flaggan -s används hindras utskrift till standard output. Kan användas när användaren inte har något behov av att få informationen.

2.1.4 TARGET

Om användaren bara vill bygga om ett eller flera specifika targets kan dessa anges som argument till programmet, om inga targets anges som argument kommer programmet att utgå ifrån reglerna i makefilen.

3 Diskussion och reflektion

Försökte att utgå ifrån den feedback jag fick på föregående uppgift där jag blev ombedd att faktorisera min kod och undvika att göra logiska operationer i main funktionen av programmet. Funktionerna i programmet är utifrån det utformade att utföra en specifik uppgift i programmet, exempelvis en funktion för att initialisera den struct som kommer användas, en funktion för att kontrollera när filer modifierads och en funktion som hanterar säker allokering av minne.

Har också lagt ner mer tid på att skriva min rapport och försöka strukturera den på ett sätt som gör den lättläst och enkel att förstå. Bad en bekant testa programmet med hjälp av den användarhandledning jag skrivit till rapporten och personen i fråga hade inga problem att förstå hur programmet användes tackvare detaljerad beskrivning av hur olika flaggor användes.

Kanske har lagt ner lite mer tid på uppgiften än vad som hade behövts vilket lett till att jag inte kommit igång med nästkommande uppgift men jag hoppas att jag inte kommer ha några större problem och inte känner för stor tidspress när jag väl sätter igång.