OU2 - make

2022-09-30

Theodor Jonsson
ens18trn@cs.umu.se

Systemnära programmering
5DV088

1. Kompilering och körning

För att kompilera programmet skriv kommandot i Figur 1 eller skriv “make” om rätt Makefile finns i directory. Makefile som har använts under kodning finns i Figur 3.

```
itchy:~/Documents$ gcc -std=gnu11 -Wall -o mmake mmake.c parser.c
```

Figur 1. Exempel kompilering utan make

För att köra programmet skriv kommandot som finns i Figur 2 detta kommer köra programmet med filen som heter “mmakefile” i ditt directory samt ta default target i denna fil.

```
itchy:~/Documents$ ./mmake
```

Figur 2. Kommando för körning av mmake

```
CC = gcc
CFLAGS = -g -std=gnu11 -Wall -Wextra -Wpedantic -Wmissing-declarations -Wmissing-prototypes -Wold-style-definition -Werror

mmake: mmake.o parser.o
	$(CC) -o mmake mmake.o parser.o $(CFLAGS)
mmake.o: mmake.c mmake.h
	$(CC) -c mmake.c $(CFLAGS)
parser: parser.o
	$(CC) -o parser parser.o $(CFLAGS)
parser.o: parser.c parser.h
	$(CC) -c parser.c $(CLAGS)
```

Figur 3. Makefile som har används

2. Användarhandledning

Detta program ska fungera som en mer begränsad version av UNIX-kommandot make. Den ska bygga filer beroende på vissa omständigheter. Programmet kollar på Target filen och kollar om den ska byggas eller byggas om. Dessa omständigheter är:

1. Target-filen existerar ej.
2. Tidpunkten för ändring av en prerequisite-file är senare än för Target-filen.
3. Flaggan -B används.

Programmet kommer också bygga alla prerequisite filer till ett Target om de finns regler för dessa filer detta kommer ske rekursivt. Programmet kommer till sist skriva ut vilket eller vilka kommandon som har blivit exekverade.

Den mest grundläggande användning av mmake är att köra utan några argument. Detta kommer köra med en fil mmakefile vilket är default filen som programmet väljer om ingen annan fil specificeras av användaren se Figur 5 för körning samt Figur 6 för hur denna fil kan se ut.

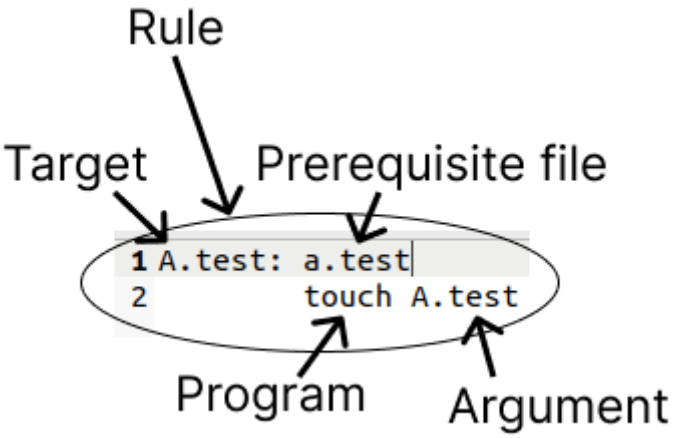
```
itchy:~/Documents$ ./mmake
touch A.test
```

Figur 5. Exempel körning med mmakefile från Figur 6

```
1 A.test: a.test
2         touch A.test
```

Figur 6. Exempel på mmakefile

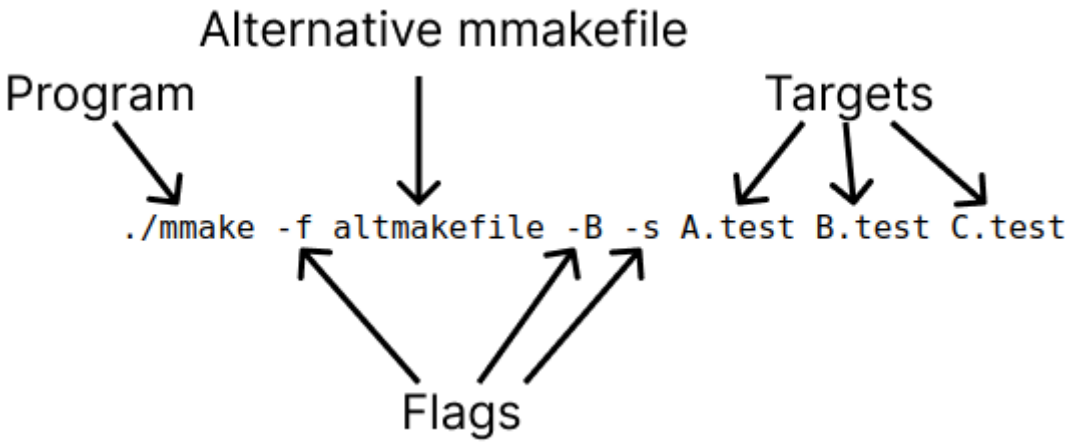
Om en fil ska användas måste den vara i ett specifikt format där en regel börjar med ett Target vilket är namnet på filen som ska byggas sen ett kolon sedan en lista av prerequisite filer dessa ska delas med ett mellanslag. På andra raden ska börja med en tab sen programmet som ska exekveras samt dess argument se Figur 7 för visualisering.



Figur 7. Visualisering av mmakefile format

Programmet stödjer tre olika flaggor. Flaggan -f visar att en annan fil än mmakefile ska användas denna fil ska specifieras i argument direkt efter -f. Flaggan -B tvingar ombyggnation av target filen ändå fast inga filer har blivit uppdaterade. Flaggan -s används för göra så programmet inte skriver ut de kommandon som har exekverats till standard output.

Programmet kan ta in en lista av targets som argument dessa argument ska bestämma vilka filer som ska byggas om inga targets kommer in från argument väljs default target som är filen högst upp i filen. Se Figur 8 för på flaggor och exempel på target argument



Figur 8. Visualisering av olika flaggor och targets

3. Diskussion

Största problemet jag hade med denna laboration var att förstå mig på specifikationen men de är nästan bara mitt eget fel att jag inte läste den ordentligt från första början. Sen att försöka pussla ihop hur den rekursiva delen skulle fungera var också ganska krångligt.

Tyckte den här laborationen var väldigt lärosam för att före detta hade jag inte riktigt en aning hur make funkade och vad som var vad i makefilen men detta har gjort att jag förstår mig på make mycket bättre.