Linköpings Universitet
Institutionen för datavetenskap (IDA)
UPP-gruppen 2016-09-28

Makefile

`make` är ett verktyg för att hantera kompilering av många källkodsfiler. Man har en fil i mappen som heter `Makefile` som `make` kommer tolka för att välja vilka filer som ska kompileras.

Exempel 1

```
main: lab.cc
    g++ lab.cc
```

I exemplet har vi

- main: som är en regel
- lab.cc: som är ett beroende
- `g++ lab.cc` kommand som ska utföras

Det som händer när ni skriver `make` i terminalen kommer `g++ lab.cc` att köras. Man kan också skriva `make main` för att utföra samma sak.

OBS!!! Det är viktigt att ha en tabb framför ett kommand och filen måste ha en tom rad annars kommer make inte att köras korrekt **OBS!!!**

Exempel 2

```
main: lab.cc
    g++ lab.cc

test: lab_test.cc
    g++ lab_test.cc
```

Det som är nytt i exempel två är att man har infört ytterligare en ny regel.

Ifall du skriver `make main` kommer `lab.cc` att kompileras och ifall du skriver `make test` kommer lab_test.cc att kompileras.

Exempel 3

En `Makefile` ska kunna vara enkelt att konfigurera. Ifall du nu behöver använda någon annan kompilator istället för g++ ska du inte behöva gå igenom hela `Makefile` och ändra g++ till någonting annat. Därför finns macron.

```
CCC=g++

main: lab.cc
    $(CCC) lab.cc

test: lab_test.cc
    $(CCC) lab_test.cc
```

Nu behöver du bara ändra på en enda rad ifall du behöver byta kompilator. Med andra ord kommer kommer `make main` att köra kommandot `g++ lab.cc` att köras.

Exempel 4

Samma sak som kompilatorn kan du göra med alla flaggor

```
CCC=g++
CFLAGS=-Wall -Wextra -pedantic

main: lab.cc
    $(CCC) $(CFLAGS) lab.cc

test: lab_test.cc
    $(CCC) $(CFLAGS) lab_test.cc
```

Nu kommer `make main` att bli `g++ -Wall -Wextra -pedantic lab.cc`.

Exempel 5

Nästa grej är att beroenden kan vara regler i sig

```
CCC=g++
CFLAGS=-Wall -Wextra -pedantic

main: lab.o
    $(CCC) $(CFLAGS) lab.o

lab.o: lab.cc
    $(CCC) $(CFLAGS) -c lab.cc

test: lab_test.cc
    $(CCC) $(CFLAGS) lab_test.cc
```

Här har ni en regel som beroende. `main` beror på `lab.o`, som i sin tur beror på `lab.cc`. En viktig sak att notera är flaggan *-c* som används för att skapa objektfiler som slutar på .o (exempel lab.o).

Exempel 6

```
CCC=g++
CFLAGS=-Wall -Wextra -pedantic

main: lab.o
    $(CCC) $(CFLAGS) lab.o

lab.o: lab.cc
    $(CCC) $(CFLAGS) -c lab.cc

test: lab_test.cc
    $(CCC) $(CFLAGS) lab_test.cc

clean:
    \rm *.o a.out
```

Här har vi lagt till regeln clean. Ifall du skriver `make clean` kommer `\rm *o a.out` att köras i terminalen och ta bort alla filer som slutar på .o samt filen a.out.