

TDP005: Introduktion till Make

Jonas Lindgren Torbjörn Lönnemark
`jonas.lindgren@liu.se` `torbjorn.lonnemark@liu.se`

Niklas Hayer

2013-11-14

1 Programmet make

Om du kör **make** letar programmet efter en fil som heter **Makefile** i arbetsmappen, för att sedan exekvera den. Har du flera make-filer i samma katalog och vill köra en specifik kan du göra det med kommandot **make -f MinMakefil**. Det finns flera andra flaggor till **make**, för mer information se **man make**.

2 Byggprocessen

1. Källfiler kompileras till objekt-filer.
2. Länkaren länkar ihop objektfilerna till en exekverbar fil.

3 Kompilera för hand

Görs enkelt med hjälp av kommandot **g++ main.cpp other.cpp -o app** vilket genererar den körbara filen **app**.

4 Grunderna i en makefil

En makefil består i grunden av:

```
mål: beroenden  
[tab]system-kommando
```

För att applicera det på vårt tidigare kommando skulle detta ge oss en makefil innehållandes:

```
all:  
    g++ main.cpp other.cpp -o app
```

Vi kan nu köra denna makefil med kommandot **make** (givet att make-filen kallas **Makefile**). I exemplet kan vi se att standardmålet heter **all**, så om man inte anger ett specifikt mål kommer **make** att bygga **all**. Vi ser även att **all** inte har några beroenden, så **make** kan direkt köra det specificerade kommandot.

5 Beroenden

Det är användbart att ange många olika mål, eftersom man då bara behöver kompilera om de modifierade filer, och inte hela projektet när man ändrat något.

```
all: app

app: main.o other.o
    g++ main.o other.o -o app
main.o: main.cpp
    g++ -c main.cpp
other.o: other.cpp
    g++ -c other.cpp
clean:
    rm -rf *.o app
```

Här så har målet `all` enbart beroenden och inga kommandon, `make` ser då till att alla dessa beroenden tillgodoses och är uppdaterade. I det här fallet betyder det att `make` ser till att både `main.o` samt `other.o` är kompillerade från senaste versionen av källkodsfilerna varefter de länkas ihop enligt målet `app`. Nyttillkommet här är även målet `clean`, som man kan använda för att snabbt ta bort alla objektfiler samt exekverbara filer. Det är användbart om man till exempel vill kompilera om alla delar i sitt projekt, eller om man vill rensa upp för att sedan packa ihop och distribuera det.

6 Variabler och kommentarer

Man kan (och bör) använda variabler för att göra sina makefiler kortare och mer lättförståeliga.

```
# Det här är en kommentar
# Kompilatorn som ska användas:
CC=g++
# Flaggor till kompilatorn:
CFLAGS=-std=c++11 -Wall -Wextra -pedantic

all: app
```

```
app: main.o other.o
    $(CC) main.o other.o -o app
main.o: main.cpp
    $(CC) $(CFLAGS) -c main.cpp
other.o: other.cpp
    $(CC) $(CFLAGS) -c other.cpp
clean:
    rm -rf *.o app
```

Som synes kan variabler ibland vara användbara. För att använda dem tilldelar man ett värde till en variabel innan man definierar sina mål, och använder den därefter med `$(VARIABEL)`.

7 Övning (Gör Detta!)

1. Hämta övningsfilerna från kurshemsidan.
2. Skriv en makefil med målen `all`, `clean` och `run` för den givna C++-koden. Varje källkodsfil ska ha sitt eget delmål.
3. Testa att:
 - Göra en ändring i `student.cpp`. Kompilera med `make`. Vilka filer kompileras om?
 - Göra en ändring i `student.h`. Kompilera med `make`. Vilka filer kompileras om då? Om inget kompileras om, kan du komma på något sätt att åtgärda det?
4. Skapa ett Eclipse projekt med standard `make`, och lägg till din egen makefil. Detta är en speciell inställning man får göra efter man skapat projektet. Kom ihåg att använda `clean` efter ni gjort den inställningen.
5. Skapa ett managed `make` projekt i Eclipse, så att en makefile genereras. Det här är default-inställningen i Eclipse. Titta igenom den, och reflektera över eventuella skillnader mot den som du själv skrivit.

Visa dina makefiler och slutsatser för labbassistent när du är klar.

8 Fördjupning

Det finns mer avancerad funktionalitet i make än den som beskrivs i det här dokumentet. Vill man fördjupa sig i dessa finns det många resurser bland annat på internet som tar upp denna. Några platser att börja leta är:

<https://www.ida.liu.se/~tao/pub/tools/make/make-CPP.pdf>

http://theory.uwinnipeg.ca/gnu/make/make_toc.html

<http://www.student.cs.uwaterloo.ca/~isg/res/unix/make/>