CC
GCC
MAKE

I più comuni compilatori C in ambiente Unix

"Programma che serve a mantenere, aggiornare e rigenerare programmi e file collegati"

`cc pippo.c` → Nome eseguibile: a.out

`cc –o pippo pippo.c` → Nome eseguibile: pippo

```
cc –o pippo pippo1.c pippo2.c
```

```
cc –c pippo1.c
cc –c pippo2.c
cc –o pippo pippo1.o pippo2.o
```

`make pippo`

Posso memorizzare comandi di compilazione in file detti makefile e chiederne l'interpretazione al programma make

---

Contenuto di un makefile

Un makefile contiene una sequenza di specifiche che rispettano la sintassi:

```
TARGET: DIPENDENZE
    COMANDO
```

(tab)

Esempio

```
pippo:  pippo1.c pippo2.c  pippo2.h
        cc –o pippo pippo1.c  pippo2.c
```

pippo (casualmente il nome di un eseguibile) dipende dai file **pippo1.c, pippo2.c e pippo2.h**

Quando viene eseguito il comando make, se anche uno solo dei file contenuto nella lista delle dipendenze risulta **più recente di pippo**, viene eseguito il comando (in questo caso di compilazione) associato

Contenuto di un makefile

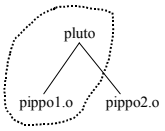
Un makefile può contenere molte regole associate a molti target differenti eventualmente collegati gli uni agli altri

Esempio

```
pippo1.o:  pippo1.c
        cc –c pippo1.c

pippo2.o:  pippo2.c
        cc –c pippo2.c –lm

pluto: pippo1.o  pippo2.o
        cc –o pluto pippo1.o  pippo2.o
```



Vantaggio:
Se modifico pippo1.c non dovrò ricompilare tutto!!

Contenuto di un makefile

Di solito si aggiungono anche due target speciali: all e clear

```
target1: lista1
        cc –o target1 …

target2: lista2
        cc –o target2 …

all: target1  target2

clear:
        rm target1 target2
```

→ Causa l'eventuale compilazione di tutti i target

→ Causa la cancellazione di tutti i target

Contenuto di un makefile

Definizione e uso di macro e inserzione di commenti

```
CC = gcc

# compilazione di pippo
pippo:  pippo.c
        $(CC)

# compilazione di pluto
pluto: pluto.c
        $(CC) –o pluto pluto.c
```

→ definisco la macro CC assegnadole come valore la stringa gcc

→ per utilizzare una macro si usa la notazione $(vrb)

→ i commenti sono preceduti da # terminano al newline

Uso di un makefile

prompt> make → di default viene cercato un file avente nome Makefile o makefile e viene processato il I target

prompt> make target1 → viene eseguito l'eventuale aggiornamento del sottoalbero avente radice target1

prompt> make target1 target2 → posso indicare più target come argomenti

prompt> make -f makefilename target → Con l'opzione –f posso indicare a make di usare le regole contenute in un file avente nome diverso da quelli di default

esempio

```
pippo1:  pippo1.c
	gcc -o pippo1 pippo1.c

pippo2:  pippo2.c
	gcc -o pippo2 pippo2.c

pippo3:  pippo3.c
	gcc -o pippo3 pippo3.c

all: pippo1 pippo2 pippo3

clear:
	rm pippo1 pippo2 pippo3
```

Provare a scrivere tre programmini che fanno una stampa, a creare il makefile riportato a lato e poi eseguire:

make pippo2

make

make all

make clear

E verificare gli effetti

Provare a parametrizzare rispetto al compilatore, alla directory in cui sono contenuti i file e aggiungere nelle dependency list makefile medesimo