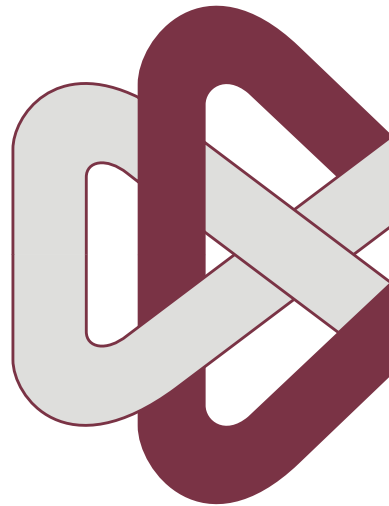




CIMAT

Makefiles, bevísimo tutorial

MC. Miguel Vargas Félix
miguelvargas@cimat.mx
<http://www.cimat.mx/~miguelvargas>

Contenido

Makefiles

El Makefile mínimo

El Makefile con dependencias

Compilando en paralelo

¿Preguntas?

Para saber más...

Makefiles

Se utilizan cuando hay que compilar varios archivos de código fuente.

Es análogo a un *project file* de Visual C++ o Builder.

Asigna un orden de compilación a un proyecto.

Sirve para establecer dependencias entre archivos de código.

Es el standard “de facto” para compilar programas de C, C++ y fortran en sistemas POSIX (Unix, Linux, FreeBSD, etc.).

El Makefile mínimo

Queremos compilar un proyecto

funciones.h

funciones.c

main.c

Makefile

El archivo makefile contendría:

```
all:
    → gcc -c funciones.c
    → gcc -c main.c
    → gcc -o program funciones.o main.o

clean:
    → rm -f funciones.o main.o program
```

Es standard que el archivo se llame “Makefile”

Se invoca con:

make

El símbolo ‘→’ indica que se inserta un tabulador.

```
all:
    → gcc -c funciones.c
    → gcc -c main.c
    → gcc -o program funciones.o main.o

clean:
    → rm -f funciones.o main.o program
```

Un texto al inicio de una línea, seguido de dos puntos define un target:

```
all:
```

define al target “all”. Las líneas siguientes que inicien con un tabulador serán las instrucciones para completar este target.

Un target puede tener cualquier nombre. Se puede ejecutar un target en particular con:

```
make clean
```

El target que se ejecuta al utilizar el comando **make** es el primero que aparece en el Makefile.

En este caso

```
make
```

es equivalente a

```
make all
```

Los targets que siempre se suelen definir en un Makefile son: “all” y “clean”.

El Makefile con dependencias

Un target puede además indicar dependencias a otros targets o archivos:

```
all: program

program: funciones.o main.o
    → gcc -o program funciones.o main.o

funciones.o: funciones.c funciones.h
    → gcc -c funciones.c

main.o: main.c funciones.h
    → gcc -c main.c

clean:
    → rm -f funciones.o main.o program
```

Si el target es un archivo, entonces éste se ejecutará solo si el archivo no existe o si es más antiguo que sus dependencias.

Compilando en paralelo

Si se trata de un proyecto grande y se cuenta con una computadora *multi-core* es útil paralelizar la compilación, por ejemplo:

```
make -j 3
```

Compilará los targets del Makefile ejecutando tres renglones de instrucciones a la vez.

¿Preguntas?

Para saber más...

man make

En la WWW:

<http://en.wikipedia.org/wiki/Makefile>

<http://www.gnu.org/software/make/manual>

<http://www.google.com/#&q=makefile+tutorial>