Makefile

El `make` es una herramienta para actualizar, en forma optimizada y automática, los diversos archivos de programas que integran un proyecto de software. Las reglas de actualización se escriben en un archivo de texto llamado usualmente `makefile` o `Makefile`. La actualización se invoca dando el comando `make` que ejecuta las reglas del archivo `makefile` recompilando sólo las partes que han sido modificadas desde la última compilación, y enlaza los módulos en código objeto construyendo el ejecutable.
`make` puede usarse con cualquier lenguaje de programación cuyas tareas puedan lanzarse mediante comandos. En realidad puede usarse para manejar cualquier conjunto de archivos en los cuales haya archivos que deban actualizarse toda vez que cambien ciertos otros.
`make` acepta parámetros para realizar sólo ciertas tareas, como ser eliminar archivos intermedios o compilar sólo un módulo. Las reglas para cada tarea deben estar escritas en el `makefile`.

Programa ejemplo

Usaremos como ejemplo un programa escrito en C que posee un programa principal y dos módulos, el código está integrado por los archivos:

- `main.c`
- `modulo1.h`
- `modulo1.c`
- `modulo2.h`
- `modulo2.c`

La compilación simple

La compilación de este programa puede hacerse con el comando:

```
g++ -o ejec main.c modulo1.c modulo2.c
```

Este comando crea el archivo ejecutable `ejec`, que puede invocarse digitando `./ejec`.
Toda vez que se modifique alguno de los archivos que integran el programa, este comando recompila todo, haya cambiado o no. En un programa grande, esto es un inconveniente.

Compilación con make

Se crea un archivo `makefile` con el siguiente contenido:

```
# makefile para programa ejemplo
# usar tabulador (no espacios) en la línea de comando
ejec: main.o modulo1.o modulo2.o
      g++ -o ejec main.o modulo1.o modulo2.o
main.o: main.c
      g++ -c main.c
modulo1.o: modulo1.h modulo1.c
      g++ -c modulo1.c
modulo2.o: modulo2.h modulo2.c
      g++ -c modulo2.c
clean:
      rm ejec
      rm *.o
```

Las líneas iniciadas con `#` son comentarios, no se ejecutan. El resto de las líneas son reglas. Cada regla empieza con un nombre seguido de "`:`" luego aparecen las dependencias, que pueden ser otras reglas o archivos con el código. Las líneas iniciadas con tabulador son continuación de la misma regla, y contienen las órdenes a ejecutarse para esa regla.

Reglas de make.

Las reglas tienen el siguiente formato:

```
destino: requisito ...
    comando
    ...
```

destino es el nombre de un archivo a crear, un ejecutable o un archivo objeto (.o). También puede ser el nombre de una tarea realizar: es usual usar "**clean**" como indicativo de la regla que ejecuta los comandos necesarios para eliminar archivos objeto y recomenzar una compilación desde cero.

requisito es el nombre de un archivo del cual depende el destino a crear. Un destino suele depender de varios archivos requisito. Cuando el destino es el nombre de una tarea no hay requisitos.

comando es una acción a realizar. La creación de un destino puede requerir varios comandos. Cada línea de comando debe comenzar con TABULADOR, no sirven los espacios. El comando es un comando normal, que puede darse en la línea de comandos del shell, y debe ejecutar correctamente, pues **make** no sabe nada de él.

Uso de variables

La siguiente versión de **makefile** muestra el uso de variables:

```
# makefile para ejemplo con uso de variables
# usar tabulador (no espacios) en la línea de comando
CC = g++
OBJECTS = main.o modulo1.o modulo2.o
ejec: $(OBJECTS)
      $(CC) -o main $(OBJECTS)
main.o: main.c
      $(CC) -c main.c
modulo1.o: modulo1.h modulo1.c
      $(CC) -c modulo1.c
modulo2.o: modulo2.h modulo2.c
      $(CC) -c modulo2.c
clean:
      rm ejec
      rm *.o
```

En esta versión resulta fácil cambiar el compilador, o agregar un nuevo nombre de archivo objeto. Existen nombres de variables comprendidas por **make**, para usos específicos, como **CC** en el ejemplo; otras son costumbre, como **OBJECTS**; también pueden crearse otras a gusto del programador.

Opciones

El comando **make** acepta entre otras estas opciones:

- **-n** *muestra comandos a ejecutar, sin ejecutarlos.*
- **-f *archivo*** *indica el nombre del archivo en caso de no usar uno llamado "**makefile**".*

Resumen

La compilación de un ejecutable a partir de programas en C y sus archivos de encabezado puede simplificarse creando un archivo ***makefile*** siguiendo los modelos mostrados.
Para compilar usando un archivo con reglas llamado ***makefile*** basta ejecutar el comando

```
make
```

Si sólo quieren verse los comandos a ejecutar, digitar

```
make -n
```

Si se quiere compilar las reglas escritas en otro archivo que no sea ***makefile*** (Ej: ***reglas.make***)

```
make -f reglas.make
```

Si se quiere ejecutar solo una regla (Ej: ***clean***)

```
make clean
```

Los comandos a ejecutar dependen del estado de actualización de los archivos; si no han habido cambios desde la última creación del archivo objeto, la regla no se ejecuta.