

Makefiles

En proyectos grandes y complejos es una buena práctica automatizar el proceso de compilación y vinculación, no solo debido a la complejidad que puedan surgir de las dependencias entre los fuentes sino también para evitar la compilación de archivos que no sufrieron cambios. Además algunos fuentes pueden necesitar distintos opcionales en la compilación o vinculación, haciendo extenso y complejo el comando para compilar dichos archivos.

Comando make

El comando make automáticamente busca en el directorio actual (también se puede especificar un path específico) un archivo llamado makefile o Makefile donde definiremos la estructura que debe seguir el proceso de compilación de nuestro proyecto.

Archivo makefile

El archivo makefile, puede contener comentarios al estilo bash scripting (#), variables y **dependencias**. Estas últimas son justamente lo mas importante ya que es lo que permite al comando make fabricar el árbol de dependencias para realizar el proceso de compilación.

Veamos un ejemplo: Supongamos que queremos compilar un fuente llamado **fork.c** y generar el binario ejecutable **ejemplo_fork**. Nuestro makefile sería así

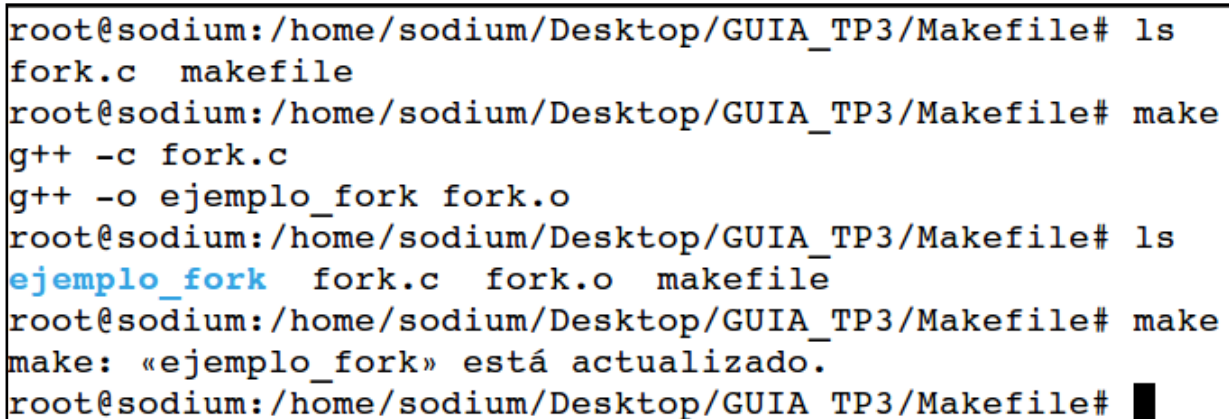


```
makefile ✕
ejemplo_fork: fork.o
    g++ -o ejemplo_fork fork.o

fork.o: fork.c
    g++ -c fork.c

clean:
    rm *.o
```

Nótese en la siguiente captura el resultado de ejecutar el comando make, se generaron el archivo fork.o y el binario **ejemplo_fork**. Además al intentar compilar de nuevo, dado que el fuente no fue modificado y el archivo fork.o se encuentra presente, el comando nos avisa que el binario está actualizado.



```
root@sodium:/home/sodium/Desktop/GUIA_TP3/Makefile# ls
fork.c  makefile
root@sodium:/home/sodium/Desktop/GUIA_TP3/Makefile# make
g++ -c fork.c
g++ -o ejemplo_fork fork.o
root@sodium:/home/sodium/Desktop/GUIA_TP3/Makefile# ls
ejemplo_fork  fork.c  fork.o  makefile
root@sodium:/home/sodium/Desktop/GUIA_TP3/Makefile# make
make: «ejemplo_fork» está actualizado.
root@sodium:/home/sodium/Desktop/GUIA_TP3/Makefile#
```

Sobre las líneas del archivo podemos decir lo siguiente:

Tenemos etiquetas; son las que están a la izquierda y finalizan con el carácter : por lo general son nombres de archivos (ejemplo_fork y fork.o), pero también se pueden utilizar para especificar utilidades opcionales (mas para mantenimiento del proyecto) mas allá del proceso de compilación (clean).

Tenemos dependencias; son las que están a la derecha del carácter : y en ellas se especifican el/los archivo/s necesario/s para construir el archivo especificado en la etiqueta.

Tenemos acciones; son las que están debajo de la etiqueta y separadas por un TAB desde el margen izquierdo, en este caso:

```
g++ -o ejemplo_fork fork.o
g++ -c fork.c
```

Ellas son las que se ejecutan, de ser necesario (si hay que actualizar el archivo etiquetado).

La acción `rm *.o` se ejecuta si se especificó al makefile dicha opción (clean), por ejemplo;

```
root@sodium:/home/sodium/Desktop/GUIA_TP3/Makefile# make clean
rm *.o
root@sodium:/home/sodium/Desktop/GUIA_TP3/Makefile# ls
ejemplo_fork  fork.c  makefile
root@sodium:/home/sodium/Desktop/GUIA_TP3/Makefile#
```

Nótese que se eliminó el archivo objeto `fork.o`.

También se podría por ejemplo hacer un backup del binario.

A continuación vemos un ejemplo donde se hace uso de una variable para especificar el compilador a utilizar en las acciones, dado un cambio de compilador, habrá que modificar la variable solamente.



```
makefile X
comp=g++ # Especificamos el compilador

ejemplo_fork: fork.o
    $(comp) -o ejemplo_fork fork.o

fork.o: fork.c
    $(comp) -c fork.c

clean:
    rm *.o
```

Phony Targets:

En el ejemplo anterior la etiqueta `clean` no funcionaría si existiera un archivo llamado “clean” en el directorio donde se ejecuta el makefile. En este caso se puede utilizar lo que se conoce como Phony Target (Objetivo Falso), entonces el comando `make` ya sabe que esa etiqueta no se corresponde con un archivo a construir, por más que exista dicho archivo. Se lo debe especificar de la siguiente manera:

```
makefile ✕  
.PHONY: clean  
  
comp=g++ # Especificamos el compilador  
  
ejemplo_fork: fork.o  
    $(comp) -o ejemplo_fork fork.o  
  
fork.o: fork.c  
    $(comp) -c fork.c  
  
clean:  
    rm *.o
```

Para mas información pueden acceder al material de la cátedra en Sisop
<http://www.sisop.com.ar/apuntes>

Programación en C – Básicos

[C Tutorial \(Remy Saville\)](#)

[Punteros en C \(Ted Jensen\)](#)

[Mis primeros garabatos en C \(Universidad de Edmonton, Canada\)](#)

[ANSI C for Programmers on UNIX Systems \(Tim Love, Cambridge\)](#)

[Using Make to Maintain Software](#)

Y también en el siguiente link encontrarán una completa referencia a make y makefiles
https://www.gnu.org/software/make/manual/html_node/