

Taller 3: Make

Investigación.

Para comprender mejor Make, realice una pequeña búsqueda para responder las siguientes preguntas:

1. ¿Qué es GNU Make?

GNU Make es una herramienta que controla la generación de ejecutables y otros archivos no fuente de un programa a partir de los archivos fuente del programa.

<https://www.gnu.org/software/make/>

2. ¿Cuáles son los componentes más importantes de un archivo Makefile?

Los Makefiles contienen cinco tipos de cosas: reglas explícitas, reglas implícitas, definiciones de variables, directivas y comentarios. Las reglas, variables y directivas se describen en detalle en capítulos posteriores.

https://www.gnu.org/software/make/manual/html_node/Makefile-Contents.html

3. ¿Cómo se definen (asignaciones) y utilizan los macros dentro de un Makefile? Brinde un ejemplo.

Una macro o variable MAKE es una cadena que se expande cuando se llama desde un fichero makefile. Las macros permiten crear ficheros makefile genéricos o plantilla que se adaptan a diferentes proyectos software. Una macro puede representar listas de nombres de ficheros, opciones del compilador, programas a ejecutar, directorios donde buscar los ficheros fuente, directorios donde escribir la salida, etc.

La sintaxis de definición de macros en un fichero makefile es la siguiente:

Nombre = texto a expandir

donde,

Nombre: es el nombre de la macro. Es sensible a las mayúsculas y no puede contener espacios en blanco. La costumbre es utilizar nombres en mayúscula.

texto a expandir: es una cadena que puede contener cualquier carácter alfanumérico, de puntuación o espacios en blanco

Para definir una macro llamada, por ejemplo, NOMLIB que representa a la cadena libejemplo.a se especificará de la siguiente manera:

NOMLIB = libejemplo.a

<https://www.it.uc3m.es/~pedmume/asignaturas/2005/LAO/Lab2/tutorial4/node6.html>

4. ¿Qué utilidad tienen los macros que hacen referencia a herramientas del toolchain?

Existen macros predefinidas por make que pueden ser muy útiles al escribir un makefile:

\$(CC): devuelve el nombre del compilador de C por defecto (puede ser cc, gcc, etc)

\$(CFLAGS): opciones para el compilador de C

\$@: nombre del destino de la regla actual

\$?: lista con los archivos más antiguos que el destino en la lista de dependencias de la regla actual

\$\$: lista con todos los archivos en la lista de dependencias de la regla actual

\$<: archivo en la lista de dependencias compuesta de un solo archivo en la regla actual

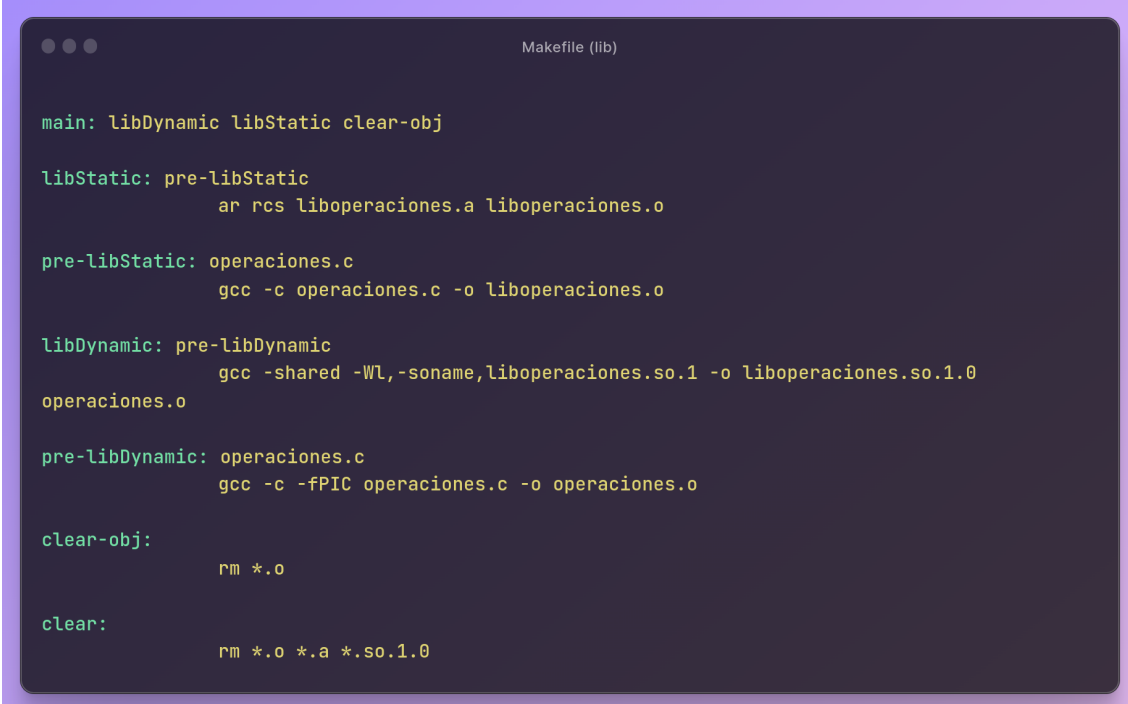
De esta manera, gracias al toolchain, se pueden obtener todo ese tipo de configuraciones sin tener que conocerlas de manera explícita.

https://www.it.uc3m.es/~pedmume/asignaturas/2005/LAO/Lab2/index_es.html

Ejercicios prácticos.

Debido a que la biblioteca se hizo en el taller 2 y el código utilizado para el presente taller es el mismo, no resulta conveniente dar detalles de la misma.

En este taller solo se va a enfocar en mostrar los archivos Makefile requeridos.



```
Makefile (lib)

main: libDynamic libStatic clear-obj

libStatic: pre-libStatic
        ar rcs liboperaciones.a liboperaciones.o

pre-libStatic: operaciones.c
        gcc -c operaciones.c -o liboperaciones.o

libDynamic: pre-libDynamic
        gcc -shared -Wl,-soname,liboperaciones.so.1 -o liboperaciones.so.1.0
operaciones.o

pre-libDynamic: operaciones.c
        gcc -c -fPIC operaciones.c -o operaciones.o

clear-obj:
        rm *.o

clear:
        rm *.o *.a *.so.1.0
```

Figura 1. Makefile de la carpeta lib

```
Makefile (src)

main: static dynamic

static:
    gcc -static calculadora.c -L ../lib -loperaciones -o ../bin/calculadora_d

dynamic:
    gcc calculadora.c -o ../bin/calculadora_e -L ../lib -loperaciones
```

Figura 2. Makefile de la carpeta src

```
Makefile (src)

main: lib-rule src-rule

lib-rule:
    cd lib && $(MAKE)

src-rule:
    cd src && $(MAKE)
```

Figura 3. Makefile principal.

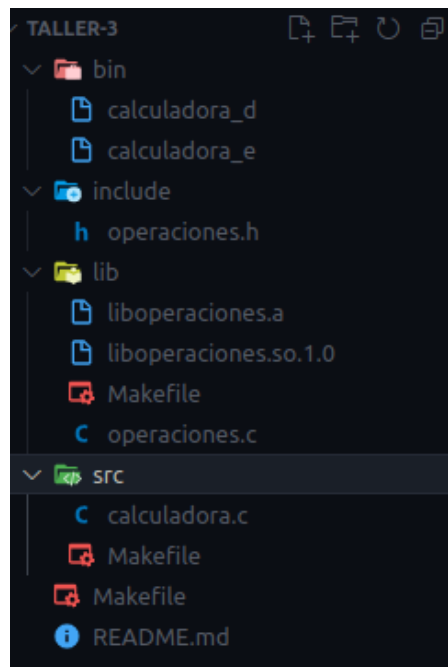


Figura 4. Estructura final del proyecto.

```

● samastua@samastua-Inspiron-15-3567:~/Documents/TEC/Embebidos/Git/Taller-3$ make
cd lib && make
make[1]: Entering directory '/home/samastua/Documents/TEC/Embebidos/Git/Taller-3/lib'
gcc -c -fPIC operaciones.c -o operaciones.o
gcc -shared -Wl,-soname,liboperaciones.so.1 -o liboperaciones.so.1.0 operaciones.o
gcc -c operaciones.c -o liboperaciones.o
ar rcs liboperaciones.a liboperaciones.o
rm *.o
make[1]: Leaving directory '/home/samastua/Documents/TEC/Embebidos/Git/Taller-3/lib'
cd src && make
make[1]: Entering directory '/home/samastua/Documents/TEC/Embebidos/Git/Taller-3/src'
gcc -static calculadora.c -L ../lib -loperaciones -o ../bin/calculadora_d
gcc calculadora.c -o ../bin/calculadora_e -L ../lib -loperaciones
make[1]: Leaving directory '/home/samastua/Documents/TEC/Embebidos/Git/Taller-3/src'

```

Figura 5. Ejecución del Makefile principal.

```

○ samastua@samastua-Inspiron-15-3567:~/Documents/TEC/Embebidos/Git/Taller-3$ ./bin/calculadora_e
Digite una opción de las siguientes:

[1] para la suma
[2] para la resta
[3] para la multiplicación
[4] para la división
[5] para la raíz cuadrada
[6] para el coseno
[0] para salir
6
Digite el número a aplicar el coseno (en grados):
180

[COSENO] ---> cos(180.000), resultado ---> -1.000

```

Figura 6. Ejecución de la calculadora con la librería estática.

```

○ samastua@samastua-Inspiron-15-3567:~/Documents/TEC/Embebidos/Git/Taller-3$ ./bin/calculadora_d
Digite una opción de las siguientes:

[1] para la suma
[2] para la resta
[3] para la multiplicación
[4] para la división
[5] para la raíz cuadrada
[6] para el coseno
[0] para salir
3
Digite el primer número:
25
Digite el segundo número:
4

[MULTIPLICACION] ---> 25.000 * 4.000, resultado ---> 100.000

```

Figura 7. Ejecución de la calculadora con la librería dinámica.