MAKEFILE

1. BASES

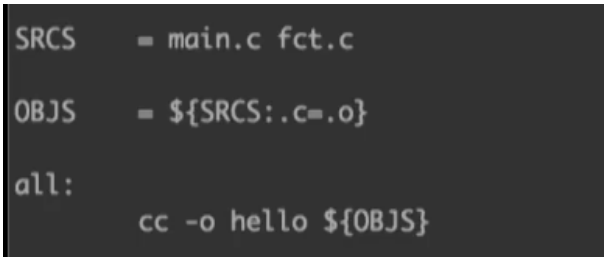
Fichero para organizar la compilación.
Compilar de manera más fácil archivos más complejos (con muchas líneas) o muchos archivos.
Declaramos una serie de reglas que nos dirán cómo hacer determinadas acciones.

1. Creamos archivo llamado “Makefile”
2. Si ponemos make en consola nos dirá que no hay objetivos (ya que todavía no hemos escrito nada)
3. Se pueden escribir comentarios con # (sirve hasta el fin de línea)
4. (Teniendo el make terminado) Si pongo en la terminal “make” se ejecutará el primer programa, si pongo “make main”, se ejecutará sólo el programa main.
5. Si ponemos (en la primera línea); SRCS = “nombre archivos que queremos que sustituya”; después en las instrucciones podemos poner: cc -o (y en vez del nombre de todos los archivos, pondremos ${SRCS} y compilara los archivos que hayamos puesto en el SRCS.

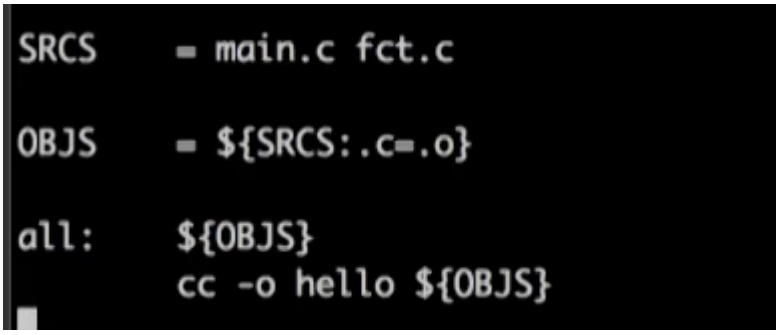
6. Creamos una variable llamada OBJS y le diremos que partiendo de SRCS transforme los ‘.c’ en ‘.o’; entonces pondremos en ‘all’ ‘OBJS’.

Estructura (regla):

- Objetivo: dependencias
  (TAB) instrucciones



En este caso le diremos a ‘all’, me hace falta ‘x’ para ejecutar esta regla (‘x’ sería nuestra dependencia). En este caso le pediremos los ‘.o’, que los creaba nuestro objetivo ‘OBJS’



- Si ya he compilado x archivos y modifico uno de ellos, al compilarlos de nuevo, sólo se compilaría el modificado ya que los anteriores ya estaban compilados, así la compilación se hace más sencilla.

Definiciones:

- **Objetivo**: Qué queremos hacer (por ej obtener programa)
- **Dependencias:** Archivos ya existentes
- **Instrucciones de código** (por ej compila este archivo con gcc…)

2. NOCIONES AVANZADAS

Es recomendable tener todos nuestros binarios en variables, las variables son lo que hemos visto antes como 'SRCD' y 'OBJS'; se declara una variable con un '=' y posteriormente podremos llamar a esa variable poniendo ${variable}.

Si tiene ':' es un objetivo y si tiene '=' es una variable.

```
SRCS    = main.c fct.c

OBJS    = ${SRCS:.c=.o}

NAME    = hello

CC      = cc

CFLAGS  = -Wall -g

.c.o:
        ${CC} ${CFLAGS} -c $< -o ${<:.c=.o}

${NAME}:    ${OBJS}
        ${CC} -o ${NAME} ${OBJS}

all:        {$NAME}
```

Ejemplos de variables y objetivos que poner en nuestro Makefile:

```
SRCS    = main.c fct.c

OBJS    = ${SRCS:.c=.o}

NAME    = hello

CC      = cc
RM      = rm -f

CFLAGS  = -Wall -g

.c.o:
        ${CC} ${CFLAGS} -c $< -o ${<:.c=.o}

${NAME}:    ${OBJS}
        ${CC} -o ${NAME} ${OBJS}

all:        ${NAME}

clean:
        ${RM} ${OBJS}

fclean:     clean
        ${RM} ${NAME}

re:         fclean all

.PHONY:     all clean fclean re
```

3. BIBLIOTECAS

Usos:

- Tener una dirección donde almacenar todas las funciones que usamos regularmente.
- Transmitir código a alguien sin darle nuestro código fuente.

Cómo crear una biblioteca:

1. Compilar archivos '.c' en '.o' (gcc -c)
2. Escribimos: ar rc "nombre librería.a" - siempre tiene que tener 'lib' delante; después le ponemos los ficheros.o
3. Para comprobar:
   gcc -Werror -Wextra -Wall main.c -L -lstr

- Si tenemos muchos archivos que compilar, gcc tardará en compilarlo; en este caso escribiremos:
  ranlib 'nombre librería'
  ej: ranlib libstr.a
  Esto hará que al compilarlo tarde menos.