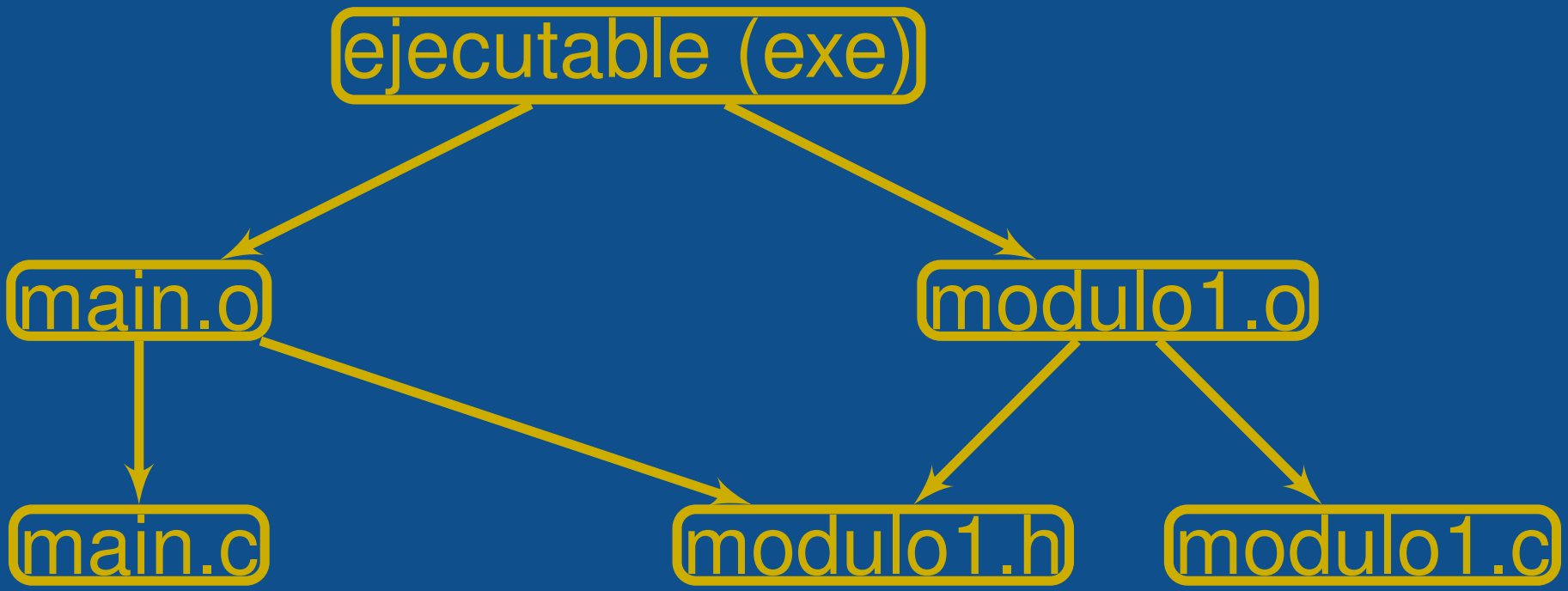
Ejemplo



Makefile

3 Reglas

```
ejecutable: main.o modulo1.o
	gcc -o ejecutable main.o sum.o

main.o: main.c modulo1.h
	gcc -c main.c

modulo1.o: modulo1.c modulo1.h
	gcc -c modulo1.c
```

*.o: *.c dependencias

- ✓ Un fichero *.o depende (por defecto) del correspodiente fichero *.c
- ✓ Ejemplo: foo.o: foo.h
  - ✗ Acción implícita: **$(cc) –c foo.c –o foo.o**

```
ejecutable: main.o modulo1.o
	gcc –o ejecutable main.o sum.o
#ERROR: main.o no se compila aunque main.c sea más reciente!!!

main.o: main.c modulo1.h
	gcc –c main.c

modulo1.o: modulo1.c modulo1.h
	gcc –c modulo1.c
```

Makefile equivalentes

```
OBJS=main.o modulo1.o

ejecutable: $(OBJS)
	gcc -o $@ $(OBJS)

%.o: %.c modulo1.h
	gcc -c $*.c
```

- ✓ **OBJS**: definición de variable
- ✓ **$(OBJS)**: obetenemos el valor de la variable
- ✓ **%.o**: patrón de regla
- ✓ **$*.c**: evalúa el correspondiente fichero *.c

Makefile: otro ejemplo

```
SRC = ../src
SRC_FILES = $(SRC)/foo.c $(COMMON_DIR)/bar.c main.c
OBJ_FILES = $(patsubst %.c,%.o,$(SRC_FILES))
CFLAGS = -Wall -g -ansi
CC = gcc
INCLUDES = -I$(SRC)

ejecutable: $(OBJ_FILES)
	$(CC) $(CFLAGS) $(OBJ_FILES) -lm -o $@
%.o:%.c
	$(CC) $(CFLAGS) $(INCLUDES) -c $*.c -o $*.o

clean:
	-rm $(OBJ_FILES) ejecutable
depend:
	@echo -e '\n' >> makefile
	$(CC) $(INCLUDES) -MM $(C_FILES) >> makefile
```

$(patsubst patrón,sustituto,texto):

Función para la sustitución de patrones en el texto dado como tercer argumento