Dispositivas auxiliares para la práctica 1

Sistemas operativos

Curso 2016-2017

Katzalin Olcoz

¿Qué es un Makefile?

- Fichero de texto que contiene un conjunto de reglas para compilar y enlazar un programa
  - La herramienta `make` procesa el fichero Makefile que está en el directorio actual y ejecuta los comandos necesarios para construir el ejecutable
  - Hay distintas variantes de make
    - En las prácticas de la asignaturas usaremos GNU make
    - Sintaxis de GNU Make en Sección 1.4.2 del guión de Práctica 1
  - No es necesario que el alumno cree Makefiles
    - Se proporciona uno con cada práctica y código de ejemplo
- Más información en la página de manual:
  - Info make

Makefile: ejemplo de la práctica 1

```
TARGET = mytar
CC = gcc
CFLAGS = -g -Wall
OBJS = mytar.o mytar_routines.o
SOURCES = $(addsuffix .c, $(basename $(OBJS)))
HEADERS = mytar.h

all: $(TARGET)

$(TARGET): $(OBJS)
	$(CC) $(CFLAGS) -o $(TARGET) $(OBJS)

.c.o:
	$(CC) $(CFLAGS) -c  $< -o $@

$(OBJS): $(HEADERS)

clean:
	-rm -f *.o $(TARGET)
```

Definiciones de variables: se sustituye $(TARGET) por mytar

Makefile: ejemplo de la práctica 1

```
TARGET = mytar
CC = gcc
CFLAGS = -g -Wall
OBJS = mytar.o mytar_routines.o
SOURCES = $(addsuffix .c, $(basename $(OBJS)))
HEADERS = mytar.h

all: $(

$(TAR

.c.o:

$(OBJ

clean
        -rm -f *.o $(TARGET)
```

Función de manipulación de texto (capítulo 8 de info make): Se ejecuta la función sobre los argumentos y su salida se escribe donde estaba la llamada a la función. Debe ir precedido de $ y entre paréntesis.

basename $(OBJS)

addsuffix .c mytar mytar_routines

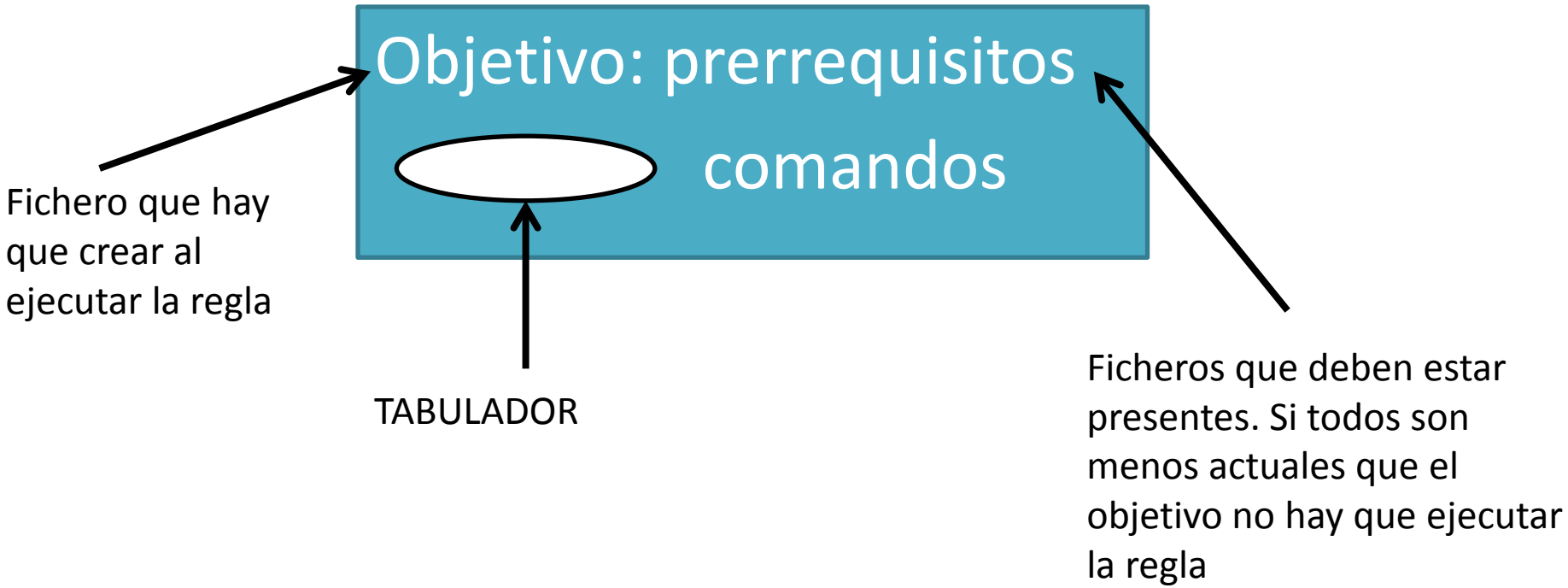
mytar.c mytar_routines.c

Makefiles: ejemplo de la práctica 1

```
TARGET = mytar
CC = gcc
CFLAGS = -g -Wall
OBJS = mytar.o mytar_routines.o
SOURCES = $(addsuffix .c, $(basename $(OBJS)))
HEADERS = mytar.h

all: $(TARGET)

$(TARGET): $(OBJS)
	$(CC) $(CFLAGS) -o $(TARGET) $(OBJS)

.c.o:
	$(CC) $(CFLAGS) -c  $< -o $@

$(OBJS): $(HEADERS)

clean:
	-rm -f *.o $(TARGET)
```

Reglas: pasos a seguir.

Formato de las reglas



Makefiles: ejemplo de la práctica 1

```
TARGET = mytar
CC = gcc
CFLAGS = -g -Wall
OBJS = mytar.o mytar_routines.o
SOURCES = $(addsuffix .c, $(basename $(OBJS)))
HEADERS = mytar.h

all: $(TARGET)

$(TARGET): $(OBJS)
	$(CC) $(CFLAGS) -o $(TARGET) $(OBJS)

.c.o:
	$(CC) $(CFLAGS) -c  $< -o $@

$(OBJS): $(HEADERS)

clean:
	-rm -f *.o $(TARGET)
```

$ make clean

Se ejecuta la regla virtual clean.

No se genera ningún fichero:

borra todos los ficheros .o y mytar

Makefiles: ejemplo de la práctica 1

```
TARGET = mytar
CC = gcc
CFLAGS = -g -Wall
OBJS = mytar.o mytar_routines.o
SOURCES = $(addsuffix .c, $(basename $(OBJS)))
HEADERS = mytar.h

all: $(TARGET)

$(TARGET): $(OBJS)
	$(CC) $(CFLAGS) -o $(TARGET) $(OBJS)

.c.o:
	$(CC) $(CFLAGS) -c  $< -o $@

$(OBJS): $(HEADERS)

clean:
	-rm -f *.o $(TARGET)
```

$ make

Se ejecuta la regla principal

Regla implícita: solo se indican las dependencias. El prerrequisito es mytar.

⇨Se procesa la regla para construir el prerrequisito.

Makefiles: ejemplo de la práctica 1

```
TARGET = mytar
CC = gcc
CFLAGS = -g -Wall
OBJS = mytar.o mytar_routines.o
SOURCES = $(addsuffix .c, $(basename $(OBJS)))
HEADERS = mytar.h

all: $(TARGET)

$(TARGET): $(OBJS)
        $(CC) $(CFLAGS) -o $(TARGET) $(OBJS)

.o: .c
        $(CC) $(CFLAGS) -c  $< -o $@

$(OBJS): $(HEADERS)

clean:
        -rm -f *.o $(TARGET)
```

Dependencia de la regla

```
gcc -g -Wall  -c  mytar.c -o mytar.o
gcc -g -Wall  -c  mytar_routines.c -o
mytar_routines.o
```

Objetivo de la regla

Makefiles: ejemplo de la práctica 1

```
TARGET = mytar
CC = gcc
CFLAGS = -g -Wall
OBJS = mytar.o mytar_routines.o
SOURCES = $(addsuffix .c, $(basename $(OBJS)))
HEADERS = mytar.h

all: $(TARGET)

$(TARGET): $(OBJS)
	$(CC) $(CFLAGS) -o $(TARGET) $(OBJS)

.c.o:
	$(CC) $(CFLAGS) -c  $< -o $@

$(OBJS): $(HEADERS)

clean:
	-rm -f *.o $(TARGET)
```

```
gcc -g -Wall  -o mytar mytar.o
mytar_routines.o
```