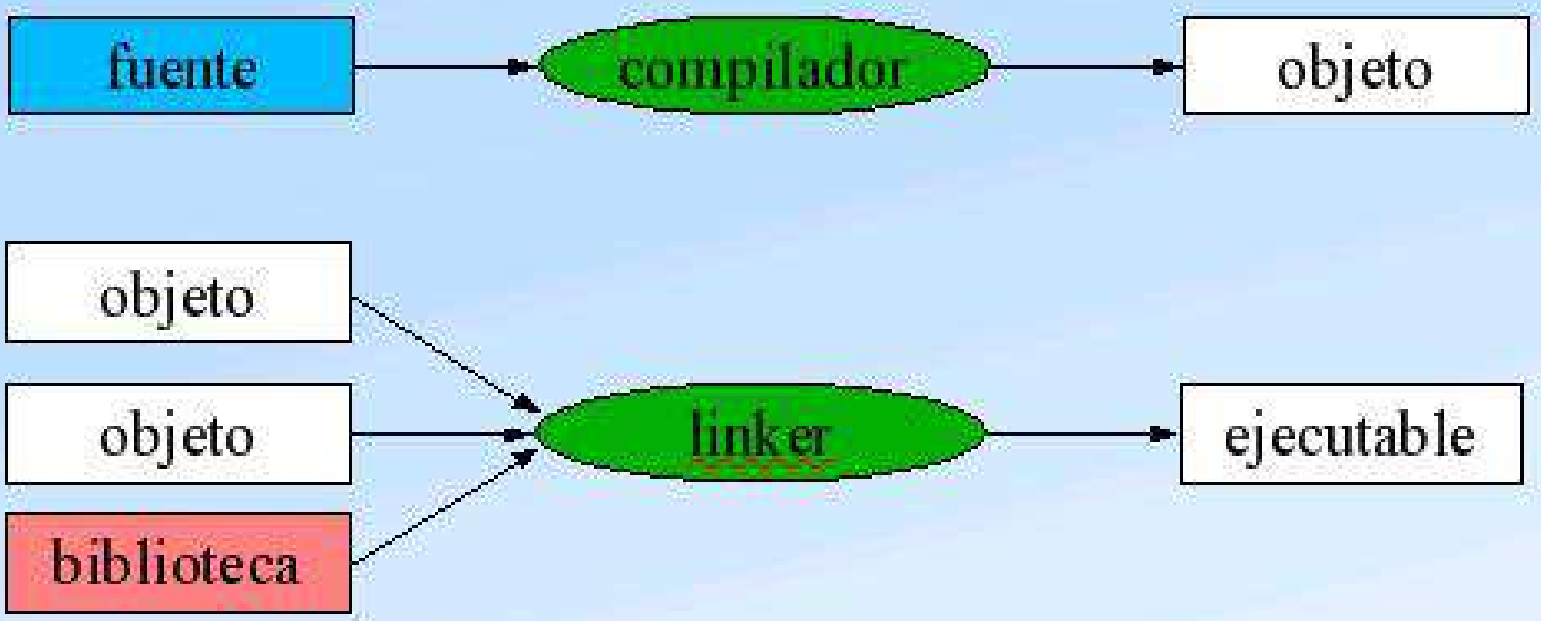
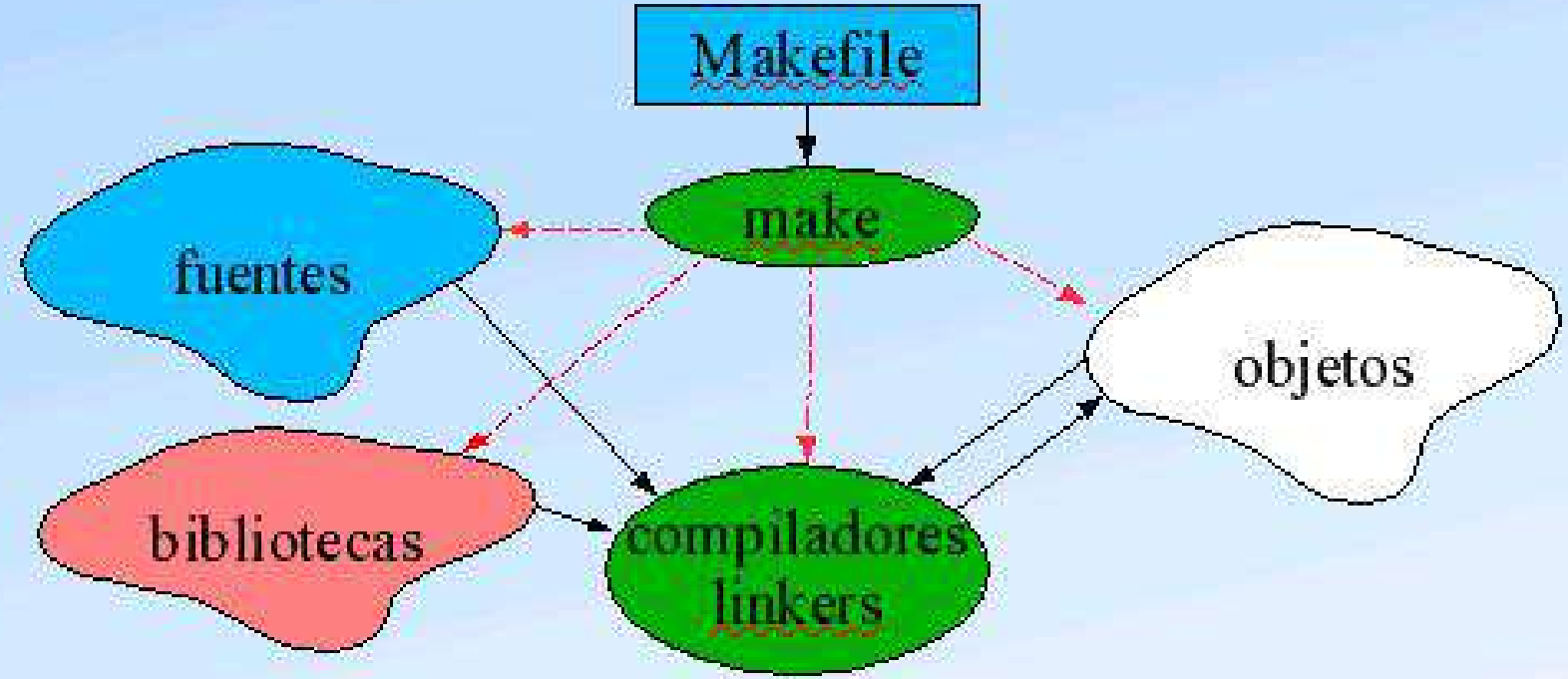
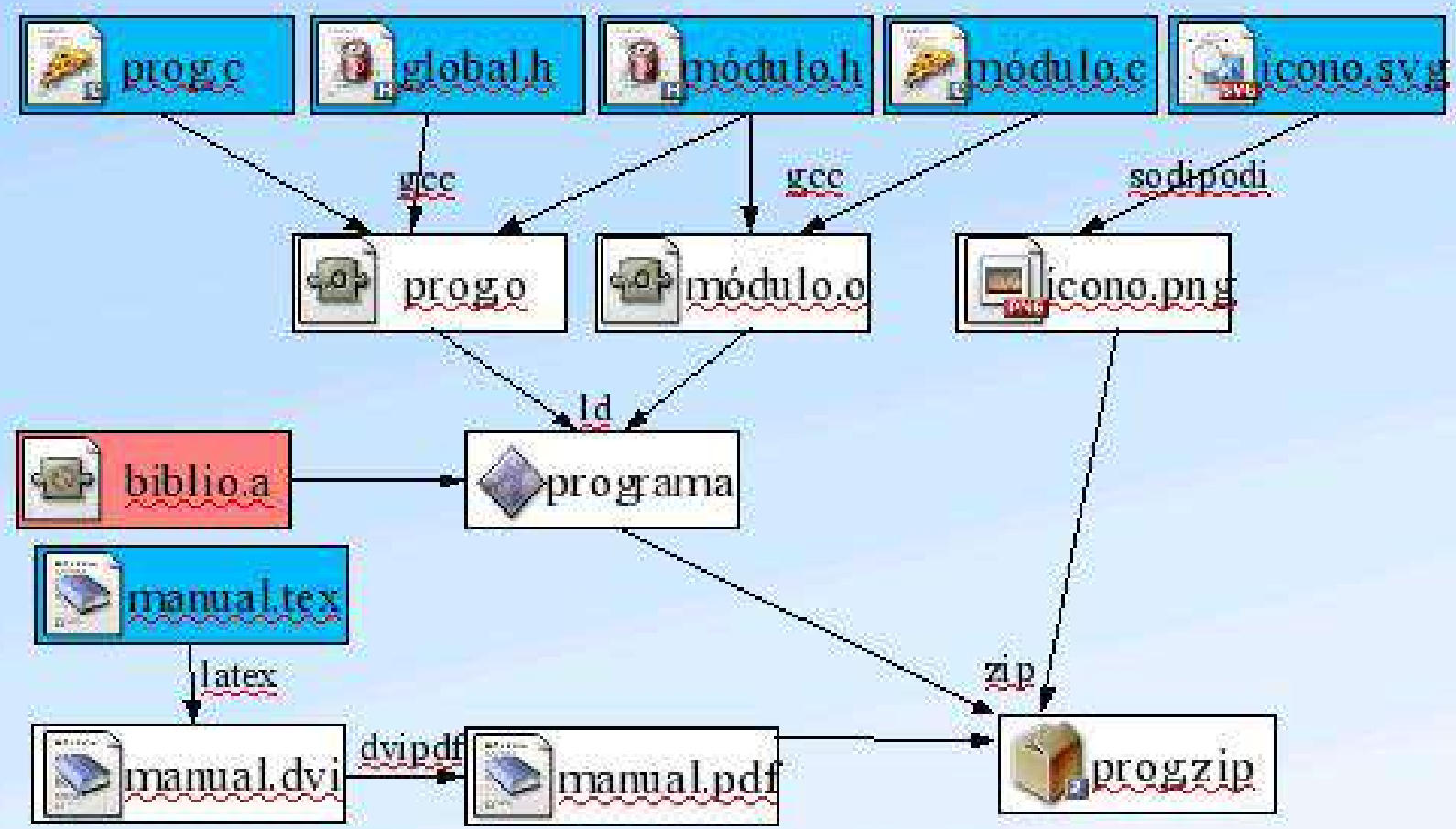
Compiladores



make


Makefile
make
fuentes
objetos
bibliotecas
compiladores
linkers

make: Resolución de dependencias



Estructura de un Makefile

```
VARIABLE=valor
VARIABLE=valor
VARIABLE=valor
VARIABLE=valor
# comentario
objetivo: dependencias
   comando
   comando

objetivo: dependencias
   comando
   comando
```

Un ejemplo

```
prog.o: prog.c global.h modulo.h
        gcc -c prog.c -o prog.o

modulo.o: modulo.c modulo.h
        gcc -c modulo.c -o modulo.o

programa: modulo.o prog.o biblio.a
        gcc -o programa modulo.o \
        prog.o biblio.a

manual.dvi: manual.tex
        latex manual.tex
```

Sustituciones

Make realiza algunas sustituciones

- `$(VAR)` por el contenido de `VAR`
- `$@` por el objetivo de la regla
- `$<` por el primer prerequisito
- `$^` por la lista de prerequisitos

Un ejemplo

```
prog.o: prog.c global.h modulo.h
        gcc -c prog.c -o prog.o

modulo.o: modulo.c modulo.h
        gcc -c modulo.c -o modulo.o

programa: modulo.o prog.o biblio.a
        gcc -o programa modulo.o \
        prog.o biblio.a

manual.dvi: manual.tex
        latex manual.tex
```

Usando sustituciones

```
prog.o: prog.c global.h modulo.h
        gcc -c $< -o $@

modulo.o: modulo.c modulo.h
        gcc -c $< -o $@

programa: modulo.o prog.o biblio.a
        gcc -o $@ $^

manual.dvi: manual.tex
        latex $<
```

Usando variables

```
CC=gcc
CFLAGS=-g

prog.o: prog.c global.h modulo.h
        $(CC) $(CFLAGS) -c $< -o $@

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

programa: modulo.o prog.o biblio.a
        $(CC) -o $@ $^
```

Reglas genéricas

```
CC=gcc
CFLAGS=-g

prog.o: prog.c global.h modulo.h
modulo.o: modulo.c modulo.h

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

programa: modulo.o prog.o biblio.a
        $(CC) -o $@ $^
```

Reglas implícitas

```
CC=gcc
CFLAGS=-g

prog.o: prog.c global.h modulo.h
modulo.o: modulo.c modulo.h

programa: modulo.o prog.o biblio.a
        $(CC) -o $@ $^
```

Dependencias automáticas

```
CC=gcc
CFLAGS=-g

include .depend

programa: modulo.o prog.o biblio.a
        $(CC) -o $@ $^
```

después, en el shell:

```
$ gcc -MM prog.c modulo.c >.depend
```

Inclusión automática

```
CC=gcc
CFLAGS=-g

-include .depend

.depend: prog.c modulo.c
        gcc -MM $^ >.depend

programa: modulo.o prog.o biblio.a
        $(CC) -o $@ $^
```

Un esquema (casi) general

```
SOURCES=prog.c modulo.c
LIBS=biblio.a
CC=gcc
CFLAGS=-g
OBJECTS=$(SOURCES:.c=.o)

-include .depend

.depend: $(SOURCES) *.h
        gcc -MM $^ >.depend

programa: $(OBJECTS) $(LIBS)
        $(CC) -o $@ $^
```

Un esquema general

```
SOURCES=prog.c modulo.c
LIBS=biblio.a
TARGET=programa
CC=gcc
CFLAGS=-g
OBJECTS=$(SOURCES:.c=.o)

all: $(TARGET)

clean:
        rm -f $(TARGET) $(OBJECTS) .depend

-include .depend

.depend: $(SOURCES) *.h
        gcc -MM $(SOURCES) >.depend

$(TARGET): $(OBJECTS) $(LIBS)
        $(CC) -o $@ $^
```