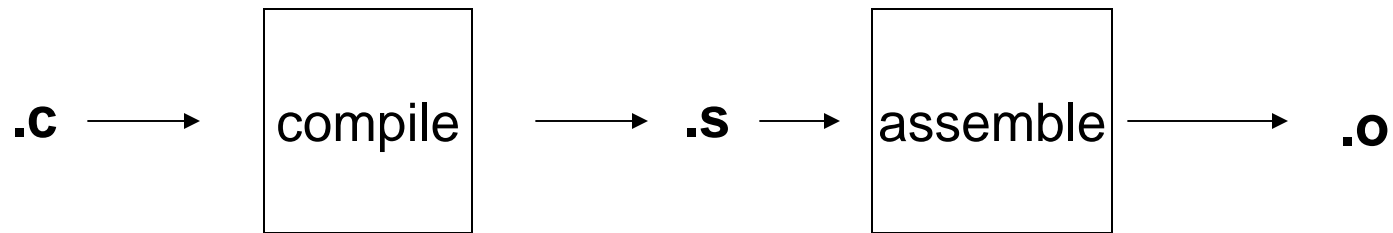


The make utility

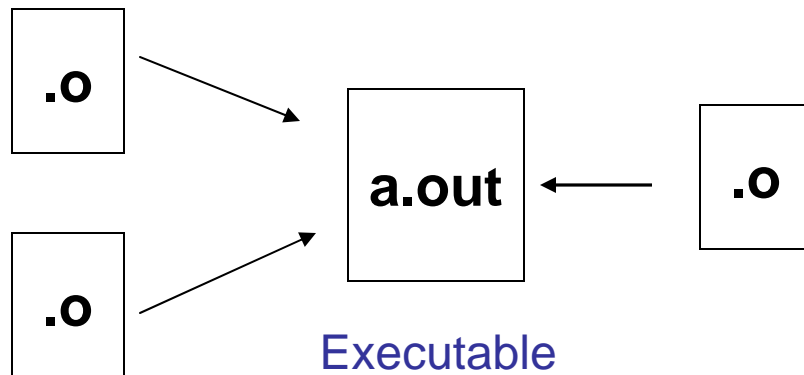
Compiling

- High level \longrightarrow Machine level



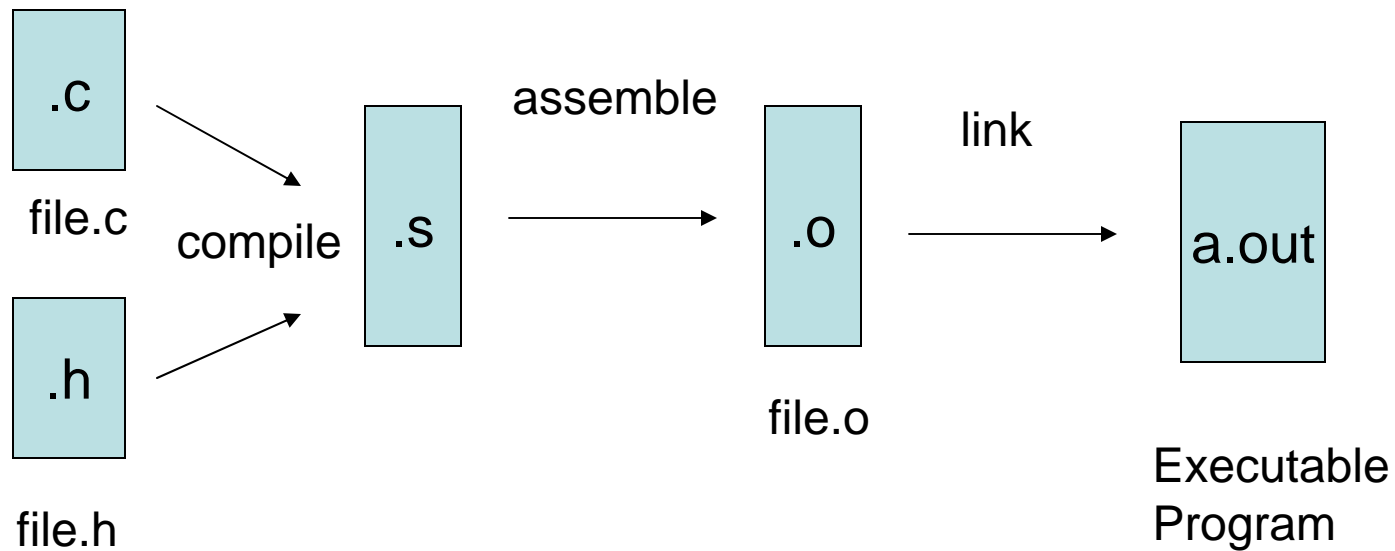
- Looks one file at a time
- Function calls not resolved
- `gcc -c file.c`

Linking



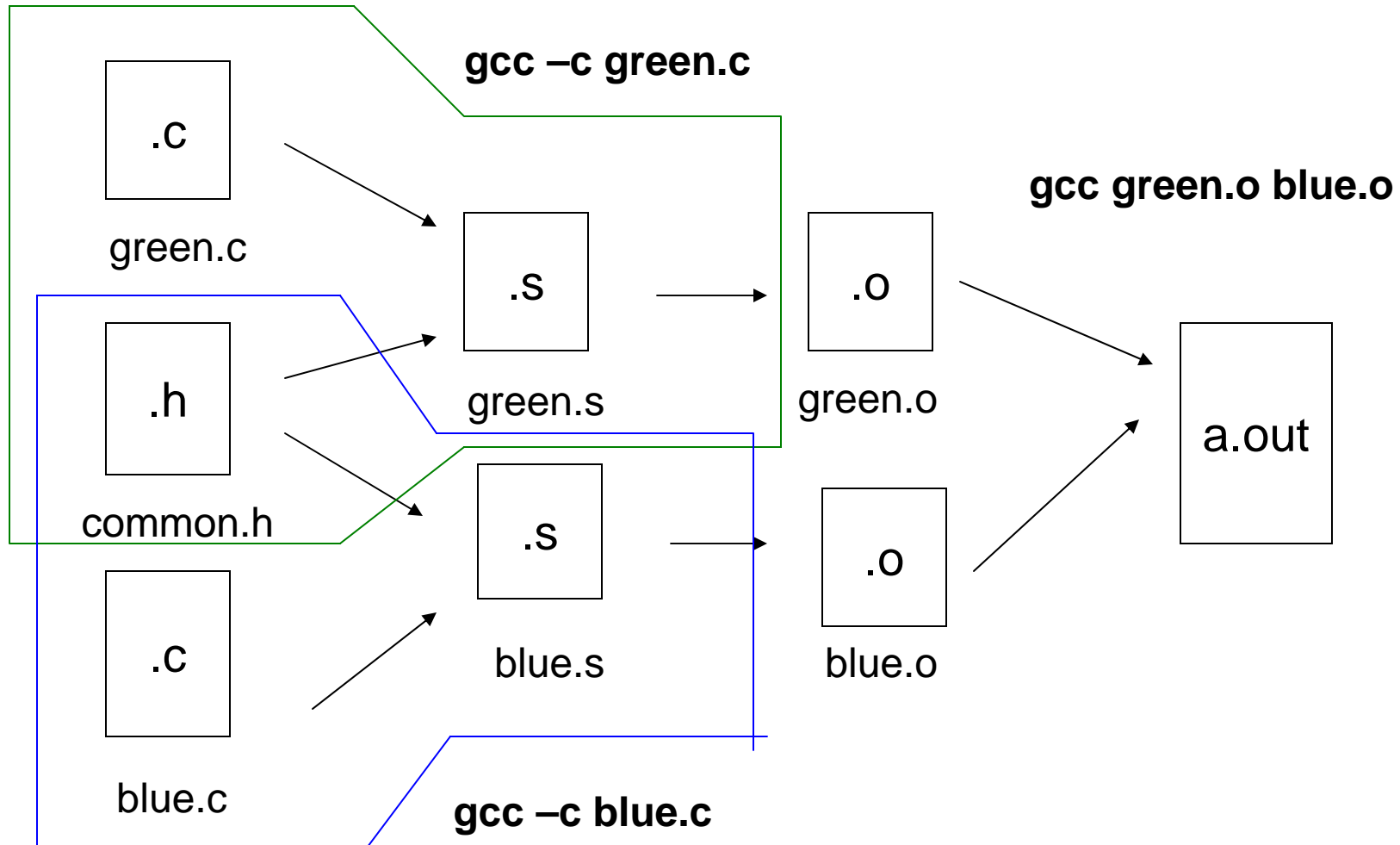
- Many files at a time
- Resolves all cross - references
- `gcc <file1.o> <file2.o> -o <output>`

A simple compilation



Command – gcc file.c

Compiling with several files



Motivation

- Small programs \longrightarrow single file
- "Not so small" programs :
 - Many lines of code
 - More than one programmer
- Problems:
 - Long files are harder to manage
 - Every change requires long compilation
 - Many programmers can not modify the same file simultaneously

Motivation - continued

- Solution : divide project to multiple files
- Targets:
 - Good **division** to components
 - **Minimum compilation** when something is changed

Multiple Source files

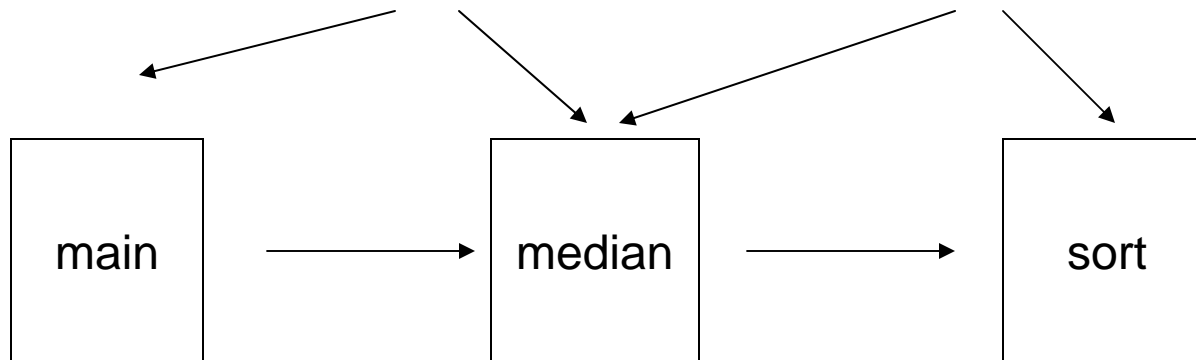
- C Programs - 2 types of files
- .c files :
 - Contain source code and global variable definitions
 - Never included
- .h files :
 - Contain function declarations, struct definitions, # define constant definitions

Project maintenance

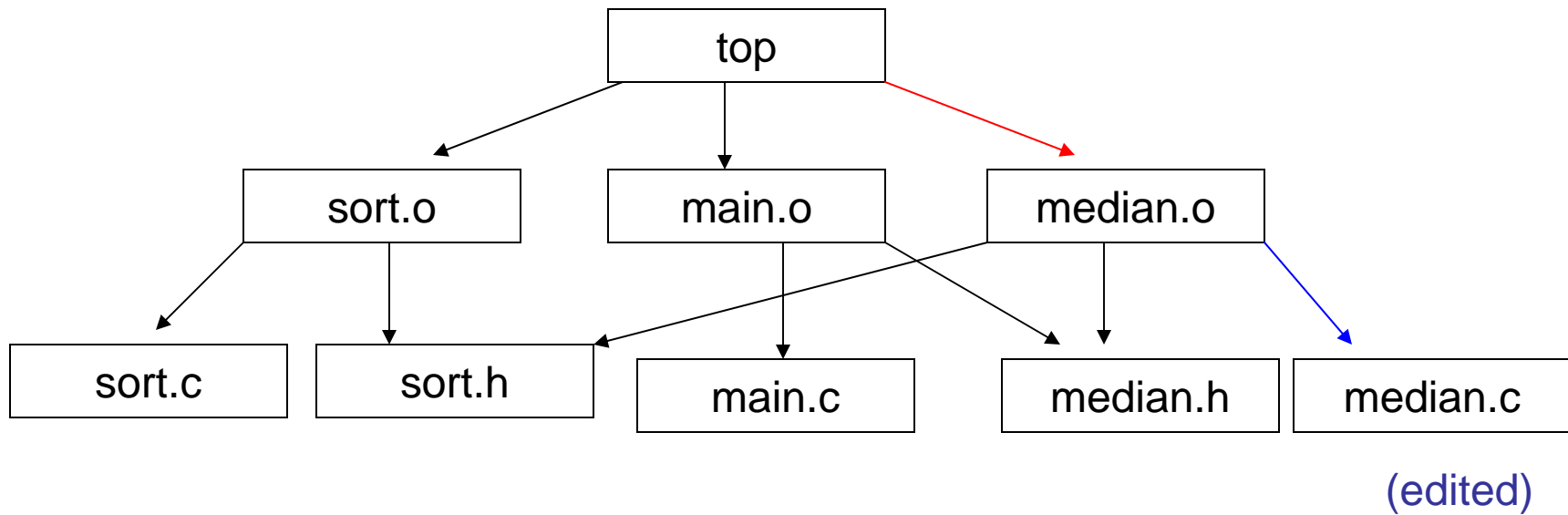
- Done in Unix by the Makefile mechanism
- A **makefile** is a file (script) containing :
 - Project **structure** (files, **dependencies**)
 - **Instructions** for files creation
- The **make** command reads a makefile, understands the project structure and makes up the executable
- Makefile mechanism **not limited to C programs**

Project structure

- Project structure and dependencies can be represented as a graph
- Example given in previous lab session :
 - Program contains 5 files
 - main.c, median.h, median.c, sort.h., sort.c

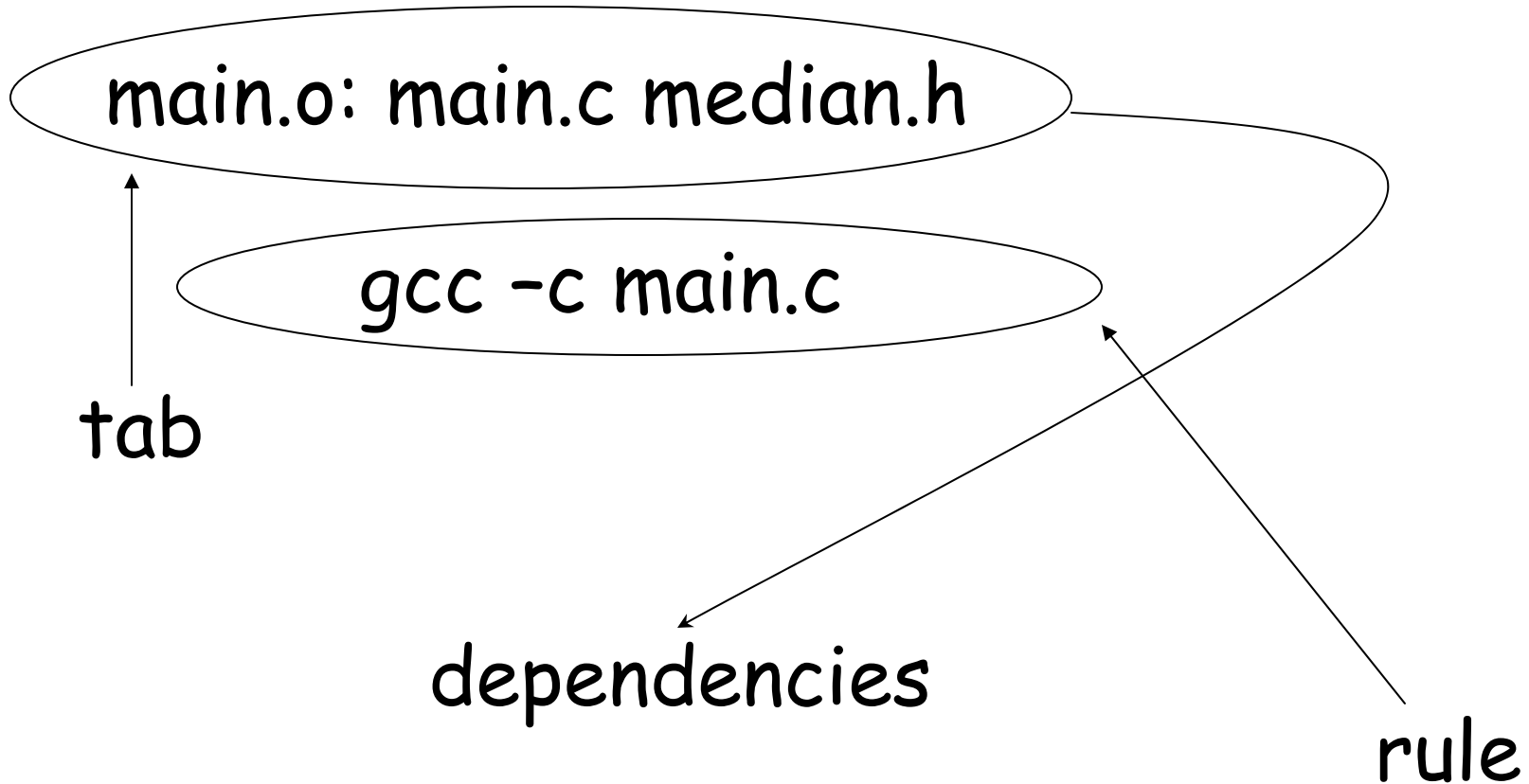


Dependency Graph

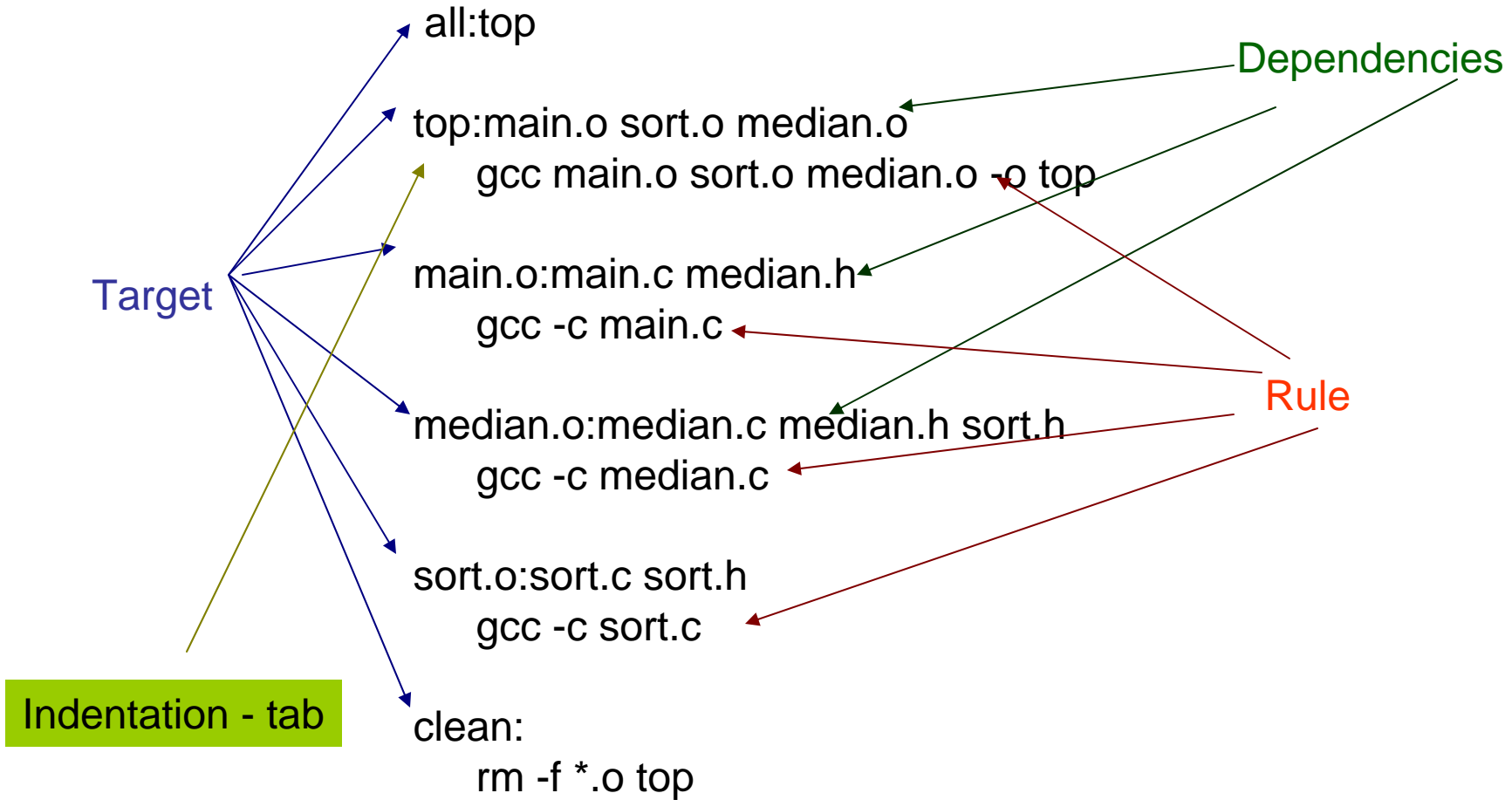


- If median.c is edited
 - gcc -c median.c
 - gcc median.o main.o sort.o -o top

Makefile syntax



Makefile Eg



Makefile (contd)

```
CC=gcc
CFLAGS=-c -Wall

all:top

top:main.o sort.o median.o
    $(CC) main.o sort.o median.o -o top

main.o:main.c
    $(CC) $(CFLAGS) main.c

sort.o:sort.h sort.c
    $(CC) $(CFLAGS) sort.c

median.o:median.h median.c
    $(CC) $(CFLAGS) median.c

clean:
    rm -f *.o top
```

Makefile (contd)

```
CC=gcc  
CFLAGS= -Wall  
OBJS=median.o main.o sort.o
```

```
all:top
```

```
top:$(OBJS)  
median.o:median.h sort.h  
sort.o:sort.h  
main.o:median.h
```

```
clean:  
    rm -f *.o top
```

References

<http://www.eng.hawaii.edu/Tutor/Make/index.html>

<http://www.student.cs.uwaterloo.ca/~isg/res/unix/make/tutorial/>