ECE 244

Make and Makefiles

Introduction to Make

Motivation

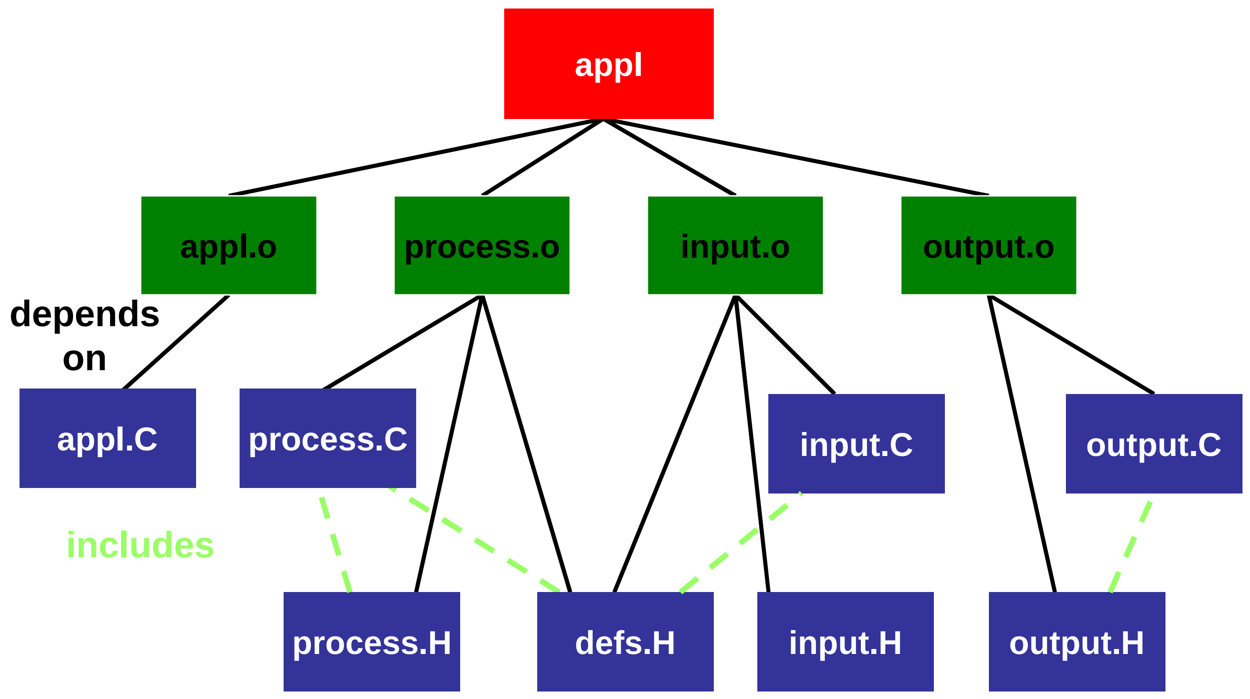
- larger programs involve many files
- hard to keep track of what exactly needs to be recompiled
- inefficient to recompile everything every time
- Need tool to help with the process

Simple Example

- appl.C → appl.o
- input.C includes input.H defs.H → input.o
- output.C includes output.H → output.o
- process.C includes process.H and defs.H → process.o

compiled and linked to create executable "appl"

Make: dependencies



Makefile: specify dependencies

```
appl:   appl.o process.o input.o output.o
```

must start in col 1

```
appl.o: appl.C

process.o: process.C process.H defs.H

input.o: input.C input.H defs.H

output.o: output.C output.H
```

Makefile: ...and command(s)

```
appl:   appl.o process.o input.o output.o
tab g++ –g –o appl appl.o process.o input.o \
                output.o

appl.o: appl.C
tab g++ -g –c –Wall appl.C

process.o: process.C process.H defs.H
tab g++ -g –c –Wall process.C

input.o: input.C input.H defs.H
tab g++ -g –c –Wall input.C

output.o: output.C output.H
tab g++ -g –c –Wall output.C
```

line continuation

Structure of each Rule

- Previous slide lists 5 rules, each with the following format:

$$\text{target: }\overbrace{\text{dependency1 dependency2 ....}}^{\text{0 or more dependencies}}$$

$$\left.\begin{array}{l}\text{command1}\\ \text{command2}\\ \text{....}\end{array}\right\}\text{ 0 or more commands}$$

- A *dependency* is a file that is used as input to create the target. A target often depends on several files.
- A *command* is an action that make carries out. A rule may have more than one command, each on its own line.

Make: Phony targets and comments

If target has no dependencies, its commands will always be executed:

```
# remove object files and other junk
clean:
        rm file1.o file2.o core
        echo cleaning completed
```

phony target

comment

Make: rules with no commands

```
all: lab5 lab5test

lab5: lab5.o LinkedList.o ....
    g++ -o lab5 lab5.o LinkedList.o ...

lab5test: lab5test.o
    g+ -l lab5test lab5test.o
```

Note: no commands

make then makes sure lab5 and lab5test are made

YOU DO NOT NEED TO KNOW THE FOLLOWING INFORMATION (BUT MAY BE INTERESTED IN IT ANYWAYS)

Makefile: …and definitions

```
OBJs = appl.o process.o input.o output.o

appl:   $(OBJs)
 tab g++ -g –o appl $(OBJs)

appl.o: appl.C
 tab g++ -g –c –Wall appl.C

process.o: process.C process.H defs.H
 tab g++ -g –c –Wall process.C

input.o: input.C input.H defs.H
 tab g++ -g –c –Wall input.C

output.o: output.C output.H
 tab g++ -g –c –Wall output.C
```

Make: Inference Rules

Note: these rules are all similar:

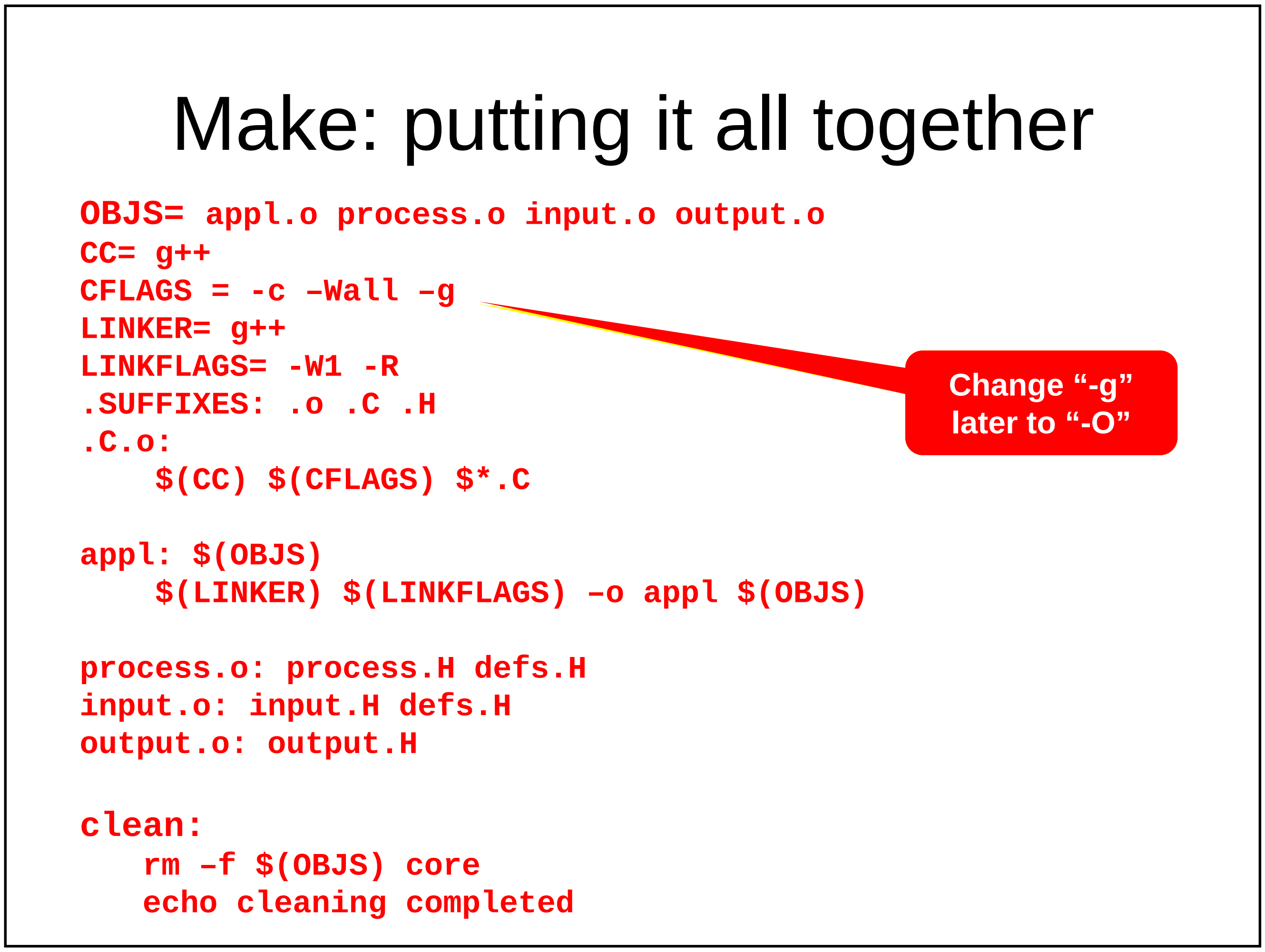
```
process.o: process.C process.H defs.H
    g++ -g –c –Wall process.C

input.o: input.C input.H defs.H
    g++ -g –c –Wall input.C

output.o: output.C output.H
    g++ -g –c –Wall output.C
```

We can define inference rule:

```
.SUFFIXES: .o .C .H
.C.o:
     g++ -g –c -Wall $*.C …
```

.o files depend on .C files

$* is current target without extension



Make: putting it all together

```
OBJS= appl.o process.o input.o output.o
CC= g++
CFLAGS = -c –Wall –g
LINKER= g++
LINKFLAGS= -W1 -R
.SUFFIXES: .o .C .H
.C.o:
    $(CC) $(CFLAGS) $*.C

appl: $(OBJS)
    $(LINKER) $(LINKFLAGS) –o appl $(OBJS)

process.o: process.H defs.H
input.o: input.H defs.H
output.o: output.H

clean:
   rm –f $(OBJS) core
   echo cleaning completed
```

Change “-g” later to “-O”

Make: Execution

```
make [options] [target-rule] [target-rule] …
```

- Make looks for file `Makefile` or `makefile`
- Make executes each specified target-rule
- If no target-rule specified, make executes first target in makefile
- Make echos each command it encounters

Make: the Algorithm

```
Let M be a makefile and X be a target file

If M does not have a rule for making X then
    if a file called X already exists
        then there is nothing to do
        else report error
else
  choose the first rule for making X
    make each dependency for that rule
    if X exists and is newer than each dependency
        then report “X is up to date”
        else make X by executing the commands
end
```