A1 remarks requests due next Monday

Midterm next Wednesday

Rooms will be posted on Website
hi, any way to discourage people from posting without searching the forum? it's super frustrating -- thanks!

course is very enjoyable and interesting so far.

Response:

Frustrating for me too! Especially when I was trying to keep up with the posts via a slow New Zealand connection. Peer encouragement/reprimand works best :)

Anonymous Says:
Anonymous Says:

Sorry, these directions (to lab 6) are too ambiguous.

Response:

Ambiguous means “is open to more than one interpretation”. Please tell me which statements you find have a double meaning and what those two (or more) interpretations are. Then I can clear up the instructions.
Makefiles

- Makefiles were originally designed to support separate compilation of C files.

\[
\text{CFLAGS} = \text{-Wall -g -std=c99 -Werror}
\]

- \text{friendme}: friendme.o friends.o
  \[
  \text{gcc $(CFLAGS) -o friendme friendme.o friends.o}
  \]

- \text{friendme.o}: friendme.c friends.h
  \[
  \text{gcc $(CFLAGS) -c friendme.c}
  \]

- \text{friends.o}: friends.c friends.h
  \[
  \text{gcc $(CFLAGS) -c friends.c}
  \]

- \text{clean:}
  \[
  \text{rm friendme *.o}
  \]
Terminology

- Target
  - reverse : reverse.c
  - gcc -Wall -o reverse reverse.c

- Prerequisite(s)
- Rule
  - May be many prerequisites
  - Rule may have many actions (one per line)
Running make

• `make`
  – with no options looks for a file called Makefile, and evaluates the first rule

• `make query`
  – Looks for a file called Makefile and looks for a rule with the target `query` and evaluates it.
How it works

• Make looks at the when the target and its prerequisites were last modified
  – It assumes targets are files and checks the dates of the files

• Make does nothing…
  – If the target exists, and
  – Is more recent than all its prerequisites

• Make executes the actions …
  – If the target doesn’t exist, or
  – If any prerequisite is more recent than the target

• Make recursively builds prerequisites FIRST
Variables

CFLAGS= -g -Wall -std=c99

reverse : reverse.c

gcc ${CFLAGS} -o reverse reverse.c

Make defines variables to represent parts of rules

<table>
<thead>
<tr>
<th>$@</th>
<th>Target</th>
</tr>
</thead>
<tbody>
<tr>
<td>$&lt;$</td>
<td>First prerequisite</td>
</tr>
<tr>
<td>$?</td>
<td>All out of date prerequisites</td>
</tr>
<tr>
<td>$^</td>
<td>All prerequisites</td>
</tr>
</tbody>
</table>
• Another example Makefile

FLAGS = -g

all : query printlog

query : query.o message.o queue.o
    gcc $(FLAGS) -o $@ $^

printlog : printlog.o message.o queue.o
    gcc $(FLAGS) -o $@ $^

# Separately compile each C file
%.o : %.c message.h
    gcc $(FLAGS) -c $<

clean :
    -rm *.o query
Pattern rules

Most files are compiled the same way
So write a pattern rule for the general case
%.o : %.c

gcc ${CFLAGS} -c $<

Use % to mark the stem of the file’s name
Like using * in commands in Unix
-c flag in gcc does compilation of file without linking.
• Another example Makefile

FLAGS = -g

all : query printlog

query : query.o message.o queue.o
    gcc ${FLAGS} -o $@ $^

printlog : printlog.o message.o queue.o
    gcc ${FLAGS} -o $@ $^

# Separately compile each C file
%.o : %.c message.h
    gcc ${FLAGS} -c $<

clean :
    -rm *.o query
Multiple Targets and Phony Targets

• Often you want one command to build a number of other targets
  
  all : query printlog
  
  printlog : ...
  ...
  query : ...

Or targets aren’t building anything

 clean:
  
  rm -f *.o query printlog