CS162: Introduction to Computer Science II

Makefile

Outline

1. Basic terminology
2. Typical contents
   - Rules
   - Targets
   - Dependencies
   - Variables
The basics

- Type in “make” – no targets specified
- “make” searches for makefile or Makefile by default

# This is a comment

target: dependencies
    rule to build

Must be a tab (not a space) otherwise it doesn’t treat it like a rule
The basics

Sample makefile:

```makefile
default: program1.cpp functions.hpp functions.cpp
    g++ -std=c++0X program1.cpp functions.cpp -o program1
```

To run the makefile, just type “make”

The basics

This is the name of the target

If any of these dependencies (ie. the three files) changes, it will rebuild the target (ie. default)

```makefile
default: program1.cpp functions.hpp functions.cpp
    g++ -std=c++0X program1.cpp functions.cpp -o program1
```
The basics

• If any of the dependencies have timestamps newer than the target, it rebuilds the target
• In our previous slide, the target was called default which is not the name of the executable
• If you type make, it will always recompile
• To make it check timestamps, change the target to be the executable name ie. program1

```
program1: program1.cpp functions.hpp functions.cpp
        g++ -std=c++0X program1.cpp functions.cpp -o program1
```

• When you type “make” the first time, it compiles everything
• When you type “make” the second time it will say program1 is up to date
Variables

CXX = g++
CXXFLAGS = -std=c++0x
OBJS = functions.o lecture1.o
SRCS = functions.cpp lecture1.cpp
HEADERS = functions.hpp

program1: program1.cpp functions.hpp functions.cpp
  g++ -std=c++0X program1.cpp functions.cpp -o program1

• You can declare variables in makefiles and use them
• (note that I’ve only declared them above and haven’t actually used them yet)

Variables

Some standard C++ makefile variables:
• CXX (C++ compiler)
• CXXFLAGS (C++ compiler flags)
• LDFLAGS (Linker flags)
• OBJS (Object files)
• SRCS (Source files)
• HEADERS (Header files)
Variables

CXX = g++
CXXFLAGS = -std=c++0x
LDFLAGS = -lboost_date_time
OBJS = functions.o lecture1.o
SRCS = functions.cpp lecture1.cpp
HEADERS = functions.hpp

program1: $(SRCS) $(HEADERS)
    $(CXX) $(CXXFLAGS) $(SRCS) -o program1

- To use a variable, you need to use the $ sign with braces or parentheses eg. ${Variable Name} or $(Variable Name)
- Notice how short and generic this makefile is!

Variables

- You can use += to concatenate to a variable eg.
  
  CXXFLAGS = -std=c++0x
  CXXFLAGS += -wall
  CXXFLAGS += -pedantic-errors

  Turn on all warnings

  Strictly enforce the C++ standard
Variables

# I removed the variables to fit this slide on the page

```
program1: ${OBJ} ${HEA}
    ${CXX} ${LDFLAGS} ${OBJ} -o program1

${OBJ}: ${SRCS}
    ${CXX} ${CXXFLAGS} -c ${(@:.0=.cpp)}
```

- Previously, if any source or header file changed, all of them were recompiled
- By adding a rule for the object files, only the source or header file that changed will be recompiled
- @(:.o=.cpp) is an implicit rule that says the target is a .o file and the source is a .cpp file

Touch

- Side note: if you want to update the timestamp on a file without doing anything to it, use “touch”
- Eg. touch functions.cpp
- This make the Makefile think it needs to recompile targets that depend on functions.cpp
Targets

• Typing “make” executes the first target it finds (which is the default)
• If you want to execute a specific target eg. foo, you need to type in “make foo”
• And introduce a target into your makefile
• To see this with the example Makefile, type in:
  touch functions.o
  make functions.o