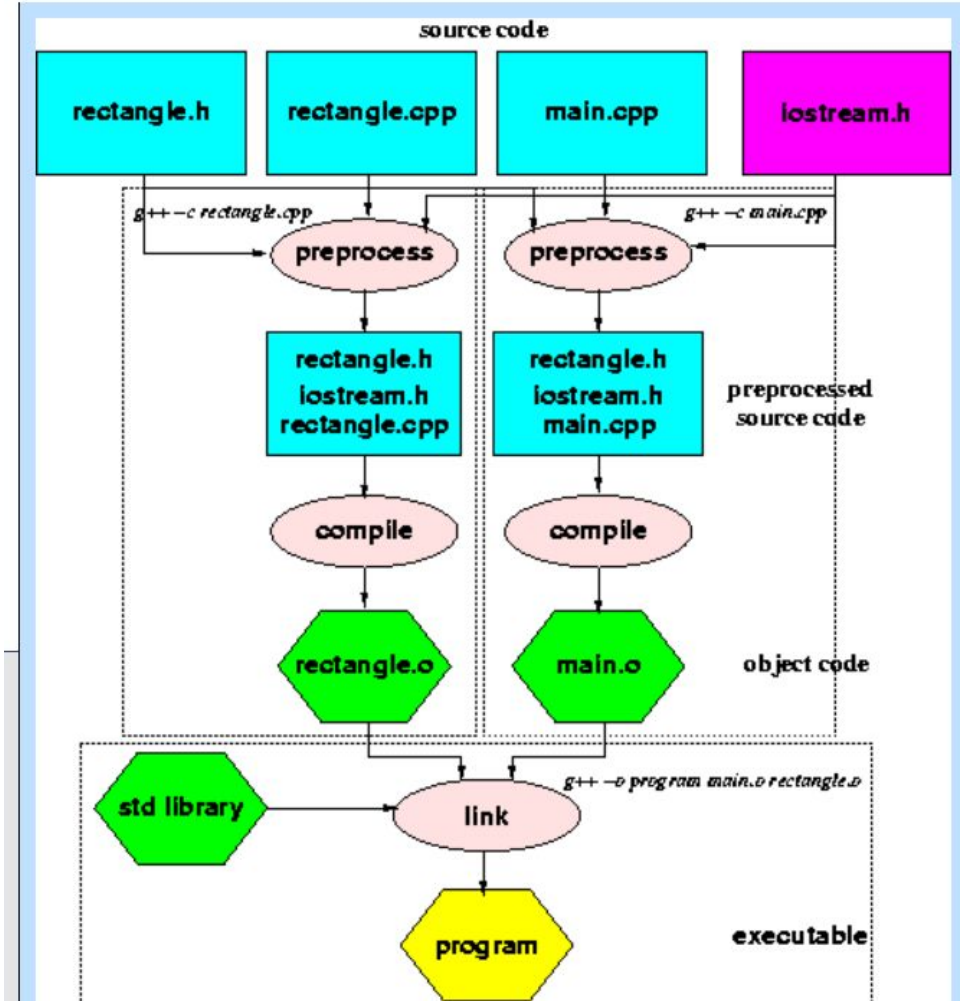
make

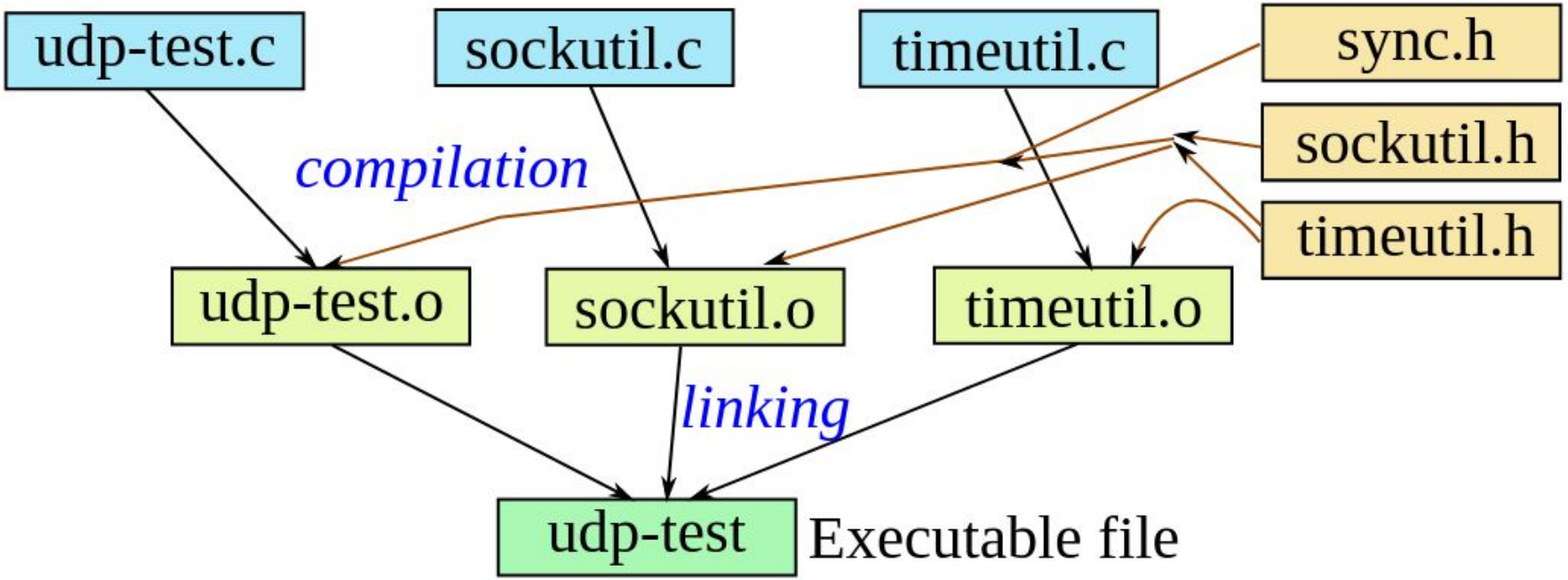
Kameswari Chebrolu

Compiling C/C++

- Popular : gcc (for C) and g++ (for C++)
  - Cc can refer to either, depends on system
- Source files: header files (.h, .hpp) and compilation units (.c, .cpp)
  - header file contains shared declarations
  - non-header files contains definitions and local (non-shared) declarations


source code
rectangle.h
rectangle.cpp
main.cpp
iostream.h
g++ -c rectangle.cpp
g++ -c main.cpp
preprocess
preprocess
rectangle.h
iostream.h
rectangle.cpp
rectangle.h
iostream.h
main.cpp
preprocessed
source code
compile
compile
rectangle.o
main.o
object code
std library
g++ -o program main.o rectangle.o
link
program
executable

Motivation Example



Makefile: a specification of the dependence DAG

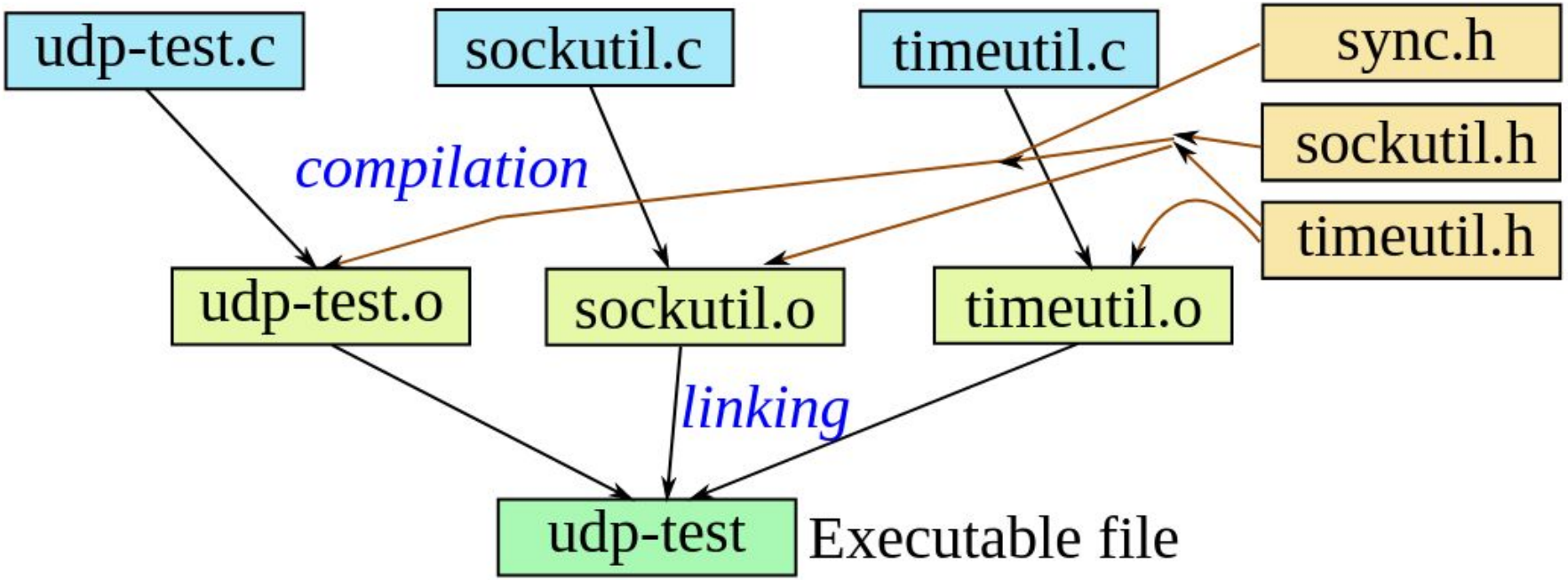
Without Makefile

- Scenario-1: Initial Compilation and Linking
  - cc -o udp-test udp-test.c sockutil.c timeutil.c
- Scenario-2: After modifying timeutil.c
  - cc -o udp-test udp-test.c sockutil.c timeutil.c

Compilation of udp-test.c and sockutil.c is unnecessary!

- Big projects have many files
- Makefile: simplifies project management
  - Minimum compilation when something is changed
    - Especially important for large projects, where compilation takes many minutes to hours
  - More error free compared to manual compiling
  - Help others build the project from source code easily
  - Gives a good overview of the project structure and dependencies

The Dependence DAG (Directed Acyclic Graph)



Makefile: a specification of the dependence DAG

Syntax

target: dependencies
    command
    command
    command

- Target: mostly name of a file that is generated by a program
  - E.g. executable or object files
  - Phony Targets also there, will cover later
- Dependencies: file names separated by spaces, need to exist before the commands for the target are run
  - target ‘clean’ does not have dependencies.
- Command: an action that needs to be carried out
  - Need to start with a tab character, not spaces (for some obscure reason)

Variables

- Variables are strings and assigned values via =
  - E.g. OBJ = udp-test.o sockutil.o timeutil.o
- Can reference variables using either ${} or $()
  - $(OBJ)

Implicit Rules

- Implicit rules: do not have to provide too much detail
  - E.g. in C, compilation takes a .c file and makes a .o file
    - No need to specify the command
    - Make applies the implicit rule when it sees this combination of file name endings

Phony Target

- Phony target: name for some commands to be executed (not name of a file)
  - a target of “.PHONY” will prevent Make from confusing the phony target with a file name

Handling errors

- Add -k when running make to continue running even in the face of errors
  - Helpful if you want to see all the errors of Make at once
- Add a - before a command to suppress the error

Naming of Makefile

- make commands looks for a makefile
  - Tries the following names in order: `GNUmakefile', `makefile' and `Makefile'
- Use a different name, use make -f

Cmake (not in syllabus)

- Makefile helps in some automation. Can we do better?
  - Can we have a tool that writes makefiles itself ?
  - Also, in the process make it compiler independent
    - C++ needs different compilers (and options) for different platforms
- CMake: open-source, cross-platform family of tools designed to build, test and package software
  - Supports compiler independent configuration files and generate native makefiles

References

https://makefiletutorial.com/#getting-started

https://www.gnu.org/software/make/manual/make.html (in depth)