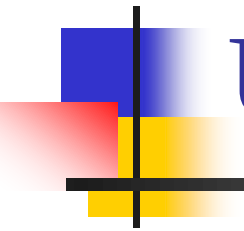
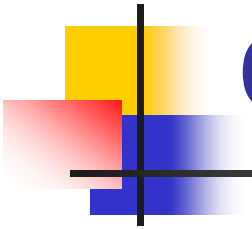
Lecture 02

Write Your First Program on UNIX

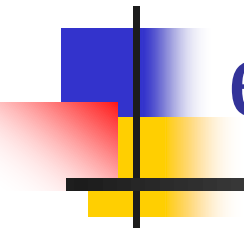


How to use compiler and debugger



Outline of today

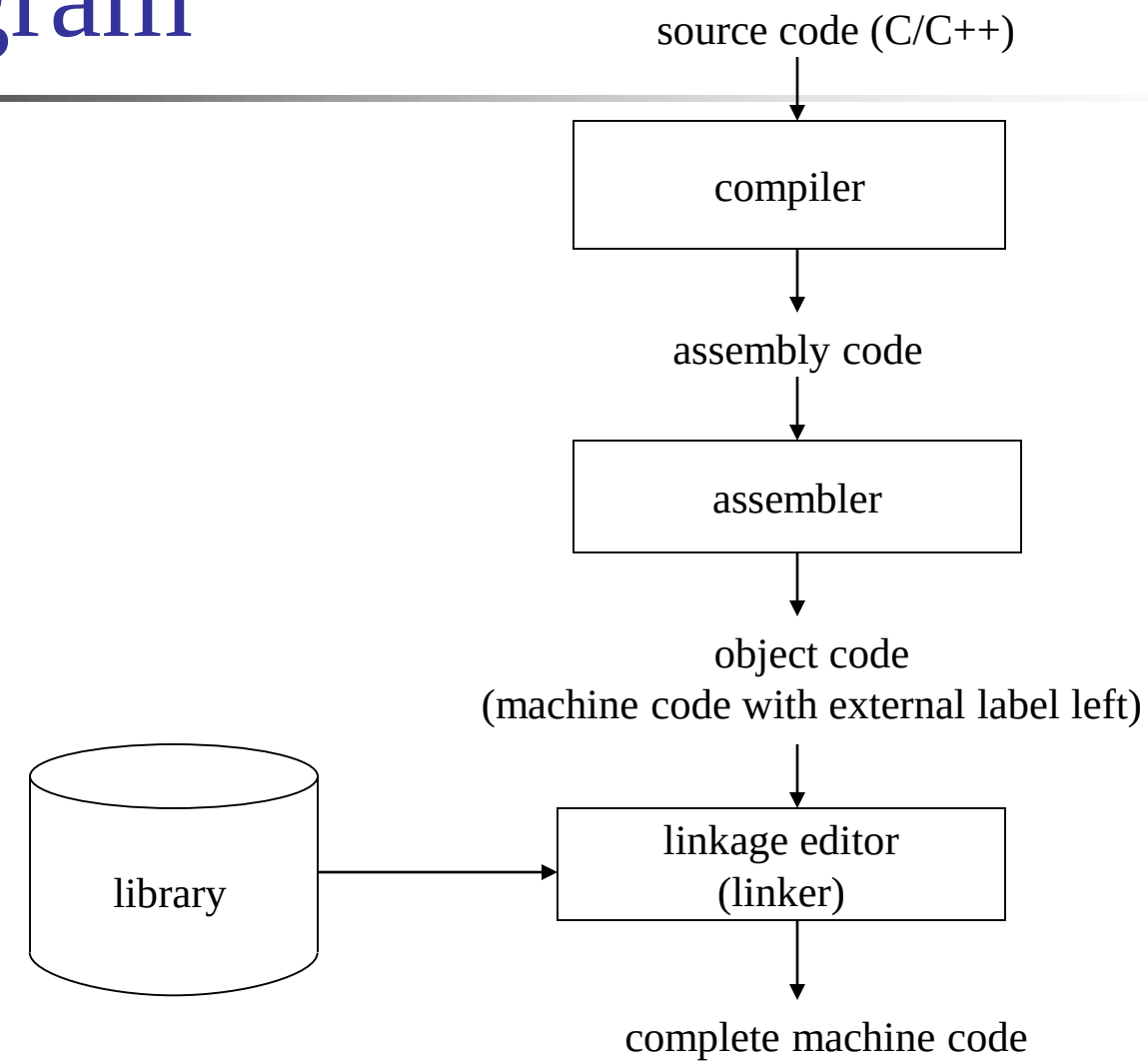
- How to compile a program – recall what you learned in system programming
- Program compilation flow on GNU/Linux
- How to debug your program
- How to compile multiple programs? the Make
- On-class exercise: write your first program



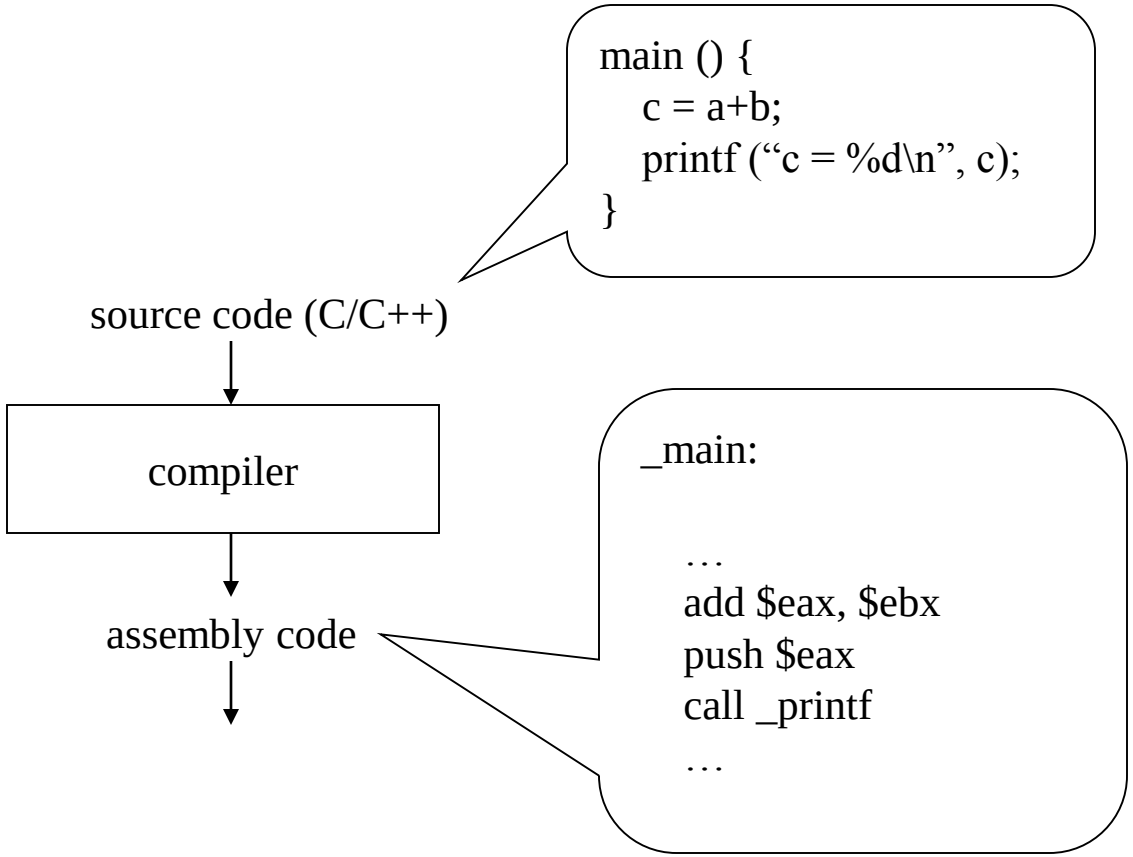
How to compile a program to executable image

Recall what you have learned in system programming

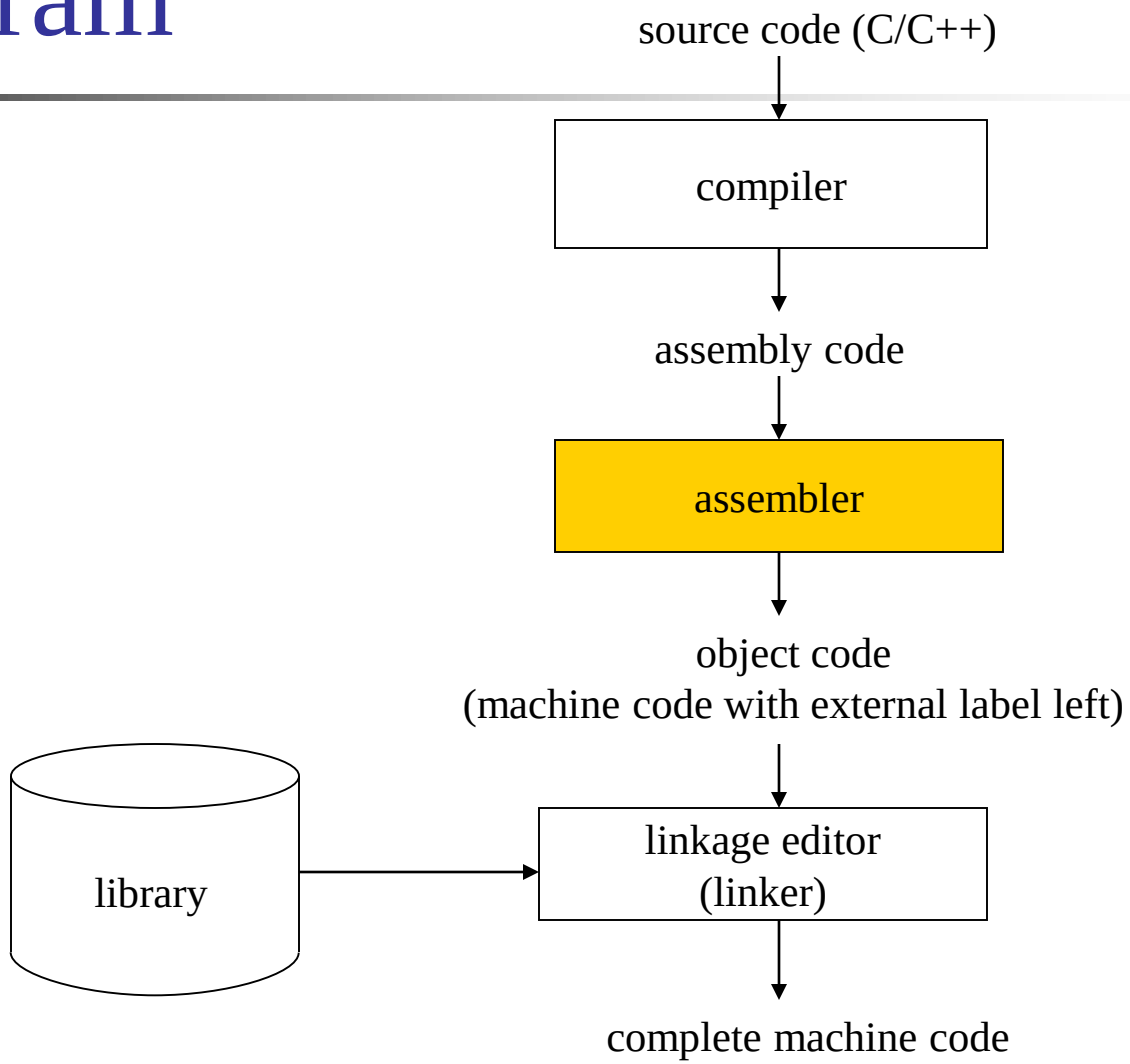
General flow of compiling a program



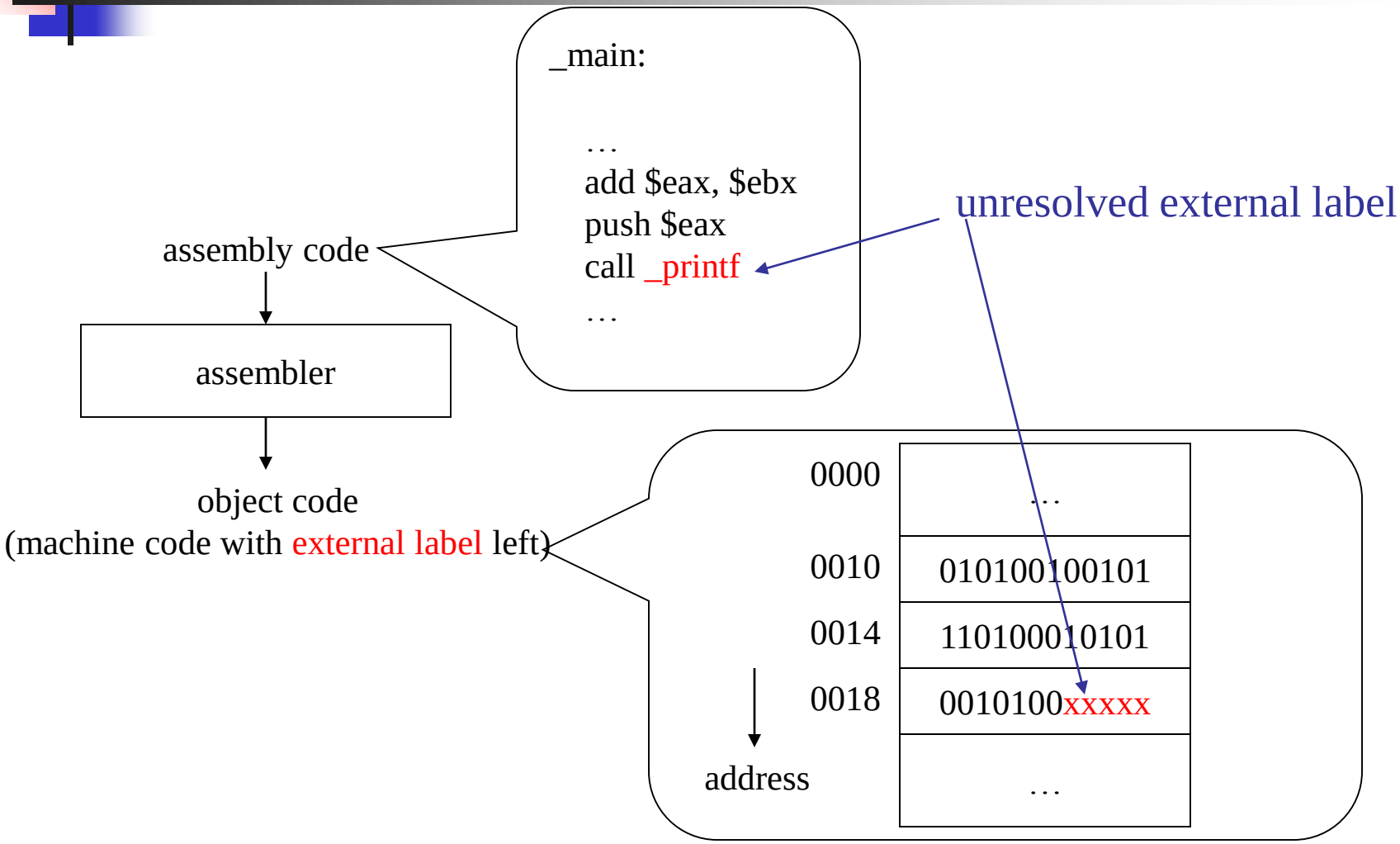
Compile source code to assembly



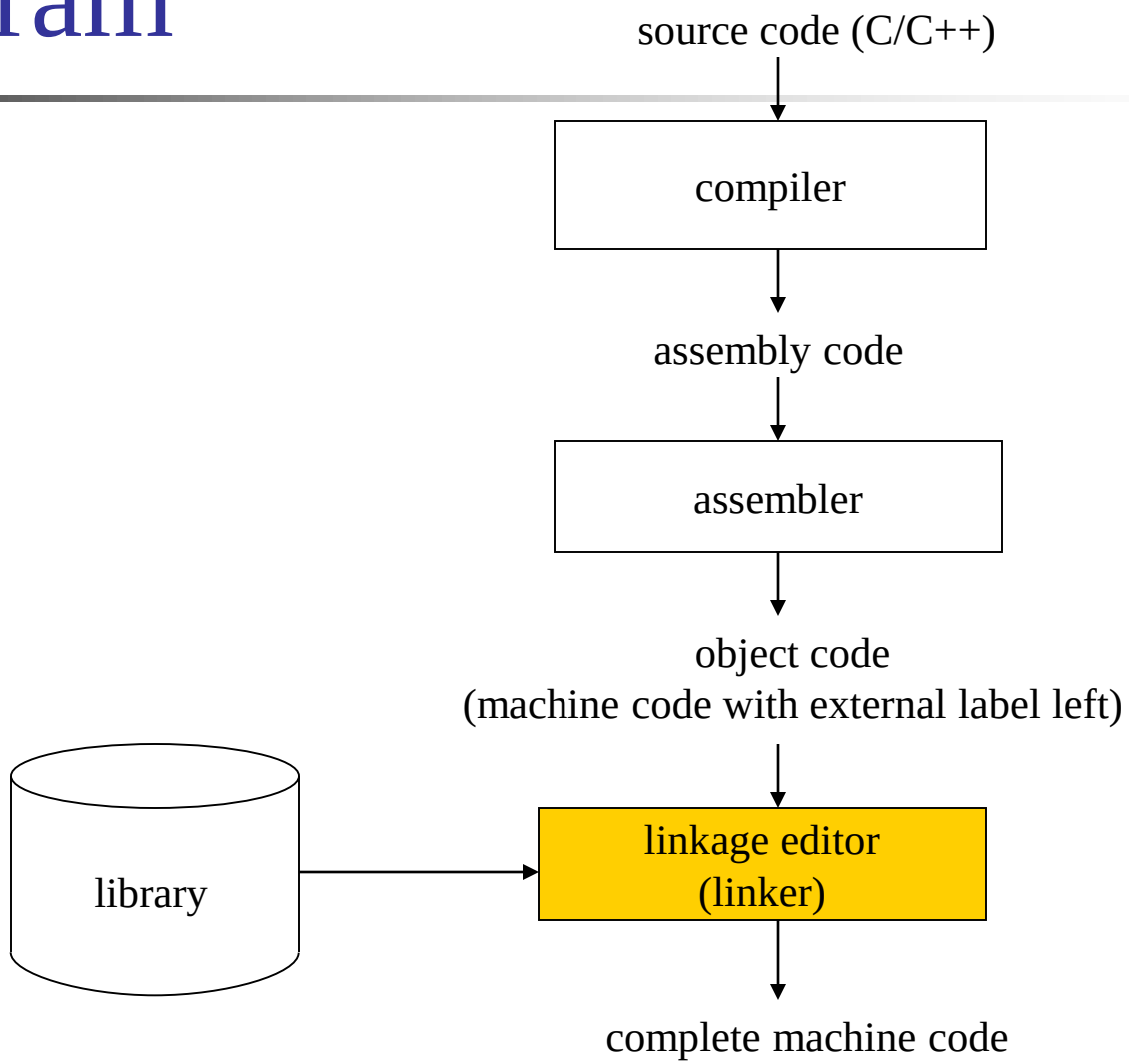
General flow of compiling a program



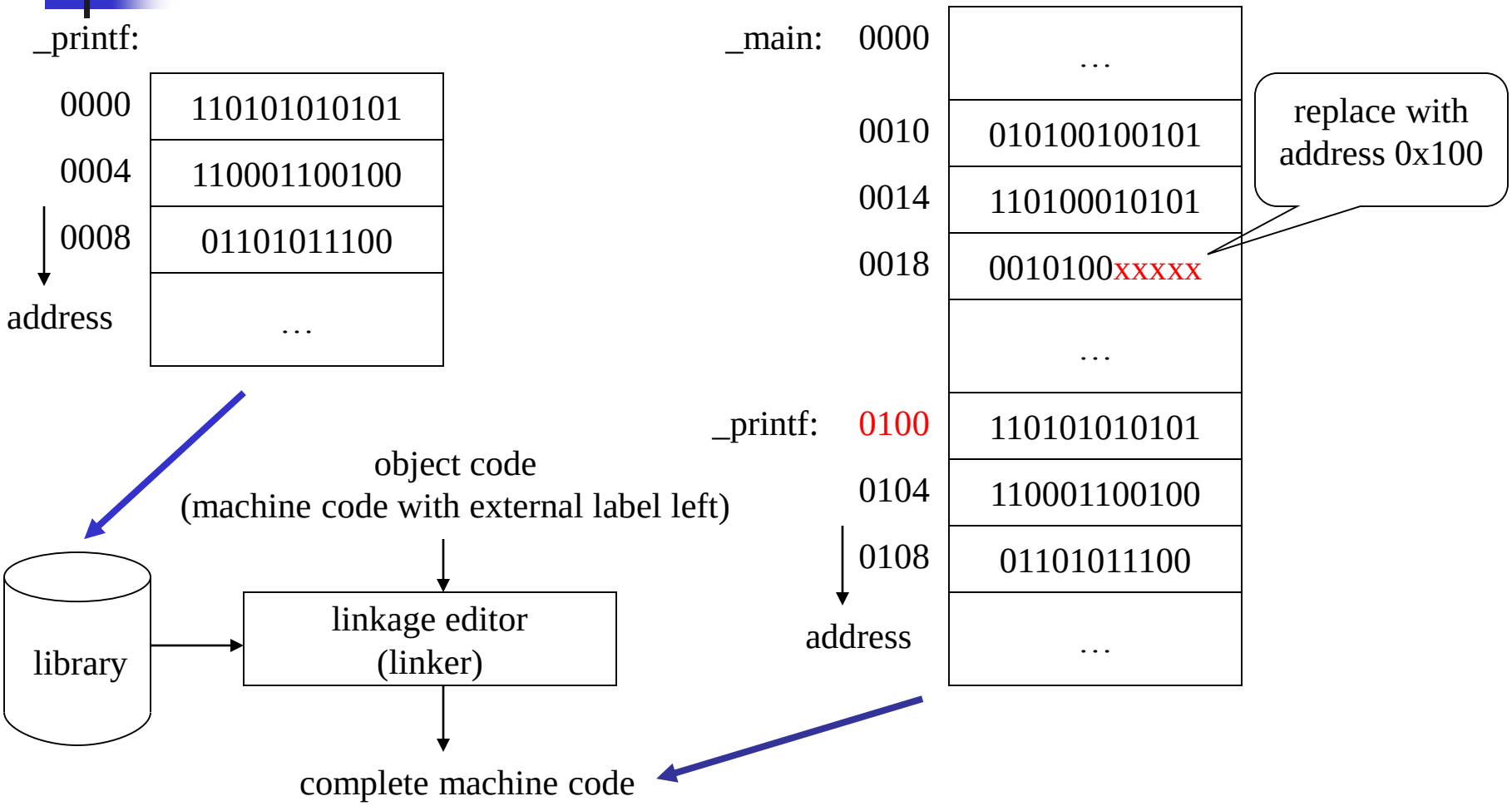
Assembly to object code

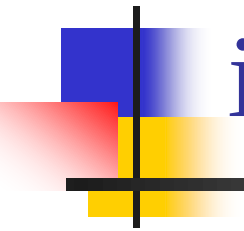


General flow of compiling a program



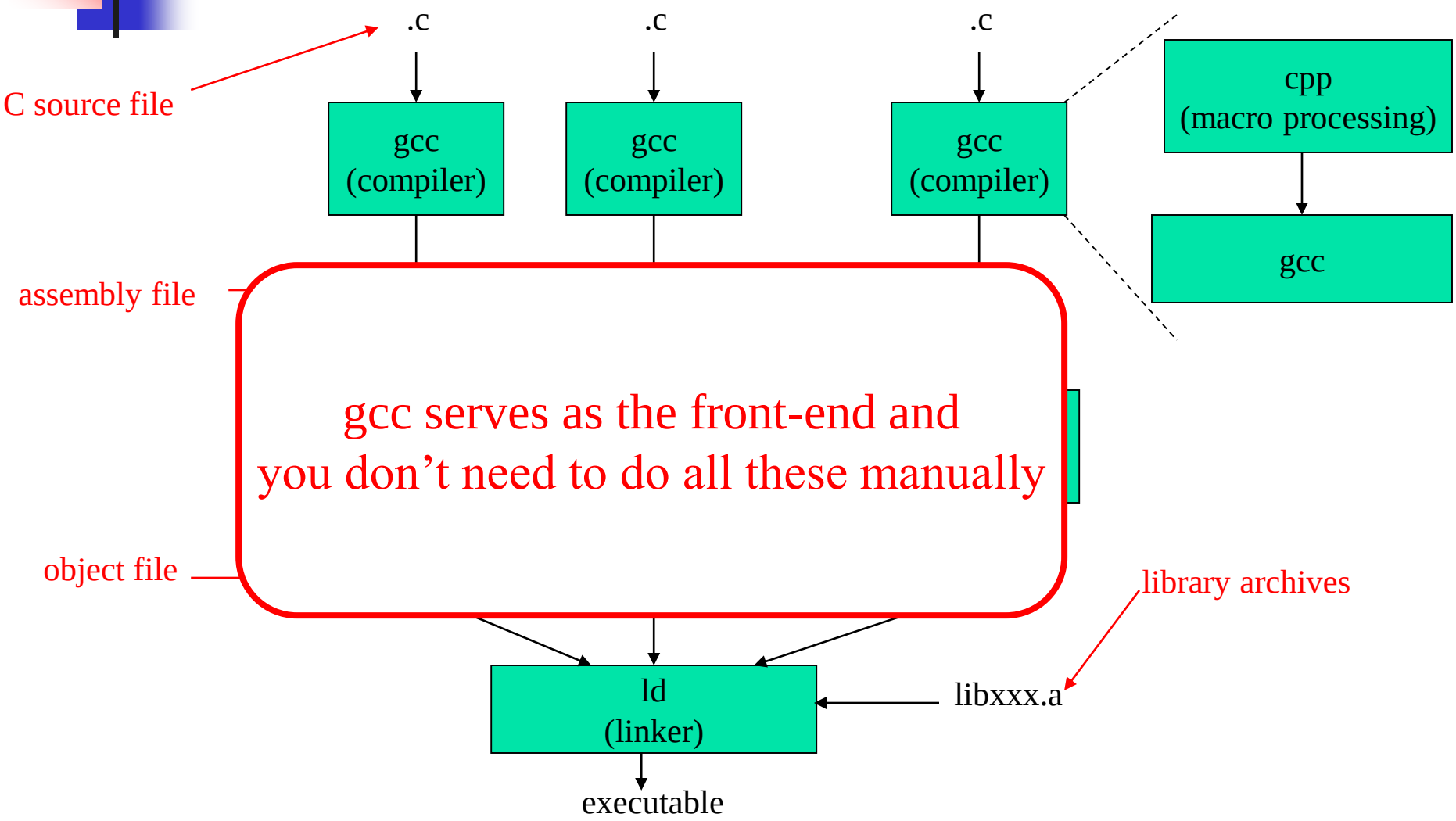
Linking: resolving relocation symbols

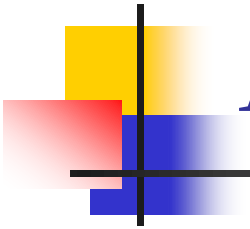




Let’s see how this concept works in GNU/Linux

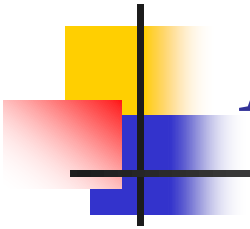
Detailed Flow to Build a Program





A simple example

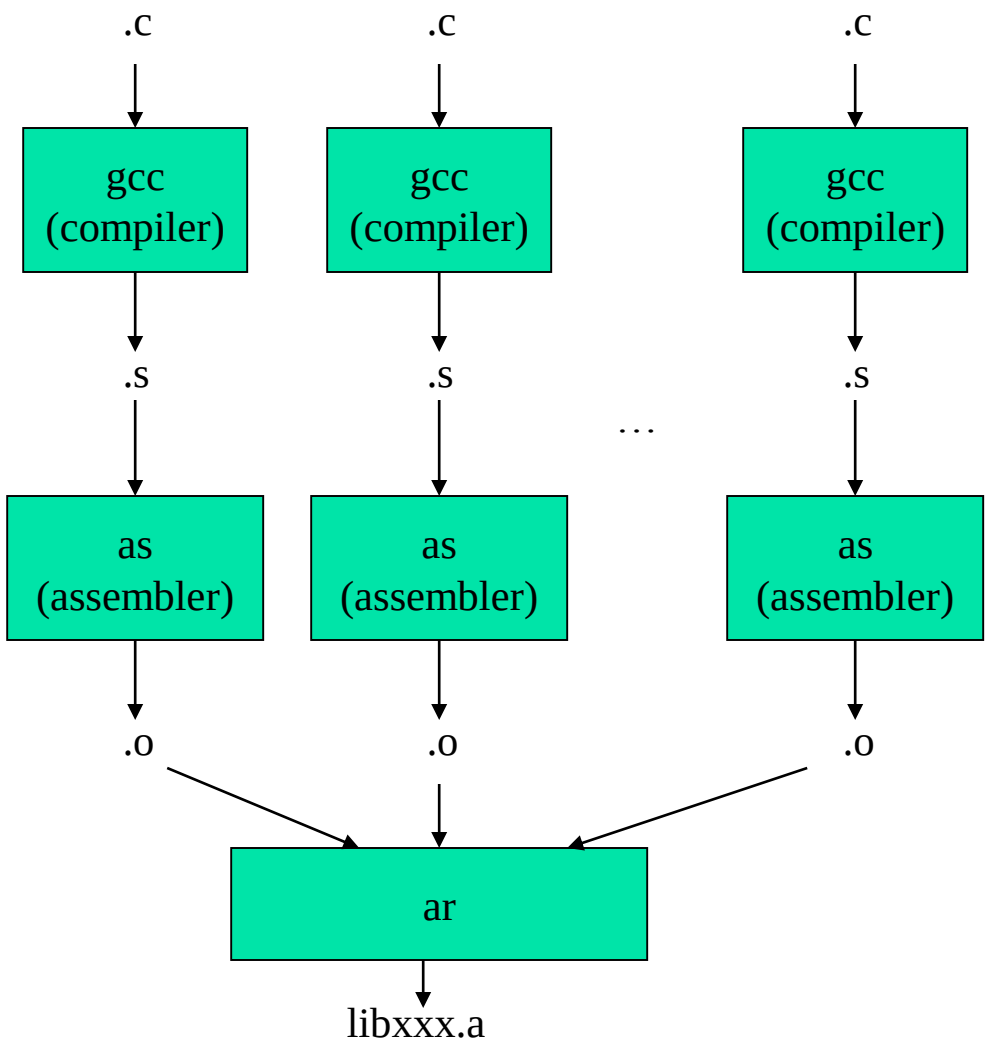
- demo/gdb_example

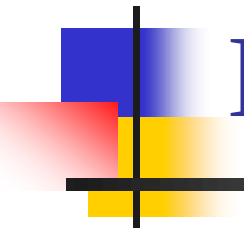


A multi-file example

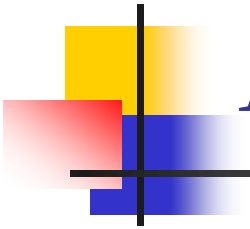
- demo/make_multifile

Build your own library





Debug Your Program

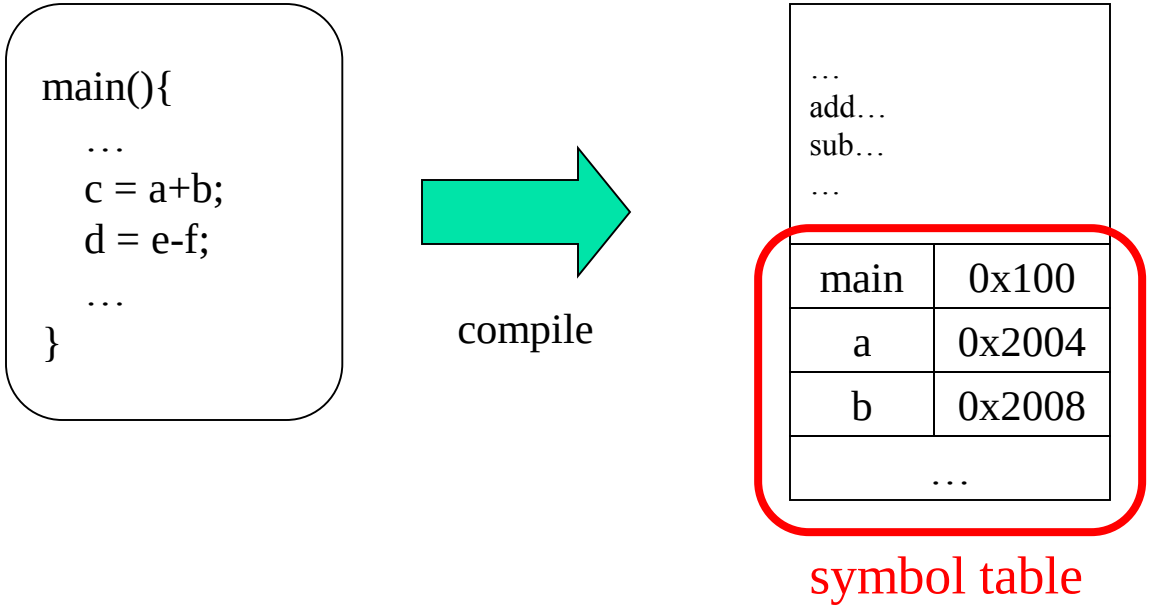


A simple program to debug

- use gcc to compile with –g
  - add debug information into the object code
- execute the program with a debugger
  - command line: *gdb*
  - GUI:…many…

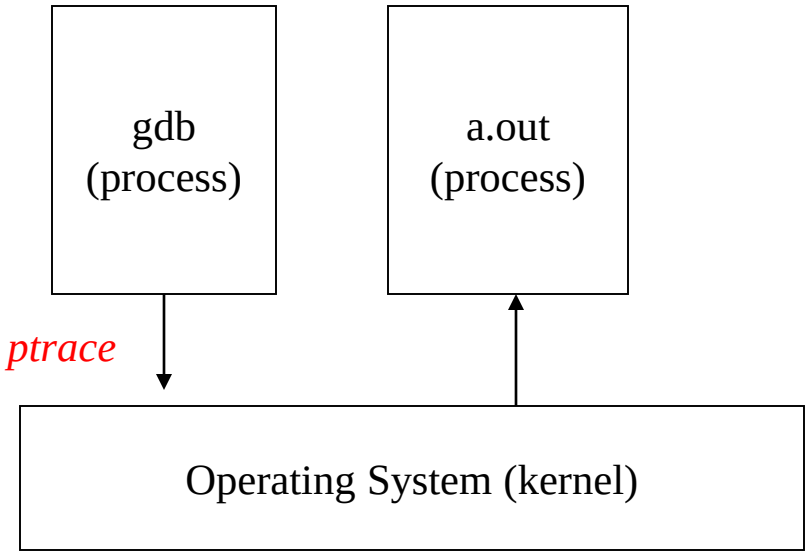
How the debugger knows program/variable location?

- program annotation

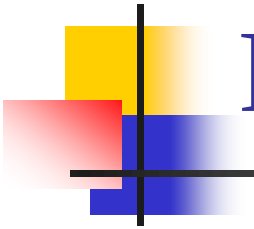


How debugger works at run-time?

- run-time poke/peek/set break points to user process space



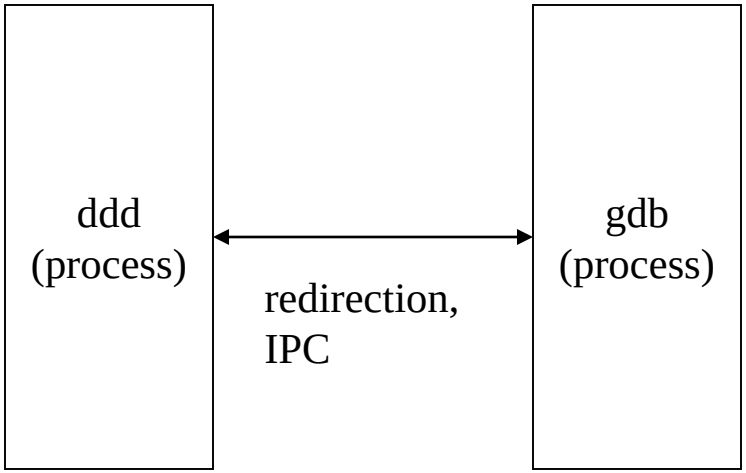
Pretty GUI for Debugging



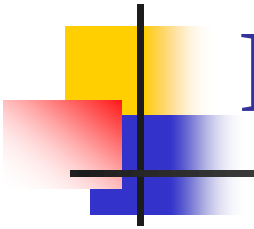
- ddd
- kdbg
- Integrated Development Environment (IDE):
  - KDevelop
  - Eclipse

- Remark: use *yum* to install the lost programs (for Fedora only!)

Structure of debugging with GUI

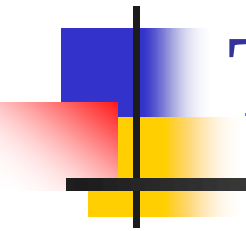


- through inter-process communication, stream redirection, etc.



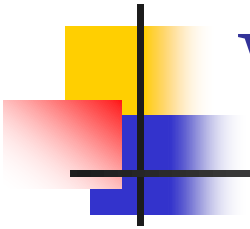
Practice by yourself

- establish how a GUI debugger works after you finish the course
  - process
  - virtual memory management
  - system call
  - inter-process communication
- try to write a debugger after you finish the course



The MAKE

useful utility to write a large program



What is MAKE?

- an utility to manage the compilation of a large-scale program

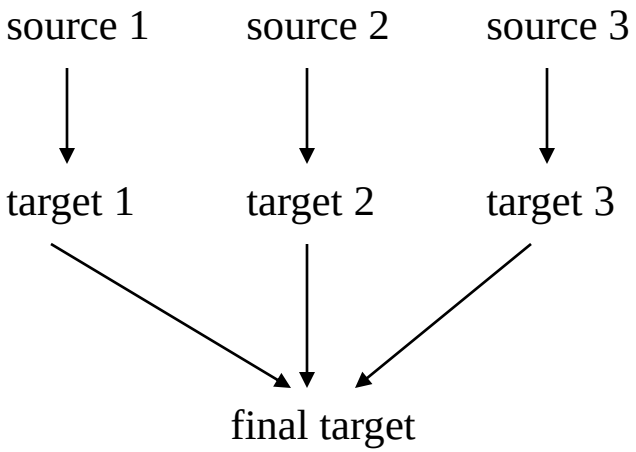
- image that you have a program with 100 C source files
  - type the 100 file names each time you want to compile it?

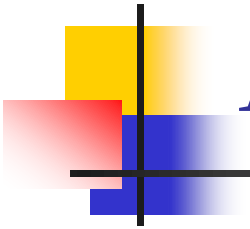
Language of MAKE

- describe dependences and rules to build targets
- Grammer:

target: *source1 source2 …*
commands_to_produce_target

- auto-check target dependence and perform required operations only



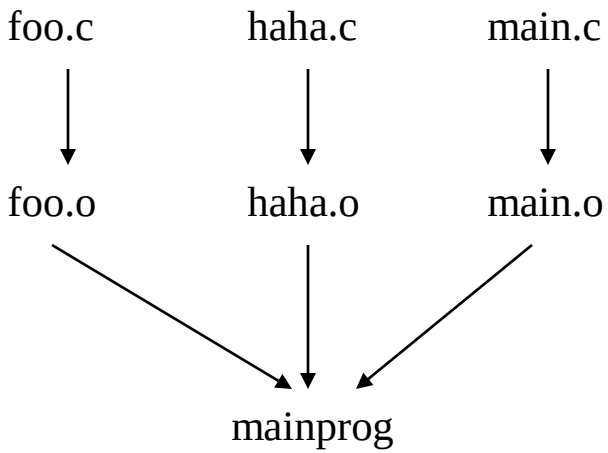


A simple make example

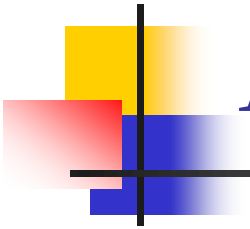
- demo/gdb_example

A Multi-File MAKE Example

- demo/make_multifile



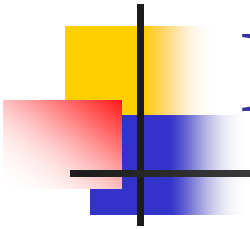
Advanced MAKE script writing



- General compilation rules
  - Example: to compile all C code (.c) to object code (.o)

```
%.o: %.c
        gcc –g –c $^ $@
```

- Variables: (example)
  - C_SRC_FILES = test.c foo.c
- String manipulation functions: (example)
  - C_SRC_FILES = $(wildcard *.c)
- please “info make”



In-Class Exercise

- write a program calling a function foo(a,b)
  - foo returns some computation on a,b (e.g. a+b)
- print the result of foo(a,b) in the main() function
- the main() and foo() function has to be in separate files
- create a Makefile to do all the compilation
  - rule to generate assembly program (e.g. make foo.s)
  - rule to build executable program with debug support
  - trying to run your program with a debugger

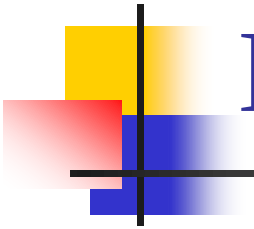
Frequently used GCC options

- *-S*: assemble only
- *-c*: stop after building the object file (.o)
- *-g*: add debug information into the object file
- *-o target_file_name*: specify the output file name

```
gcc –o mainprog test.c
```

Homework #1 (due10/31)

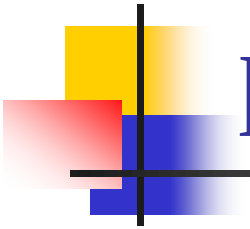
- Paper work and on M$-Windows
- Write a simple C/C++ program with macros on a separate .h file
- Build the executable file of the program
  - Using M$ Visual Studio or devC
  - Using “Command Line” tool (no GUI)
- Answer the following questions:
  (1) What’s the command to invoke macro processor?
  (2) How to transform a C/C++ program to assembly?
  (3) How to transform an assembly program to object code?
  (4) How to combine all object files to build the complete executable file?



Next Lecture

- UNIX system structure
  - Chap 1. of [Stevens]
- Please also read Chap. 3 & 4 of your OS textbook [Silberschatz et.al]
  - Chap. 3: Operating-System Structures
  - Chap. 4: Processes

How to learn UNIX programming



- Everything you've already learned in "Operating System" and "System Programming" course
- This course just present you concrete examples
- Please "man" and "info"