Lecture 1:Introduction To UNIX & C

Youjip Won and Kyungsoo Park

KAIST EE

Course Overview

- Time: Friday 11:00-12:00 (1-unit credit)
- Class A: Professor Youjip Won @ EE, ywon@kaist.ac.kr
- Class B: Professor Kyungsoo Park, kyungsoo@kaist.ac.kr
- Syllabus carefully designed for students taking EE209
  - Already took EE209? no real need to take this course
  - Do not take EE209 this semester? Better take it with EE209
- Grading – pass or fail (S/U)
  - Attendance (60%) + hands-on practice (20%) + final exam (20%)
  - No mid-term exam
  - Designed to be easy
    - We expect all to pass if you regularly attend the classes.
    - Fail if you are absent for 3 or more times

Goals and Recommended books

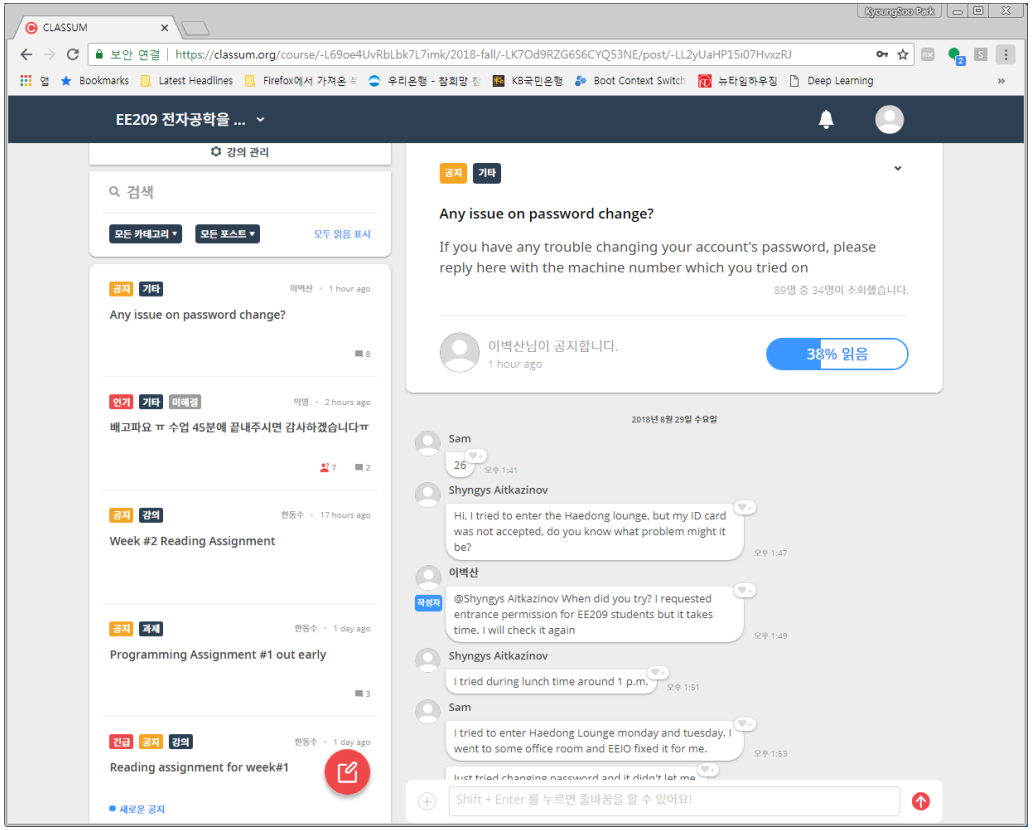
- Goal: getting familiar with the Linux programming environment and tools
  - Corollary: feel comfortable with tools for EE209 programming assignments
  - Tools: shell commands, editor, ctags/cscope, compiler, debugger, source code management, commands, script languages, trace tools

- Recommended books
  - No official textbooks
  - William Shotts, “The Linux Command Line”, 2nd edition
    - Linux commands, Bash shell, ssh/scp, etc.
    - Free PDF version available for download
  - Neil Matthew, Beginning Linux Programming, 4th Edition, **ISBN-13:** 978-0470147627
  - *Richard Blum*, *Christine Bresnahan*, Linux Command Line and Shell Scripting Bible

Contents

- Classum
- Setting up programming environment
  - Unix (Linux) and Bash
- Using text editor
- Executing a simple program
- Building a program

Q&A on Classum



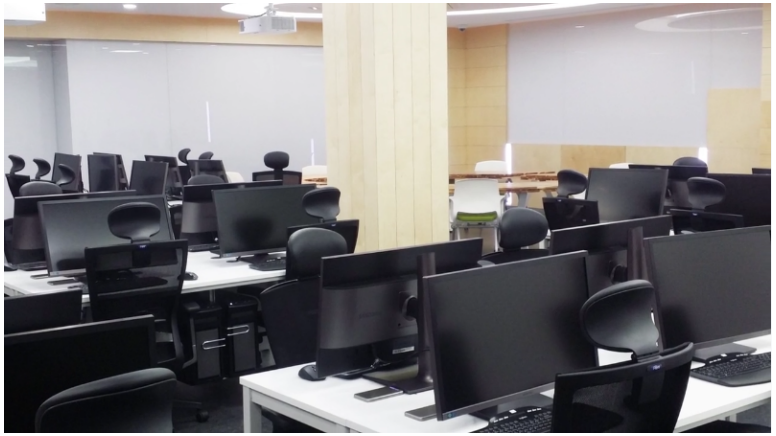
- Download the app & join our class!
  - https://classum.org/EXTPLY
  - All profs and TAs joined it.
- Ask questions & we answer.
  - Class materials.
  - EE209 programming assignments.
  - Be careful: do not show your own code for an assignment.

Linux Programming Environment

- Linux
  - Free, open-sourced operating system (OS)
  - We use a Ubuntu distribution (Ubuntu Linux)
- How to use Linux?
  - ~~Option1. Visit Haedong Lounge (PC room)~~
  - Option2. Remote access to Haedong Lounge (eelab5, eelab6)
  - Option3. Install Ubuntu on your own PC in a virtual machine
    - If you use Windows 10, consider windows subsystem for Linux (WSL)
    - Personally, I use Ubuntu on WSL on windows 10

Visit Haedong Lounge

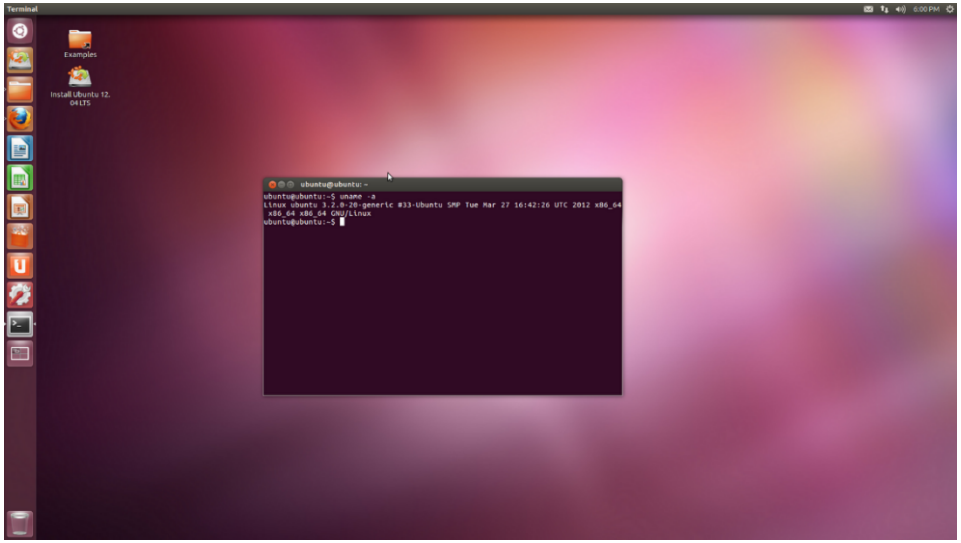
- E3-4 Room #1412
  - https://ee.kaist.ac.kr/en/node/15084
- 36 machines already set up with your accounts
  - Change your password as soon as possible,
    or your account will be automatically banned from access



- Consult with a lounge TA for machine access
- Make sure to logout after usage
- Don't turn off machines for remote accessed users (eelab5,6)
- Don't abuse them for other use

Use Ubuntu Linux Command Line Interface

- Example: on a Ubuntu machine (Haedong Lounge)
- Open Terminal program



Lab Machines & Your Accounts

- Linux is installed on these machines
  - `eelab1.kaist.ac.kr ~ eelab36.kaist.ac.kr`
- Remote-accessible machines (always on)
  - eelab5.kaist.ac.kr, eelab6.kaist.ac.kr
  - Other machines are supposed to be turned off.
- Your account is supposed to be already created.
  - ID: <your_student_id>
  - Password: KQ5yfG'ID‘ (but no ‘ in the ID) (EE209B/EE485B)
    - e.g, ID: 20201234, passwd: KQ5yfG20201234
  - If your id does not exist, or you forgot your password, please contact your class TA.

Remote Access to Haedong Lounge

- You can use tools like ‘ssh’ to access the machines remotely
  - SSH: secure shell (over network)

    Unix shell: A command line interpreter
    - You invoke a shell on a machine, and securely access it over a network.
    - Network communication on ssh is ‘encrypted’ and ‘authenticated’ (secure).
  - For Windows users: use PuTTY
    - Details in the following slide
  - For MAC OS or Linux: use Terminal
    - Search and open ‘Terminal’ program.
    - Type ‘`% ssh <Student ID>@eelabX.kaist.ac.kr`’.
      - X : 1 ~ 36

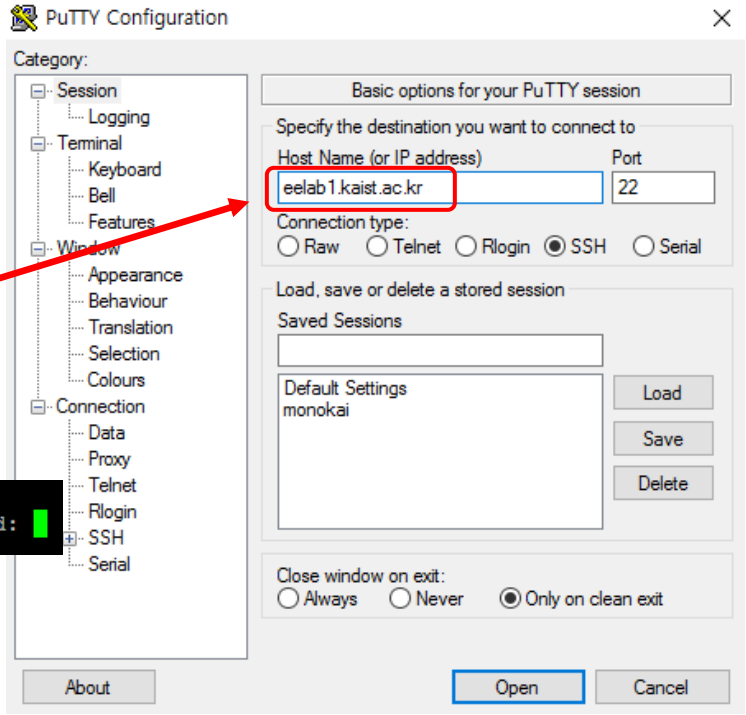
```
[Changho-Hwangs-MacBook-Air:~ chang$ ssh 20160001@eelab1.kaist.ac.kr
The authenticity of host 'eelab1.kaist.ac.kr (143.248.154.128)' can't be established.
ECDSA key fingerprint is SHA256:gzRAg3DW9rnLfTHibX6SJaR7Ypphefqrq19zIhOlRC8.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'eelab1.kaist.ac.kr,143.248.154.128' (ECDSA) to the list of known hosts.
20160001@eelab1.kaist.ac.kr's password:
```

Remote Access via PuTTY

- Download PuTTY program
  - http://www.putty.org/
  - Single file: putty.exe
- Open putty.exe
- Input “Host Name”
  - eelabX.kaist.ac.kr
- Press Open →

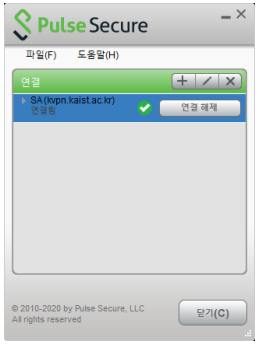
```
login as: 20160001
20160001@eelab1.kaist.ac.kr's password:
```

- Type in password and press ENTER
  - Enter “yes” at first login



Accessing KAIST machines from Outside

- KAIST allows access from outside via VPN (virtual private network)
  - Accessing eelab5 from your laptop in KAIST ('local' access: no VPN needed)
  - Accessing eelab5 from our home at Seoul ('remote' access: VPN needed)
- Install Google OTP on your mobile device (e.g. smartphone).
- Access `https://kvpn.kaist.ac.kr`
  - Login with your KAIST account
  - Google OTP: type in one time password
  - PulseSecure runs (or you need to install and run it)



Your First Successful Login

- You will see a command line interface (CLI).
  - Shell: we use “bash” shell on eelabX (more details in the next lecture)

```
20177054@eelab1:~$
```

- Now you can enter a command via typing it.
  - e.g. change your password via command ‘passwd’

```
20177054@eelab1:~$ passwd
(current) LDAP Password:
New password:
Retype new password:
passwd: password updated successfully
20177054@eelab1:~$
```

If you are working remotely …

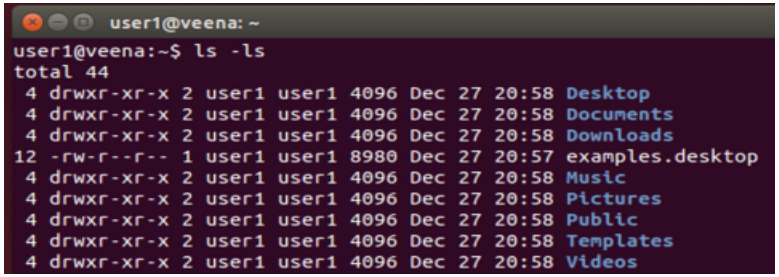
- Remote access could be unstable.
  - Machines could be turned off or network communication could be slow.
- Lab machines could be unavailable (hardware or network failures).
  - Keep your important code and files on your own machine.
  - Our recommendation:
    - Use Linux or WSL on windows10 locally.
    - Test your code on lab machines before submission.
- How to move files between machines?
  - Details in the following slide

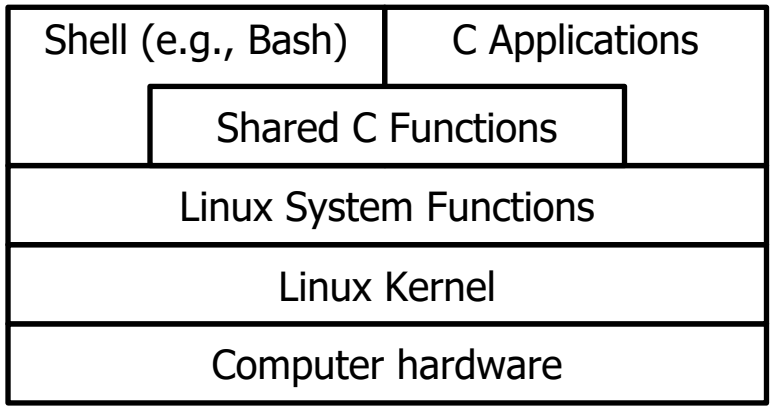
Copying Files between Machines with `scp`

- Download file from lab machine to your machine
  - `$ scp <user name>@<host_name>:<file path> .`
    - ':' → end of host name (home folder)
    - '.' → current working directory
  - e.g. `$ scp 20160001@eelab5.kaist.ac.kr:assign1/assign1.c .`
- Upload file to lab machine
  - `$ scp <file path> <user name>@<host name>:<optional path>`
  - e.g. `$ scp assign1.c 20160001@eelab5.kaist.ac.kr:assign1/`
- Copy many files
  - e.g. `$ scp *.c 20160001@eelab5.kaist.ac.kr:assign1/`
    - '*' → matches any characters more than zero
- FYI, scp uses the same 'protocol' as ssh
  - Download is secure – encrypted and authenticated

Linux (OS) and Shell

- What is OS?
  - system software that helps applications to run by providing services that require interacting with hardware, by protecting from other (malicious) program, etc.
  - “kernel”: core program that manages resources of a computer system
  - “system function”: request a “privileged” service to kernel
- Shell: Takes keyboard commands and passes them to OS to carry out
  - sh (Bourne shell), bash (bourne again sh), ksh, csh/tcsh,
  - Why shell? make difficult tasks possible!
    - GUI: make easy tasks easy





Sample Bash Commands

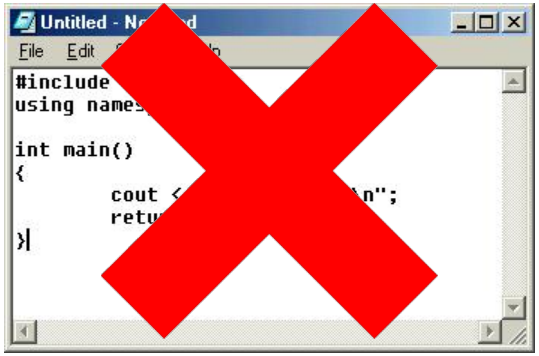
- More commands in bash_manual.pdf file
- Not comprehensive, but should be enough in most cases
- Will revisit them later

Command	Description
man <function name>	open manual page
cd <directory name>	change directory
ls [-al]	list files in the directory
mkdir <directory name>	make directory
rmdir <directory name>	remove directory
less/cat/more <file name>	print file content
cp <source> <target>	copy source file to target
mv <source> <target>	rename source file to target
rm <file name>	delete file

Be very careful using 'rm' command

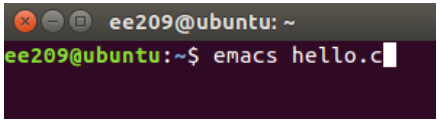
Text Editor for Programming

- We use text editors to write a program.
- Popular text editors for programming
  - Emacs, Vim, Sublime, Atom, Visual Studio, …
- Useful functionalities
  - Syntax highlighting – easy keyword check
  - Indentation (e.g., ‘tab’ to align code)
  - Easy to use tools to analyze errors in code
    - Debugging (gdb), compiling (make), code hierarchy, …
  - Hotkeys for many functionalities
- Will be a separate lecture on Emacs (KyoungSoo) or Vi (Youjip)

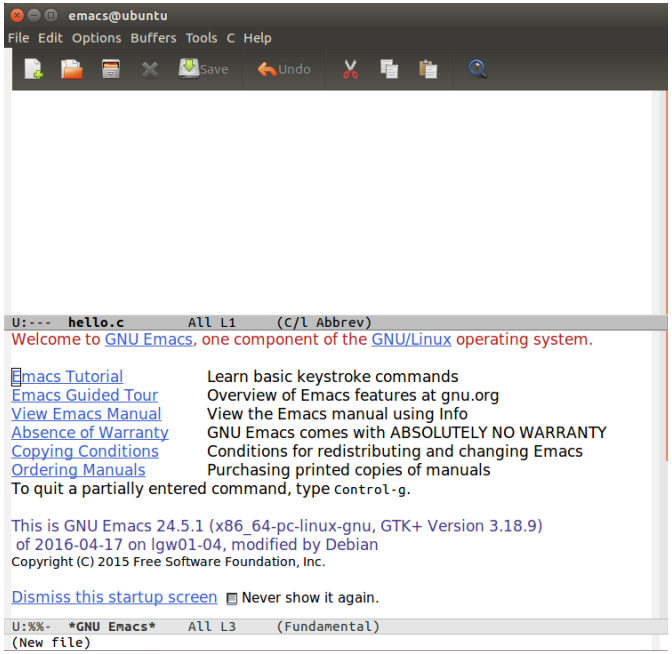


Example: Emacs

- Emacs: one of the most popular code editors on Linux
  - Open Terminal
  - Type command to open Emacs editor
    - emacs <filename>
    - e.g. emacs hello.c



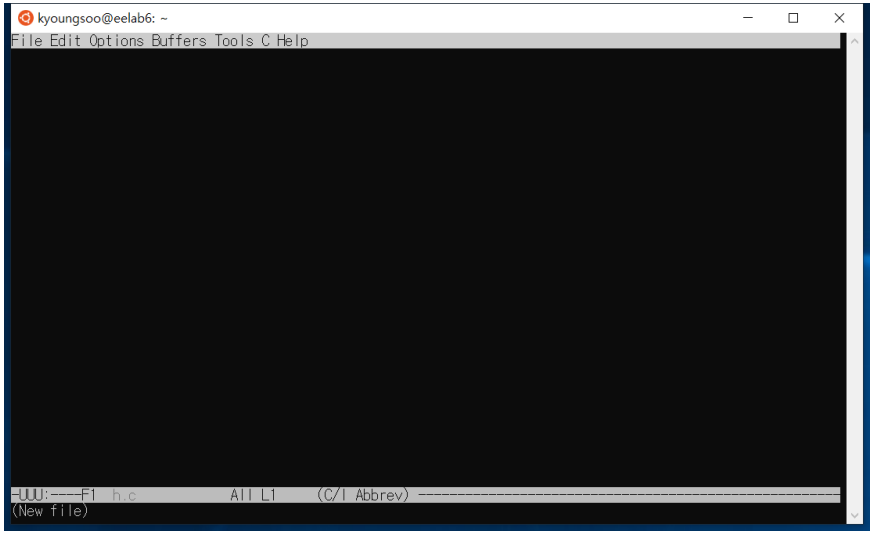
- Start programming with Emacs
  - You can remove Welcome message
  - Click on “Never show it again.”
  - Then click “Dismiss this startup screen”



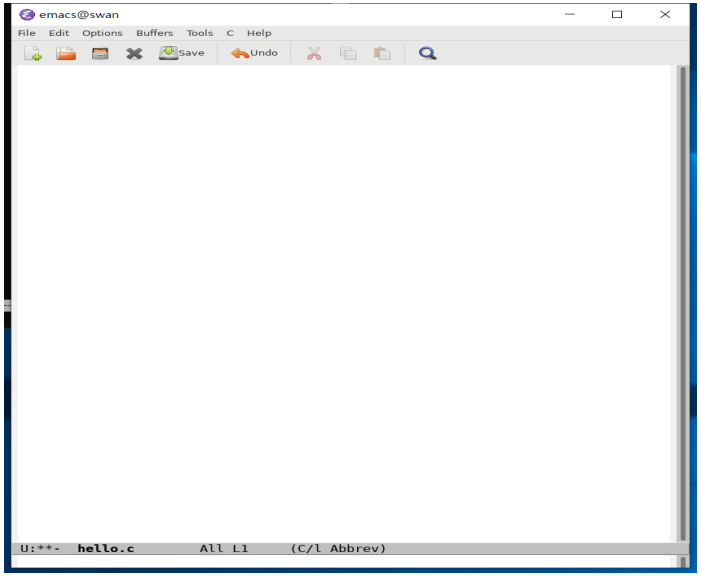
Simple Terminal vs. X Window System

- What is X Window System?
  - A Window system on UNIX-like OS
  - Provides GUI environment

Emacs on Simple Terminal



Emacs on X Window System



How To Run Linux Apps on X Window?

- Local machine must support X window
  - If you use Linux for local machine, you're likely to run X window system already.
  - If you use WSL, run helper app like 'Xming' first.
- Remote login with –X option
  - ssh –X eelab6.kaist.ac.kr // called X forwarding
  - emacs hello.c & // emacs now runs in an X window
    - '&' means that emacs runs as a background process (vs. foreground process)
    - Why run it as background?
      - Shell can accept the next command even before the previous one has not finished.
      - Very useful for editing, compiling, debugging at the same time

Emacs Hotkeys

- File, window

Hotkey	Description
Ctrl-x Ctrl-s	Save file
Ctrl-x-c	Close file
Ctrl-x-f <file>	Open file from editor
Ctrl-x 2/3	Split window vertically/horizontally
Ctrl-x o	Move to different window
Ctrl-x 1/0	Close other/this window

- Text edit

Hotkey	Description
Ctrl-k	Cut line
Ctrl-[SPACE]	Mark cursor (move with arrow key)
Ctrl/alt-w	cut/copy region
Ctrl-y	paste cut/copied region
Ctrl-s/r <string>	search/recursive string

- Special

Ctrl-x u OR Ctrl-/	undo
Ctrl-g	abort command

Building a C Program

- hello.c

```
#include <stdio.h>
int main(void)
{
    /* Write "hello, world\n" to stdout. */
    printf("hello, world\n");
    return 0;
}
```

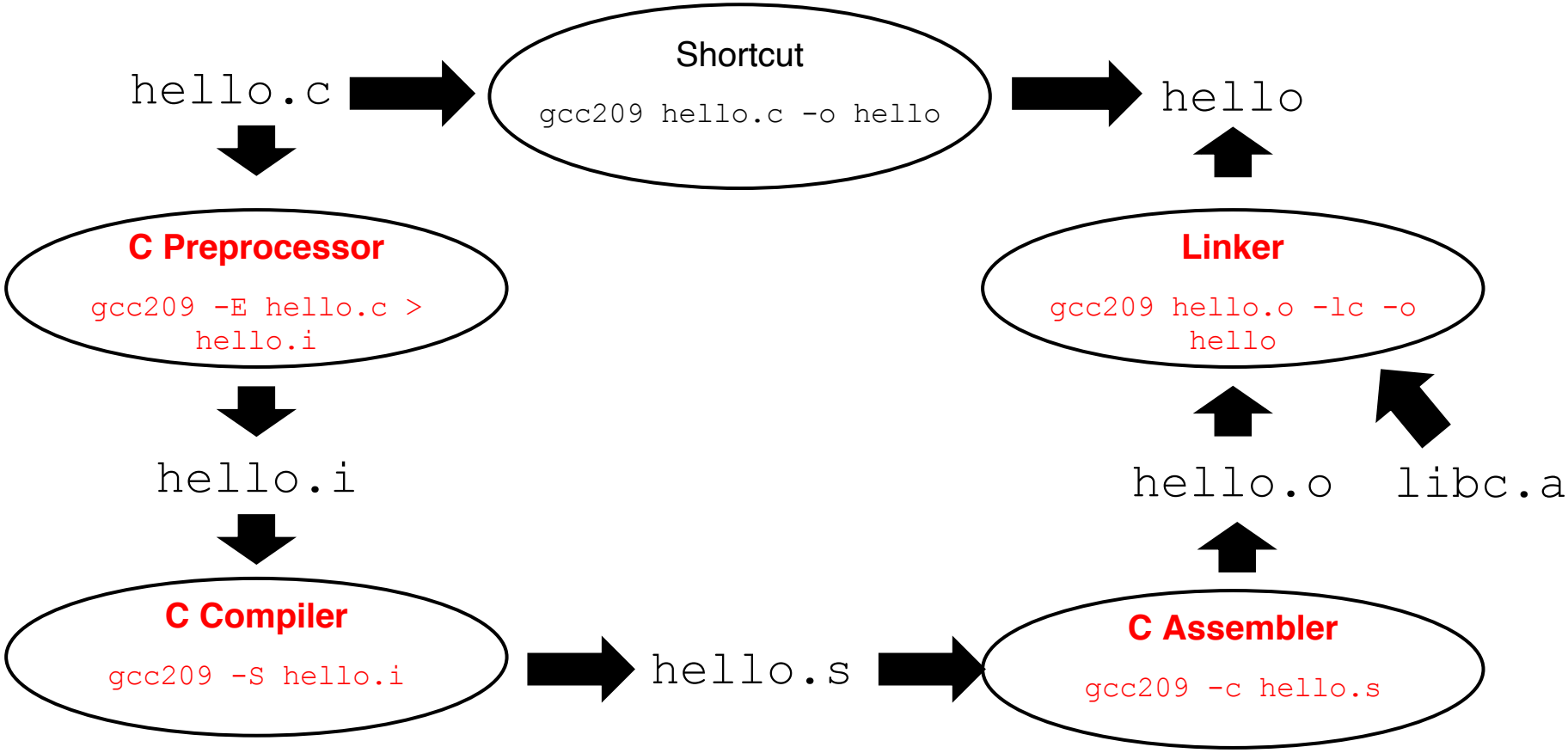
- Source code
- C language
- Contains preprocessor directives

- Compile and execute hello.c

```
ee209@ubuntu:~$ gcc209 hello.c -o hello
ee209@ubuntu:~$ ./hello
hello, world
```

`gcc209` is a script that executes
`gcc -Wall -Werror -ansi -pedantic -std=c99`

Steps to Make Executable File



Preprocess C Code

- hello.i

C Preprocessor

```
gcc209 -E hello.c > hello.i
```

```
…
int printf(char *format, …);
…
int main(void)
{
    printf("hello, world\n");
    return 0;
}
```

- Source code
- C language
- Contains declaration of printf() function
- Missing definition of printf() function
- Remove comments

Compile Assembly Language

- hello.s

C Compiler

```
gcc209 -S hello.i
```

```
.section .rodata
cGreeting:
.asciz "hello, world\n"
.section .text
.global main
.type main, @function
main:
puchl %ebp
movl %esp, %ebp
pushl %cGreeting
call printf
addl $4, $esp
movl $0, %eax
movl %ebp, %esp
popl %ebp
ret
```

- Source code
- Assembly language specific to computer architecture
- Missing definition of printf() function

Generate Object File

- hello.o

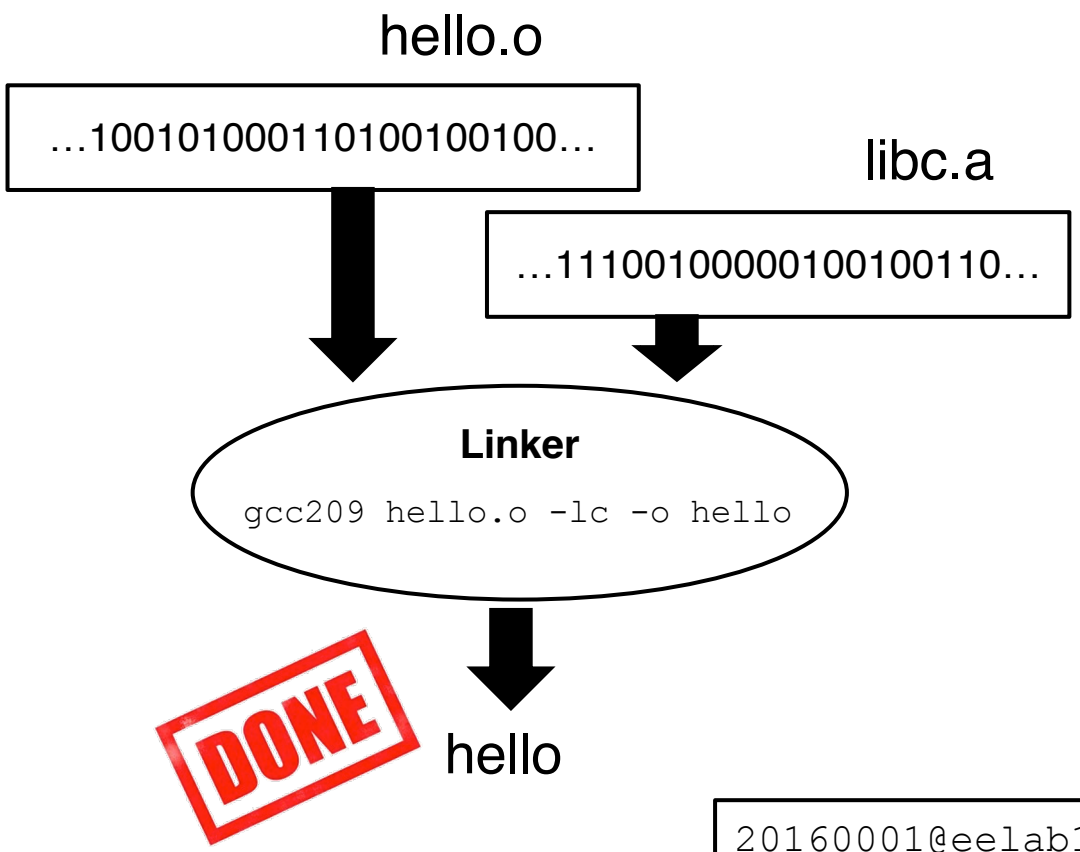
C Assembler

```
gcc209 -c hello.s
```

… 10010100011010010010 0 …

- Object file
- Machine language
- Unreadable by human
- Missing definition of printf() function
- libc.a = library containing machine language definition of printf() function (and many others)

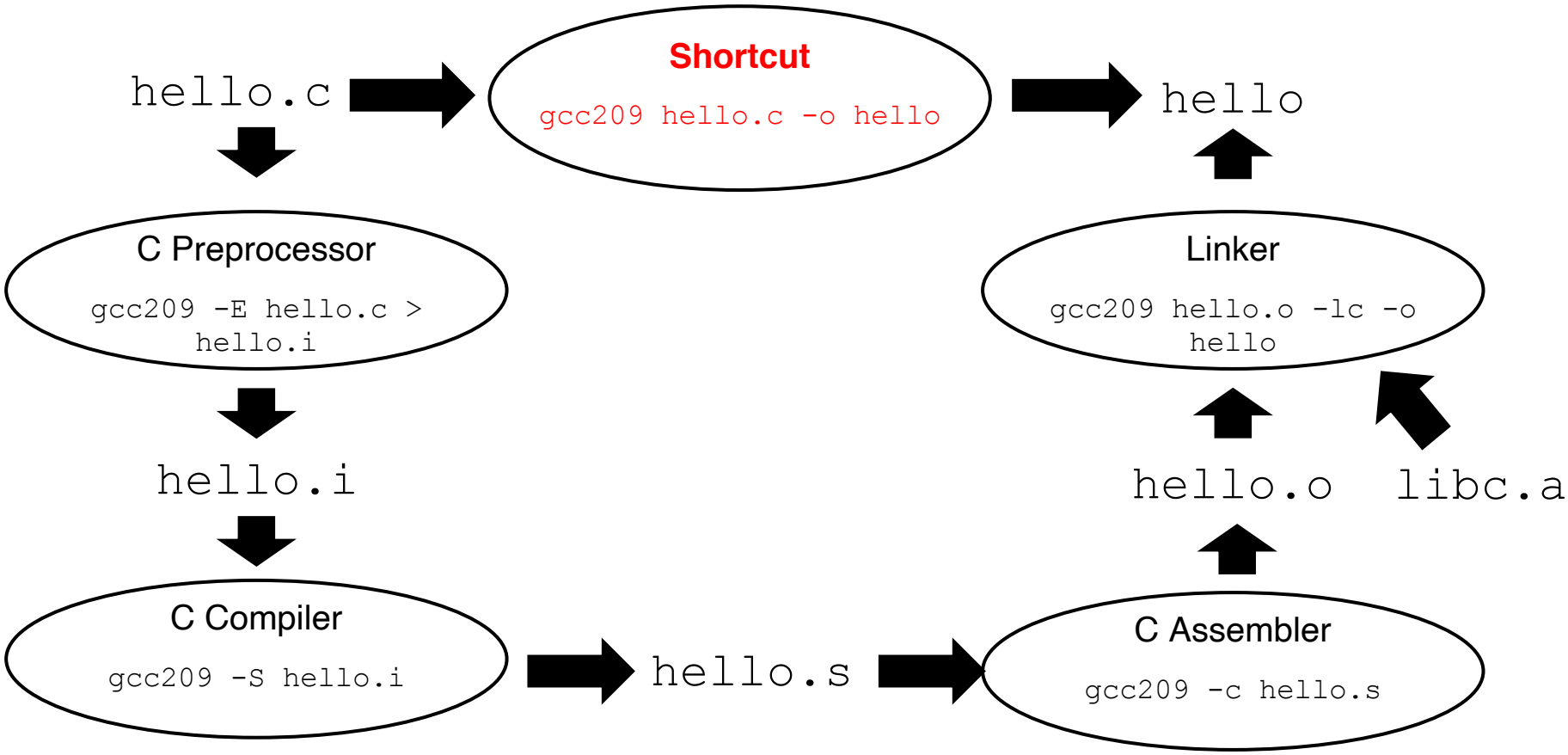
Generate Executable Binary



- Executable
- Machine language
- libc.a = library containing machine language definition of printf() function (and many others)

```
20160001@eelab1:~$ ./hello
hello, world
```

Shortcut of All Processes



gcc209 vs. gcc?

- `gcc209` is a special script made for EE209.
  - Script: a text file that contains commands to execute a series of tasks.
  - `gcc209` adds a number of options to catch improper programming
    - e.g. warns unused variable, use variable without initializing it, etc.
- You can make this script by yourself using emacs editor (or whatever editor you like).

```
$ emacs gcc209
```

```
#!/bin/bash
gcc -Wall -Werror -ansi -pedantic -std=c99 "$@"
```

  - Make this script executable:

```
$chmod +x gcc209
```

  - Move this file to folder that can be accessed globally

```
$sudo mv gcc209 /usr/bin/gcc209
```

Assignment for Lecture 1

1. `ssh` into one of the lab machines (eelab5 or eelab6)

2. Run `emacs` (preferably as background process in X window) or `vim` or any text editor at your choice.

3. Type in the C code for `hello.c` (that prints "`hello world`")

4. Compile it with `gcc209` and ***name*** the executable as hello

5. Run `hello`

- Main to-do: take a snapshot of these steps and upload it to KLMS.
  - One picture (.jpg) that shows all these commands should be enough
- Deadline: 10:59am on 3/12 (next class)
  - All assignments are due before the start of the next class in the following week.

Welcome to Programming World!

Any questions?