Lecture Note 2.
Programming Environment

September 4, 2020

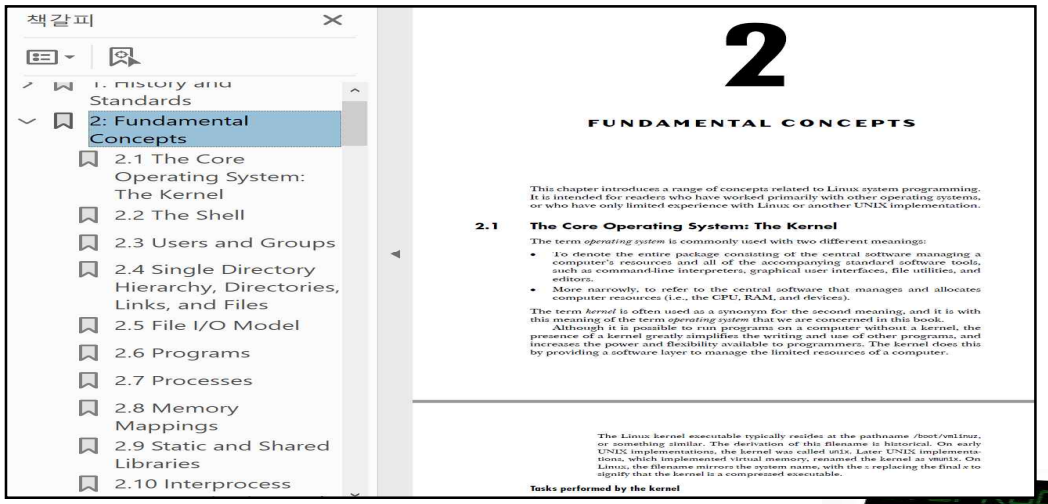
Jongmoo Choi
Dept. of Software
Dankook University
http://embedded.dankook.ac.kr/~choijm

(Copyright © 2020 by Jongmoo Choi, All Rights Reserved. Distribution requires permission)

Objectives

- Discuss the history of Linux
- Understand key concepts of Linux
- Learn how to access Linux
- Learn how to use commands in Linux
- Learn how to make programs in Linux

- Refer to Chapter 1, 2 in the LPI



Linux Introduction (1/7)

- Operating System
  - ✓ Definition: Resource Manager
  - ✓ Examples: Linux, Windows, OS X and so on.

(Source: IEEE Spectrum, 2001)

(source: https://www.deviantart.com/nick-os/art/Os-war-choose-your-poison-110510677)

(source: https://maxhemingway.com/2015/10/21/iot-device-security-considerations-and-security-layers-operating-system/)

Linux Introduction (2/7)

- Linux Definition
  - ✓ Linux is a clone of the UNIX Operating System
  - ✓ Written from scratch by Linus B. Torvalds, with assistance from a loosely-knit team of Hackers across the Network

Linus Benedict Torvalds 91. 8. 26. 답장하려면 로그인

한국어로 메시지 번역

Hello everybody out there using minix -

I'm doing a (free) operating system (just a hobby, won't be big and professional like gnu) for 386(486) AT clones. This has been brewing since april, and is starting to get ready. I'd like any feedback on things people like/dislike in minix, as my OS resembles it somewhat (same physical layout of the file-system (due to practical reasons) among other things).

I've currently ported bash(1.08) and gcc(1.40), and things seem to work. This implies that I'll get something practical within a few months, and I'd like to know what features most people would want. Any suggestions are welcome, but I won't promise I'll implement them :-)

Linus (torv...@kruuna.helsinki.fi)

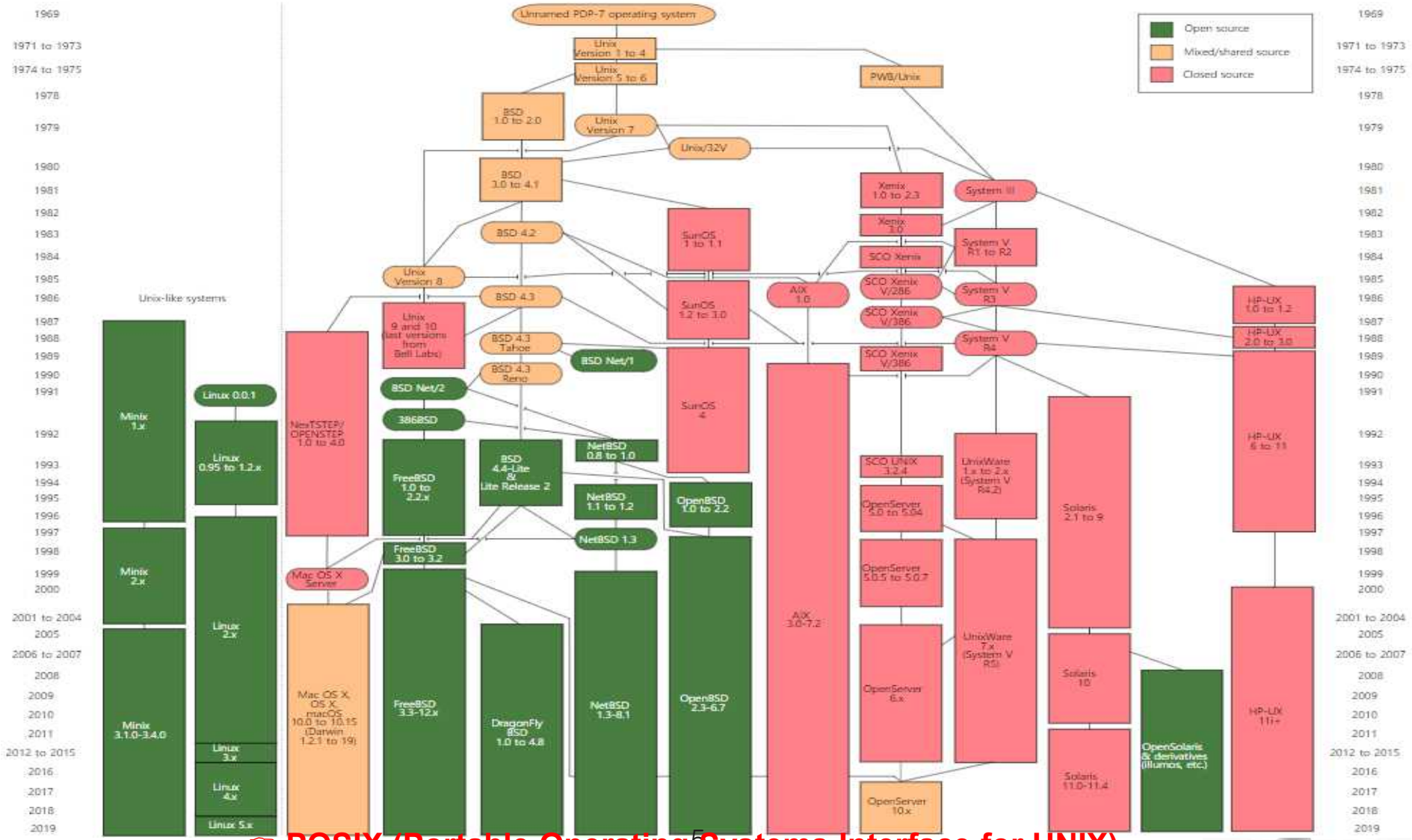
PS. Yes - it's free of any minix code, and it has a multi-threaded fs. It is NOT protable (uses 386 task switching etc), and it probably never will support anything other than AT-harddisks, as that's all I have :-(.

  - ✓ Univ. of Helsinki in Finland
  - ✓ May, 1991: Release 0.0.1 version
  - ✓ 4. September, 2020: Release 5.8.5 (refer to https://www.kernel.org/)

Linux Introduction (3/7)

- Unix-like OSes

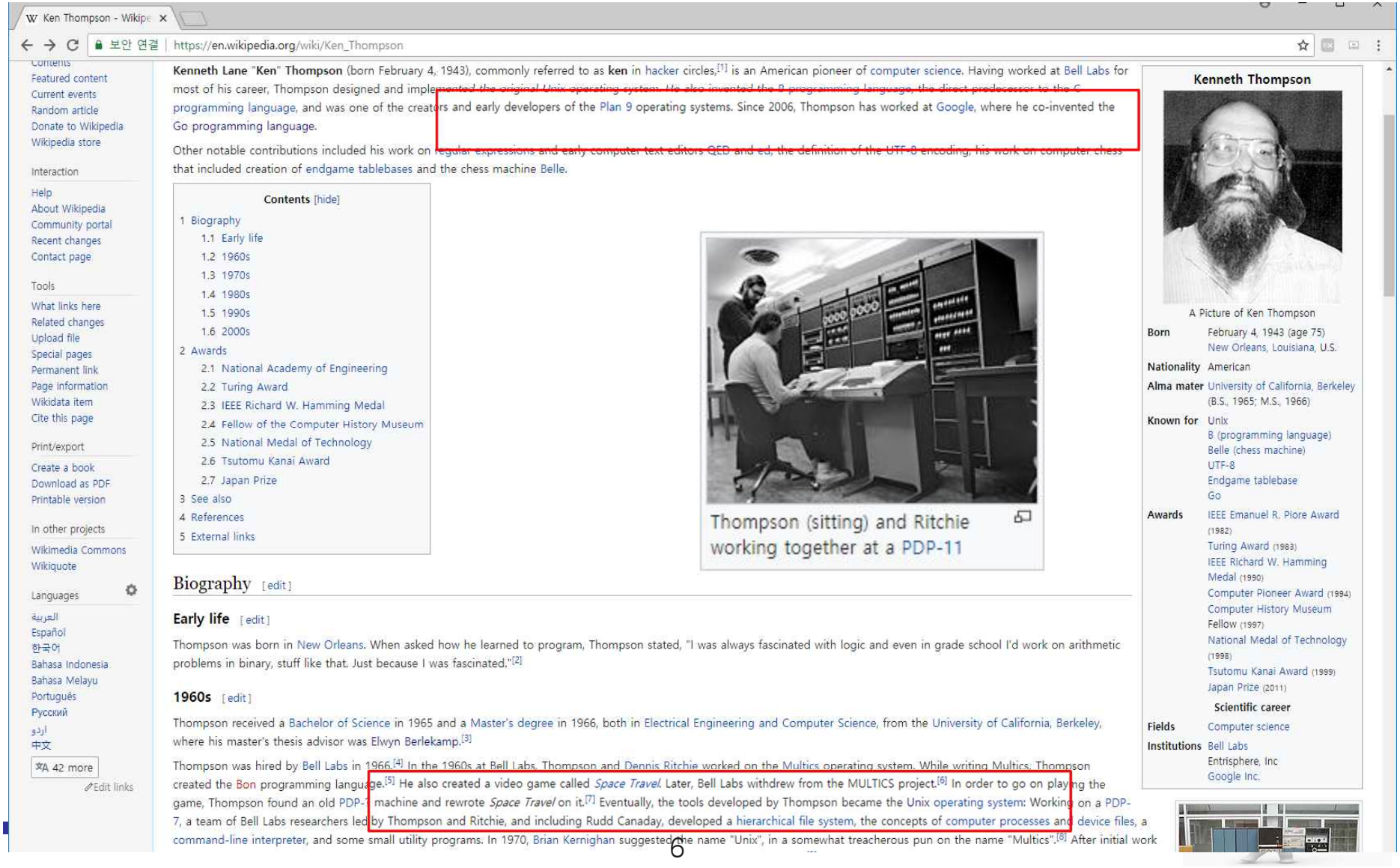
(Source: wikipedia.org)



☞ POSIX (Portable Operating Systems Interface for UNIX)

Linux Introduction (4/7)

- Ken and Dennis



Ken Thompson - Wikipe

https://en.wikipedia.org/wiki/Ken_Thompson

Contents
Featured content
Current events
Random article
Donate to Wikipedia
Wikipedia store

Interaction
Help
About Wikipedia
Community portal
Recent changes
Contact page

Tools
What links here
Related changes
Upload file
Special pages
Permanent link
Page information
Wikidata item
Cite this page

Print/export
Create a book
Download as PDF
Printable version

In other projects
Wikimedia Commons
Wikiquote

Languages
العربية
Español
한국어
Bahasa Indonesia
Bahasa Melayu
Português
Русский
اردو
中文
42 more
Edit links

Kenneth Lane "Ken" Thompson (born February 4, 1943), commonly referred to as **ken** in hacker circles,[1] is an American pioneer of computer science. Having worked at Bell Labs for most of his career, Thompson designed and implemented the original Unix operating system. He also invented the B programming language, the direct predecessor to the C programming language, and was one of the creators and early developers of the Plan 9 operating systems. Since 2006, Thompson has worked at Google, where he co-invented the Go programming language.

Other notable contributions included his work on regular expressions and early computer text editors QED and ed, the definition of the UTF-8 encoding, his work on computer chess that included creation of endgame tablebases and the chess machine Belle.

Contents [hide]

1 Biography
- 1.1 Early life
- 1.2 1960s
- 1.3 1970s
- 1.4 1980s
- 1.5 1990s
- 1.6 2000s

2 Awards
- 2.1 National Academy of Engineering
- 2.2 Turing Award
- 2.3 IEEE Richard W. Hamming Medal
- 2.4 Fellow of the Computer History Museum
- 2.5 National Medal of Technology
- 2.6 Tsutomu Kanai Award
- 2.7 Japan Prize

3 See also
4 References
5 External links

Thompson (sitting) and Ritchie working together at a PDP-11

Biography [edit]

Early life [edit]

Thompson was born in New Orleans. When asked how he learned to program, Thompson stated, "I was always fascinated with logic and even in grade school I'd work on arithmetic problems in binary, stuff like that. Just because I was fascinated."[2]

1960s [edit]

Thompson received a Bachelor of Science in 1965 and a Master's degree in 1966, both in Electrical Engineering and Computer Science, from the University of California, Berkeley, where his master's thesis advisor was Elwyn Berlekamp.[3]

Thompson was hired by Bell Labs in 1966.[4] In the 1960s at Bell Labs, Thompson and Dennis Ritchie worked on the Multics operating system. While writing Multics, Thompson created the Bon programming language.[5] He also created a video game called *Space Travel*. Later, Bell Labs withdrew from the MULTICS project.[6] In order to go on playing the game, Thompson found an old PDP-7 machine and rewrote *Space Travel* on it.[7] Eventually, the tools developed by Thompson became the Unix operating system: Working on a PDP-7, a team of Bell Labs researchers led by Thompson and Ritchie, and including Rudd Canaday, developed a hierarchical file system, the concepts of computer processes and device files, a command-line interpreter, and some small utility programs. In 1970, Brian Kernighan suggested the name "Unix", in a somewhat treacherous pun on the name "Multics".[8] After initial work

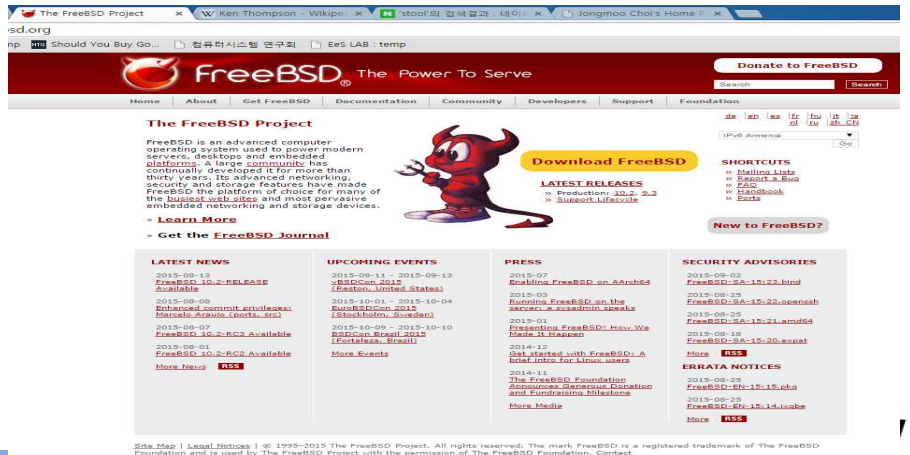
Kenneth Thompson

A Picture of Ken Thompson

Born	February 4, 1943 (age 75) New Orleans, Louisiana, U.S.
Nationality	American
Alma mater	University of California, Berkeley (B.S., 1965; M.S., 1966)
Known for	Unix B (programming language) Belle (chess machine) UTF-8 Endgame tablebase Go
Awards	IEEE Emanuel R. Piore Award (1982) Turing Award (1983) IEEE Richard W. Hamming Medal (1990) Computer Pioneer Award (1994) Computer History Museum Fellow (1997) National Medal of Technology (1998) Tsutomu Kanai Award (1999) Japan Prize (2011)
Scientific career	
Fields	Computer science
Institutions	Bell Labs Entrisphere, Inc Google Inc.

Linux Introduction (5/7)

- Contributors
  - ✓ GNU (www.gnu.org)
    - Richard M. Stallman (rms)
    - Free software
  - ✓ Minix
    - Andrew Tanenbaum
  - ✓ BSD
    - Bill Joy (cofounder of Sun Microsystems), FFS, TCP/IP, …
    - Linus Torvalds has said that if 386BSD had been available at the time, he probably would not have created Linux



Linux Introduction (6/7)

- Applications

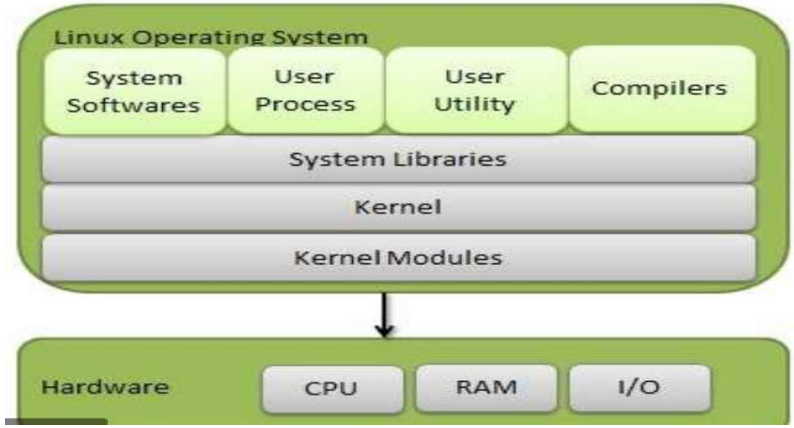
(Source: images at google)

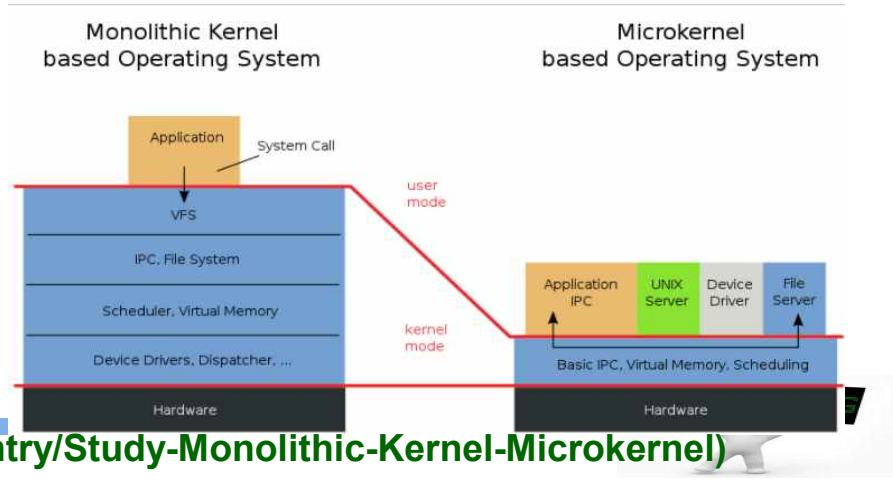
Linux Introduction (7/7)

- Some notes about UNIX and Linux (From LPI Chapter 1)
  - ✓ Linux is a member of the UNIX family
  - ✓ One feature of the UNIX is that its development is contributed by many groups, both commercial and noncommercial
  - ✓ History
    - 1969~ : UNIX Invented by Ken and Dennis, UNIX 1~7 edition at AT&T
    - 1975~ : popularly used at universities include Berkeley, MIT and CMU.
    - 1979~ : BSD and new features (FFS, TCP/IP, C shell, ...)
    - 1981~ : System III and System V from AT&T
    - 1985~ : UNIX golden ages (IBM, HP, Sun, NeXTStep, SCO, ...) ➔ UNIX War
    - 1990~ : Standardization (POSIX, FIPS, X/Open, SUS (Single UNIX Spec.)
    - 2020: Three representative OSes + Vendor proprietary OSes + New OSes
    - 1984~ : GNU by R. Stallman (gcc, Emacs, bash, ...), GPL (General Public License)
    - 1991~ : Linux by L. Torvalds, Minix + Intel optimization, GNU incorporation
    - 2020: Linux kernel version 5.8.5
  - ✓ Linux version number
    - x.y.z: Major.Minor.Revision
    - even minor: stable, odd minor: development (but NOT strict today)

Fundamental Concepts of Linux (1/7)

- From LPI Chapter 2
- 2.1 The Core of Operating System: kernel
  - ✓ OS: Computing environments vs. Kernel: Central part of OS
    - OS = Kernel + Other System Programs (GUI, Shell, GCC, Packages, …)
    - Kernel's role: 1) Process mgmt., 2) VM, 3) FS, 4) Device access, 5) Networking, 6) system call, 7) multi-user support
    - Kernel module: dynamic loadable SW runs in kernel mode
  - ✓ User mode vs kernel mode (also called as supervisor mode)
    - To protect kernel from applications
    - Monolithic kernel vs. Microkernel (u-kernel)
  - ✓ System: process's viewpoint vs. Kernel's viewpoint





(Source: https://talkingaboutme.tistory.com/entry/Study-Monolithic-Kernel-Microkernel)

Fundamental Concepts of Linux (2/7)

- 2.2 The shell
  - ✓ Special-purpose program designed to read commands typed by a user and execute them ➔ command interpreter
  - ✓ Examples: Bourne shell (Bell Lab.), C shell (BSD), Korn Shell (AT&T), bash (GNU)
- 2.3 Users and Groups
  - ✓ 3 categories: user, group, others
  - ✓ Superuser: has special privileges (User ID: 0, login name: root)

Unix Shell application comparison table

Application	sh	csh	ksh	bash	tcsh
Job control	N	Y	Y	Y	Y
Aliases	N	Y	Y	Y	Y
Input/Output redirection	Y	N	Y	Y	N
Command history	N	Y	Y	Y	Y
Command line editing	N	N	Y	Y	Y
Vi Command line editing	N	N	Y	Y	Y
Underlying Syntax	sh	csh	ksh	sh	csh

(Source: https://stackoverflow.com/questions/5725296/difference-between-sh-and-bash)

Fundamental Concepts of Linux (3/7)

- 2.4 Directory and Links
  - ✓ file types: regular, directory, link, device, ... (almost everything is file)
  - ✓ directory: a set of related file, support hierarchical structure
  - ✓ Home directory, root directory, current directory
- 2.5 File I/O Model
  - ✓ stdio library: fopen(), fread(), fwrite(), fclose(), printf(), scanf(), …
  - ✓ system call: open(), read(), write(), close(), … ➔ LN3
  - ✓ After open(): file name → file descriptor

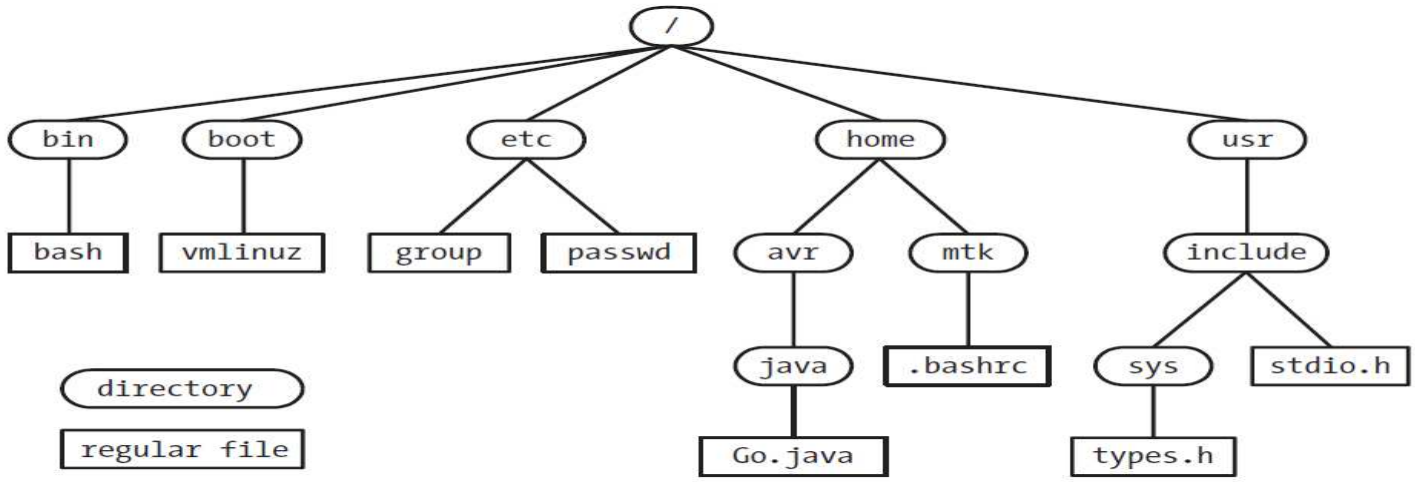


Figure 2-1: Subset of the Linux single directory hierarchy

Quiz for 3rd-Week 1st-Lesson

- Quiz
  - ✓ 1) Who invented the UNIX? Answer two persons (hint: One developed the Go language at Google and the other invented the C)
  - ✓ 2) Explain what are differences between monolithic kernel and microkernel?
  - ✓ Due: until 6 PM Friday of this week (18th, September)

1.1 A Brief History of UNIX and C

The first UNIX implementation was developed in 1969 (the same year that Linus Torvalds was born) by Ken Thompson at Bell Laboratories, a division of the telephone corporation, AT&T. It was written in assembler for a Digital PDP-7 minicomputer. The name UNIX was a pun on MULTICS (*Multiplexed Information and Computing Service*), the name of an earlier operating system project in which AT&T collaborated with Massachusetts Institute of Technology (MIT) and General Electric. (AT&T had by this time withdrawn from the project in frustration at its initial failure to develop an economically useful system.) Thompson drew several ideas for his new operating system from MULTICS, including a tree-structured file system, a separate program for interpreting commands (the shell), and the notion of files as unstructured streams of bytes.

In 1970, UNIX was rewritten in assembly language for a newly acquired Digital PDP-11 minicomputer, then a new and powerful machine. Vestiges of this PDP-11 heritage can be found in various names still used on most UNIX implementations, including Linux.

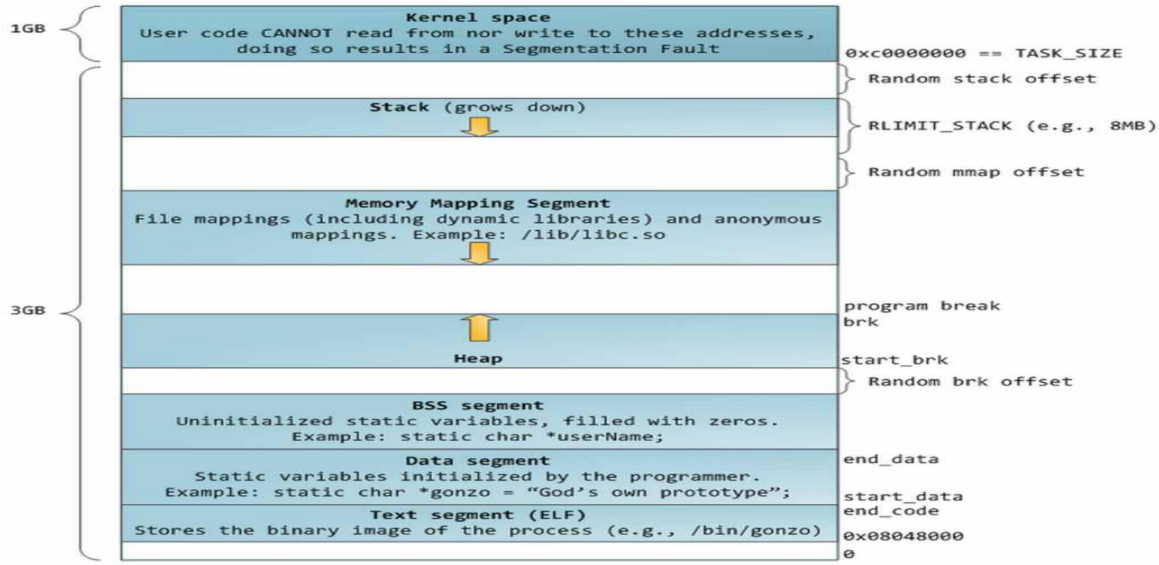
A short time later, Dennis Ritchie, one of Thompson's colleagues at Bell Laboratories and an early collaborator on UNIX, designed and implemented the C programming language. This was an evolutionary process; C followed an earlier interpreted language, B. B was initially implemented by Thompson and drew many of its ideas from a still earlier programming language named BCPL. By 1973, C had matured to a point where the UNIX kernel could be almost entirely rewritten in the new language. UNIX thus became one of the earliest operating systems to be written in a high-level language, a fact that made subsequent porting to other hardware architectures possible.

The genesis of C explains why it, and its descendant C++, have come to be used so widely as system programming languages today. Previous widely used languages were designed with other purposes in mind: FORTRAN for mathematical tasks performed by engineers and scientists; COBOL for commercial systems processing streams of record-oriented data. C filled a hitherto empty niche, and unlike FORTRAN and COBOL (which were designed by large committees), the design of C arose from the ideas and needs of a few individuals working toward a single goal: developing a high-level language for implementing the UNIX kernel and associated software. Like the UNIX operating system itself, C was designed by professional programmers for their own use. The resulting language was small, efficient, powerful, terse, modular, pragmatic, and coherent in its design.

2 Chapter 1

Fundamental Concepts of Linux (4/7)

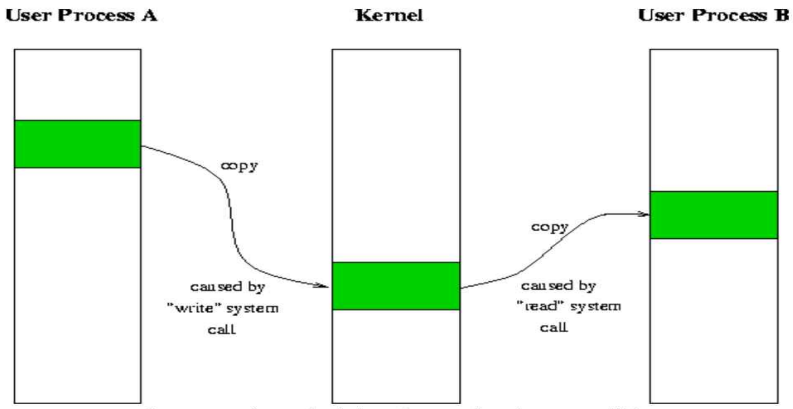
- 2.6 Programs
  - ✓ A set of instructions that describes how to perform a specific task
  - ✓ Two forms: source code, binary (machine language)
- 2.7 Processes
  - ✓ An instance of an executing program ➔ LN4, 5
  - ✓ Has its own virtual memory (layout: text, data, heap, stack, map)
- 2.8 Memory Mappings
  - ✓ mmap(): maps a file into the calling process's virtual memory
  - ✓ Access file using a pointer instead of open()/read()/write()

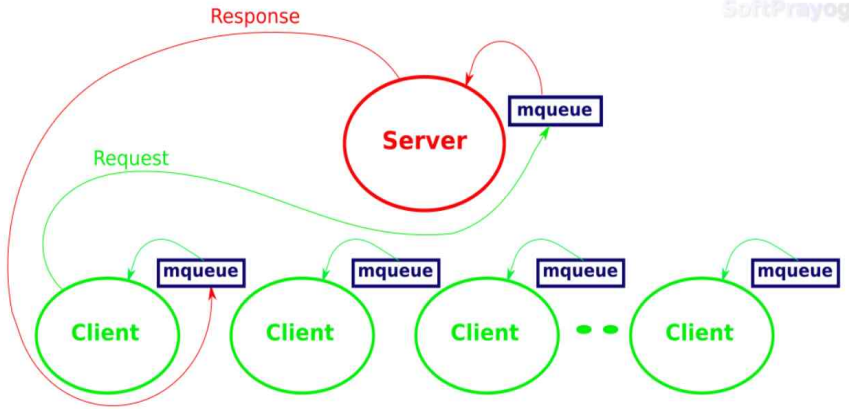


(Source: brunch.co.kr/@alden/13)

Fundamental Concepts of Linux (5/7)

- 2.9 Static and Shared Libraries
  - ✓ Compiled objects (relocatable and logically related)
  - ✓ Static libraries (also called as archive): compile-time linking
    - extracts copies of the required object modules from the library and copies these into an executable file
  - ✓ Shared libraries: run-time linking
    - instead of copying object modules from library into executable, just write a record, which allows shared libraries to be linked on-demand
- 2.10 IPC and Synchronization
  - ✓ Inter Process Communication and Process orchestration
  - ✓ Examples: signal, pipe, socket, message queue, shared memory, semaphore, …





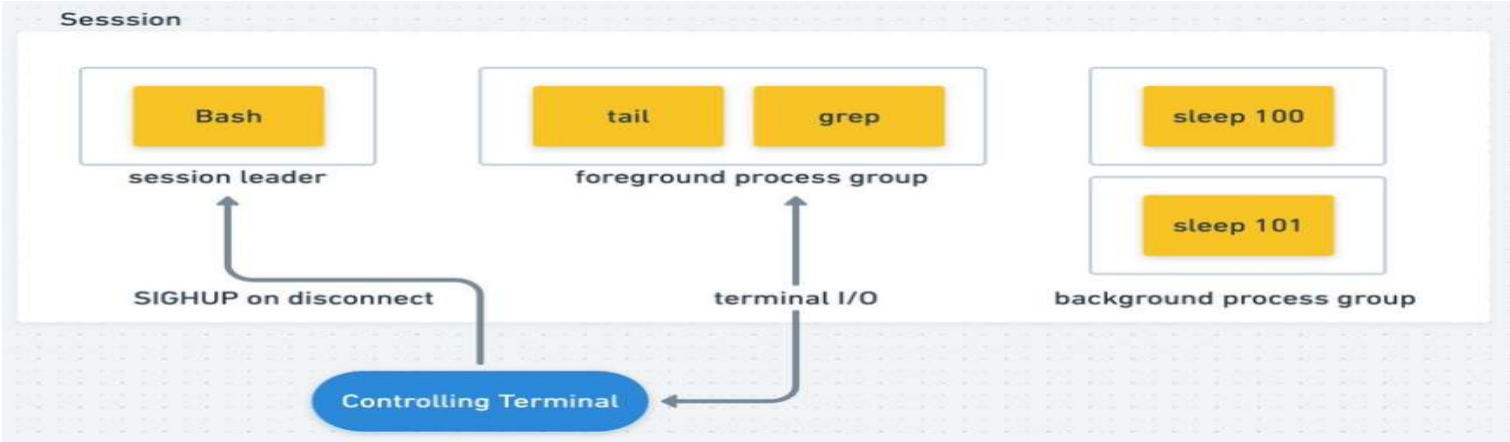
(Source: http://www.gerhardmueller.de/docs/UnixCommunicationFacilities/ip/node6.html, https://www.softprayog.in/programming/interprocess-communication-using-system-v-message-queues-in-linux)

Fundamental Concepts of Linux (6/7)

- 2.11 Signal
  - ✓ User-level interrupt: inform to a process (^C)
  - ✓ c.f.) Interrupt: a mechanism to inform an event to kernel
- 2.12 Thread
  - ✓ A flow control in a process (threads share virtual memory) ➔ LN5
- 2.13 Job control (Process group)
  - ✓ allows the user to simultaneously execute and manipulate multiple commands or pipelines.

```
$ ls -l | sort -k5n | less
```

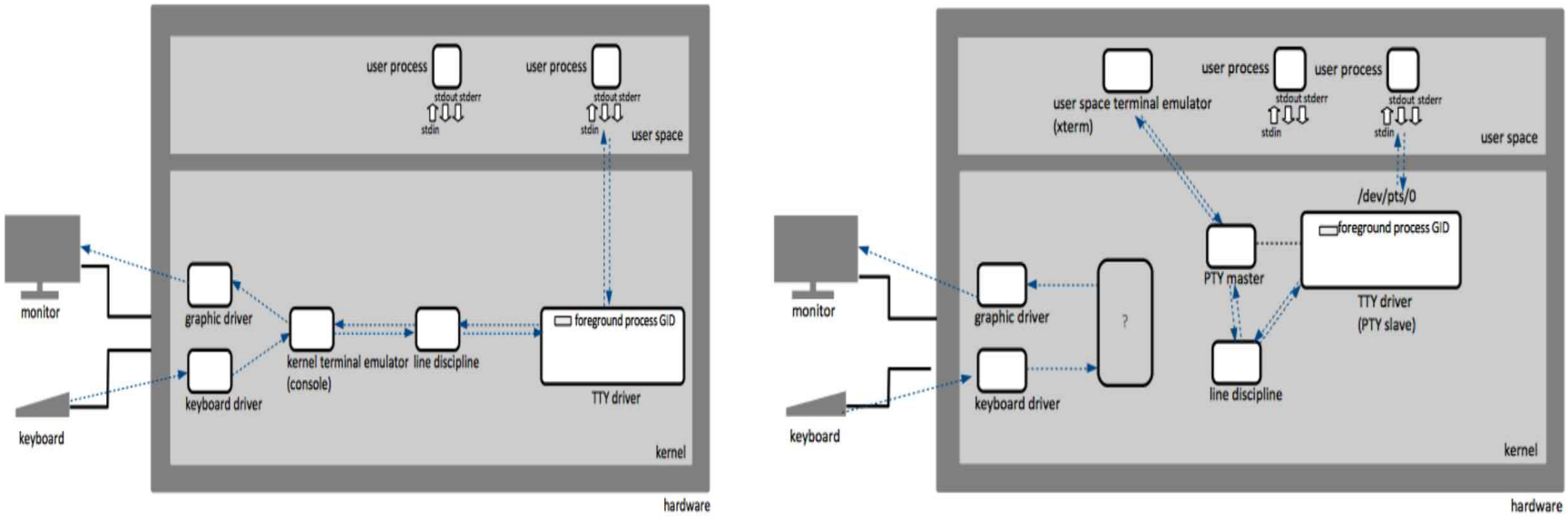
- 2.14 Session
  - ✓ A session is a collection of process groups (jobs).
  - ✓ Related with a terminal (controlling terminal, usually login terminal)
    - One foreground job and multiple background jobs



(Source: https://twitter.com/igor_sarcevic/status/1157349076809191425)

Fundamental Concepts of Linux (7/7)

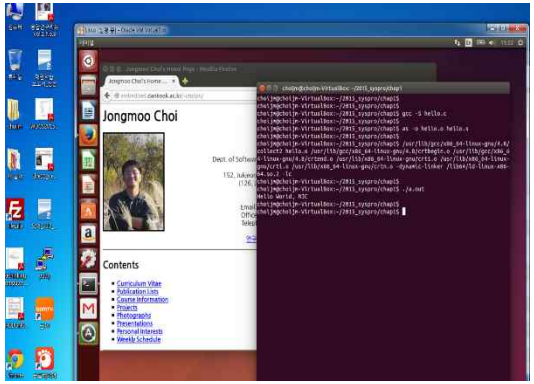
- 2.15 Pseudo-terminal
  - ✓ Connected virtual devices (e.g. terminal emulator)
- 2.16 Date and time
  - ✓ Real time (also called as epoch time): Since 1st January, 1970.
  - ✓ Process time (also called as CPU time)
    - Total amount of CPU time that a process has used since starting
    - system CPU time, user CPU time
- Others
  - ✓ Client-Server architecture, Realtime, /proc file system

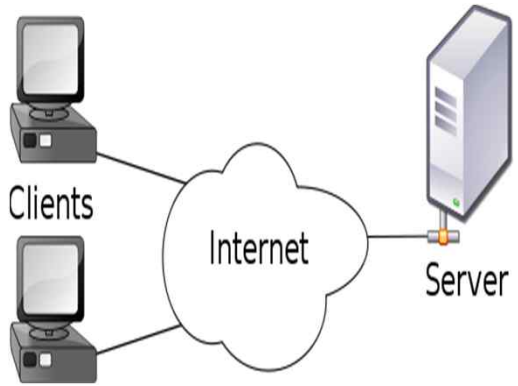


(Source: https://kb.novaordis.com/index.php/Linux_TTY)

How to access Linux (1/4)

- 1) Standalone (usually with multi-boot)
- 2) Virtualization (or WSL)
- 3) Client-Server

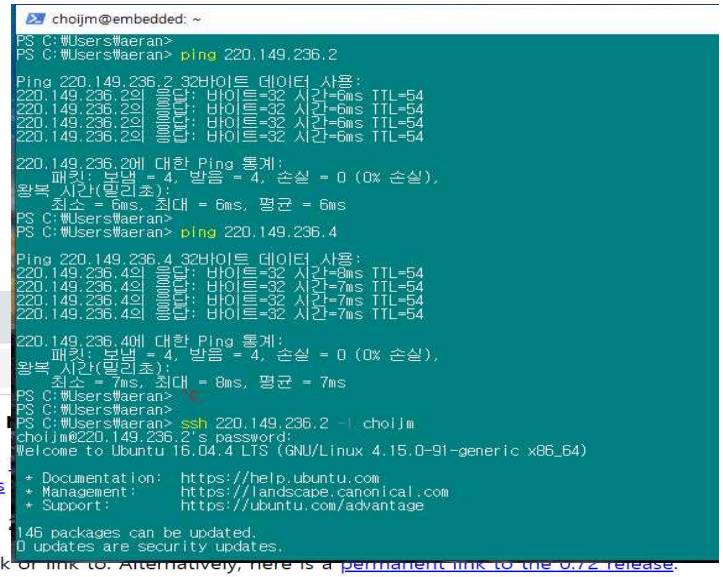


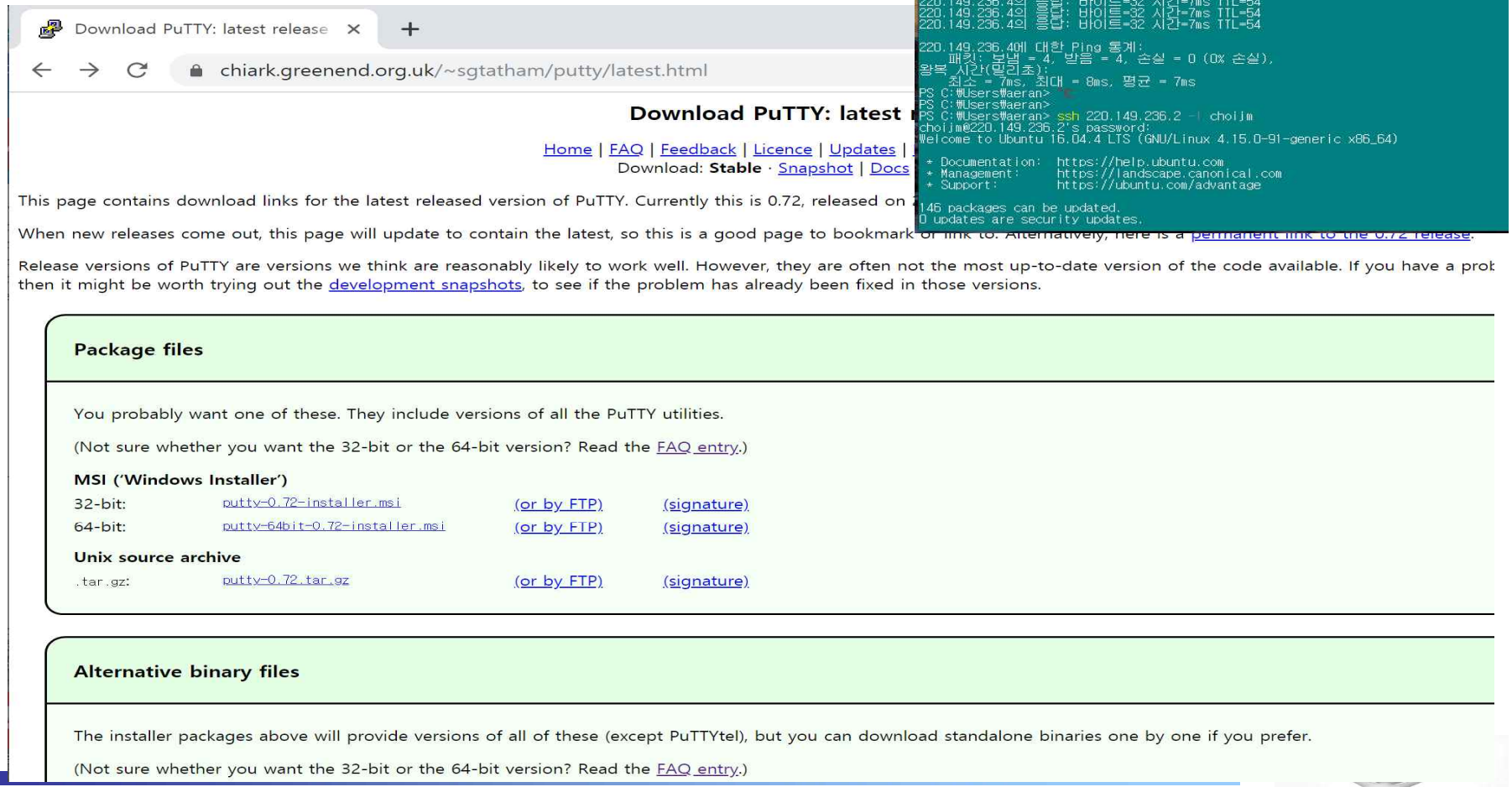


- ✓ In our course
  - Client: terminal emulator (telnet/ssh client, putty, …)
  - Server: Linux system (PC)
    - IP: 220.149.236.2 (primary), 220.149.236.4 (secondary)
  - Alternative: Amazon EC2, Google Cloud, MS Azure or ToastCloud

How to access Linux (2/4)

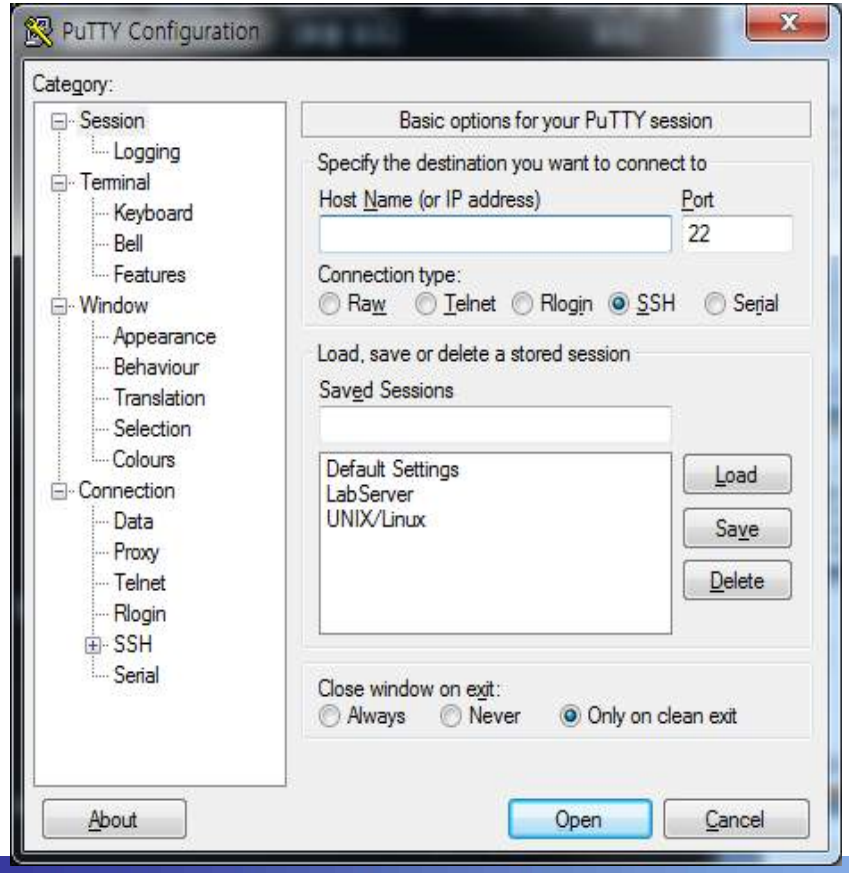
- Client
  - ✓ telnet, ssh, ping, …
  - ✓ putty, SecureCRT, powershell, …

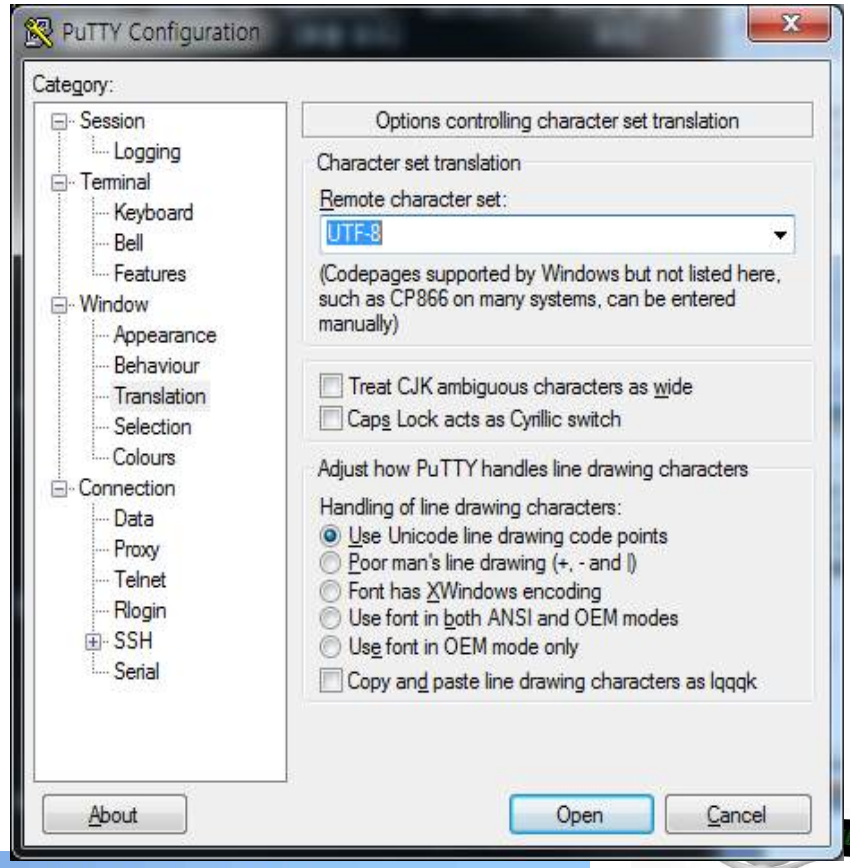




How to access Linux (3/4)

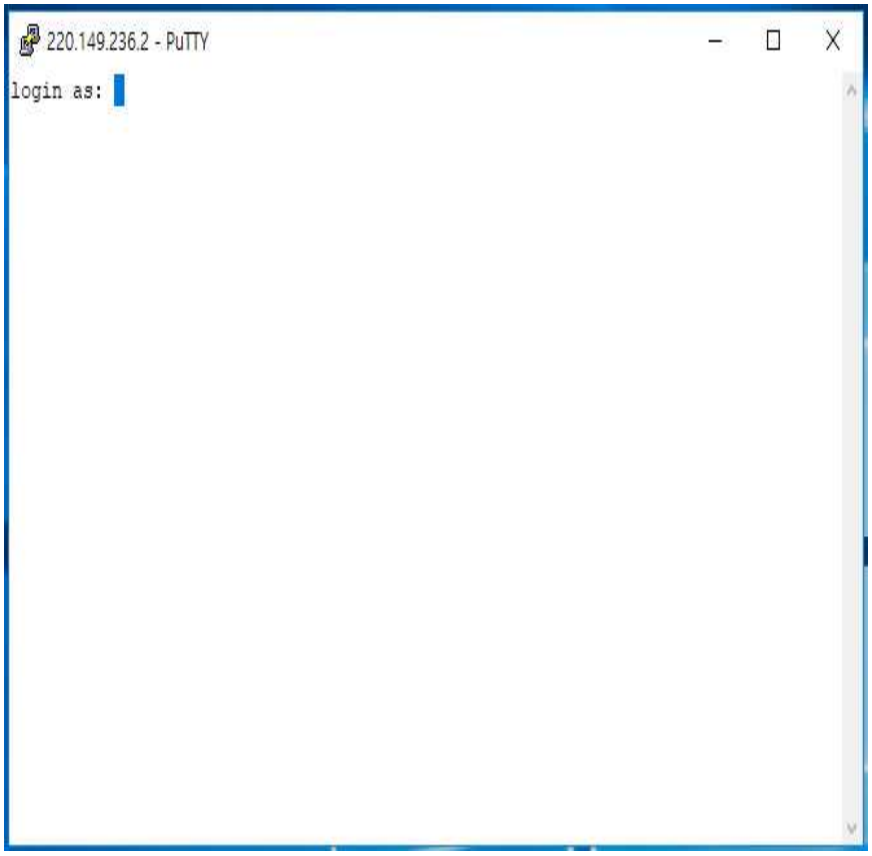
- Putty with ssh
  - ✓ IP: 220.149.236.2 (check that “type is ssh” and “port is 22”)
  - ✓ Colours: click “Use system colours
  - ✓ Translation: choose “UTF-8”

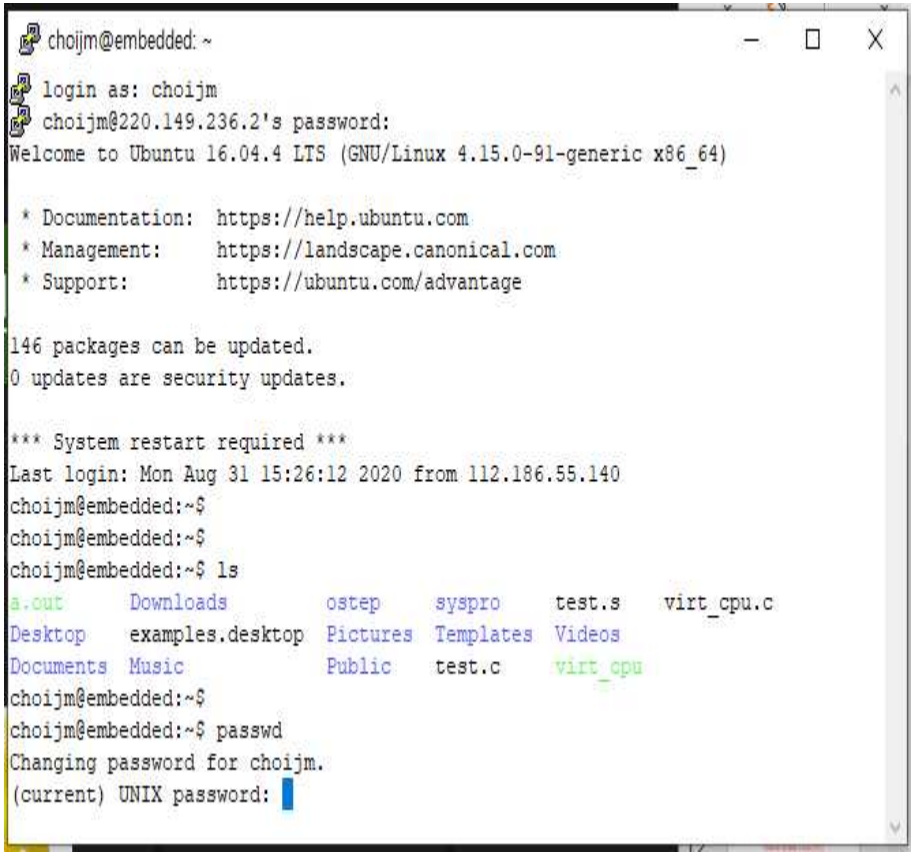




How to access Linux (4/4)

- Login and shell





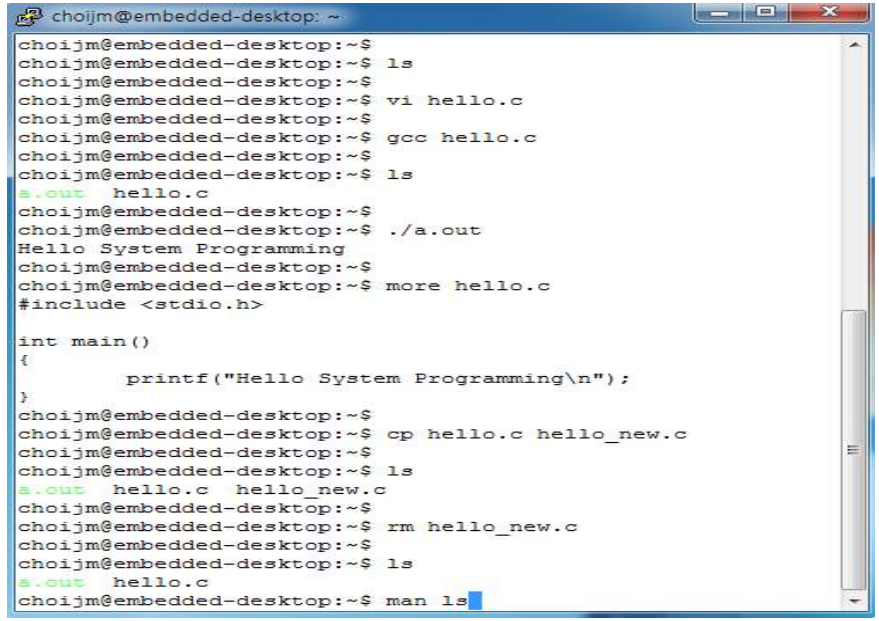
- ✓ ID: sys학번 (8 numbers of Student ID)
- ✓ Default passwd: sys****** (change using the “passwd” command)

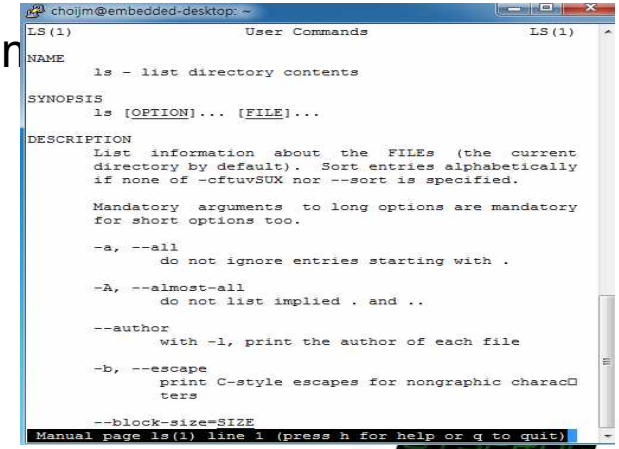
How to use commands in Linux (1/13)

- UNIX
  - ✓ Two key objects in UNIX: file as a "place" and process (task) as a "life" (by M. Bach, The Design of the UNIX Operating Systems)
- File
  - ✓ Array of bytes, stream of character (attributes: start, size, current offset)
  - ✓ Associated with disk blocks
  - ✓ Supports a variety of objects using file concept (eg. device, network, memory, and even process)
- Process (Task)
  - ✓ Program in execution
  - ✓ Associated with CPUs (Scheduling entity)
  - ✓ Having context such as memory space and CPU registers

How to use commands in Linux (2/13)

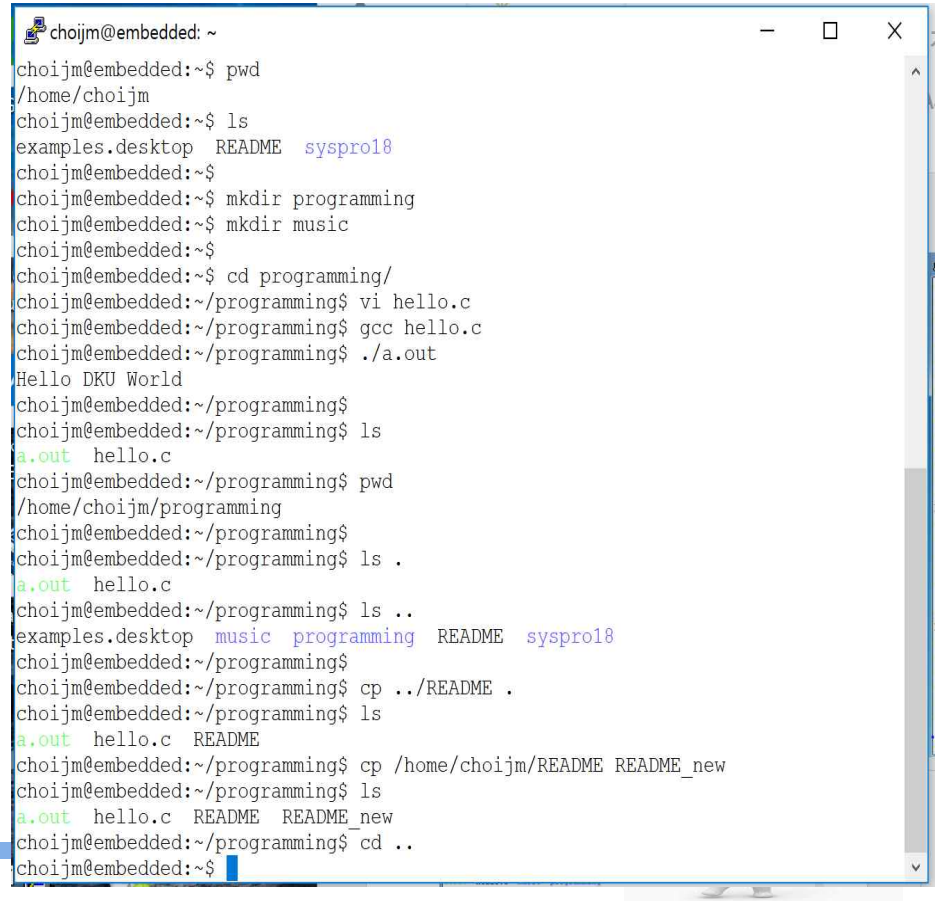
- file related command
  - ✓ create
    - vi, gcc, mknod, …
  - ✓ copy/move
    - cp, mv, ln, …
  - ✓ delete
    - rm
  - ✓ listing
    - ls
  - ✓ file content view
    - cat, more, less, head, tail, objdump, hexdu
  - ✓ file attributes manipulation
    - chmod, chown, chgrp, touch
  - ✓ redirection
    - >





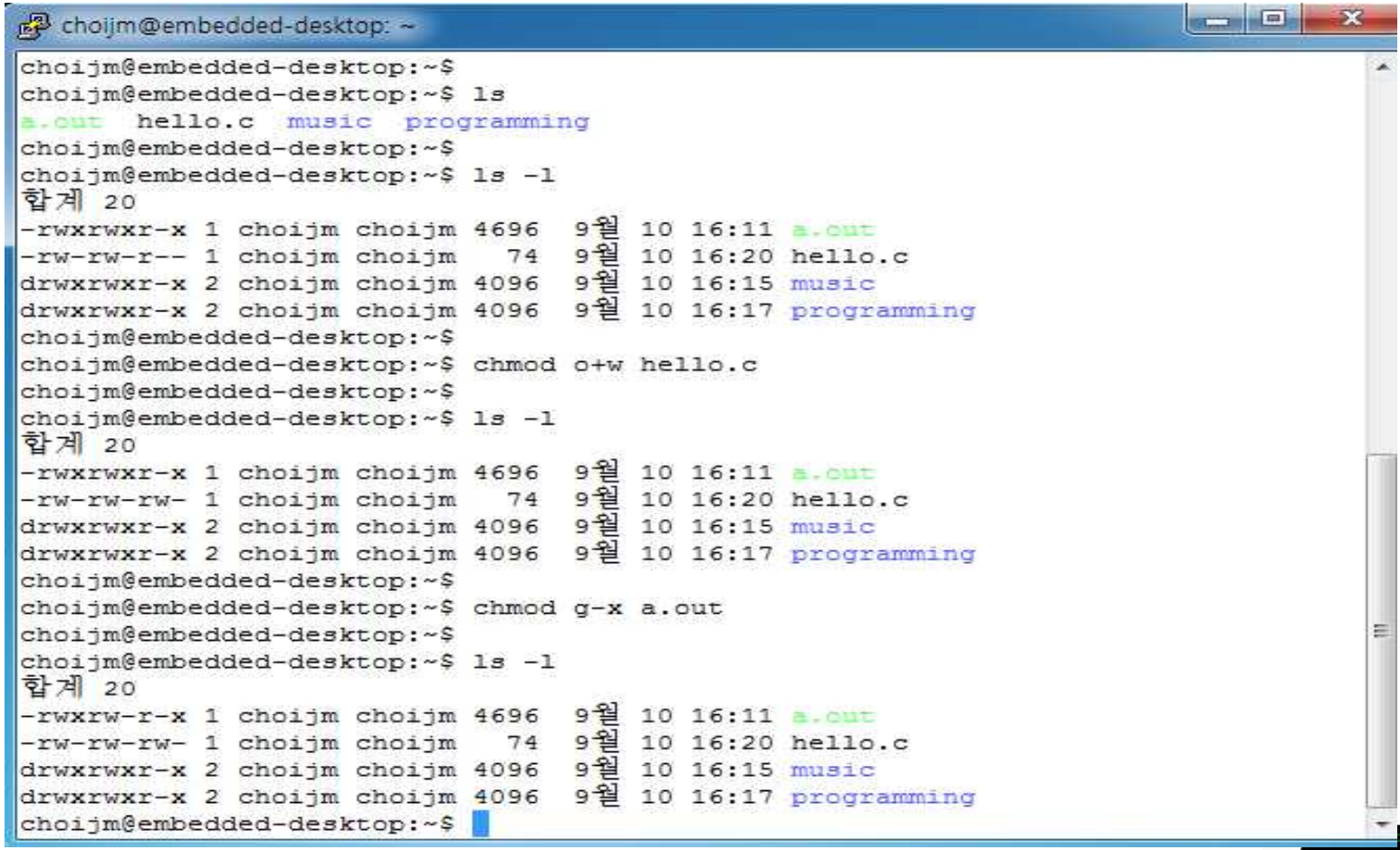
How to use commands in Linux (3/13)

- directory
  - ✓ a set of files
  - ✓ provide hierarchical structure of files
  - ✓ home directory, root directory, current directory
  - ✓ relative path, absolute path
- directory related command
  - ✓ create
    - mkdir
  - ✓ change
    - cd
  - ✓ delete
    - rmdir
  - ✓ current position
    - pwd



How to use commands in Linux (4/12)

- file attribute manipulation
  - ✓ Permission and owner
  - ✓ cf. Command format: 1) command, 2) option, 3) argument

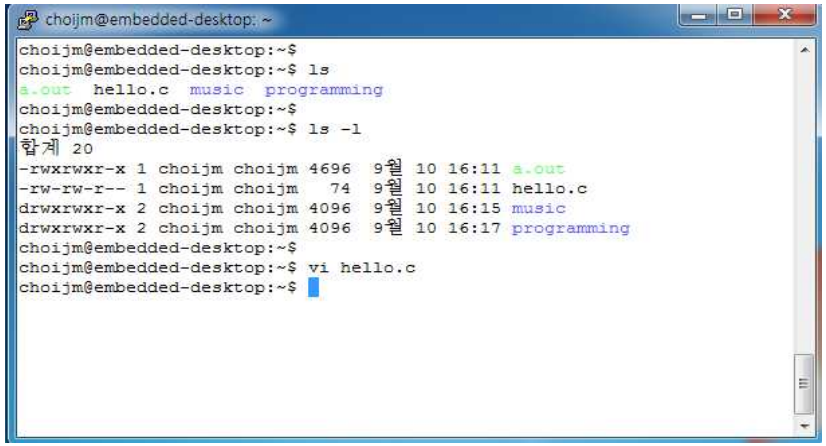


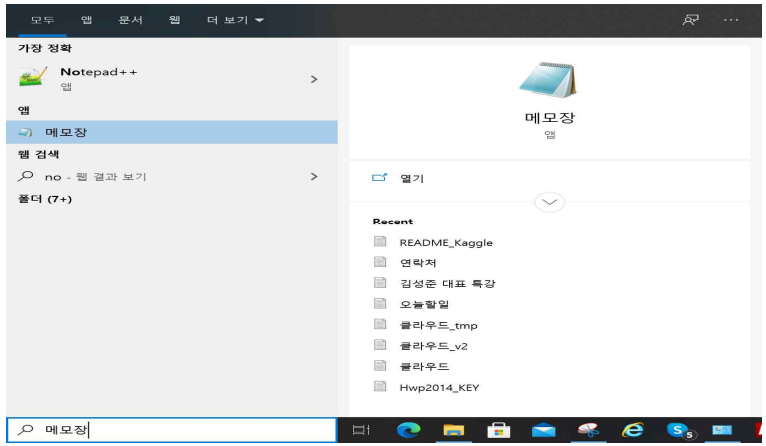
Quiz for 3rd-Week 2nd-Lesson

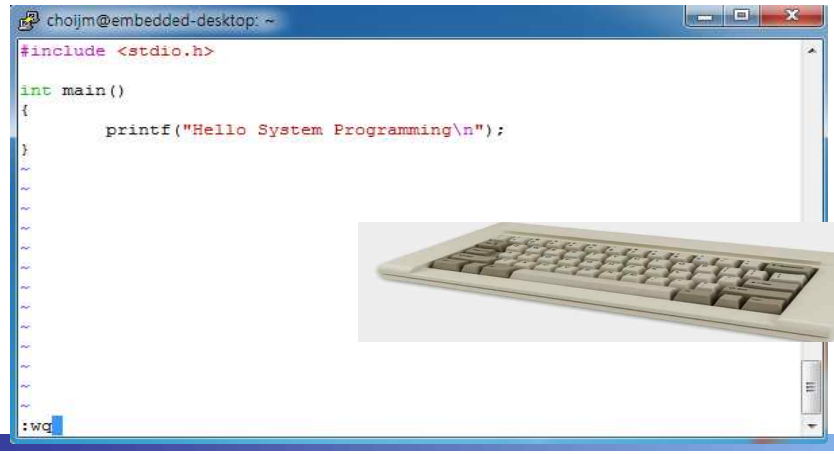
- Quiz
  - ✓ 1) Discuss how to change your password in Linux.
  - ✓ 2) Explain all information that we can see using the Linux command of “ls –l”. (at least 6)
  - ✓ Due: until 6 PM Friday of this week (18th, September)

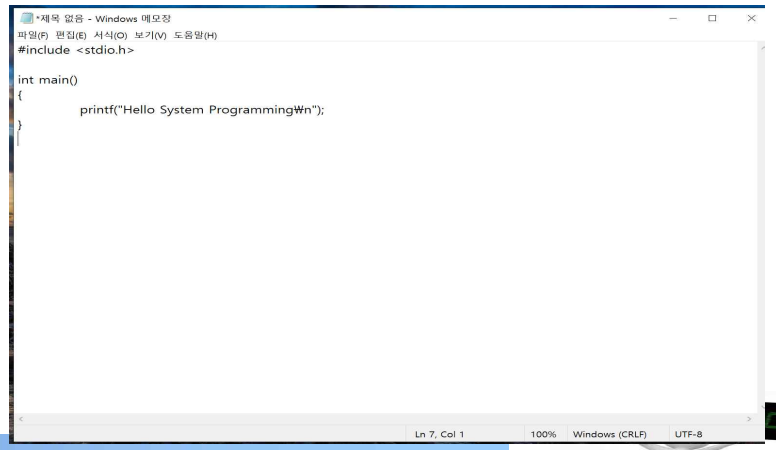
How to use commands in Linux (5/12)

- vi editor (vim)
  - ✓ What are the differences between vi and nodepad (or VS code)
    - Instant editable (explicit input mode)
    - No “file” or “Format” button (need line command mode)



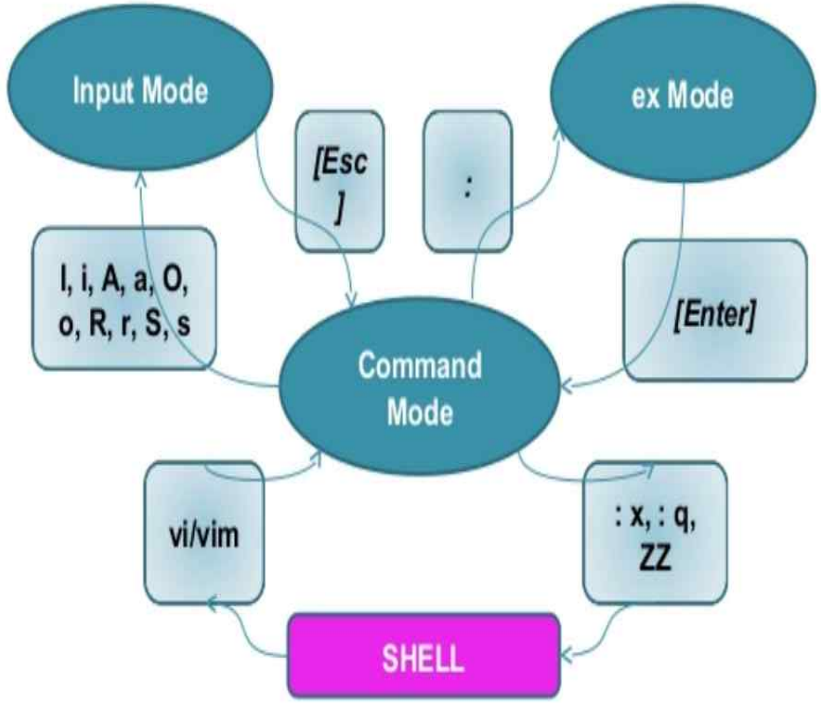






How to use commands in Linux (6/12)

- vi editor (vim)
  - ✓ 3 modes
    - command/input/line command(a.k.a. execution mode)
  - ✓ At first (just before loading vi): command mode
  - ✓ Switch to the input mode
    - a (append), i (insert), o, r, …
  - ✓ Switch to the command mode
    - ESC
  - ✓ Switch to the line command mode
    - : at command mode
  - ✓ Switch to the command mode
    - Enter or ESC



(Source: https://www.slideshare.net/TusharadriSarkar/vim-vi-improved-23917134)

How to use commands in Linux (7/12)

- vi editor (vim)
  - ✓ Actions allowed at the command/line command mode
    - Navigation (cursor movement): up/down, begin/end of word/line, …
    - File management: save, quit (e.g. :wq or :q), open, …
    - Editing: delete, change, substitute, transpose, …
    - Multiple windows, files, shell interaction, …

Vim: Navigation

Keystroke	Function
B/b	Move cursor to bottom of page *
E/e	Move cursor to end of word *
0 (Zero) / |	Move cursor to beginning of line *
$	Move cursor to end of line
)	Move cursor to beginning of next sentence
(	Move cursor to beginning of current sentence
G	Move cursor to end of file *
%	Move cursor to the matching bracket; Place cursor on {}[]()
'. (Apostrophe dot)	Move cursor to previously modified line
'a (Apostrophe a)	Move cursor to line mark "a" generated by marking "ma"

Advanced editing: Multiple Windows

This is a Vim only feature

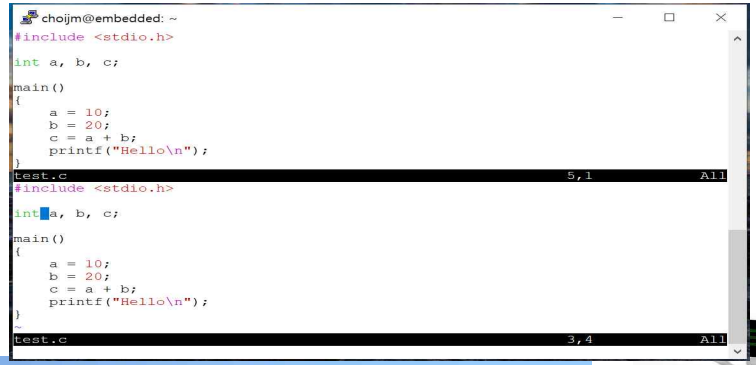
Command	Function
:sp	Split current window horizontally in two
:vsp	Split current window vertically into two
vim –O [n | files…]	Opens n windows, files split vertically
:new	Open a new blank window
:on	Make current window the only window
:q	Quit current window
:qa	Quit all windows
:xa	Save and quit all windows
[Ctrl+w]+/-	Increase/decrease window size
[Ctrl+w] [Ctrl+w]	Toggle between windows

Pattern Substitutions

- General format of substitution
  :[.|$|%]s/s1/s1[switches] or :n1,n2s/s1/s2/[switches]
- [switches] are: g|c|i|I meaning
  global/confirmation/ignore-case/no-ignore-case

Some interesting examples of pattern substitutions

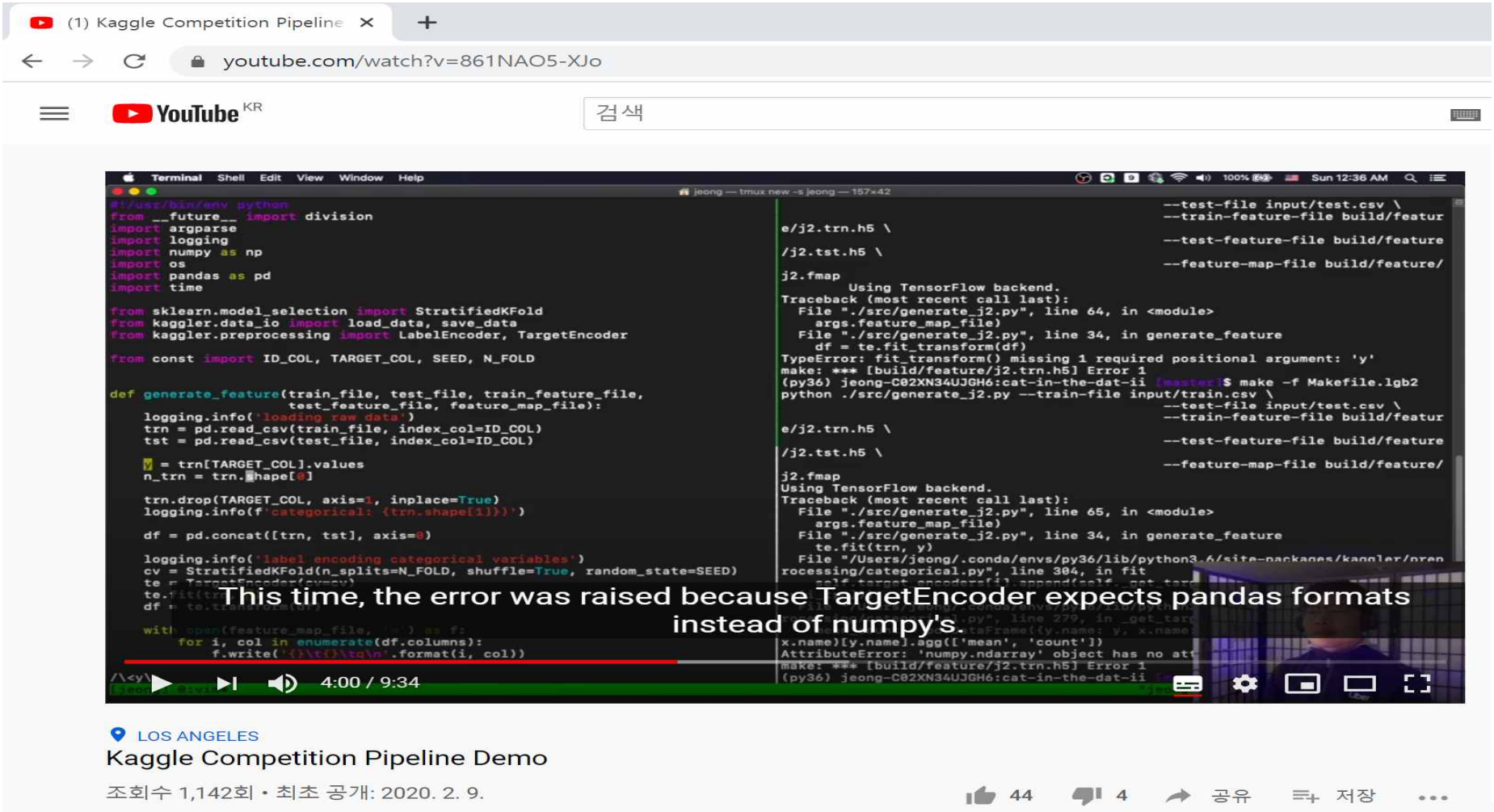
Command	Function
:1,$s/#//g	Globally remove #
:3,10s/^/#/	Insert # at the beginning of line 3 to 10
:$s/$/;/	Insert a ; at the end of last line
:%s/abc/xyz/gc	Globally replace abc by xyz interactively
:1,$s/include/<&>/g	Globally replace include by <include>



(Source: https://www.slideshare.net/TusharadriSarkar/vim-vi-improved-23917134)

How to use commands in Linux (8/12)

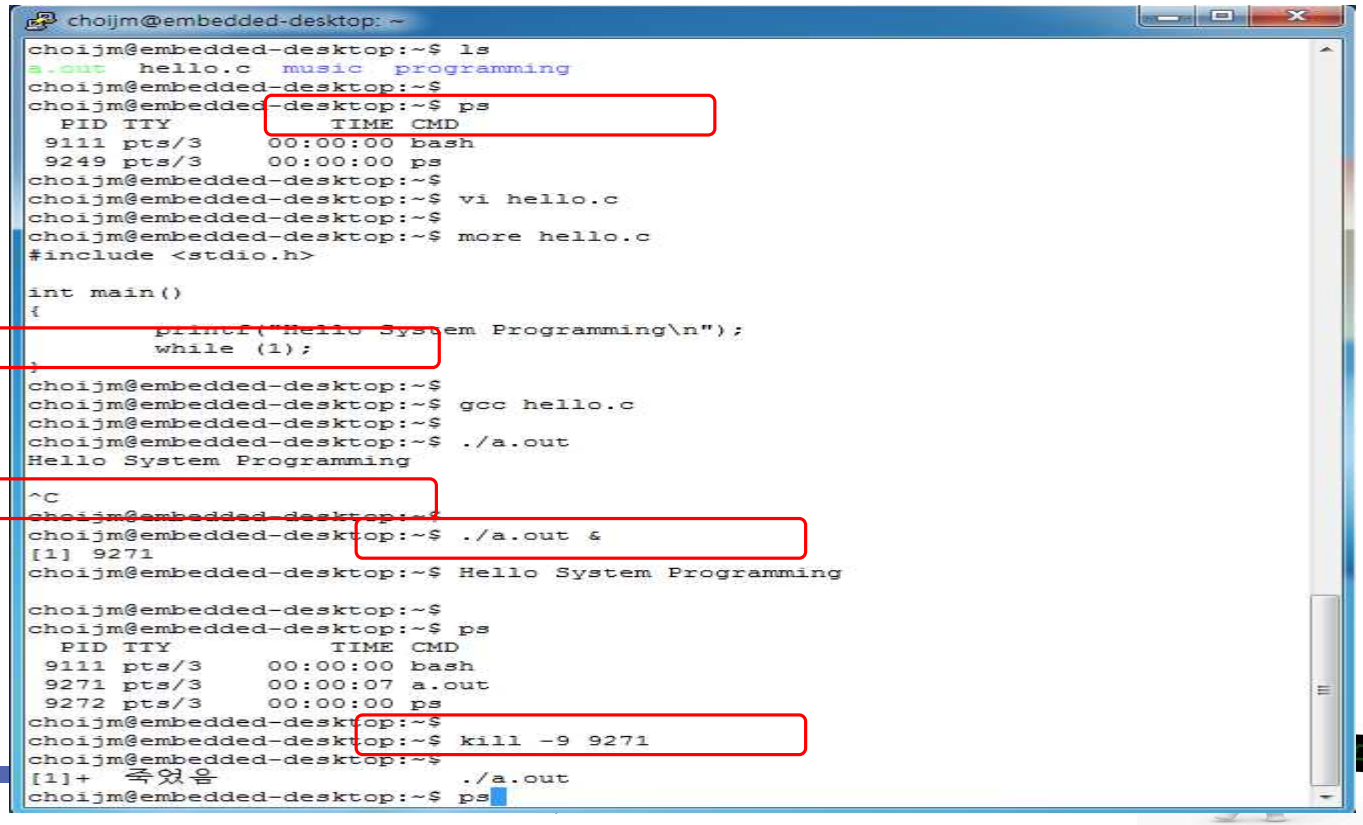
- Reference: Dr. Jeong-Yoon Lee's Kaggle demo (terminal mode)



(Source: https://www.youtube.com/watch?v=861NAO5-XJo)

How to use commands in Linux (9/12)

- process related commands
  - ✓ process status
    - ps, pstree, top, /proc
  - ✓ Creation and deletion
    - Implicitly: using shell (fork(), execve() and exit() internally)
    - Explicitly: signal, kill command



How to use commands in Linux (10/12)

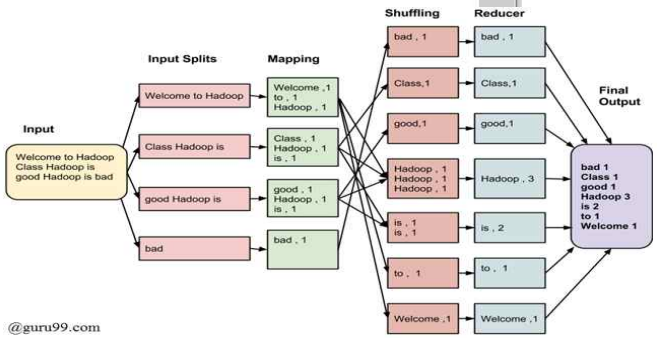
- Advanced commands: pipe

```
choijm@embedded: ~
choijm@embedded:~$ pwd
/home/choijm
choijm@embedded:~$ ls -l
total 56
-rwxrwxr-x  1 choijm choijm 4676 11월  19  2018 a.out
drwxr-xr-x  9 choijm choijm 4096  9월   5 11:51 Desktop
-rw-r--r--  1 choijm choijm 8980  8월  31  2018 examples.desktop
drwxrwxr-x  2 choijm choijm 4096  3월  13 09:22 OSTEP
drwxr-xr-x  2 choijm choijm 4096  9월   5  2019 Public
drwxrwxr-x 10 choijm choijm 4096 11월  20  2019 Syspro
-rw-rw-r--  1 choijm choijm   95  9월  17  2019 test.c
-rw-rw-r--  1 choijm choijm  517  9월  17  2019 test.s
-rwxrwxr-x  1 choijm choijm 4880  3월  13 09:27 virt_cpu
-rw-rw-r--  1 choijm choijm  269  3월  13 09:27 virt_cpu.c
choijm@embedded:~$
choijm@embedded:~$ ls -l | sort
drwxrwxr-x 10 choijm choijm 4096 11월  20  2019 Syspro
drwxrwxr-x  2 choijm choijm 4096  3월  13 09:22 OSTEP
drwxr-xr-x  2 choijm choijm 4096  9월   5  2019 Public
drwxr-xr-x  9 choijm choijm 4096  9월   5 11:51 Desktop
-rw-r--r--  1 choijm choijm 8980  8월  31  2018 cxamplcs.dcsktop
-rw-rw-r--  1 choijm choijm  269  3월  13 09:27 virt_cpu.c
-rw-rw-r--  1 choijm choijm  517  9월  17  2019 test.s
-rw-rw-r--  1 choijm choijm   95  9월  17  2019 test.c
-rwxrwxr-x  1 choijm choijm 4676 11월  19  2018 a.out
-rwxrwxr-x  1 choijm choijm 4880  3월  13 09:27 virt_cpu
total 56
choijm@embedded:~$ ls -l | sort -k5n
total 56
-rw-rw-r--  1 choijm choijm   95  9월  17  2019 test.c
-rw-rw-r--  1 choijm choijm  269  3월  13 09:27 virt_cpu.c
-rw-rw-r--  1 choijm choijm  517  9월  17  2019 test.s
drwxrwxr-x 10 choijm choijm 4096 11월  20  2019 Syspro
drwxrwxr-x  2 choijm choijm 4096  3월  13 09:22 OSTEP
drwxr-xr-x  2 choijm choijm 4096  9월   5  2019 Public
drwxr-xr-x  9 choijm choijm 4096  9월   5 11:51 Desktop
-rwxrwxr-x  1 choijm choijm 4676 11월  19  2018 a.out
-rwxrwxr-x  1 choijm choijm 4880  3월  13 09:27 virt_cpu
-rw-r--r--  1 choijm choijm 8980  8월  31  2018 examples.desktop
choijm@embedded:~$
choijm@embedded:~$ ls -l | sort -k5n | wc -l
11
choijm@embedded:~$
```

How to use commands in Linux (11/12)

- Advanced commands: pipe, redirection and background

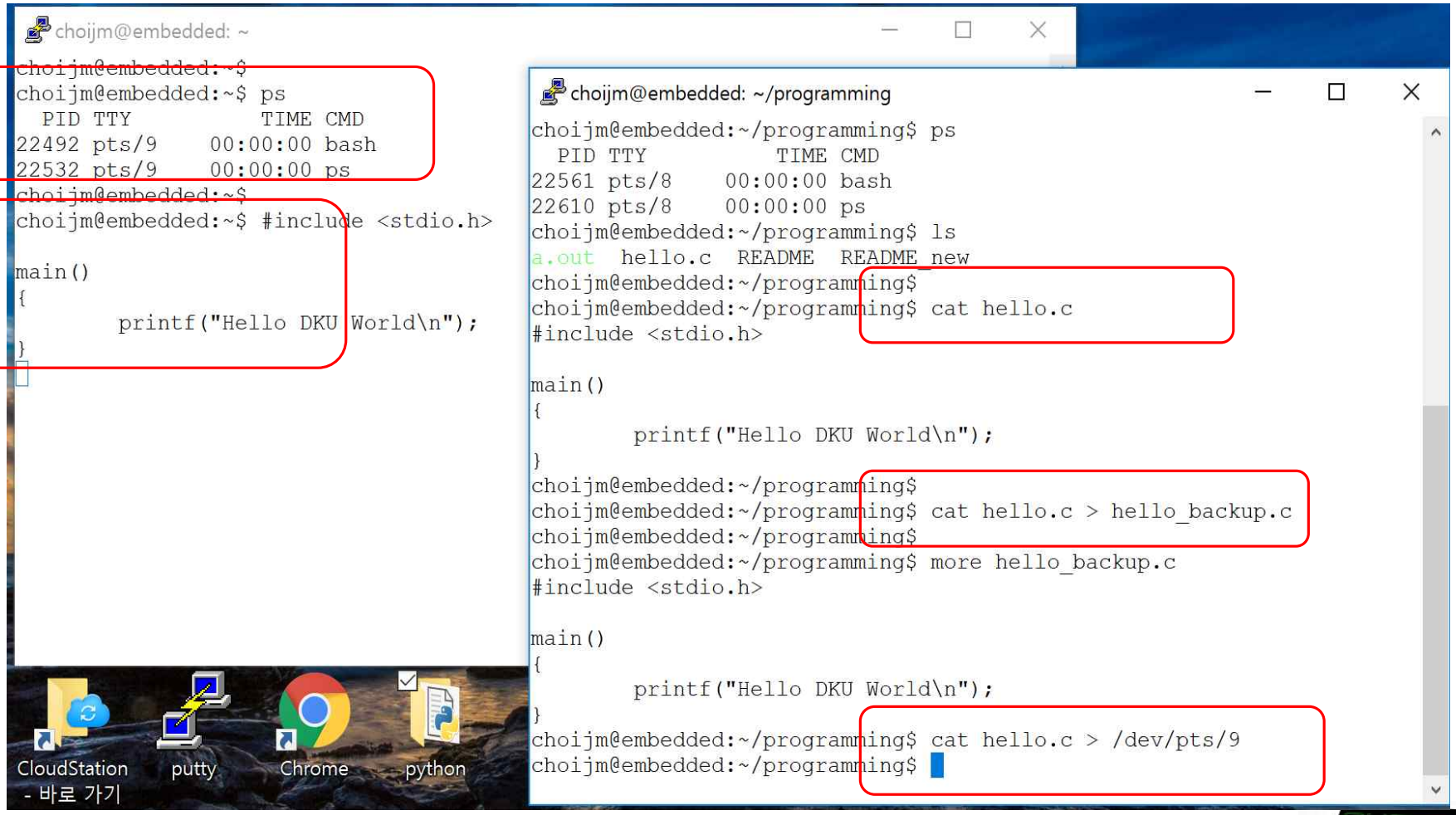
```
choijm@embedded: ~
choijm@embedded:~$ ls
a.out     examples.desktop  Public  test.c  virt_cpu
Desktop   OSTEP             Syspro  test.s  virt_cpu.c
choijm@embedded:~$
choijm@embedded:~$ man pipe
choijm@embedded:~$
choijm@embedded:~$ man pipe > man_pipe_output.txt
choijm@embedded:~$
choijm@embedded:~$ ls
a.out     examples.desktop     OSTEP   Syspro  test.s     virt_cpu.c
Desktop   man_pipe_output.txt  Public  test.c  virt_cpu
choijm@embedded:~$
choijm@embedded:~$ grep -o process man_pipe_output.txt | wc -l
4
choijm@embedded:~$ grep -o file man_pipe_output.txt | wc -l
7
choijm@embedded:~$ grep -o O_NONBLOCK man_pipe_output.txt | wc -l
2
choijm@embedded:~$
choijm@embedded:~$ grep -o process man_pipe_output.txt | wc -l &
[1] 4283
choijm@embedded:~$ 4

[1]+  Done                    grep --color=auto -o process man_pipe_output.txt |
 wc -l
choijm@embedded:~$
choijm@embedded:~$ (grep -o process man_pipe_output.txt | wc -l) & (grep -o file
 man_pipe_output.txt | wc -l) & (grep -o O_NONBLOCK man_pipe_output.txt | wc -l)
 &
[1] 4290
[2] 4291
[3] 4292
choijm@embedded:~$ 4
7
2

[1]   Done                    ( grep --color=auto -o
 | wc -l )
[2]-  Done                    ( grep --color=auto -o
wc -l )
[3]+  Done                    ( grep --color=auto -o
txt | wc -l )
choijm@embedded:~$
```



(➔ See LN1)

How to use commands in Linux (12/12)

- Generalization of file concept
  - ✓ Treat device, socket, IPC as a file



How to make and run a program in Linux (1/6)

- Overall

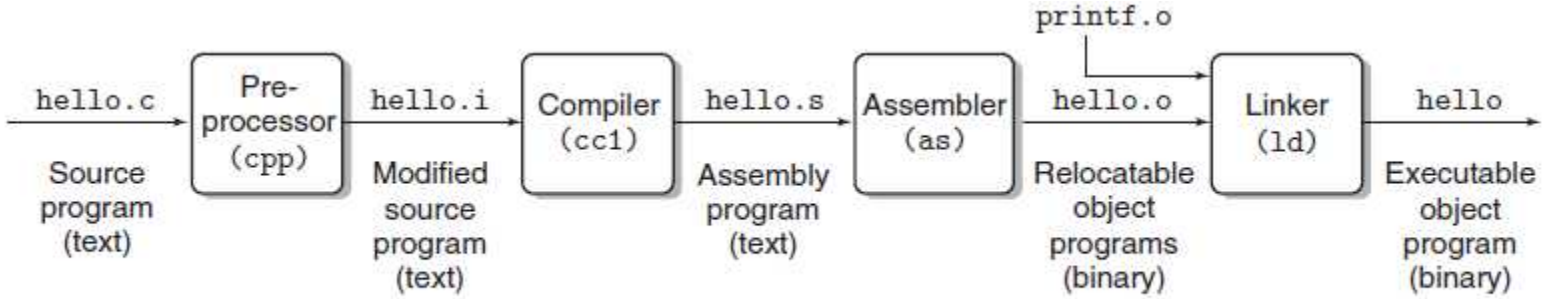
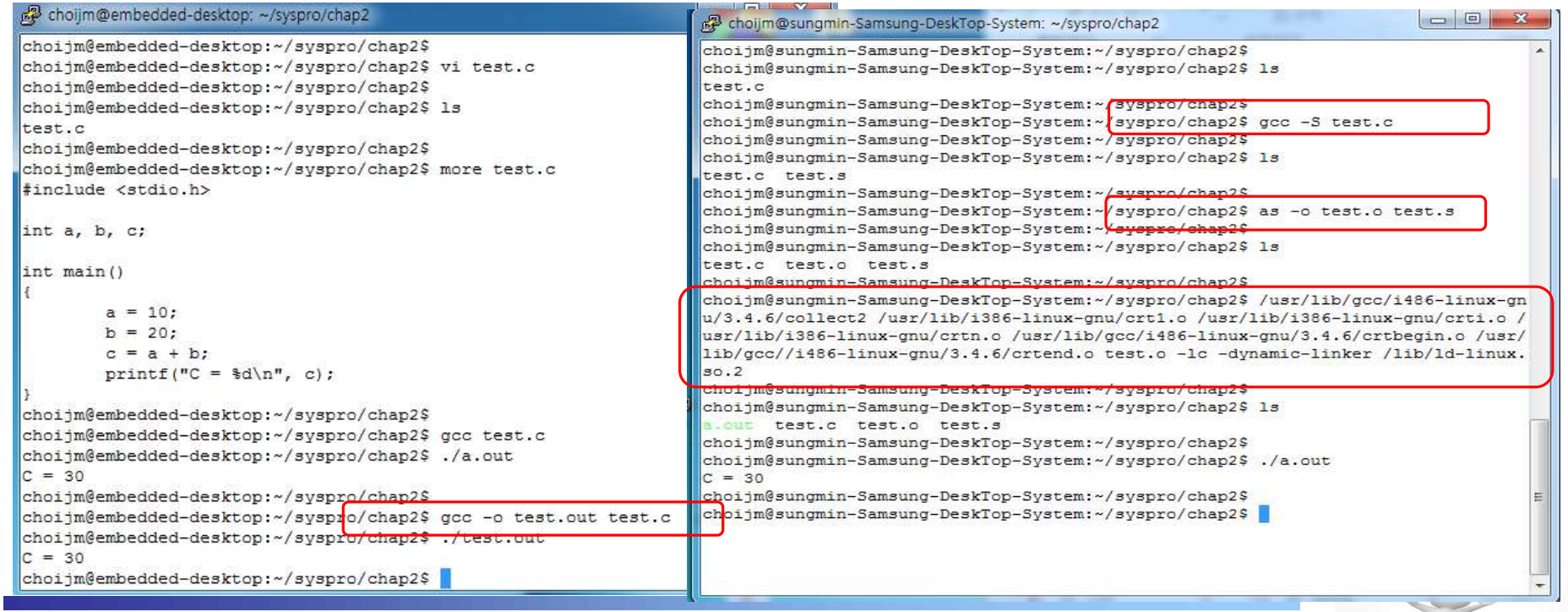


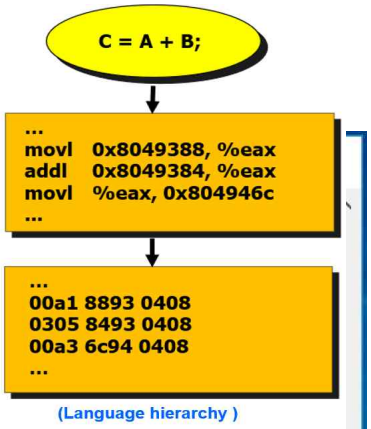
Figure 1.3 The compilation system.

(Source: computer systems: a programmer perspective, Figure 1.3)



How to make and run a program in Linux (2/6)

- Assembly code



choijm@embedded: ~/syspro18/chap2

```
choijm@embedded:~/syspro18/chap2$ gcc -S test.c
choijm@embedded:~/syspro18/chap2$ more test.s
        .file   "test.c"
        .section        .rodata
.LC0:
        .string "C = %d\n"
        .text
.globl main
        .type   main, @function
main:
        pushl   %ebp
        movl    %esp, %ebp
        subl    $8, %esp
        andl    $-16, %esp
        movl    $0, %eax
        addl    $15, %eax
        addl    $15, %eax
        shrl    $4, %eax
        sall    $4, %eax
        subl    %eax, %esp
        movl    $10, a
        movl    $20, b
        movl    b, %eax
        addl    a, %eax
        movl    %eax, c
        movl    c, %eax
        movl    %eax, 4(%esp)
        movl    $.LC0, (%esp)
        call    printf
        leave
        ret
        .size   main, .-main
        .comm   a,4,4
        .comm   b,4,4
        .comm   c,4,4
        .section        .note.GNU-stack,"",@progbits
        .ident  "GCC: (GNU) 3.4.6 (Debian 3.4.6-5)"
choijm@embedded:~/syspro18/chap2$
choijm@embedded:~/syspro18/chap2$
```

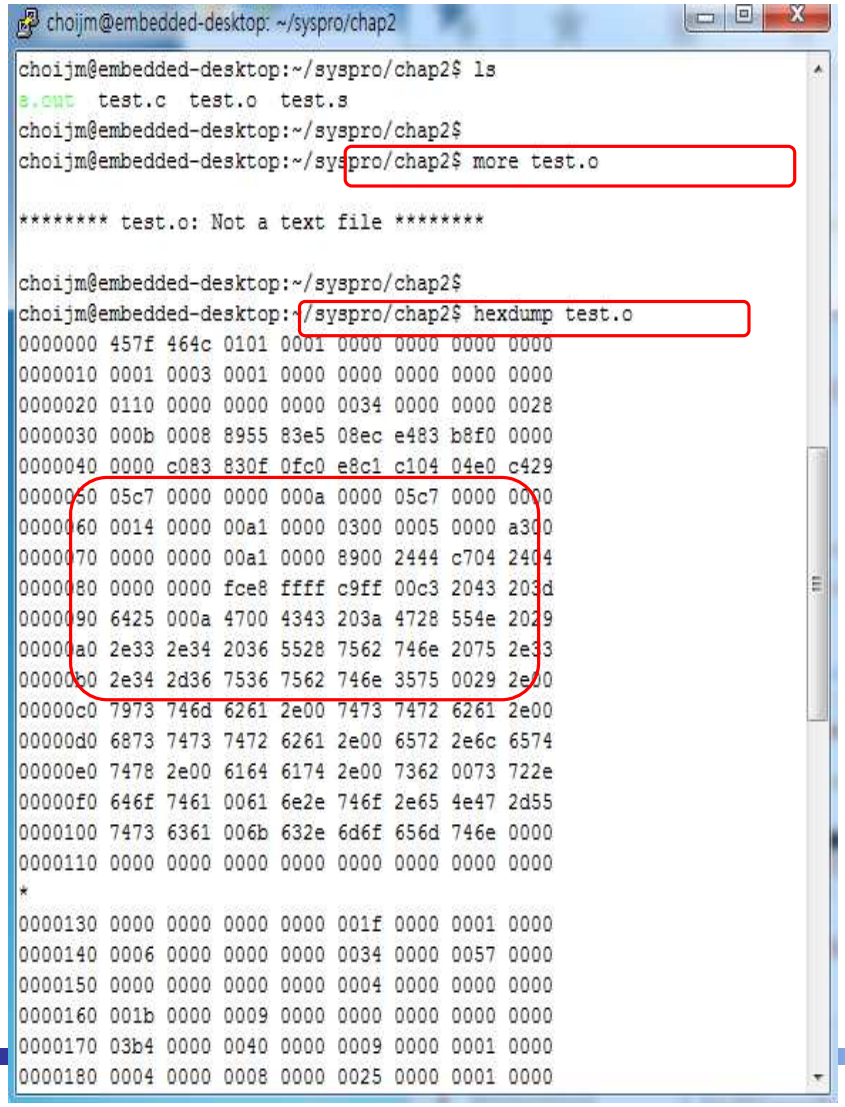
choijm@sysprog1: ~

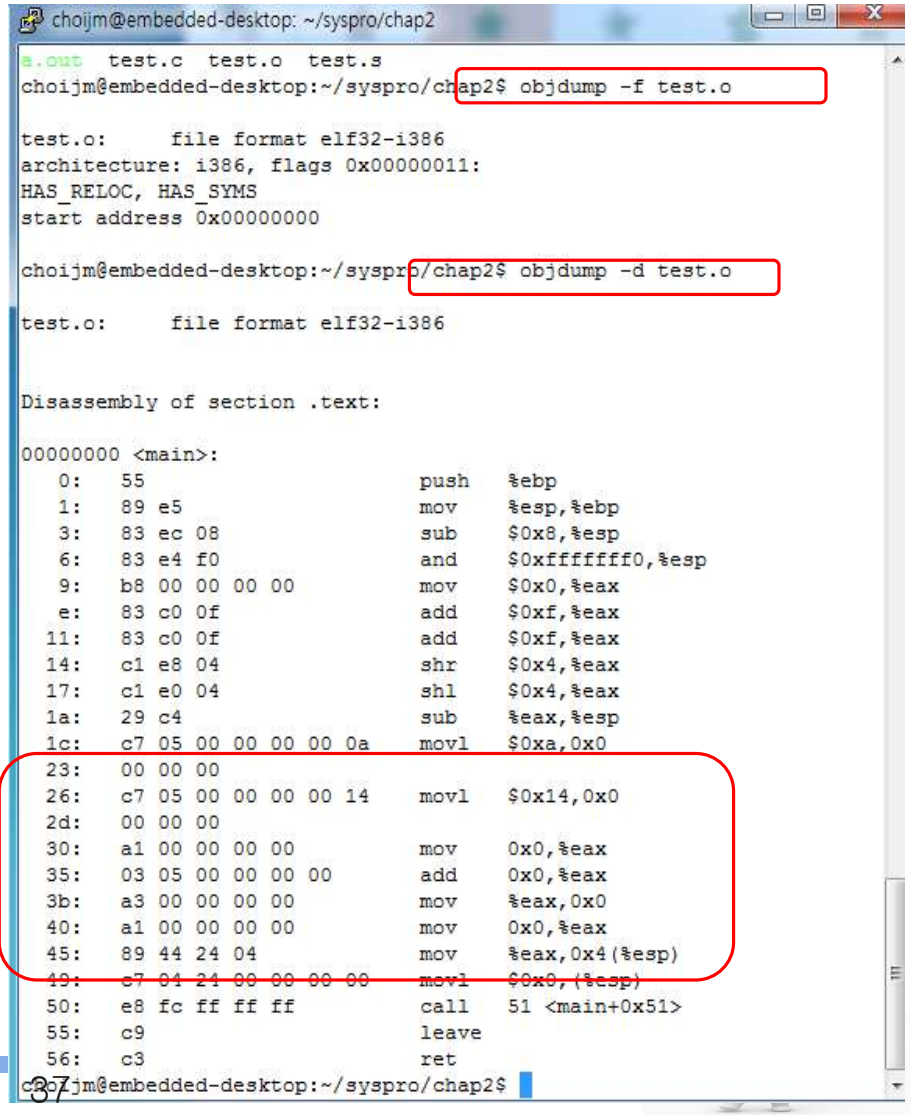
```
choijm@sysprog1:~$ more test.s
        .file   "test.c"
        .section        .rodata
.LC0:
        .string "C=%d\n"
        .text
.globl main
        .type   main, @function
main:
.LFB2:
        pushq   %rbp
.LCFI0:
        movq    %rsp, %rbp
.LCFI1:
        movl    $10, a(%rip)
        movl    $20, b(%rip)
        movl    b(%rip), %eax
        addl    %eax, c(%rip)
        movl    c(%rip), %esi
        movl    $.LC0, %edi
        movl    $0, %eax
        call    printf
        leave
        ret
.LFE2:
        .size   main, .-main
        .comm   a,4,4
        .comm   b,4,4
        .comm   c,4,4
        .section        .eh_frame,"a",@progbits
.Lframe1:
        .long   .LECIE1-.LSCIE1
.LSCIE1:
        .long   0x0
        .byte   0x1
        .string ""
        .uleb128 0x1
        .sleb128 -8
---More--(54%)
```

☞ Can be different based on the version of kernel and compiler

How to make and run a program in Linux (3/6)

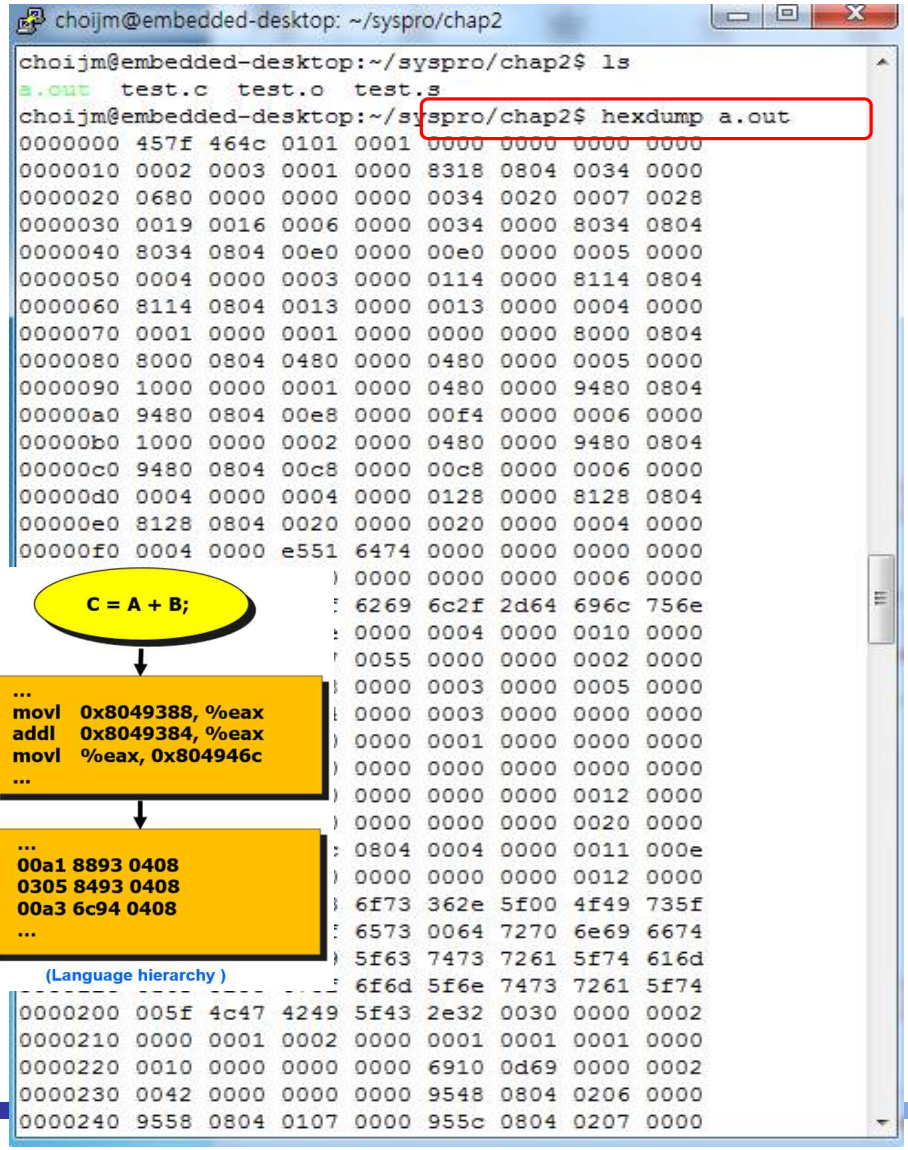
- Relocatable code
  - ✓ Hexdump (or xxd), objdump

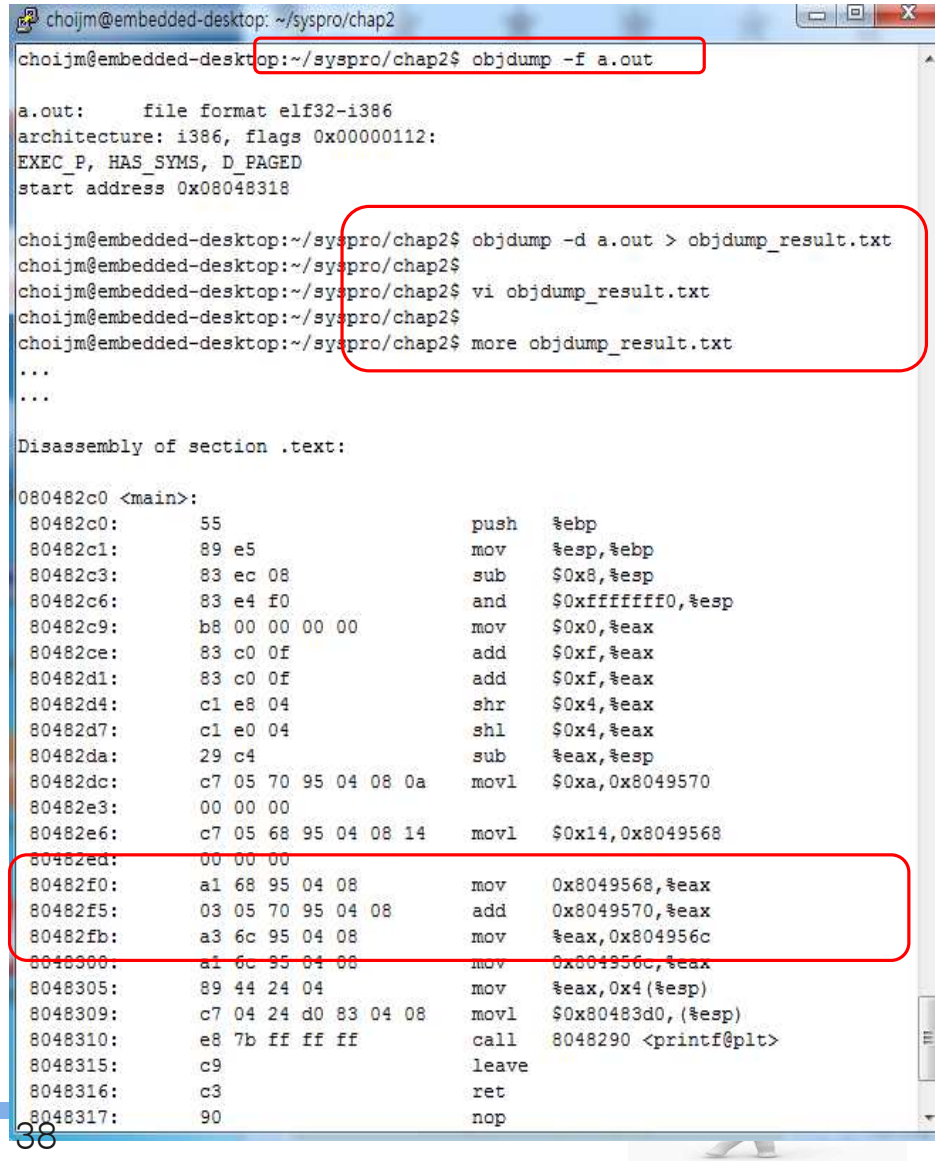




How to make and run a program in Linux (4/6)

- Executable code





How to make and run a program in Linux (5/6)

- What are the execution results of this program?

How to make and run a program in Linux (6/6)

- debugger

```
choijm@embedded-desktop: ~/syspro/gdb_exam
choijm@embedded-desktop:~/syspro/gdb_exam$
choijm@embedded-desktop:~/syspro/gdb_exam$ vi gdb_test.c
choijm@embedded-desktop:~/syspro/gdb_exam$
choijm@embedded-desktop:~/syspro/gdb_exam$ cat gdb_test.c
#include <stdio.h>

int a[4] = {5, 6, 7, 8};
int *pa;

main()
{
        printf("%d\n", a[0]);
        printf("%d\n", a[2]);
        printf("%d\n", *a);
        printf("%d\n", *(a+2));
        printf("%d\n", *pa);
        printf("%d\n", *(pa+2));
}
choijm@embedded-desktop:~/syspro/gdb_exam$
choijm@embedded-desktop:~/syspro/gdb_exam$ gcc -o gdb_test.out gdb_test.c
choijm@embedded-desktop:~/syspro/gdb_exam$
choijm@embedded-desktop:~/syspro/gdb_exam$ ./gdb_test.out
5
7
5
7
세그멘테이션 오류 (core dumped)
choijm@embedded-desktop:~/syspro/gdb_exam$
choijm@embedded-desktop:~/syspro/gdb_exam$
```

```
choijm@embedded-desktop: ~/syspro/gdb_exam
choijm@embedded-desktop:~/syspro/gdb_exam$
choijm@embedded-desktop:~/syspro/gdb_exam$ gcc -g -o gdb_test.out gdb_test.c
choijm@embedded-desktop:~/syspro/gdb_exam$
choijm@embedded-desktop:~/syspro/gdb_exam$ gdb gdb_test.out
GNU gdb (Ubuntu/Linaro 7.4-2012.04-0ubuntu2.1) 7.4-2012.04
Copyright (C) 2012 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.  Type "show copying"
and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
For bug reporting instructions, please see:
<http://bugs.launchpad.net/gdb-linaro/>...
Reading symbols from /home/choijm/syspro/gdb_exam/gdb_test.out...done.
(gdb)
(gdb) run
Starting program: /home/choijm/syspro/gdb_exam/gdb_test.out
warning: no loadable sections found in added symbol-file system-supplied DSO at
 0x7ffff7ffa000
5
7
5
7

Program received signal SIGSEGV, Segmentation fault.
0x0000000000400567 in main () at gdb_test.c:12
12              printf("%d\n", *pa);
(gdb) list
7       {
8               printf("%d\n", a[0]);
9               printf("%d\n", a[2]);
10              printf("%d\n", *a);
11              printf("%d\n", *(a+2));
12              printf("%d\n", *pa);
13              printf("%d\n", *(pa+2));
14      }
(gdb)
Line number 15 out of range; gdb_test.c has 14 lines.
(gdb) break 10
Breakpoint 1 at 0x40052c: file gdb_test.c, line 10.
(gdb) run
The program being debugged has been started already.
Start it from the beginning? (y or n) y

Starting program: /home/choijm/syspro/gdb_exam/gdb_test.out
warning: no loadable sections found in added symbol-file system-supplied DSO at
 0x7ffff7ffa000
5
7

Breakpoint 1, main () at gdb_test.c:10
10              printf("%d\n", *a);
(gdb) n
5
11              printf("%d\n", *(a+2));
(gdb)
```

☞ **There are various valuable debugger commands such as breakpoint, step, info reg, ...**
See http://beej.us/guide/bggdb/

Summary

- Discuss the features of Linux
- Understand the commands related to file and process
- Explore the language hierarchy in Linux (UNIX)

☞ **Homework 2.**

1.1 Make a file using vi editor that contains your favorite poem

1.2 Make a snapshot that

- has at least 10 commands.

- shows student's ID and date (using whoami and date)

- Server IP: 220.149.236.2 or 220.149.236.4

1.3 Bonus: includes the compilation practice (eg. gcc, as, gdb)

1.4 Deadline: 6 PM Friday of the next week (2nd, October)

Appendix 1. Snapshot Example

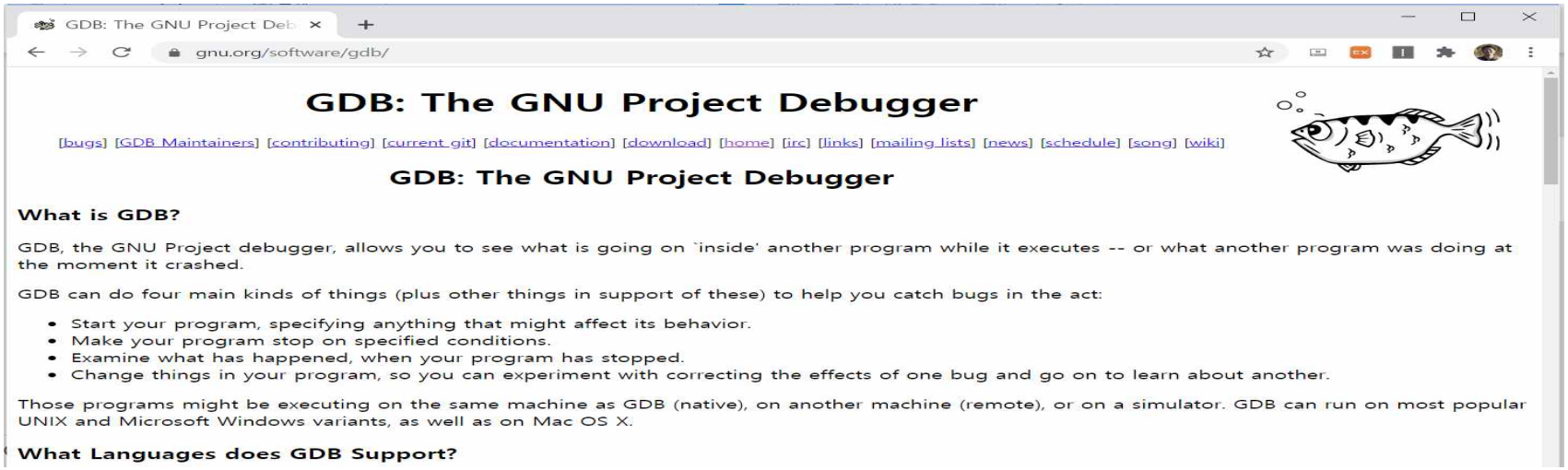
- Example

```
choijm@localhost:~/syspro_20130902/reports
[choijm@localhost reports]$
[choijm@localhost reports]$ ls
a.out  hello.c  music  my_favorite_poem.txt  subdir
[choijm@localhost reports]$
[choijm@localhost reports]$ more my_favorite_poem.txt
나 하늘로 돌아가리라.
새벽빛 와 닿으면 스러지는
이슬 더불어 손에 손을 잡고,

나 하늘로 돌아가리라.
노을빛 함께 단 둘이서
기슭에서 놀다가 구름 손짓하며는,

나 하늘로 돌아가리라.
아름다운 이 세상 소풍 끝내는 날,
가서, 아름다웠더라고 말하리라.
[choijm@localhost reports]$
[choijm@localhost reports]$
```

```
choijm@embedded: ~/Syspro/chap2
choijm@embedded:~/Syspro/chap2$ vi gdb_test.c
choijm@embedded:~/Syspro/chap2$ gcc -g gdb_test.c
choijm@embedded:~/Syspro/chap2$ ./a.out
5
7
5
7
Segmentation fault (core dumped)
choijm@embedded:~/Syspro/chap2$ gdb a.out
GNU gdb (Ubuntu 7.11.1-0ubuntu1~16.5) 7.11.1
Copyright (C) 2016 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.  Type "show copying"
and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.
For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from a.out...done.
(gdb) run
Starting program: /home/choijm/Syspro/chap2/a.out
5
7
5
7

Program received signal SIGSEGV, Segmentation fault.
0x080483f7 in main () at gdb_test.c:12
12
       printf("%d\n", *pa);
(gdb) quit
A debugging session is active.

        Inferior 1 [process 4497] will be killed.

Quit anyway? (y or n) y
choijm@embedded:~/Syspro/chap2$ mkdir reports
choijm@embedded:~/Syspro/chap2$ vi my_favorite_poem.txt
choijm@embedded:~/Syspro/chap2$ grep include gdb_test.c | wc
      1       2      19
choijm@embedded:~/Syspro/chap2$ whoami
choijm
choijm@embedded:~/Syspro/chap2$ date
2020. 09. 05. (토) 20:57:56 KST
choijm@embedded:~/Syspro/chap2$
```

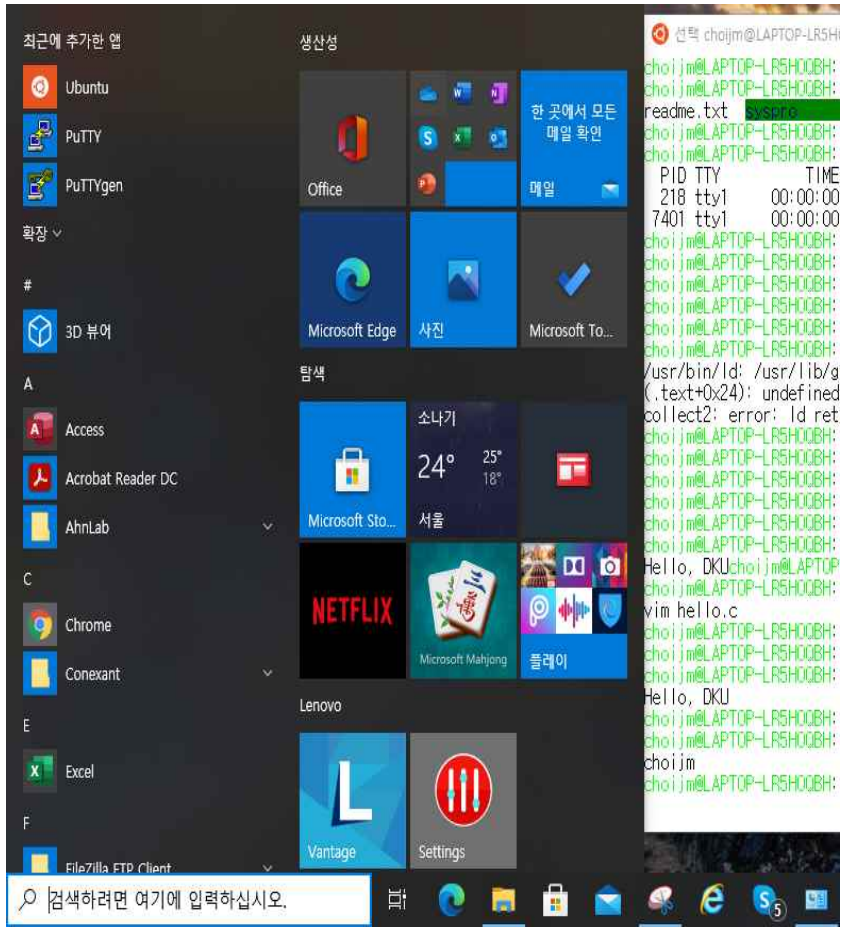
Quiz for 4th-Week 1st-Lesson

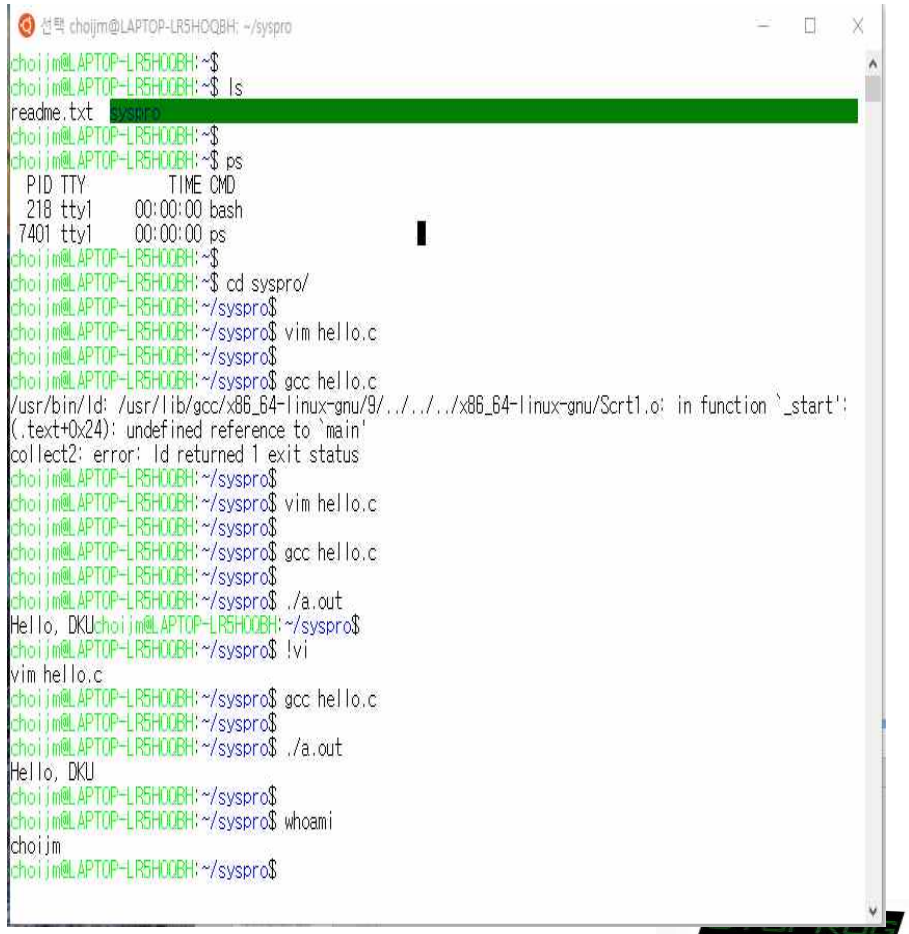
- Quiz
  - ✓ 1) What are the differences when we do “$cat hello.c > a.txt” and “$cat hello.c > /dev/pts/2”?
  - ✓ 2) What is the background music in “Dr Jeong-Joon Lee’s Kaggle Demo” in Page 30”?
  - ✓ Bonus) What commands can you find in the Kaggle Demo? (at least 5, only what you have learned in the LN2.)
  - ✓ Due: until 6 PM Friday of this week (25th, September)



Appendix 2: How to access Linux: Alternative

- WSL (Windows Subsystem for Linux)
  - ✓ a compatibility layer for running Linux binary executables (in ELF format) natively on Windows OS





Appendix 2: How to access Linux: Alternative

- Toast Cloud (or Amazon EC2 or Google)
  - ✓ Supported by NHN (like Amazon EC2 or Google Compute Engine)
  - ✓ Using PaaS in this course
    - IP: 133.186.152.119 (May be different per each VM instance)
    - For general users: same as the Linux server