Lecture Note 2.
Programming Environment

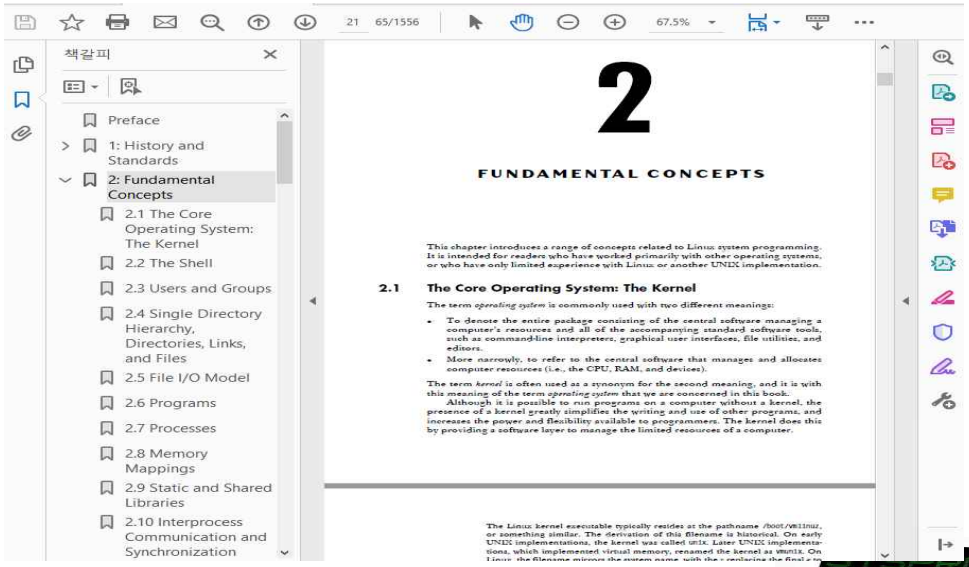
September 7, 2021

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

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

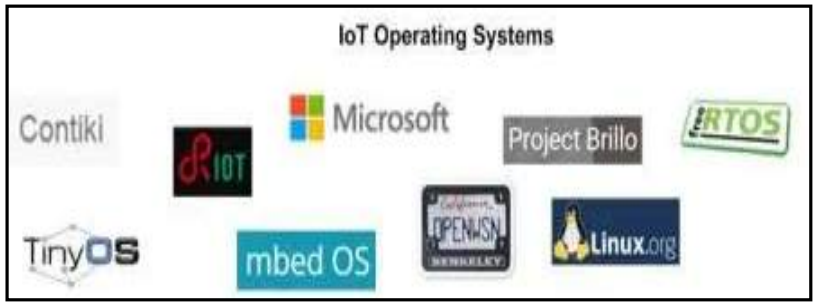


Linux Introduction (1/7)

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

(Source: IEEE Spectrum, 2001)

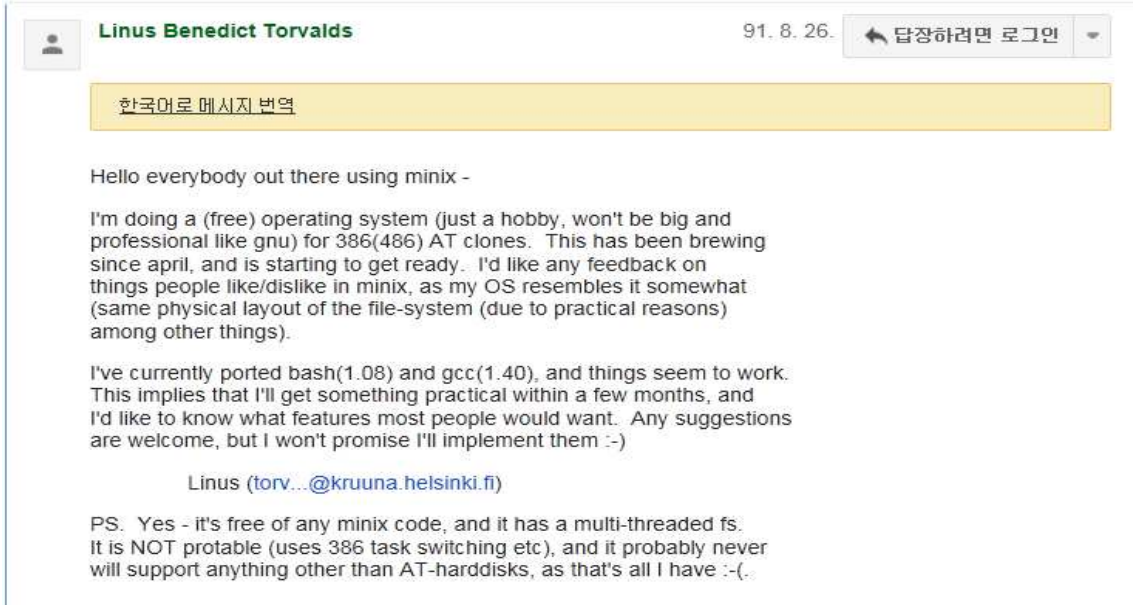
(source: https://www.deviantart.com/nick-os/art/Os-war-choose-your-poison-110510677)



(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 Developers 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)

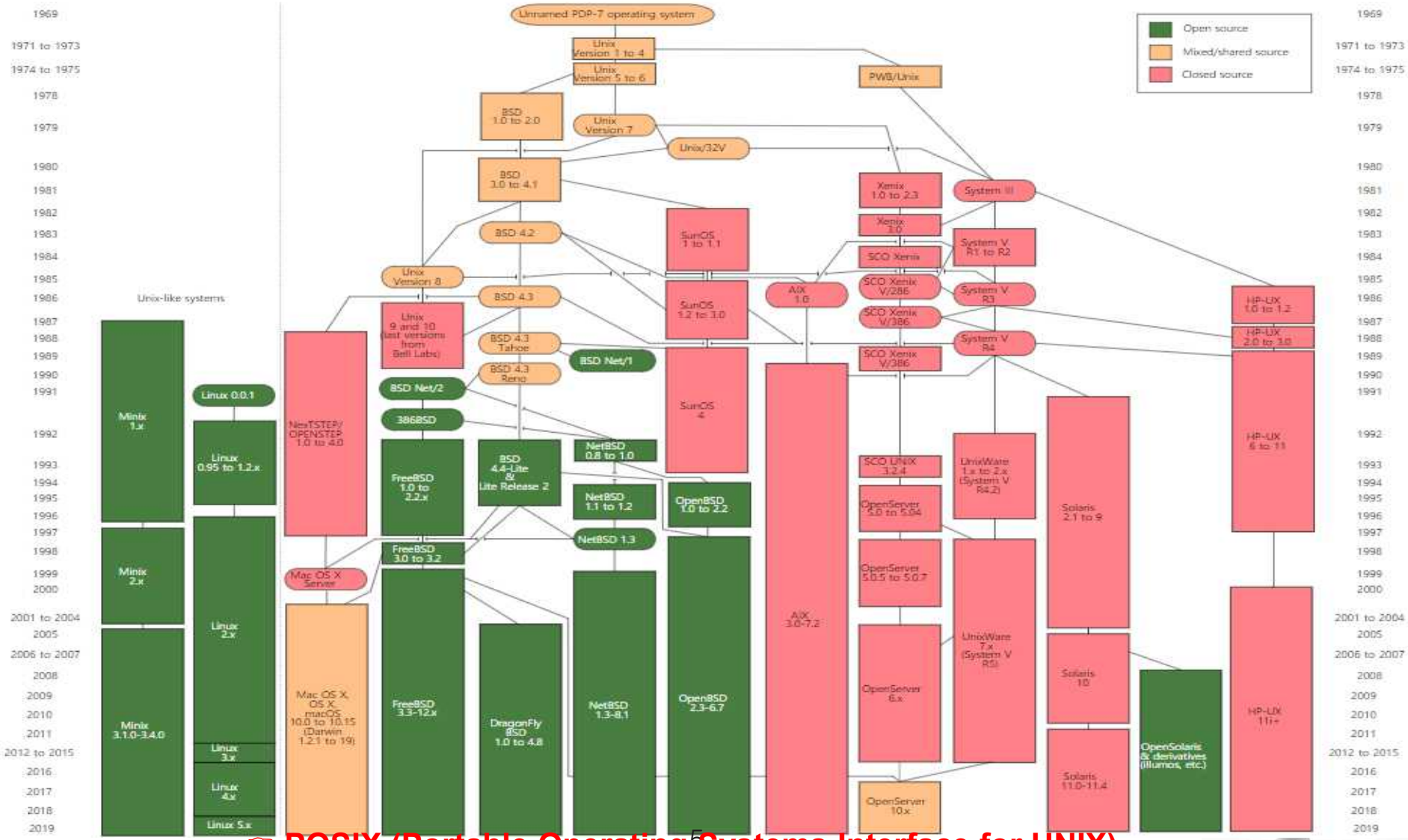
PS. Yes - it's free of any minix code, and it has a multi-threaded fs. It is NOT protable (uses 386 task switching etc), and it probably never will support anything other than AT-harddisks, as that's all I have :-(.

  - ✓ Univ. of Helsinki in Finland
  - ✓ May, 1991: Release 0.0.1 version
  - ✓ 7. September, 2021: Release 5.14.1 (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

보안 연결 | 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]

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

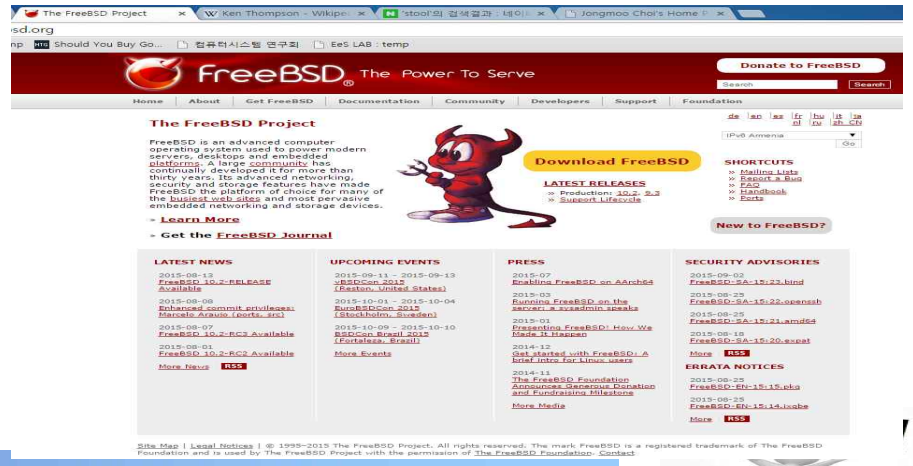
Kenneth Thompson

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

ROS
Robot Operating System Logo

(Source: images at google)

Linux Introduction (7/7)

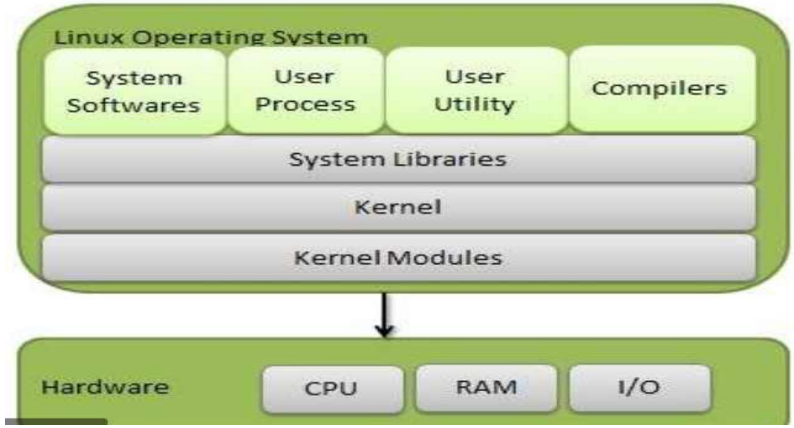
- Some notes about UNIX and Linux (From LPI Chapter 1)
  - ✓ Linux is a member of the UNIX family
  - ✓ History
    - 1969~ : UNIX Invented by Ken and Dennis, UNIX 1~7 edition at AT&T
    - 1975~ : popularly used at universities include Berkeley, MIT and CMU.
    - 1979~ : BSD and new features (FFS, 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.)
    - 2021: Three representative OSes + Vendor proprietary OSes + New OSes

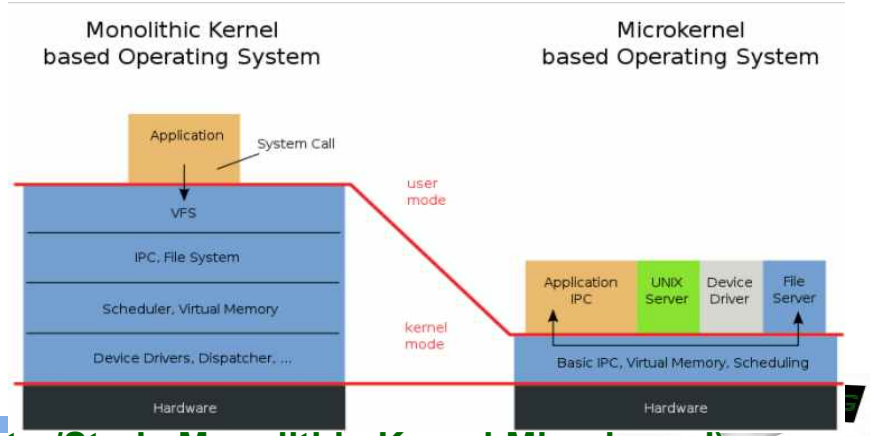
    - 1984~ : GNU by R. Stallman (gcc, Emacs, bash, …), GPL (General Public License)
    - 1991~ : Linux by L. Torvalds, Minix + Intel optimization, GNU incorporation
    - 2021: Linux kernel version 5.14.1

  - ✓ 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

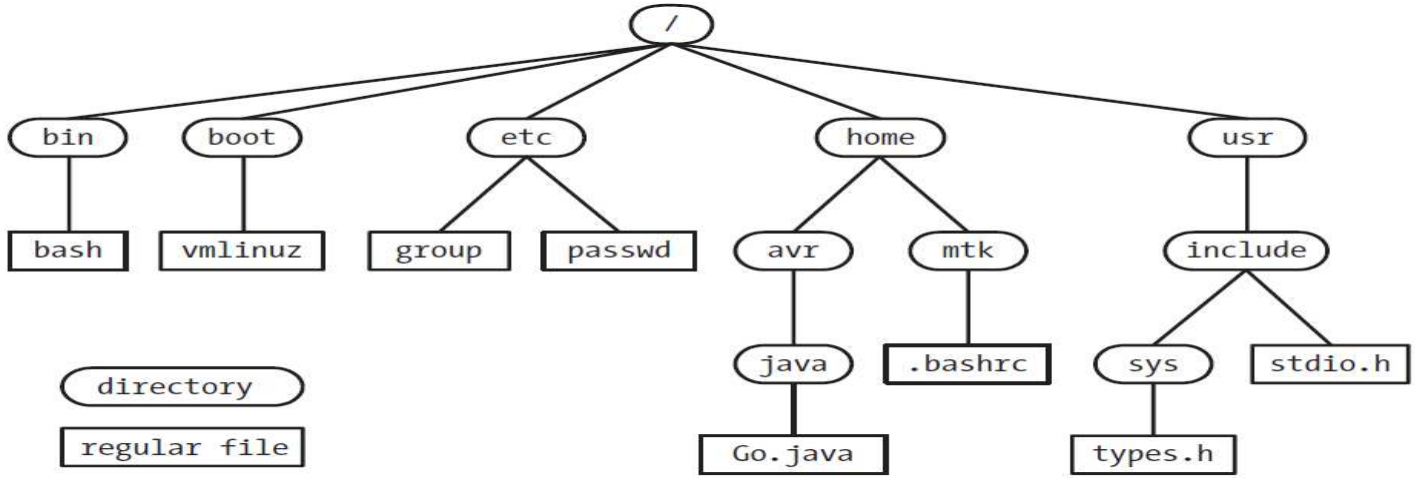


Figure 2-1: Subset of the Linux single directory hierarchy

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) Discuss the difference between OS (Operating System) and Kernel (using the below figure)
  - ✓ Due: until 6 PM Friday of this week (17th, September)

1.1 A Brief History of UNIX and C

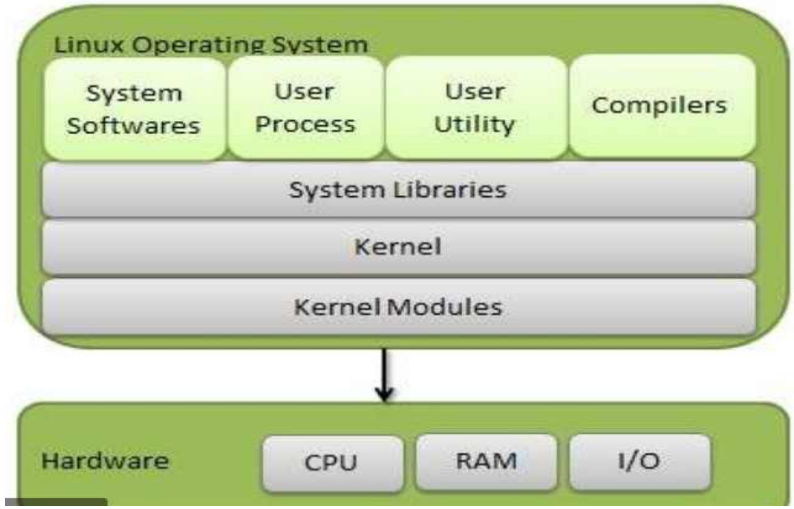
The first UNIX implementation was developed in 1969 (the same year that Linus Torvalds was born) by Ken Thompson at Bell Laboratories, a division of the telephone corporation, AT&T. It was written in assembler for a Digital PDP-7 minicomputer. The name UNIX was a pun on MULTICS (*Multiplexed Information 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.

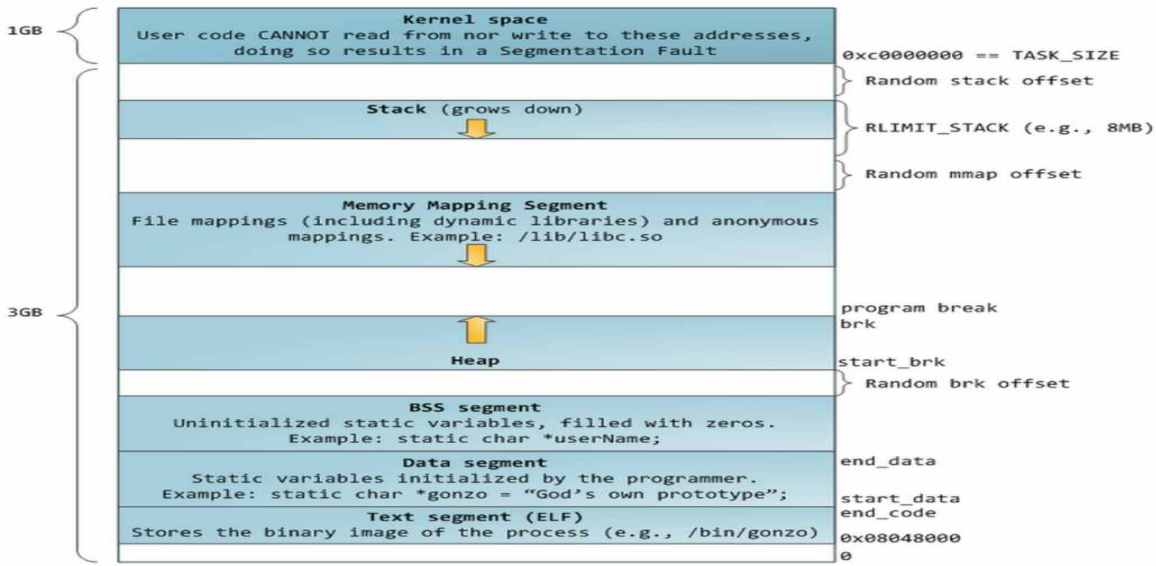
The genesis of C explains why it, and its descendant C++, have come to be used so widely as system programming languages today. Previous widely used languages were designed with other purposes in mind: FORTRAN for mathematical tasks performed by engineers and scientists; COBOL for commercial systems 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)

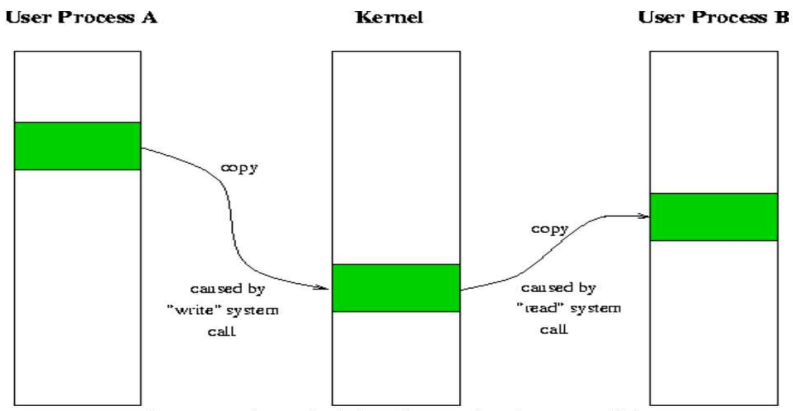
- 2.6 Programs
  - ✓ A set of instructions that describes how to perform a specific task
  - ✓ Two forms: source code, binary (machine language)
- 2.7 Processes
  - ✓ An instance of an executing program ➔ LN4, 5
  - ✓ Has its own virtual memory (layout: text, data, heap, stack, map)
- 2.8 Memory Mappings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

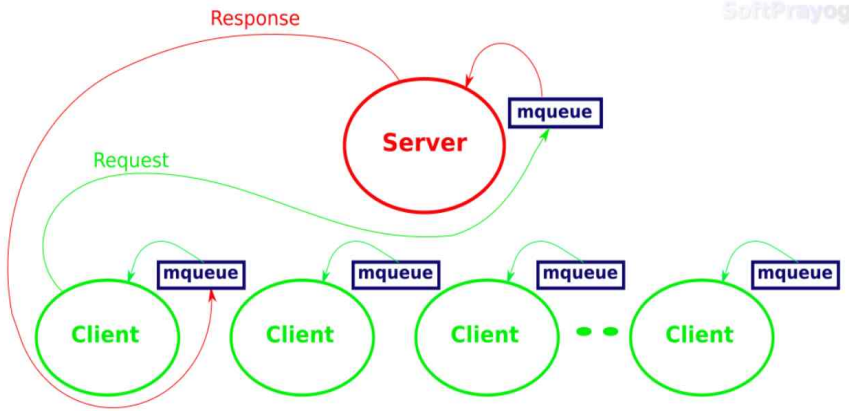


(Source: brunch.co.kr/@alden/13)

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, …





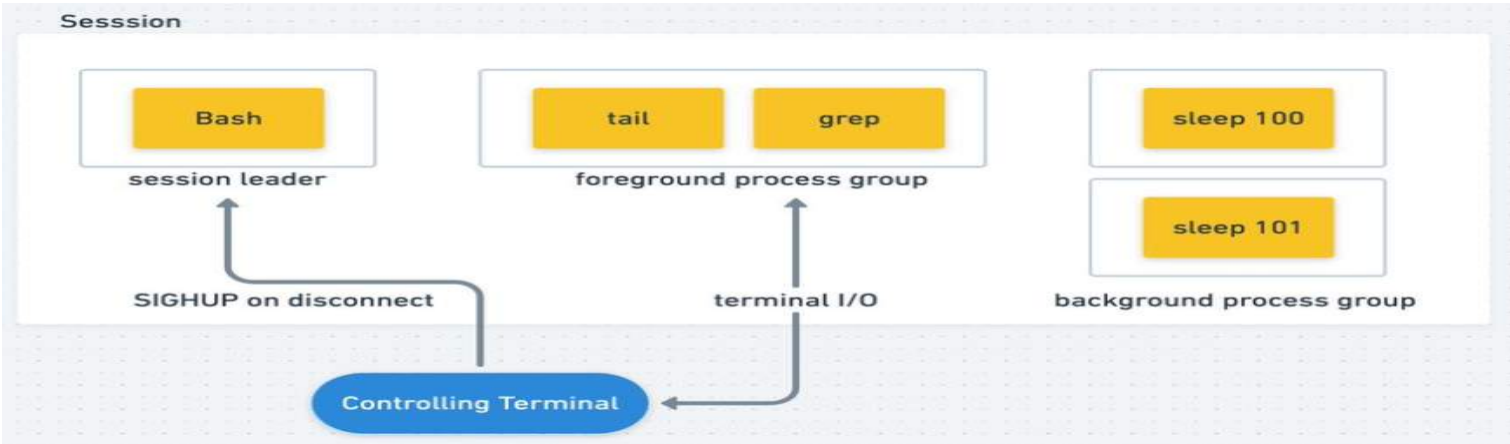
(Source: http://www.gerhardmueller.de/docs/UnixCommunicationFacilities/ip/node6.html, https://www.softprayog.in/programming/interprocess-communication-using-system-v-message-queues-in-linux)

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
```

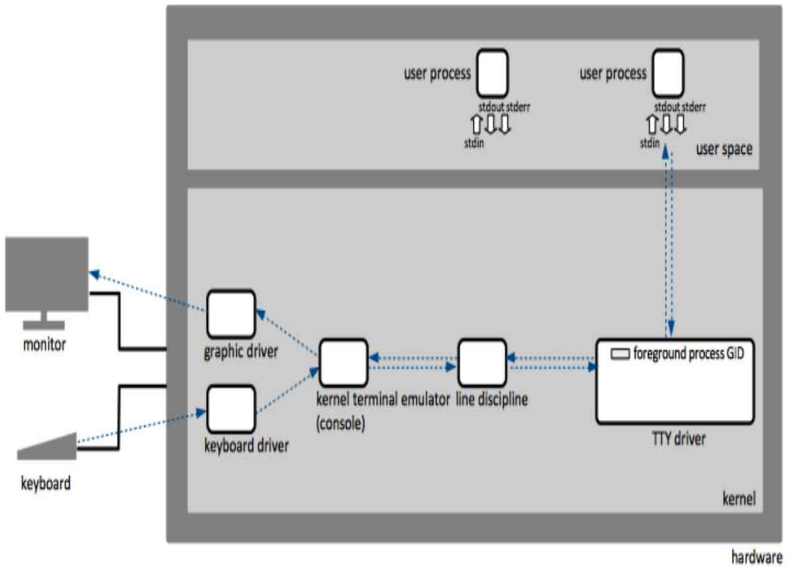
- 2.14 Session
  - ✓ A session is a collection of process groups (jobs).
  - ✓ Related with a terminal (controlling terminal, usually login terminal)
    - One foreground job and multiple background jobs

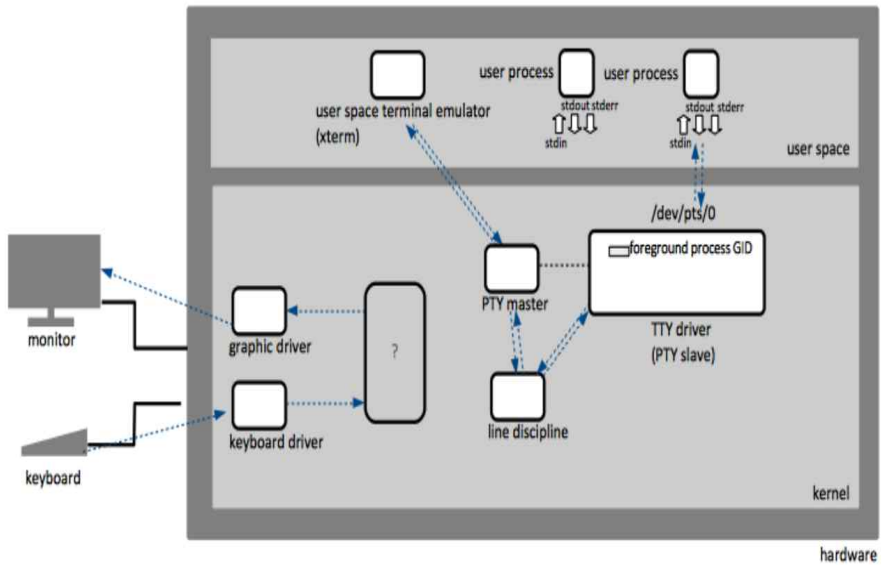


(Source: https://twitter.com/igor_sarcevic/status/1157349076809191425)

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

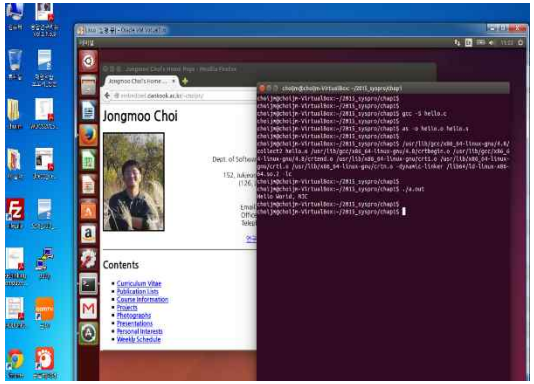


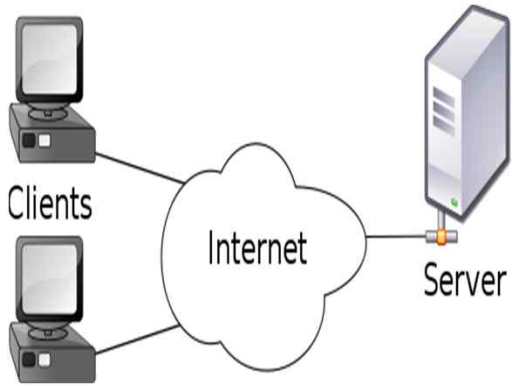


(Source: https://kb.novaordis.com/index.php/Linux_TTY)

How to access Linux (1/4)

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

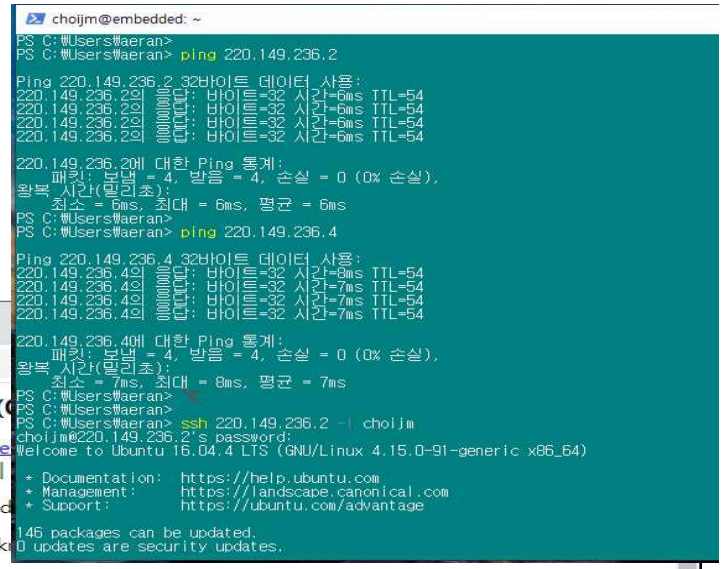


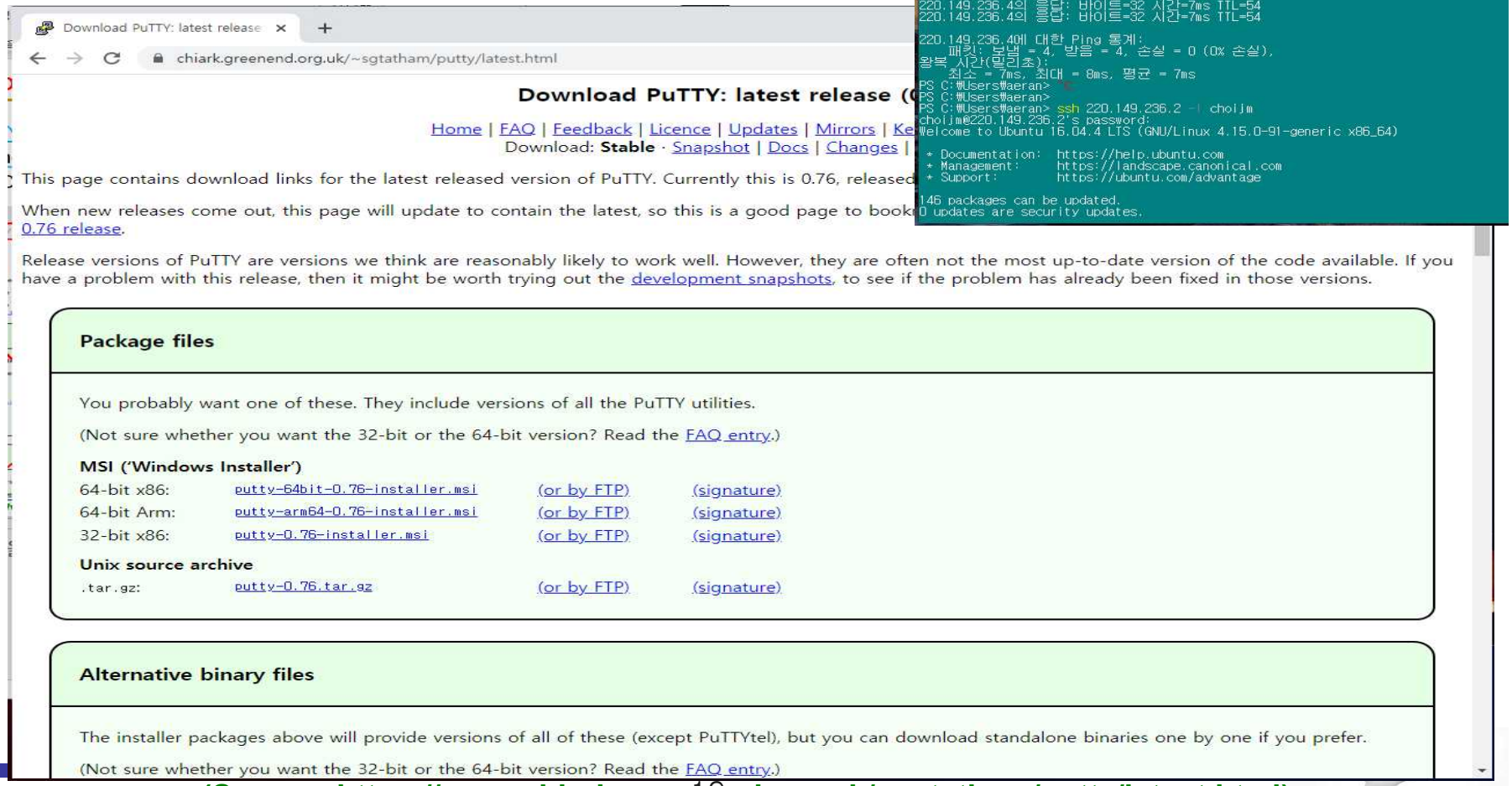


✓ In our course
- Client: terminal emulator (telnet/ssh client, putty, …)
- Server: Linux system (PC)
  - IP: 220.149.236.2 (primary), 220.149.236.4 (secondary)
- Alternative: Amazon EC2, Google Cloud, MS Azure or ToastCloud

How to access Linux (2/4)

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

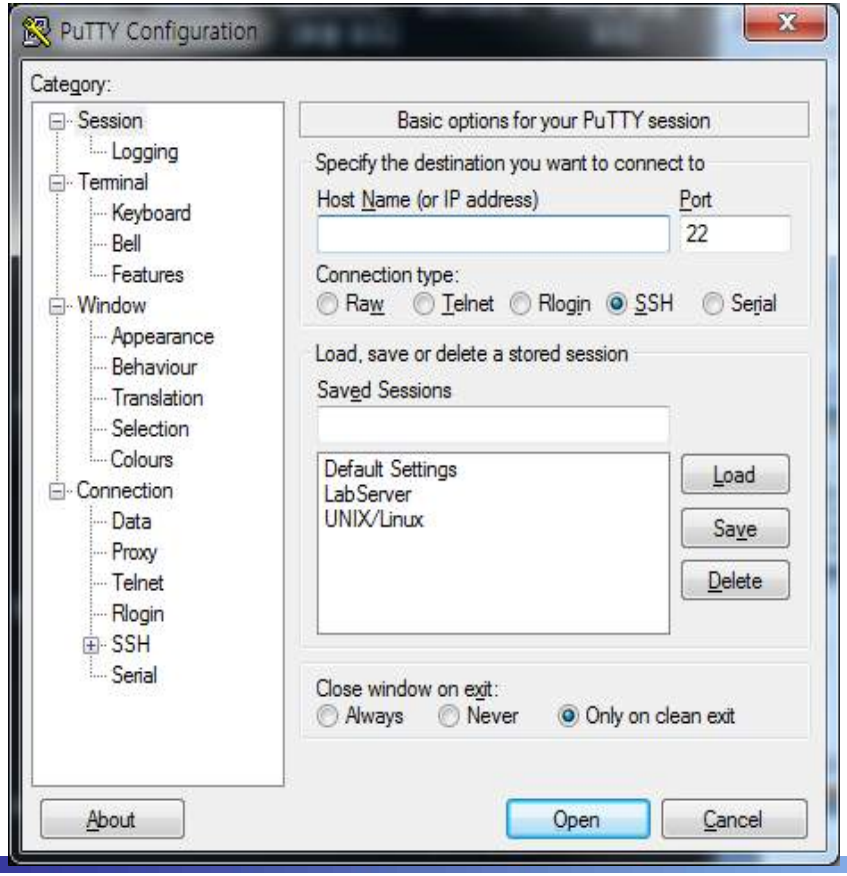


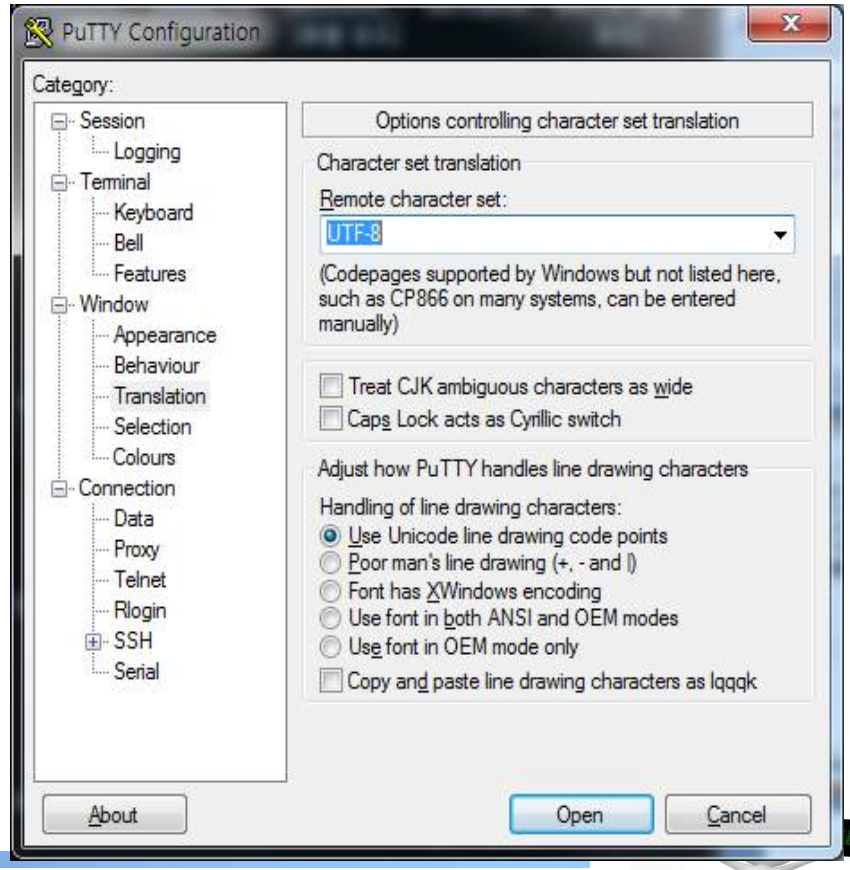


(Source: https://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html)

How to access Linux (3/4)

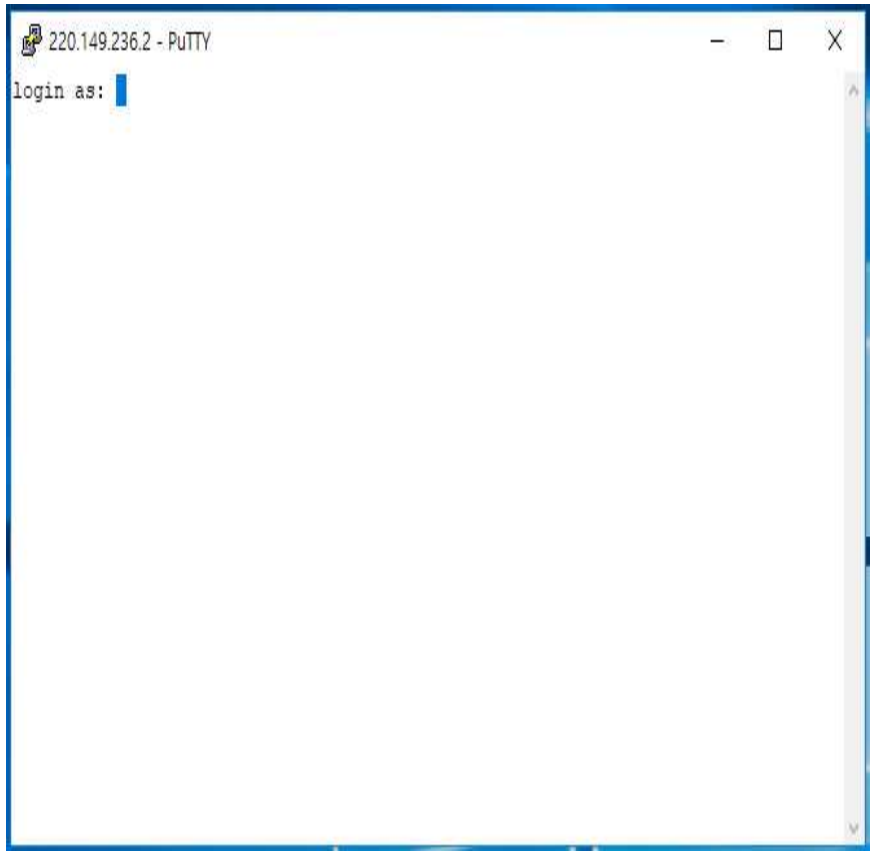
- Putty with ssh
  - ✓ IP: 220.149.236.2 (check that “type is ssh” and “port is 22”)
  - ✓ Colours: click “Use system colours
  - ✓ Translation: choose “UTF-8”

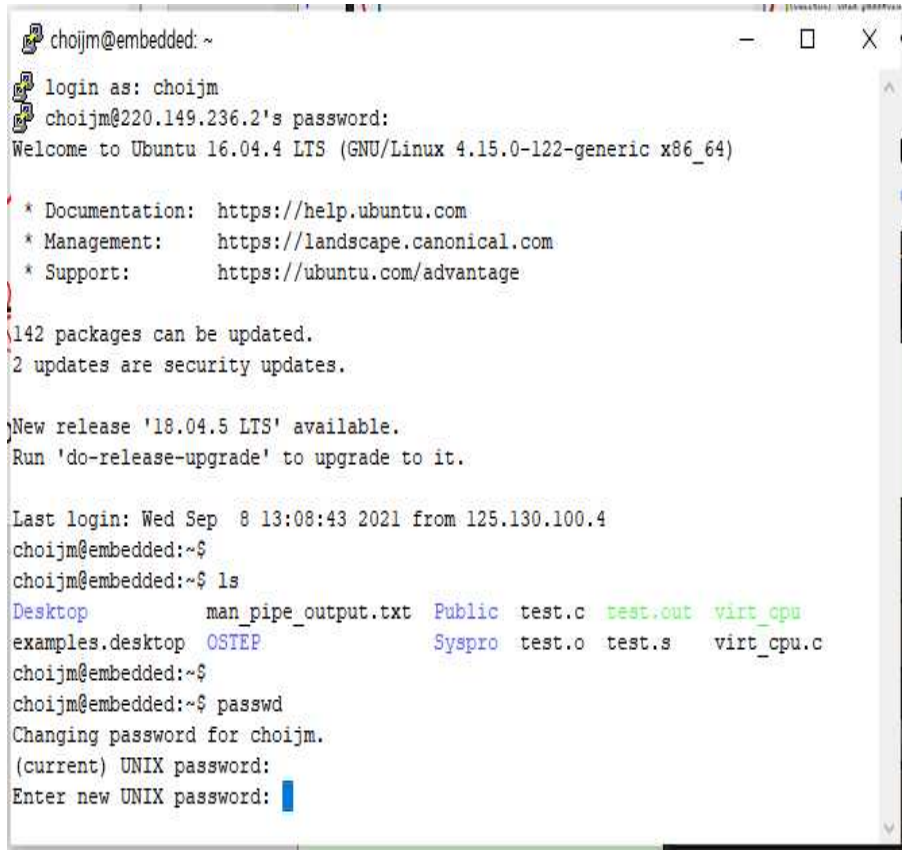




How to access Linux (4/4)

- 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, hexdun
  - ✓ file attributes manipulation
    - chmod, chown, chgrp, touch
  - ✓ redirection
    - >

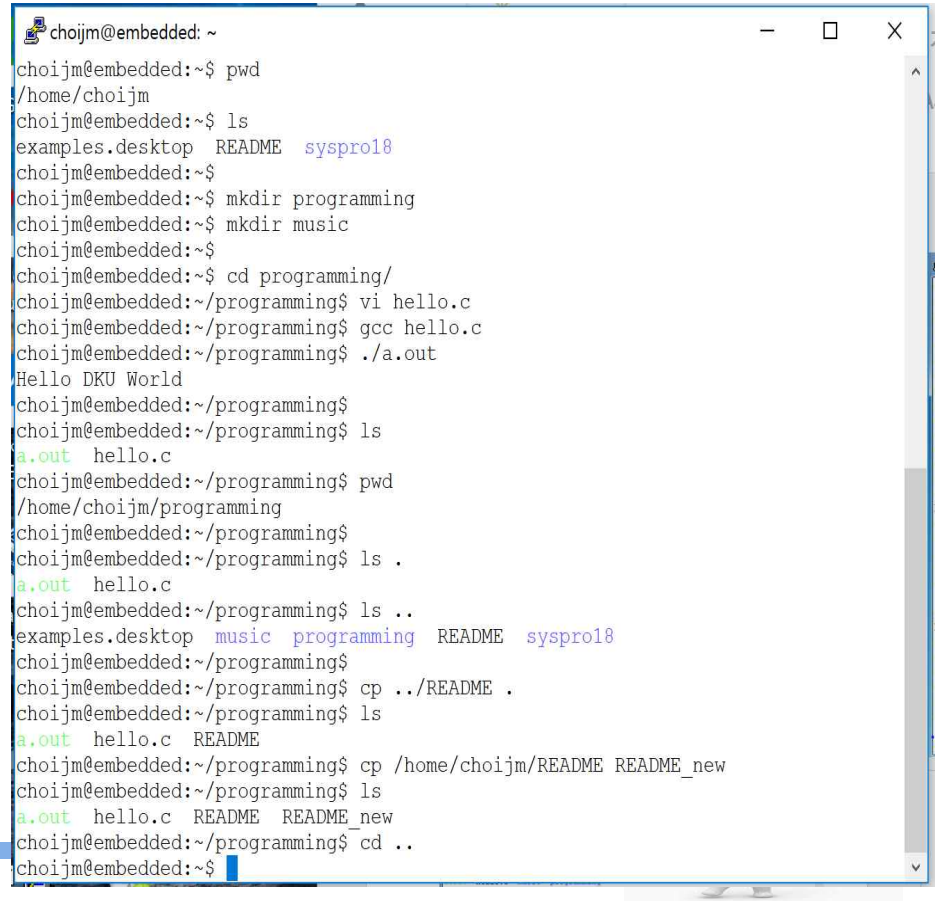
```
choijm@embedded-desktop: ~
choijm@embedded-desktop:~$
choijm@embedded-desktop:~$ ls
choijm@embedded-desktop:~$
choijm@embedded-desktop:~$ vi hello.c
choijm@embedded-desktop:~$
choijm@embedded-desktop:~$ gcc hello.c
choijm@embedded-desktop:~$
choijm@embedded-desktop:~$ ls
a.out  hello.c
choijm@embedded-desktop:~$
choijm@embedded-desktop:~$ ./a.out
Hello System Programming
choijm@embedded-desktop:~$
choijm@embedded-desktop:~$ more hello.c
#include <stdio.h>

int main()
{
        printf("Hello System Programming\n");
}
choijm@embedded-desktop:~$
choijm@embedded-desktop:~$ cp hello.c hello_new.c
choijm@embedded-desktop:~$
choijm@embedded-desktop:~$ ls
a.out  hello.c  hello_new.c
choijm@embedded-desktop:~$
choijm@embedded-desktop:~$ rm hello_new.c
choijm@embedded-desktop:~$
choijm@embedded-desktop:~$ ls
a.out  hello.c
choijm@embedded-desktop:~$ man ls
```

```
choijm@embedded-desktop: ~
LS(1)                    User Commands                    LS(1)

NAME
       ls - list directory contents

SYNOPSIS
       ls [OPTION]... [FILE]...

DESCRIPTION
       List  information  about  the  FILEs  (the  current
       directory by default).  Sort entries alphabetically
       if none of -cftuvSUX nor --sort is specified.

       Mandatory  arguments  to long options are mandatory
       for short options too.

       -a, --all
              do not ignore entries starting with .

       -A, --almost-all
              do not list implied . and ..

       --author
              with -l, print the author of each file

       -b, --escape
              print C-style escapes for nongraphic charac
              ters

       --block-size=SIZE
 Manual page ls(1) line 1 (press h for help or q to quit)
```

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



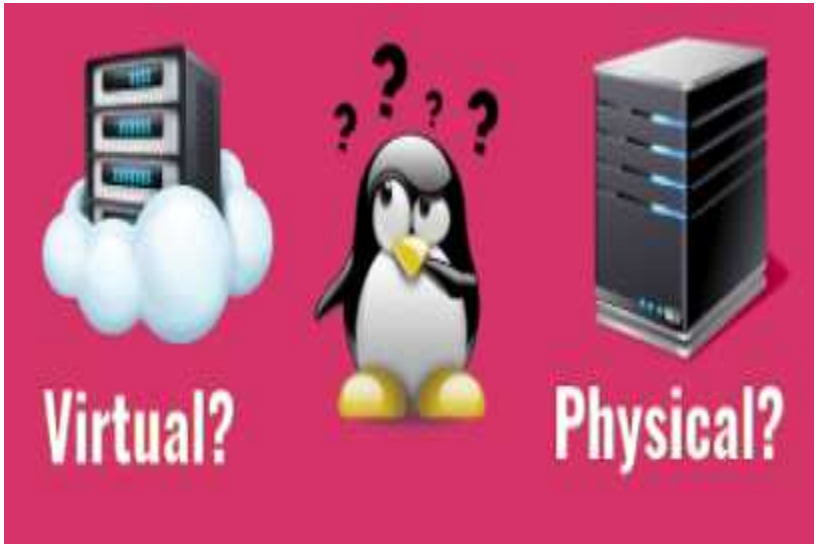
How to use commands in Linux (4/12)

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

```
choijm@embedded-desktop: ~
choijm@embedded-desktop:~$
choijm@embedded-desktop:~$ ls
a.out  hello.c  music  programming
choijm@embedded-desktop:~$
choijm@embedded-desktop:~$ ls -l
합계 20
-rwxrwxr-x 1 choijm choijm 4696  9월 10 16:11 a.out
-rw-rw-r-- 1 choijm choijm   74  9월 10 16:20 hello.c
drwxrwxr-x 2 choijm choijm 4096  9월 10 16:15 music
drwxrwxr-x 2 choijm choijm 4096  9월 10 16:17 programming
choijm@embedded-desktop:~$
choijm@embedded-desktop:~$ chmod o+w hello.c
choijm@embedded-desktop:~$
choijm@embedded-desktop:~$ ls -l
합계 20
-rwxrwxr-x 1 choijm choijm 4696  9월 10 16:11 a.out
-rw-rw-rw- 1 choijm choijm   74  9월 10 16:20 hello.c
drwxrwxr-x 2 choijm choijm 4096  9월 10 16:15 music
drwxrwxr-x 2 choijm choijm 4096  9월 10 16:17 programming
choijm@embedded-desktop:~$
choijm@embedded-desktop:~$ chmod g-x a.out
choijm@embedded-desktop:~$
choijm@embedded-desktop:~$ ls -l
합계 20
-rwxrw-r-x 1 choijm choijm 4696  9월 10 16:11 a.out
-rw-rw-rw- 1 choijm choijm   74  9월 10 16:20 hello.c
drwxrwxr-x 2 choijm choijm 4096  9월 10 16:15 music
drwxrwxr-x 2 choijm choijm 4096  9월 10 16:17 programming
choijm@embedded-desktop:~$
```

Quiz for 3rd-Week 2nd-Lesson

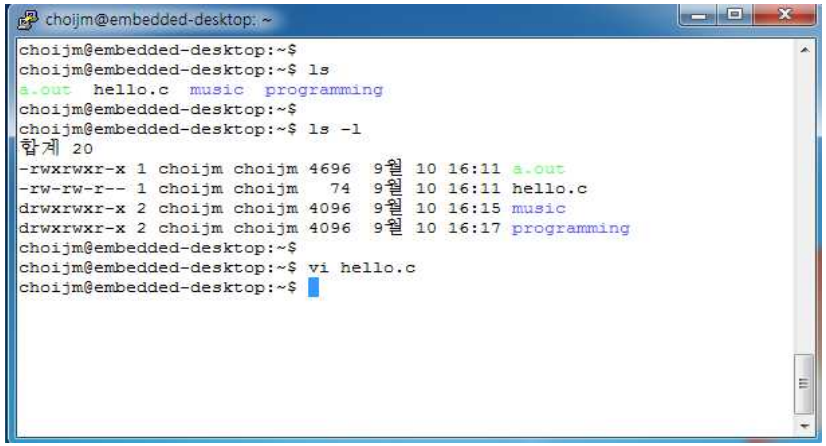
- Quiz
  - ✓ 1) Discuss three ways about how to access Linux.
  - ✓ 2) Explain differences between “$ls .” and “$ls ..”. Also, explain differences between “ls” and “ls –l”.
  - ✓ Due: until 6 PM Friday of this week (17th, September)

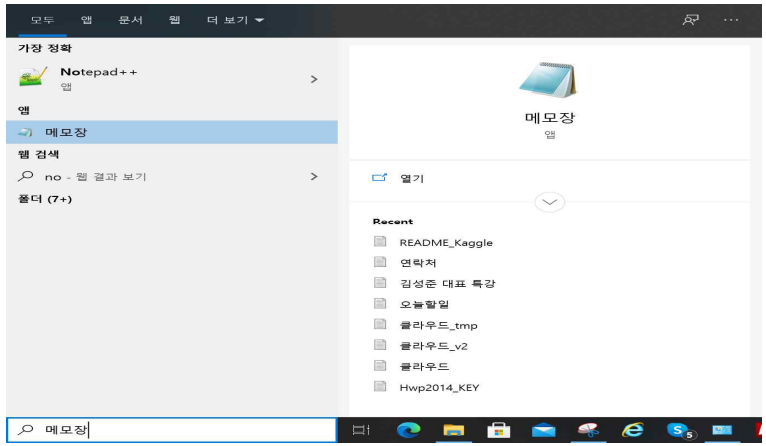


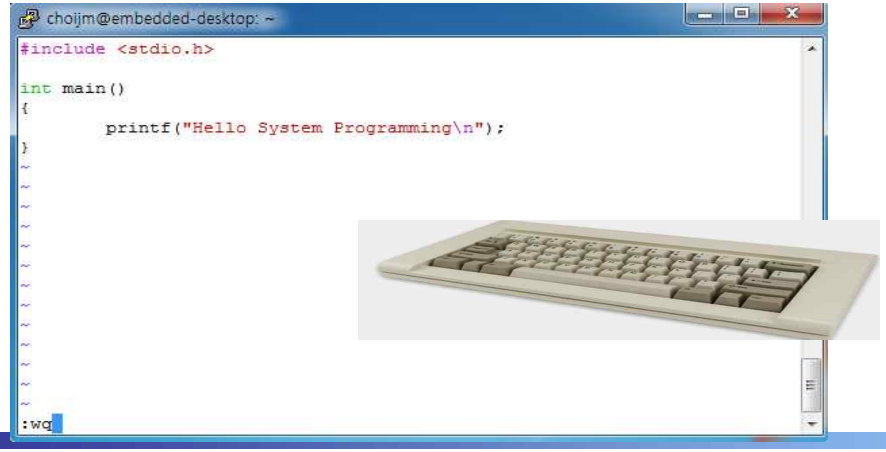
(Source: https://ostechnix.com/check-linux-system-physical-virtual-machine/ and https://www.tecmint.com/15-basic-ls-command-examples-in-linux/)

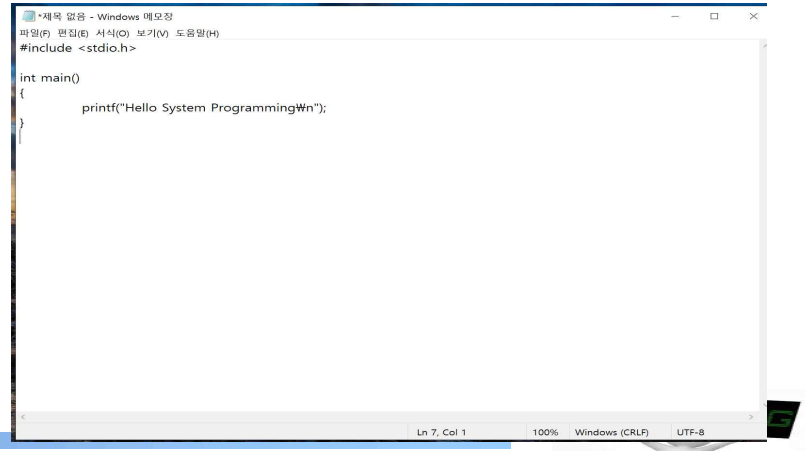
How to use commands in Linux (5/12)

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



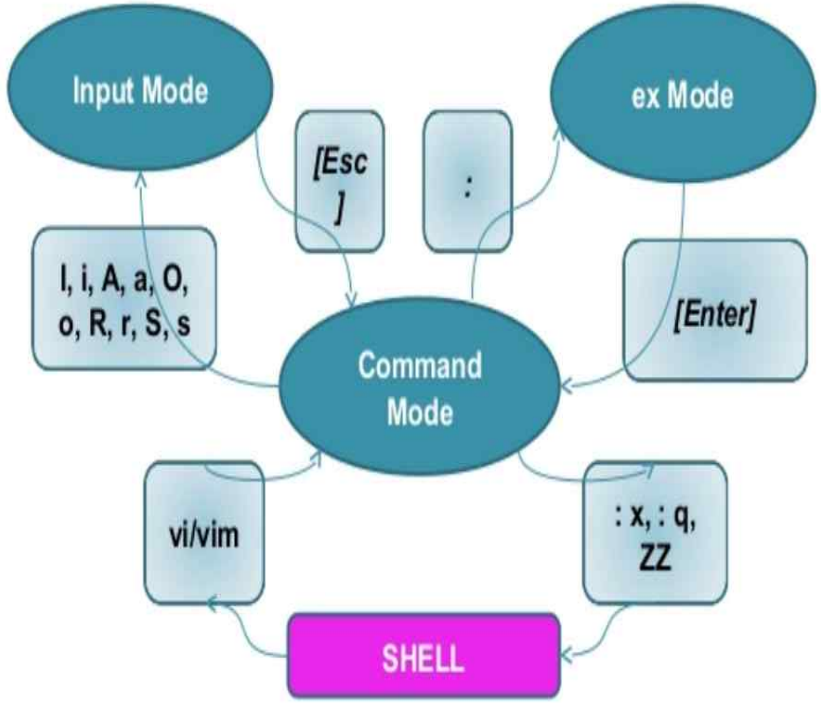






How to use commands in Linux (6/12)

- 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

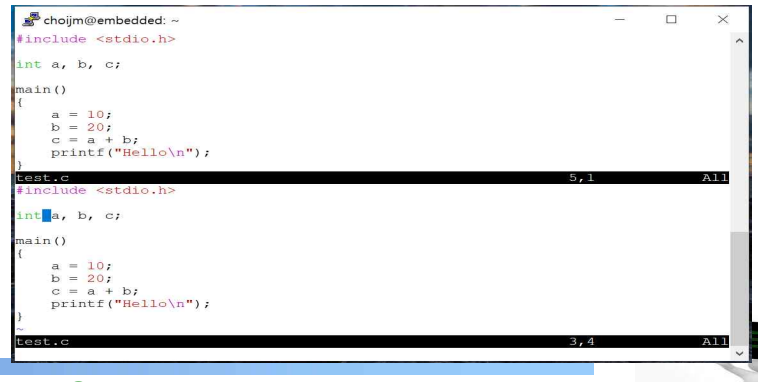
Command	Function
:sp	Split current window horizontally in two
:vsp	Split current window vertically into two
vim –O [n | files…]	Opens n windows, files split vertically
:new	Open a new blank window
:on	Make current window the only window
:q	Quit current window
:qa	Quit all windows
:xa	Save and quit all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

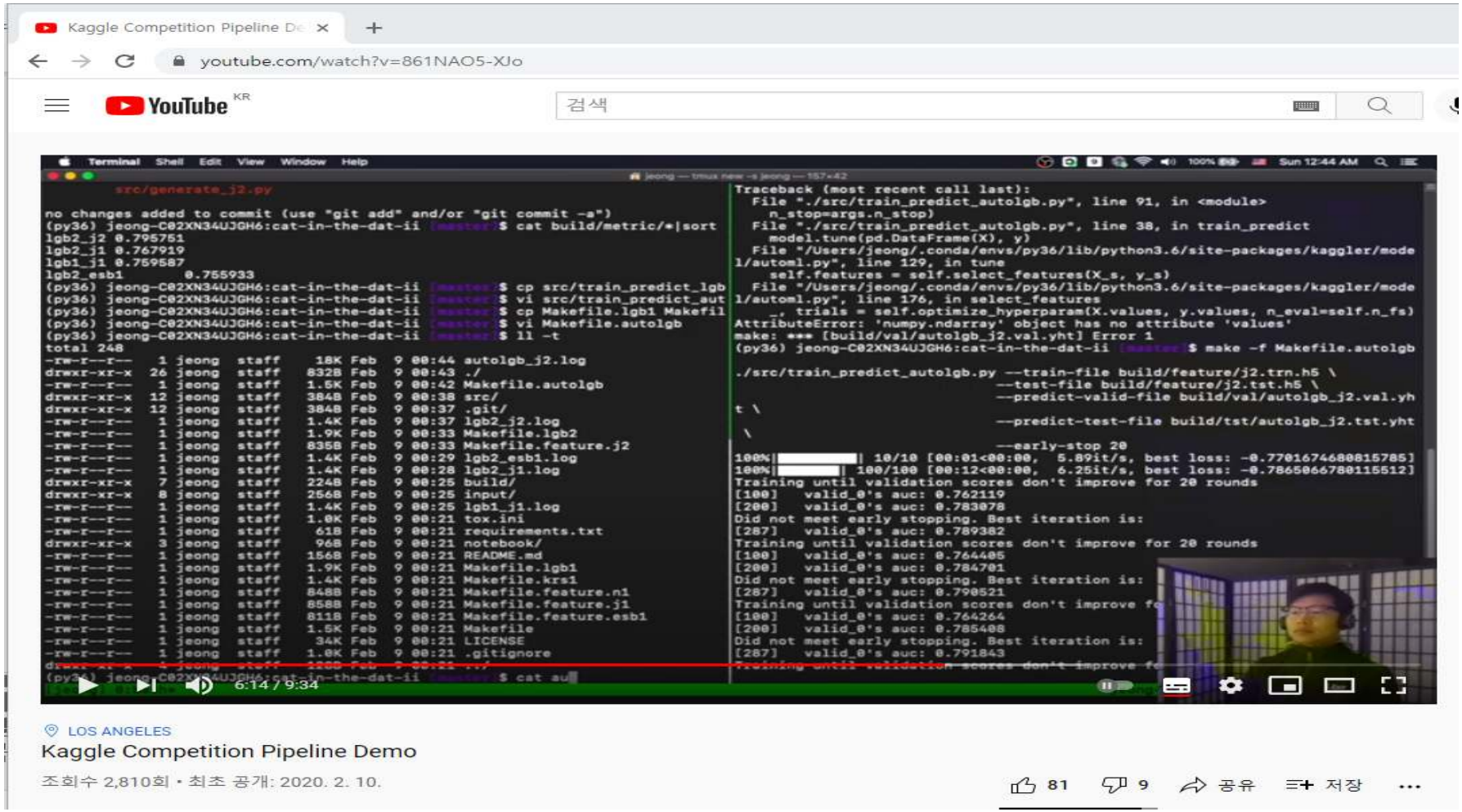
Command	Function
:1,$s/#//g	Globally remove #
:3,10s/^/#/	Insert # at the beginning of line 3 to 10
:$s/$/;/	Insert a ; at the end of last line
:%s/abc/xyz/gc	Globally replace abc by xyz interactively
:1,$s/include/<&>/g	Globally replace include by <include>



(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

```
choijm@embedded-desktop: ~
choijm@embedded-desktop:~$ ls
a.out  hello.c  music  programming
choijm@embedded-desktop:~$
choijm@embedded-desktop:~$ ps
  PID TTY          TIME CMD
 9111 pts/3    00:00:00 bash
 9249 pts/3    00:00:00 ps
choijm@embedded-desktop:~$
choijm@embedded-desktop:~$ vi hello.c
choijm@embedded-desktop:~$
choijm@embedded-desktop:~$ more hello.c
#include <stdio.h>

int main()
{
        printf("Hello System Programming\n");
        while (1);
}
choijm@embedded-desktop:~$
choijm@embedded-desktop:~$ gcc hello.c
choijm@embedded-desktop:~$
choijm@embedded-desktop:~$ ./a.out
Hello System Programming

^C
choijm@embedded-desktop:~$
choijm@embedded-desktop:~$ ./a.out &
[1] 9271
choijm@embedded-desktop:~$ Hello System Programming

choijm@embedded-desktop:~$
choijm@embedded-desktop:~$ ps
  PID TTY          TIME CMD
 9111 pts/3    00:00:00 bash
 9271 pts/3    00:00:07 a.out
 9272 pts/3    00:00:00 ps
choijm@embedded-desktop:~$
choijm@embedded-desktop:~$ kill -9 9271
choijm@embedded-desktop:~$
[1]+  죽었음                  ./a.out
choijm@embedded-desktop:~$ ps
```

How to use commands in Linux (10/12)

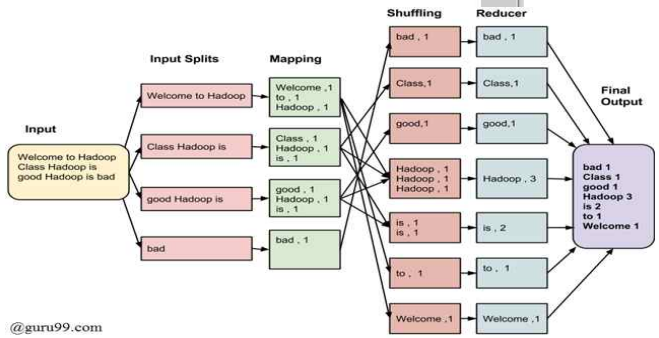
- Advanced commands: pipe

```
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 examples.desktop
-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

choijm@embedded: ~

```
choijm@embedded:~$
choijm@embedded:~$ ps
  PID TTY          TIME CMD
22492 pts/9    00:00:00 bash
22532 pts/9    00:00:00 ps
choijm@embedded:~$
choijm@embedded:~$ #include <stdio.h>

main()
{
        printf("Hello DKU World\n");
}
```

choijm@embedded: ~/programming

```
choijm@embedded:~/programming$ ps
  PID TTY          TIME CMD
22561 pts/8    00:00:00 bash
22610 pts/8    00:00:00 ps
choijm@embedded:~/programming$ ls
a.out  hello.c  README  README_new
choijm@embedded:~/programming$
choijm@embedded:~/programming$ cat hello.c
#include <stdio.h>

main()
{
        printf("Hello DKU World\n");
}
choijm@embedded:~/programming$
choijm@embedded:~/programming$ cat hello.c > hello_backup.c
choijm@embedded:~/programming$
choijm@embedded:~/programming$ more hello_backup.c
#include <stdio.h>

main()
{
        printf("Hello DKU World\n");
}
choijm@embedded:~/programming$ cat hello.c > /dev/pts/9
choijm@embedded:~/programming$
```

CloudStation - 바로 가기 | putty | Chrome | python

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

- Overall

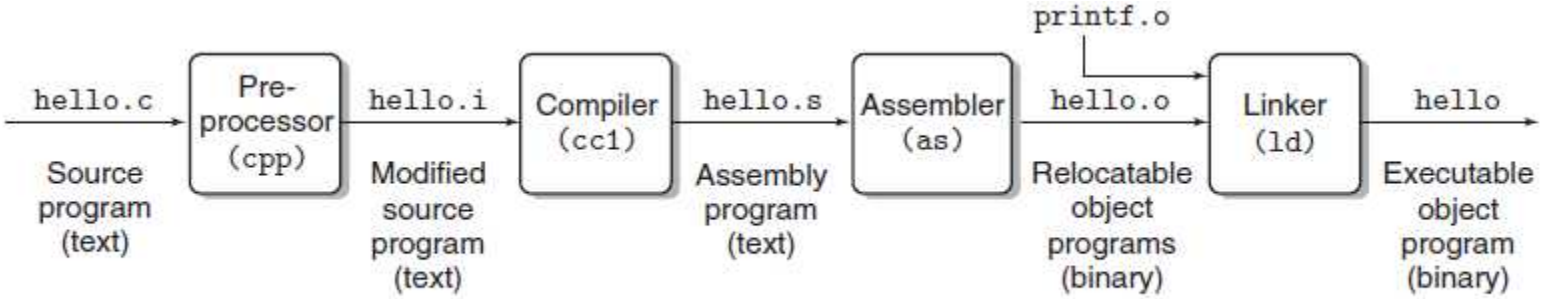


Figure 1.3 The compilation system.

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

choijm@embedded-desktop: ~/syspro/chap2

```
choijm@embedded-desktop:~/syspro/chap2$
choijm@embedded-desktop:~/syspro/chap2$ vi test.c
choijm@embedded-desktop:~/syspro/chap2$
choijm@embedded-desktop:~/syspro/chap2$ ls
test.c
choijm@embedded-desktop:~/syspro/chap2$
choijm@embedded-desktop:~/syspro/chap2$ more test.c
#include <stdio.h>

int a, b, c;

int main()
{
        a = 10;
        b = 20;
        c = a + b;
        printf("C = %d\n", c);
}
choijm@embedded-desktop:~/syspro/chap2$
choijm@embedded-desktop:~/syspro/chap2$ gcc test.c
choijm@embedded-desktop:~/syspro/chap2$ ./a.out
C = 30
choijm@embedded-desktop:~/syspro/chap2$
choijm@embedded-desktop:~/syspro/chap2$ gcc -o test.out test.c
choijm@embedded-desktop:~/syspro/chap2$ ./test.out
C = 30
choijm@embedded-desktop:~/syspro/chap2$
```

choijm@sungmin-Samsung-DeskTop-System: ~/syspro/chap2

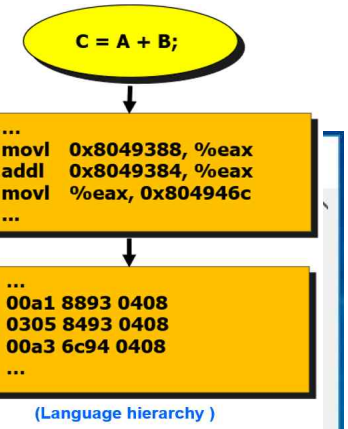
```
choijm@sungmin-Samsung-DeskTop-System:~/syspro/chap2$
choijm@sungmin-Samsung-DeskTop-System:~/syspro/chap2$ ls
test.c
choijm@sungmin-Samsung-DeskTop-System:~/syspro/chap2$
choijm@sungmin-Samsung-DeskTop-System:~/syspro/chap2$ gcc -S test.c
choijm@sungmin-Samsung-DeskTop-System:~/syspro/chap2$
choijm@sungmin-Samsung-DeskTop-System:~/syspro/chap2$ ls
test.c  test.s
choijm@sungmin-Samsung-DeskTop-System:~/syspro/chap2$
choijm@sungmin-Samsung-DeskTop-System:~/syspro/chap2$ as -o test.o test.s
choijm@sungmin-Samsung-DeskTop-System:~/syspro/chap2$
choijm@sungmin-Samsung-DeskTop-System:~/syspro/chap2$ ls
test.c  test.o  test.s
choijm@sungmin-Samsung-DeskTop-System:~/syspro/chap2$
choijm@sungmin-Samsung-DeskTop-System:~/syspro/chap2$ /usr/lib/gcc/i486-linux-gn
u/3.4.6/collect2 /usr/lib/i386-linux-gnu/crt1.o /usr/lib/i386-linux-gnu/crti.o /
usr/lib/i386-linux-gnu/crtn.o /usr/lib/gcc/i486-linux-gnu/3.4.6/crtbegin.o /usr/
lib/gcc//i486-linux-gnu/3.4.6/crtend.o test.o -lc -dynamic-linker /lib/ld-linux.
so.2
choijm@sungmin-Samsung-DeskTop-System:~/syspro/chap2$
choijm@sungmin-Samsung-DeskTop-System:~/syspro/chap2$ ls
a.out  test.c  test.o  test.s
choijm@sungmin-Samsung-DeskTop-System:~/syspro/chap2$
choijm@sungmin-Samsung-DeskTop-System:~/syspro/chap2$ ./a.out
C = 30
choijm@sungmin-Samsung-DeskTop-System:~/syspro/chap2$
choijm@sungmin-Samsung-DeskTop-System:~/syspro/chap2$
```

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

```
choijm@embedded-desktop: ~/syspro/chap2
choijm@embedded-desktop:~/syspro/chap2$ ls
a.out  test.c  test.o  test.s
choijm@embedded-desktop:~/syspro/chap2$
choijm@embedded-desktop:~/syspro/chap2$ more test.o

******** test.o: Not a text file ********

choijm@embedded-desktop:~/syspro/chap2$
choijm@embedded-desktop:~/syspro/chap2$ hexdump test.o
0000000 457f 464c 0101 0001 0000 0000 0000 0000
0000010 0001 0003 0001 0000 0000 0000 0000 0000
0000020 0110 0000 0000 0000 0034 0000 0000 0028
0000030 000b 0008 8955 83e5 08ec e483 b8f0 0000
0000040 0000 c083 830f 0fc0 e8c1 c104 04e0 c429
0000050 05c7 0000 0000 000a 0000 05c7 0000 0000
0000060 0014 0000 00a1 0000 0300 0005 0000 a300
0000070 0000 0000 00a1 0000 8900 2444 c704 2404
0000080 0000 0000 fce8 ffff c9ff 00c3 2043 203d
0000090 6425 000a 4700 4343 203a 4728 554e 2029
00000a0 2e33 2e34 2036 5528 7562 746e 2075 2e33
00000b0 2e34 2d36 7536 7562 746e 3575 0029 2e00
00000c0 7973 746d 6261 2e00 7473 7472 6261 2e00
00000d0 6873 7473 7472 6261 2e00 6572 2e6c 6574
00000e0 7478 2e00 6164 6174 2e00 7362 0073 722e
00000f0 646f 7461 0061 6e2e 746f 2e65 4e47 2d55
0000100 7473 6361 006b 632e 6d6f 656d 746e 0000
0000110 0000 0000 0000 0000 0000 0000 0000 0000
*
0000130 0000 0000 0000 0000 001f 0000 0001 0000
0000140 0006 0000 0000 0000 0034 0000 0057 0000
0000150 0000 0000 0000 0000 0004 0000 0000 0000
0000160 001b 0000 0009 0000 0000 0000 0000 0000
0000170 03b4 0000 0040 0000 0009 0000 0001 0000
0000180 0004 0000 0008 0000 0025 0000 0001 0000
```

```
choijm@embedded-desktop: ~/syspro/chap2
a.out  test.c  test.o  test.s
choijm@embedded-desktop:~/syspro/chap2$ objdump -f test.o

test.o:     file format elf32-i386
architecture: i386, flags 0x00000011:
HAS_RELOC, HAS_SYMS
start address 0x00000000

choijm@embedded-desktop:~/syspro/chap2$ objdump -d test.o

test.o:     file format elf32-i386


Disassembly of section .text:

00000000 <main>:
   0:   55                      push   %ebp
   1:   89 e5                   mov    %esp,%ebp
   3:   83 ec 08                sub    $0x8,%esp
   6:   83 e4 f0                and    $0xfffffff0,%esp
   9:   b8 00 00 00 00          mov    $0x0,%eax
   e:   83 c0 0f                add    $0xf,%eax
  11:   83 c0 0f                add    $0xf,%eax
  14:   c1 e8 04                shr    $0x4,%eax
  17:   c1 e0 04                shl    $0x4,%eax
  1a:   29 c4                   sub    %eax,%esp
  1c:   c7 05 00 00 00 00 0a    movl   $0xa,0x0
  23:   00 00 00
  26:   c7 05 00 00 00 00 14    movl   $0x14,0x0
  2d:   00 00 00
  30:   a1 00 00 00 00          mov    0x0,%eax
  35:   03 05 00 00 00 00       add    0x0,%eax
  3b:   a3 00 00 00 00          mov    %eax,0x0
  40:   a1 00 00 00 00          mov    0x0,%eax
  45:   89 44 24 04             mov    %eax,0x4(%esp)
  49:   c7 04 24 00 00 00 00    movl   $0x0,(%esp)
  50:   e8 fc ff ff ff          call   51 <main+0x51>
  55:   c9                      leave
  56:   c3                      ret
choijm@embedded-desktop:~/syspro/chap2$
```

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

- Executable code

```
choijm@embedded-desktop: ~/syspro/chap2
choijm@embedded-desktop:~/syspro/chap2$ ls
a.out  test.c  test.o  test.s
choijm@embedded-desktop:~/syspro/chap2$ hexdump a.out
0000000 457f 464c 0101 0001 0000 0000 0000 0000
0000010 0002 0003 0001 0000 8318 0804 0034 0000
0000020 0680 0000 0000 0000 0034 0020 0007 0028
0000030 0019 0016 0006 0000 0034 0000 8034 0804
0000040 8034 0804 00e0 0000 00e0 0000 0005 0000
0000050 0004 0000 0003 0000 0114 0000 8114 0804
0000060 8114 0804 0013 0000 0013 0000 0004 0000
0000070 0001 0000 0001 0000 0000 0000 8000 0804
0000080 8000 0804 0480 0000 0480 0000 0005 0000
0000090 1000 0000 0001 0000 0480 0000 9480 0804
00000a0 9480 0804 00e8 0000 00f4 0000 0006 0000
00000b0 1000 0000 0002 0000 0480 0000 9480 0804
00000c0 9480 0804 00c8 0000 00c8 0000 0006 0000
00000d0 0004 0000 0004 0000 0128 0000 8128 0804
00000e0 8128 0804 0020 0000 0020 0000 0004 0000
00000f0 0004 0000 e551 6474 0000 0000 0000 0000
                  0000 0000 0000 0006 0000
                  6269 6c2f 2d64 696c 756e
                  0000 0004 0000 0010 0000
                  0055 0000 0000 0002 0000
                  0000 0003 0000 0005 0000
                  0000 0003 0000 0000 0000
                  0000 0001 0000 0000 0000
                  0000 0000 0000 0000 0000
                  0000 0000 0000 0012 0000
                  0000 0000 0000 0020 0000
                  0804 0004 0000 0011 000e
                  0000 0000 0000 0012 0000
                  6f73 362e 5f00 4f49 735f
                  6573 0064 7270 6e69 6674
                  5f63 7473 7261 5f74 616d
                  6f6d 5f6e 7473 7261 5f74
0000200 005f 4c47 4249 5f43 2e32 0030 0000 0002
0000210 0000 0001 0002 0000 0001 0001 0001 0000
0000220 0010 0000 0000 0000 6910 0d69 0000 0002
0000230 0042 0000 0000 0000 9548 0804 0206 0000
0000240 9558 0804 0107 0000 955c 0804 0207 0000
```

C = A + B;

```
...
movl   0x8049388, %eax
addl   0x8049384, %eax
movl   %eax, 0x804946c
...
```

```
...
00a1 8893 0408
0305 8493 0408
00a3 6c94 0408
...
```

(Language hierarchy)

```
choijm@embedded-desktop: ~/syspro/chap2
choijm@embedded-desktop:~/syspro/chap2$ objdump -f a.out

a.out:     file format elf32-i386
architecture: i386, flags 0x00000112:
EXEC_P, HAS_SYMS, D_PAGED
start address 0x08048318

choijm@embedded-desktop:~/syspro/chap2$ objdump -d a.out > objdump_result.txt
choijm@embedded-desktop:~/syspro/chap2$
choijm@embedded-desktop:~/syspro/chap2$ vi objdump_result.txt
choijm@embedded-desktop:~/syspro/chap2$
choijm@embedded-desktop:~/syspro/chap2$ more objdump_result.txt
...
...

Disassembly of section .text:

080482c0 <main>:
 80482c0:       55                      push   %ebp
 80482c1:       89 e5                   mov    %esp,%ebp
 80482c3:       83 ec 08                sub    $0x8,%esp
 80482c6:       83 e4 f0                and    $0xfffffff0,%esp
 80482c9:       b8 00 00 00 00          mov    $0x0,%eax
 80482ce:       83 c0 0f                add    $0xf,%eax
 80482d1:       83 c0 0f                add    $0xf,%eax
 80482d4:       c1 e8 04                shr    $0x4,%eax
 80482d7:       c1 e0 04                shl    $0x4,%eax
 80482da:       29 c4                   sub    %eax,%esp
 80482dc:       c7 05 70 95 04 08 0a    movl   $0xa,0x8049570
 80482e3:       00 00 00
 80482e6:       c7 05 68 95 04 08 14    movl   $0x14,0x8049568
 80482ed:       00 00 00
 80482f0:       a1 68 95 04 08          mov    0x8049568,%eax
 80482f5:       03 05 70 95 04 08       add    0x8049570,%eax
 80482fb:       a3 6c 95 04 08          mov    %eax,0x804956c
 8048300:       a1 6c 95 04 08          mov    0x804956c,%eax
 8048305:       89 44 24 04             mov    %eax,0x4(%esp)
 8048309:       c7 04 24 d0 83 04 08    movl   $0x80483d0,(%esp)
 8048310:       e8 7b ff ff ff          call   8048290 <printf@plt>
 8048315:       c9                      leave
 8048316:       c3                      ret
 8048317:       90                      nop
```

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

- What are the execution results of this program?

```
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$
```

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 (e.g. ls –l, ps, pipe, redirection, …) including compilation practice (e.g. gcc, as, gdb, …)

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

- Server IP: 220.149.236.2 or 220.149.236.4

1.3 Deadline: 6 PM Friday of the next week (8th, October)

Appendix 1. Snapshot Example

- Example

choijm@localhost:~/syspro_20130902/reports

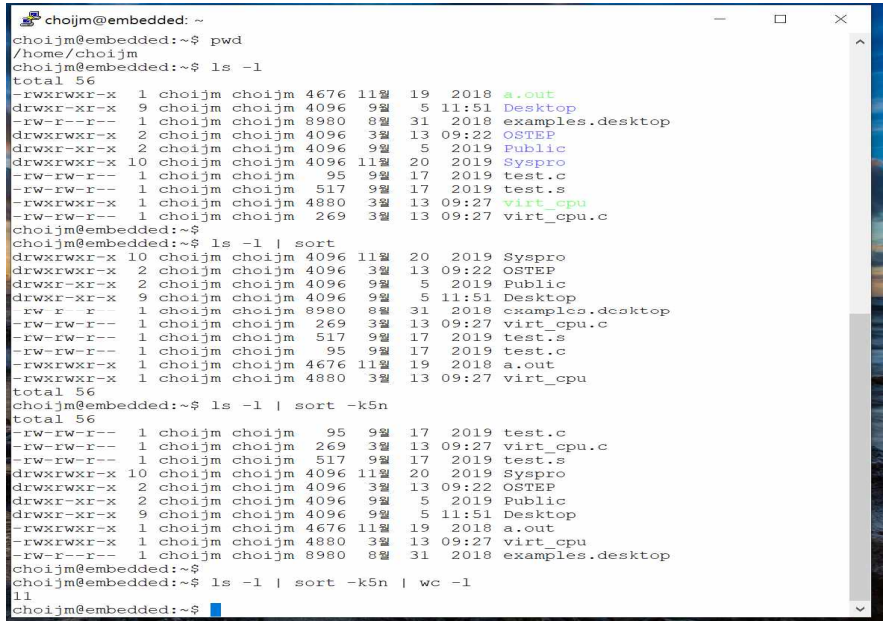
```
[choijm@localhost reports]$
[choijm@localhost reports]$ ls
a.out  hello.c  music  my_favorite_poem.txt  subdir  test.txt  test2.txt
[choijm@localhost reports]$
[choijm@localhost reports]$ more my_favorite_poem.txt
나 하늘로 돌아가리라.
새벽빛 와 닿으면 스러지는
이슬 더불어 손에 손을 잡고,

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

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

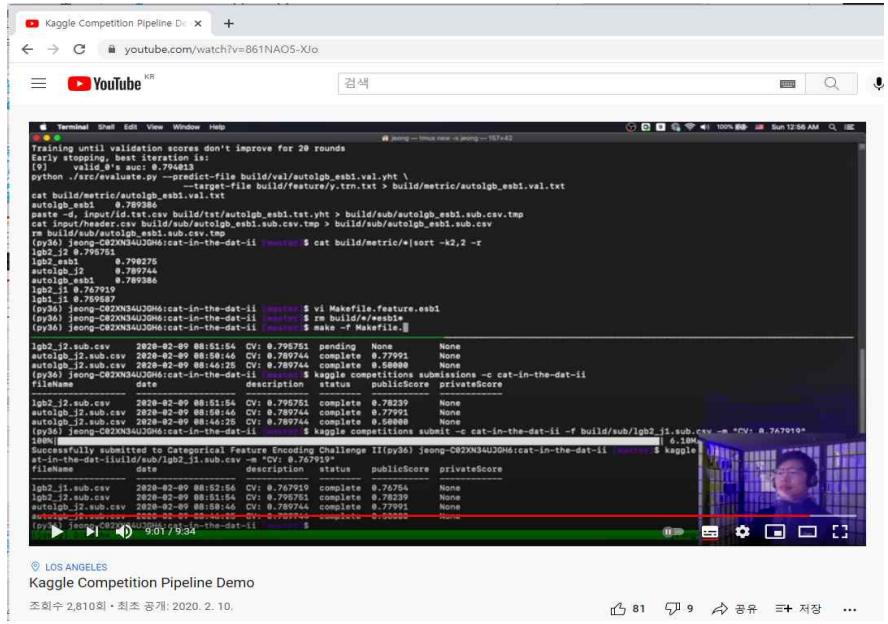
choijm@embedded: ~

```
choijm@embedded:~$ ps
  PID TTY          TIME CMD
15864 pts/8    00:00:00 bash
16309 pts/8    00:00:00 ps
choijm@embedded:~$ vi test.c
choijm@embedded:~$ ls
Desktop           man_pipe_output.txt  Public  test.c  test.out  virt_cpu
examples.desktop  OSTEP                Syspro  test.o  test.s    virt_cpu.c
choijm@embedded:~$ cat test.c
#include <stdio.h>

int a, b, c;

main()
{
        a = 10;
        b = 20;
        c = a + b;
        printf("Hello: C = %d\n", c);
}
choijm@embedded:~$ cat test.c > test1.c
choijm@embedded:~$
choijm@embedded:~$ gcc -o test.out test.c
choijm@embedded:~$ ./test.out
Hello: C = 30
choijm@embedded:~$ gcc -S test.c
choijm@embedded:~$ gdb test.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 test.out...(no debugging symbols found)...done.
(gdb) run
Starting program: /home/choijm/test.out
Hello: C = 30
[Inferior 1 (process 16388) exited with code 016]
(gdb) quit
choijm@embedded:~$
choijm@embedded:~$ whoami
choijm
choijm@embedded:~$ date
2021. 09. 09. (목 ) 17:17:06 KST
choijm@embedded:~$
```

Quiz for 4th-Week 1st-Lesson

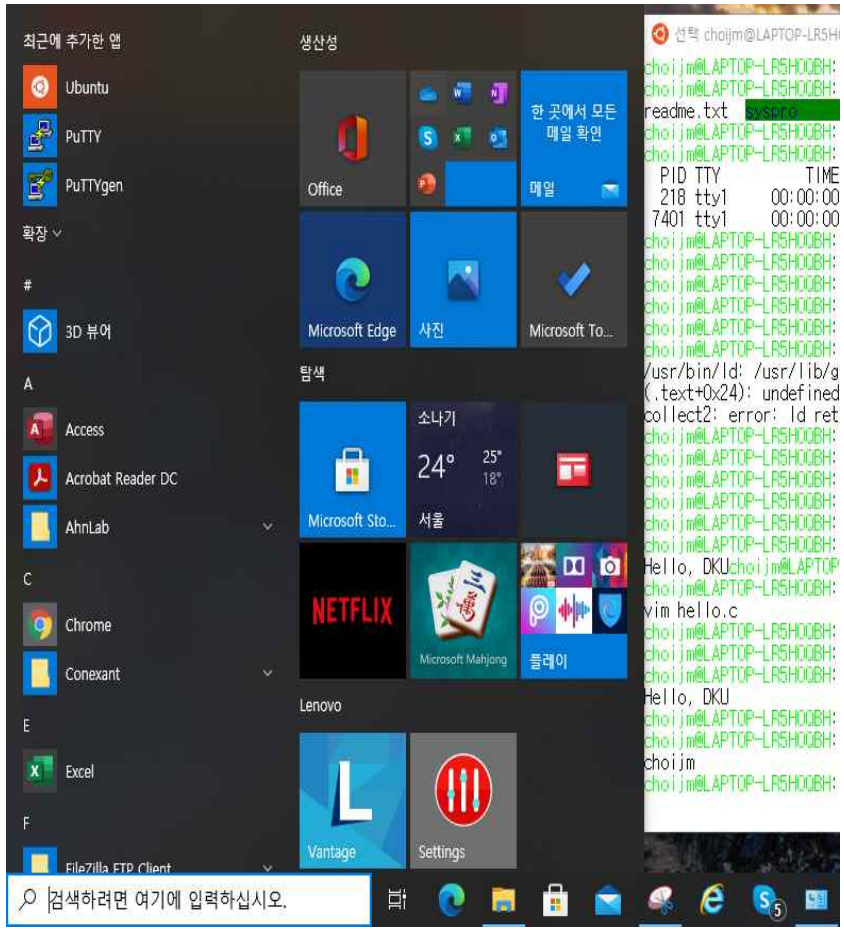
- Quiz
  - ✓ 1) Explain the command “$ls –l | sort” in the below left figure.
  - ✓ 2) What is the background music in “Dr Jeong-Joon Lee’s Kaggle Demo” in Page 30”? What commands can you find in the Kaggle Demo? (at least 5 that you have learned in the LN2.)
  - ✓ Due: until 6 PM Friday of this week (1st, October)

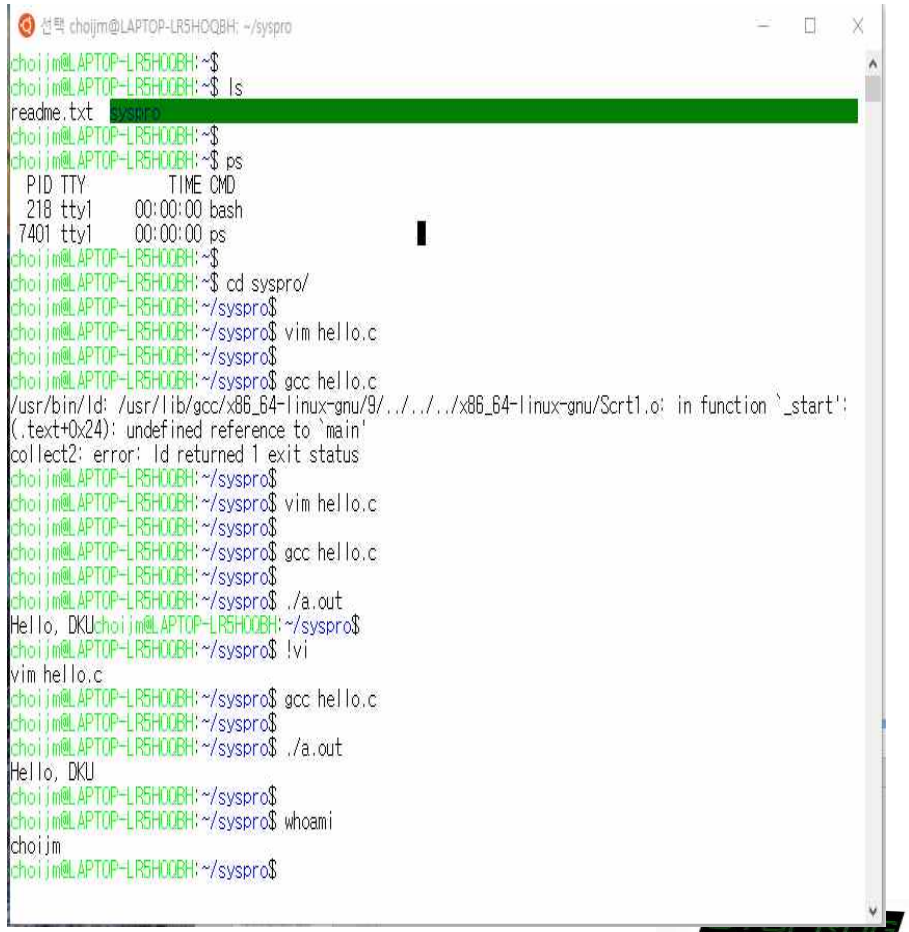




Appendix 2: How to access Linux: Alternative

- 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