Linux

Lecture 24

Hyung-Sin Kim

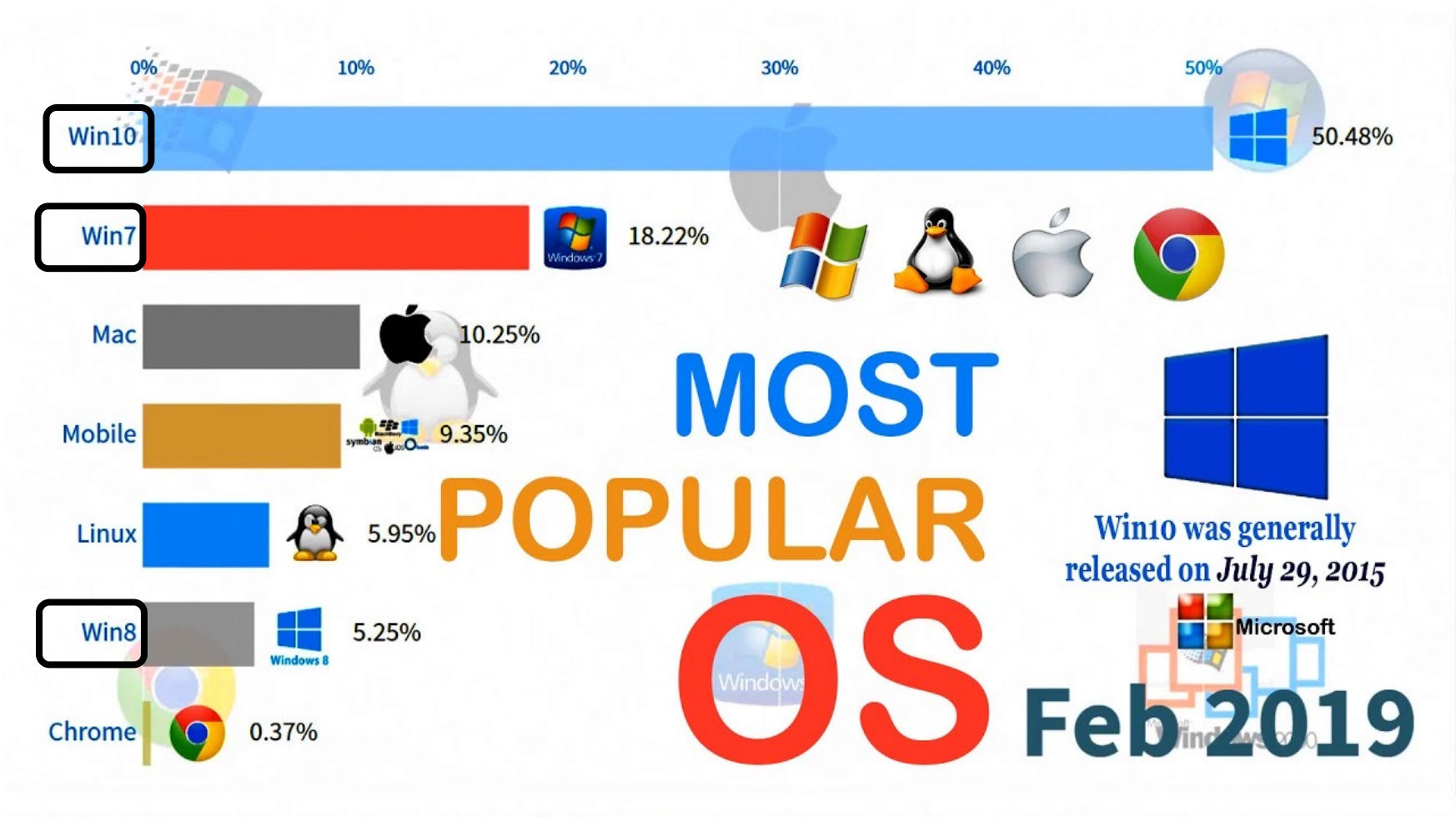
SNU Graduate School of Data Science

Contents

- Why Linux?
- History of Linux
- Quick overview of Ubuntu

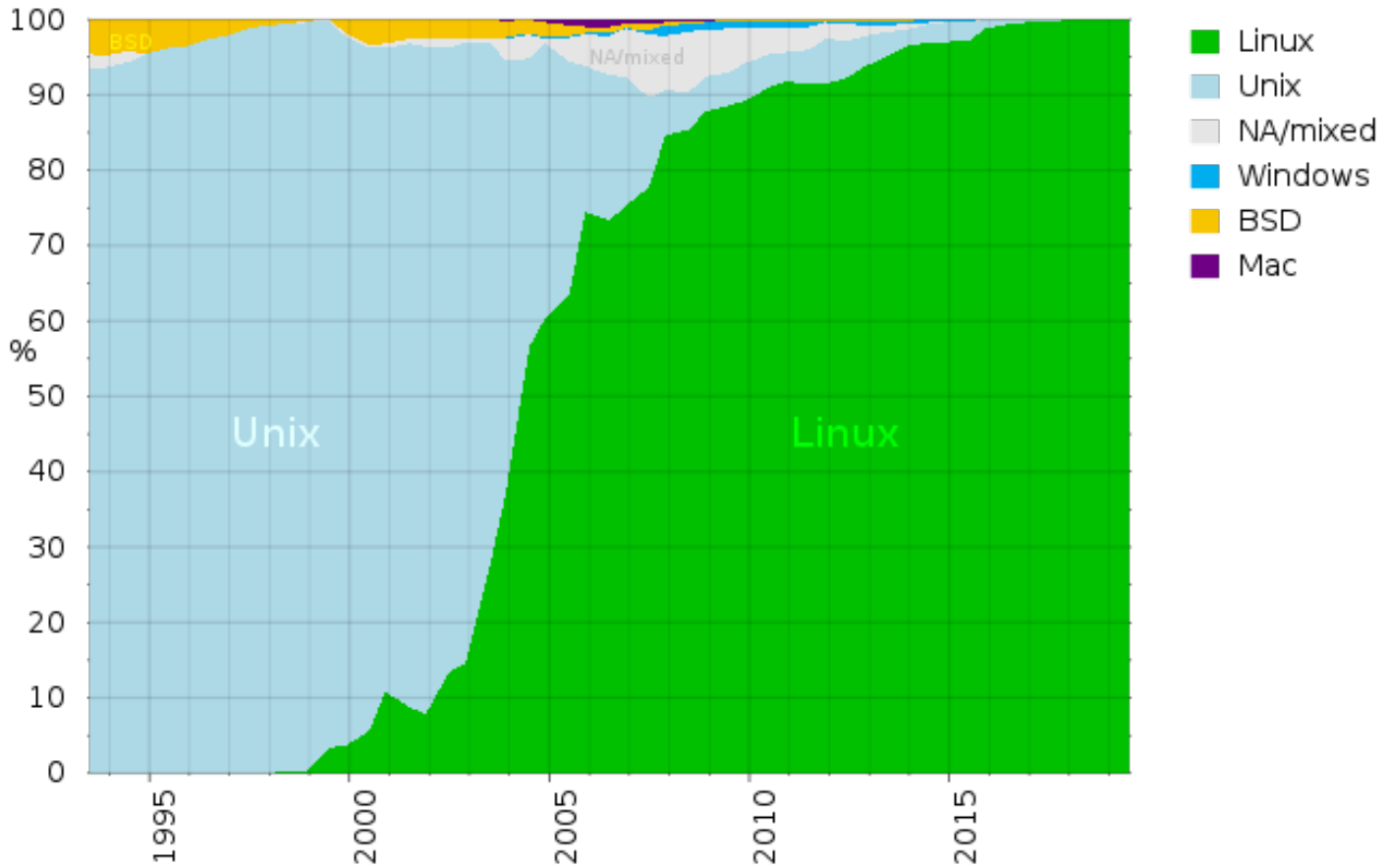
Why Linux?

Operating Systems on All Computing Devices



Then… why should you know Linux?

Operating Systems on Supercomputers



Why Linux?

- What operating system should you use? The answer depends on what kind of application software **YOU** need to use.
  - Many application software is developed for Windows (MS office, game, iexplorer)
  - Some application software (especially those for developers) is developed for Linux

- If you are heading toward data science, you will encounter Linux very soon
  - Simply because application software you want to use supports Linux better than Windows
  - Especially when you want to access a supercomputer remotely for processing **big data** …

Why Ubuntu?

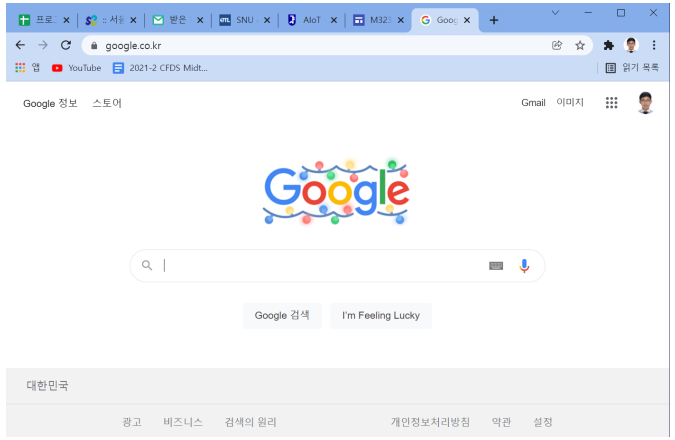
- There are lots of Linux-based operating systems

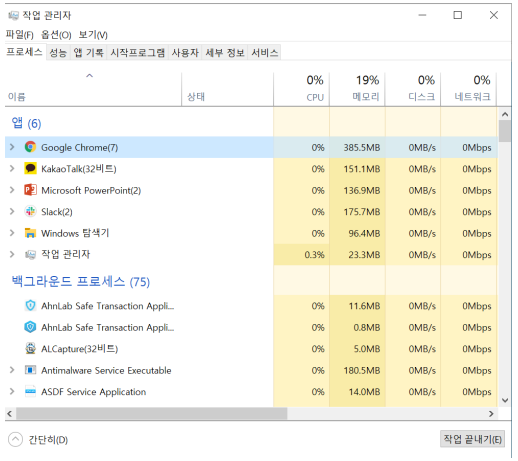
- Important things to consider
  - Well maintained
  - Easy to use
  - Large community (what others use…)

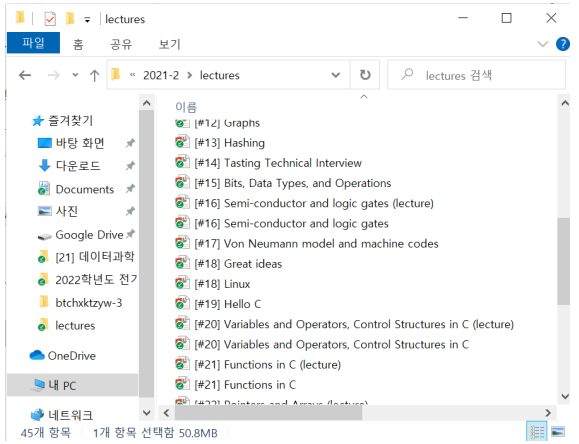
- Ubuntu is the **most popular** desktop Linux distribution these days

What is “Knowing and Using Linux/Ubuntu”?

- What is “knowing and using Windows” in practice?
  - Install/use/delete various application software on Windows for your tasks
    - Install and use application software - MS office, Abobe, …
    - File management
    - Networking, web browsing, email
    - Process management







It does not necessarily mean that you can play with Windows operating system software

What is “Knowing and Using Linux/Ubuntu”?

- What is “knowing and using Windows” in practice?
  - Install/use/delete various application software on Windows for your tasks
    - Install and use application software - MS office, Abobe, …
    - File management
    - Networking, web browsing, email
    - Process management

- You can be familiar with Ubuntu by doing the same things
  - Being familiar with application software on Ubuntu first

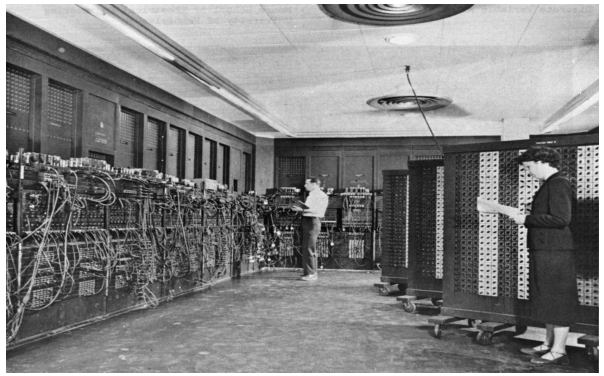
History of Linux

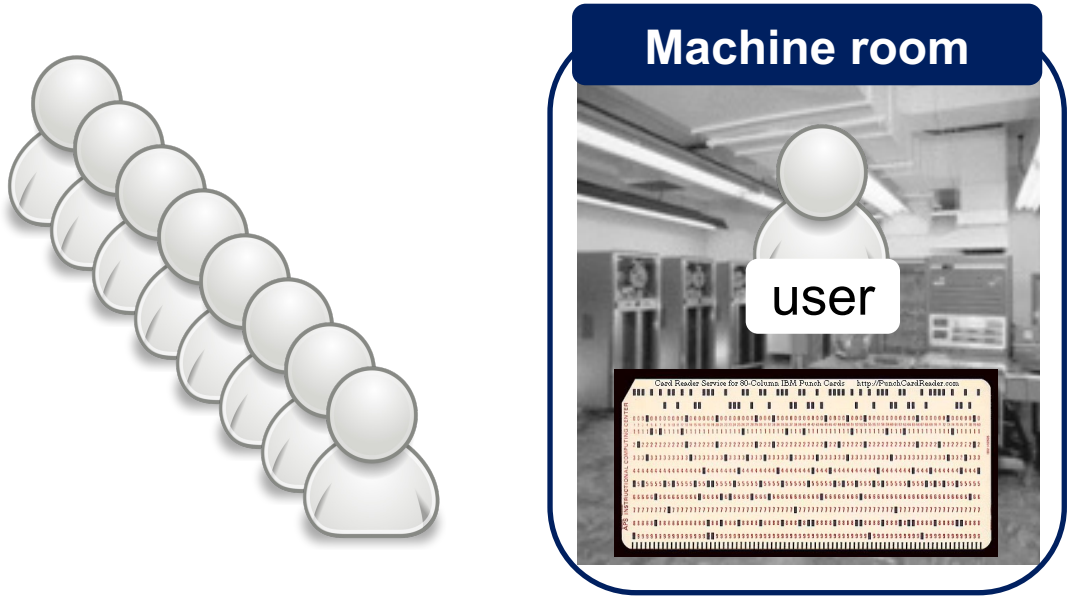
Computing Bootcamp

In the Beginning…

- There was no PCs (personal computers) but huge and expensive mainframes
  - Only governments and huge companies could own the mainframes

- A computer could run only one program at a time
  - Next user should wait in front of the machine room until the current user finishes her job

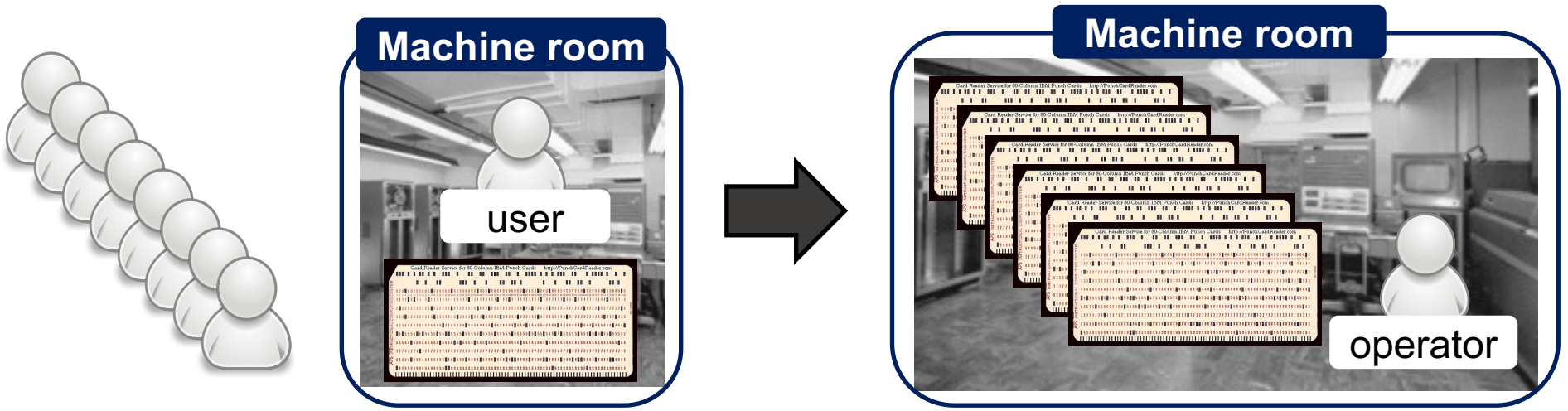



Machine room
user

Hardware Became More Powerful

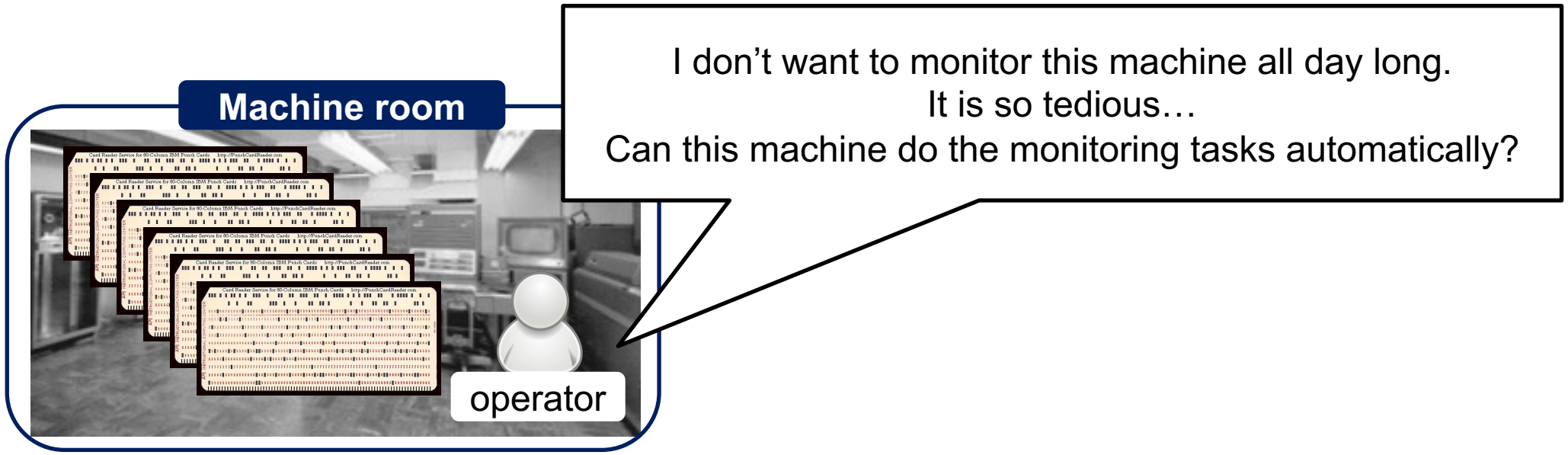
- Less time to run a program
  - Time wastage for physically switching users became relatively large

- Stack of people ➡ Stack of jobs (punch cards or disks)
  - A dedicated machine operator monitors if the machine does its jobs well



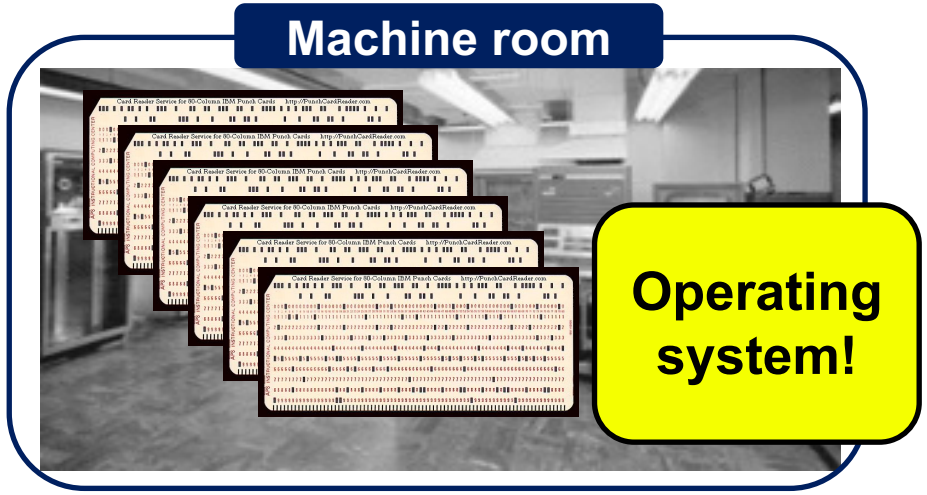
Monitoring is Boring …

- Monitoring task
  - Assign hardware resource for a customer job, execute the customer job, monitor during the execution, record its usage, when the job ends, reassign hardware resource to the next customer job …



Monitoring is Boring …

- Monitoring task
  - Assign hardware resource for a customer job, execute the customer job, monitor during the execution, record its usage, when the job ends, reassign hardware resource to the next customer job …

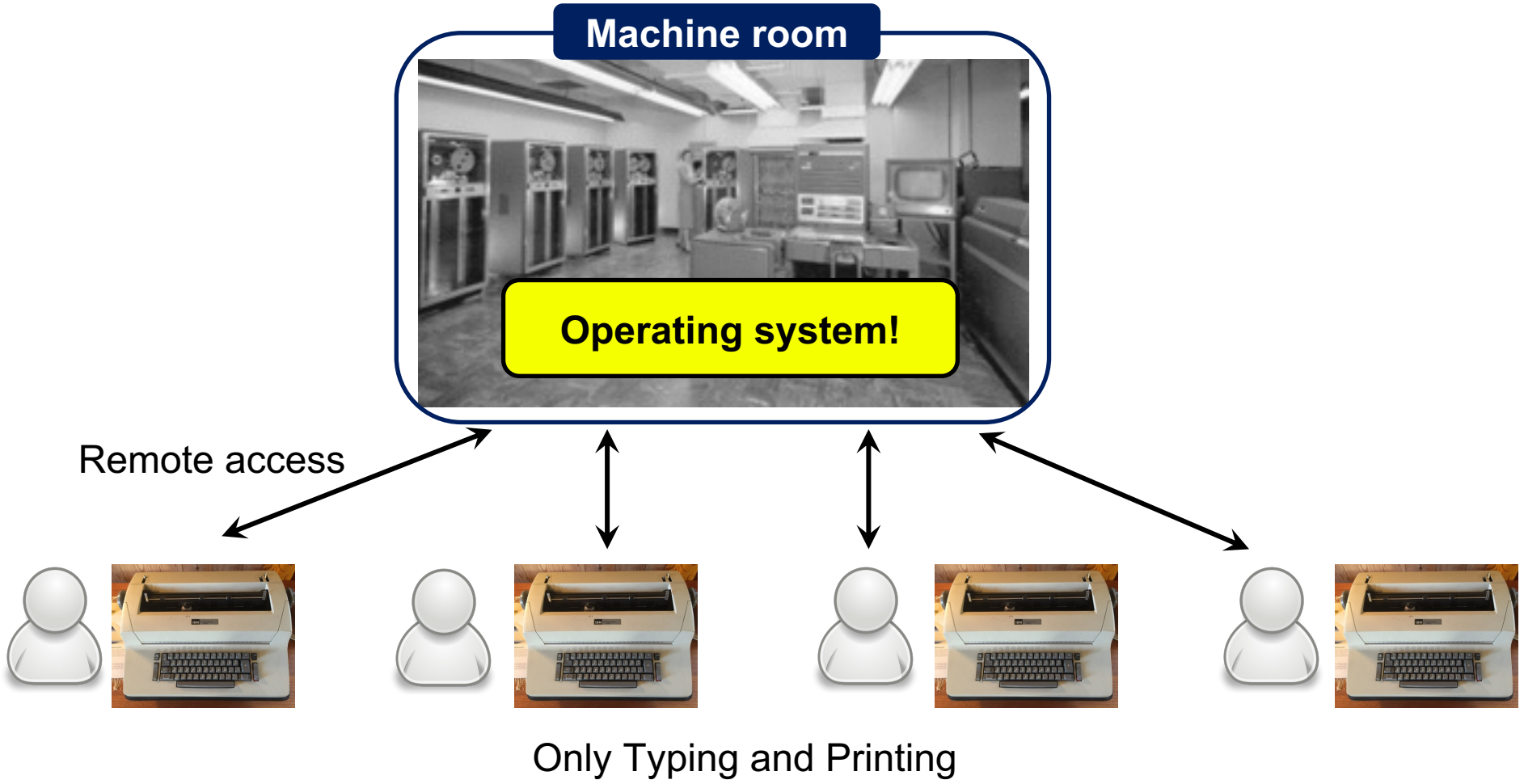


Hardware management

Software (program) scheduling

Resource monitoring

Toward Multi-user and Multi-tasking

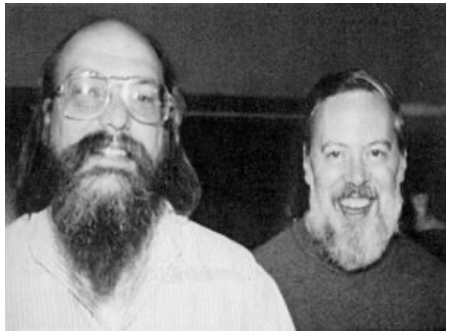


1964 ~ 1969: Multics Project

- Multics: **Multiplexed** Information and Computing Service
  - Time-sharing operating system, allowing multiple users to access a mainframe simultaneously
- AT&T Bell Labs, MIT, CSRG, and GE
- Failed but there were many novel ideas!

1970: Unix Operating System

- Unix: **Uniplexed** Information and Computing Service
  - Single task operating system
- Ken Thompson and Dennis Ritchie (Bell Labs) who were part of Multics project
- Original Unix was written in Assembly language (i.e., hardware-dependent code) and ran only on a single-type of machine, PDP-11
- A very primitive OS without portability and multi-tasking
  - But importantly, it worked!
- It has gradually been improved, now a portable, multi-tasking, and multi-user OS



US Gov, AT&T, Unix Spread, and Open Source

- AT&T became **too big** and the US Gov started regulating this company
  - AT&T was prevented from entering the computer business and forced to license Unix source code to anyone who asked
  - Unix grew quickly and became **widely adopted** by academic institutions and businesses

- In 1982, however, the US Gov broke up AT&T into 7 smaller companies
  - And after the break-up, Unix was allowed to be sold as a **proprietary product**

- In 1983, GNU Project started to create a "complete Unix-compatible **FREE** software system" – **"GNU's Not Unix!"**
  - By the early 1990s, however, it was not still 100% complete

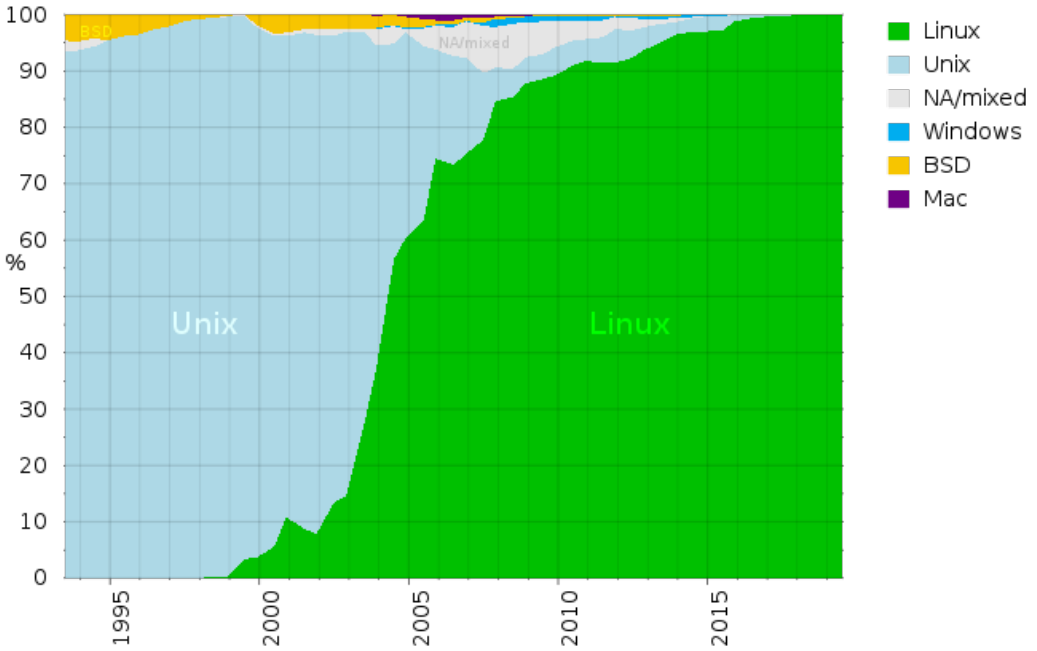
1991: Linux by Linus Torvalds

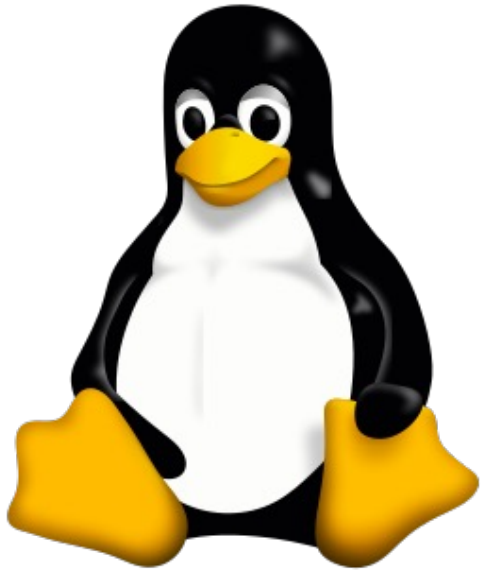
- In 1987, Prof. Andrew Tanenbaum built MINIX, a **minimal** Unix-like operating system for educational purpose

- As a graduate student, Linus Torvalds wrote Linux, a Unix-like operating system based on MINIX, which works on **PC**s, and made it free
  - Unix was designed for supercomputers (mainframes)
  - At that time, Microsoft Windows and Apple Mac OS were already out there for PCs

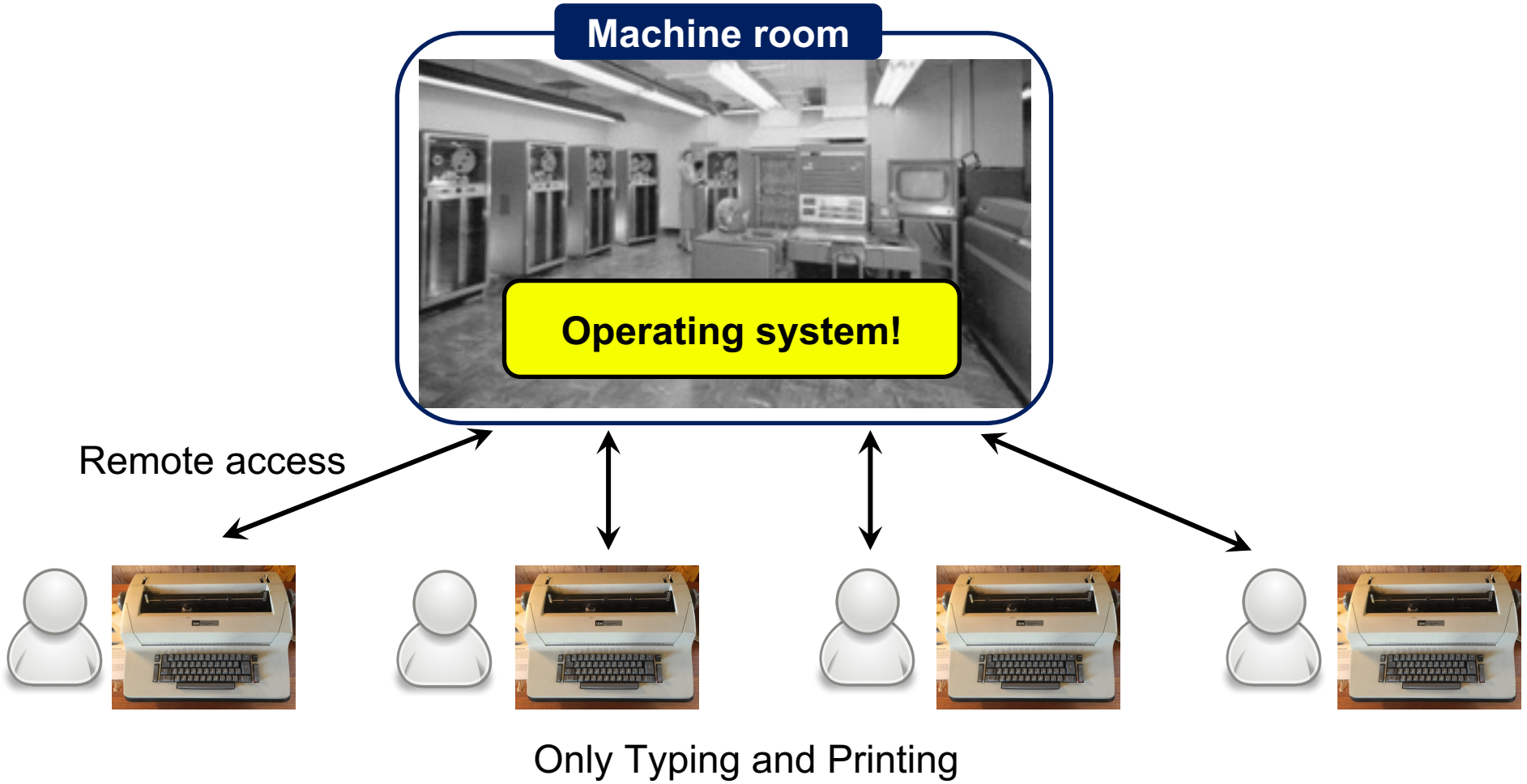
- Nevertheless, Linux attracted a lot of people
  - People who already know Unix and want to run it on PC-type hardware
  - People who want to experiment with operating system principles
  - People who need or want to control their operating system
  - People who have personal problems with Microsoft

Now Linux is the most widely used Unix-based operating system

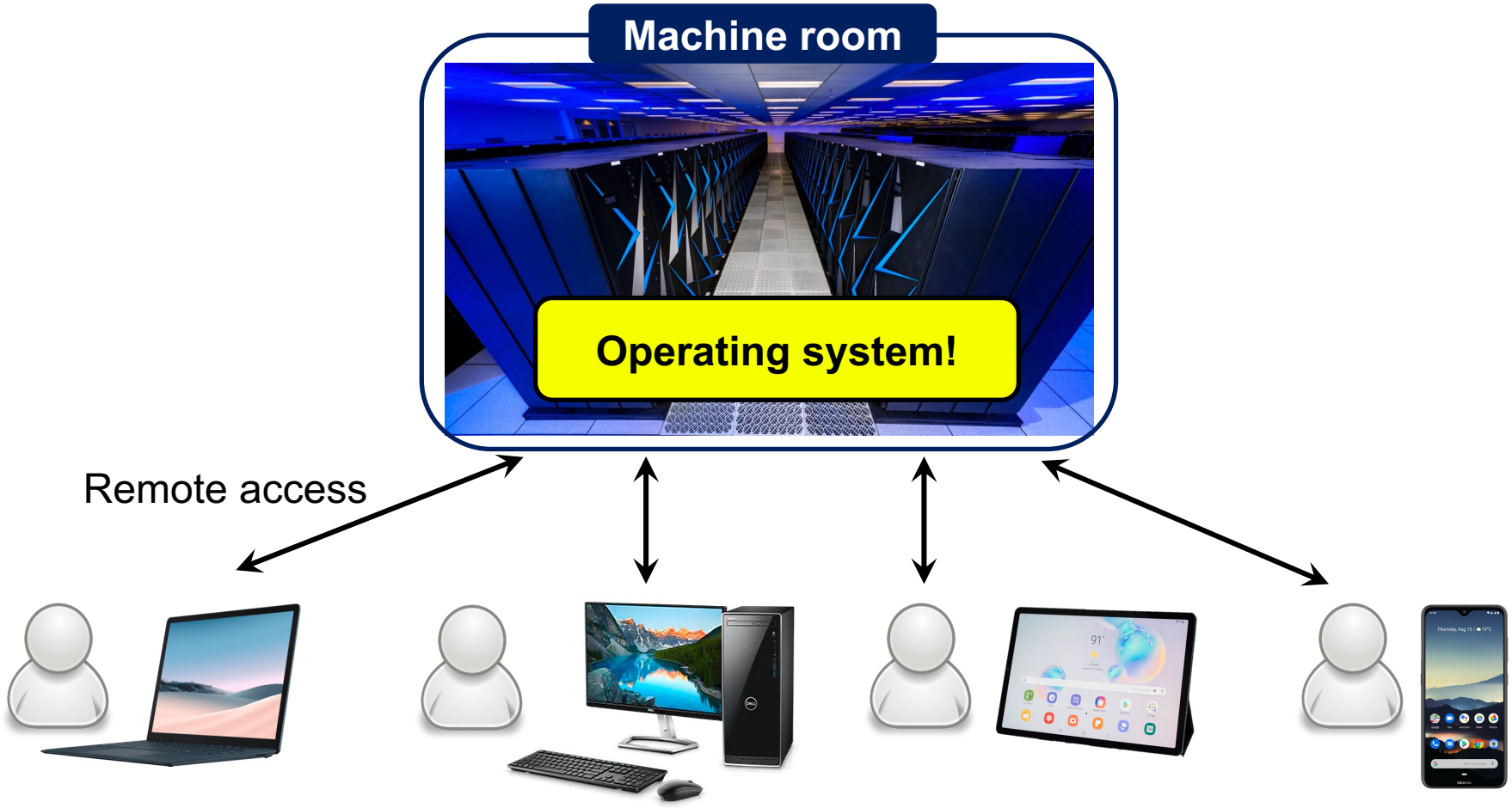




User Scenario that Unix Supports Well



User Scenario for Data Science (Good with Linux)



This is the reason why you need to learn Linux (Ubuntu)
Windows is not enough!

Quick overview of Ubuntu

Computing Bootcamp

Linux Shell – Bridging a User and Linux OS

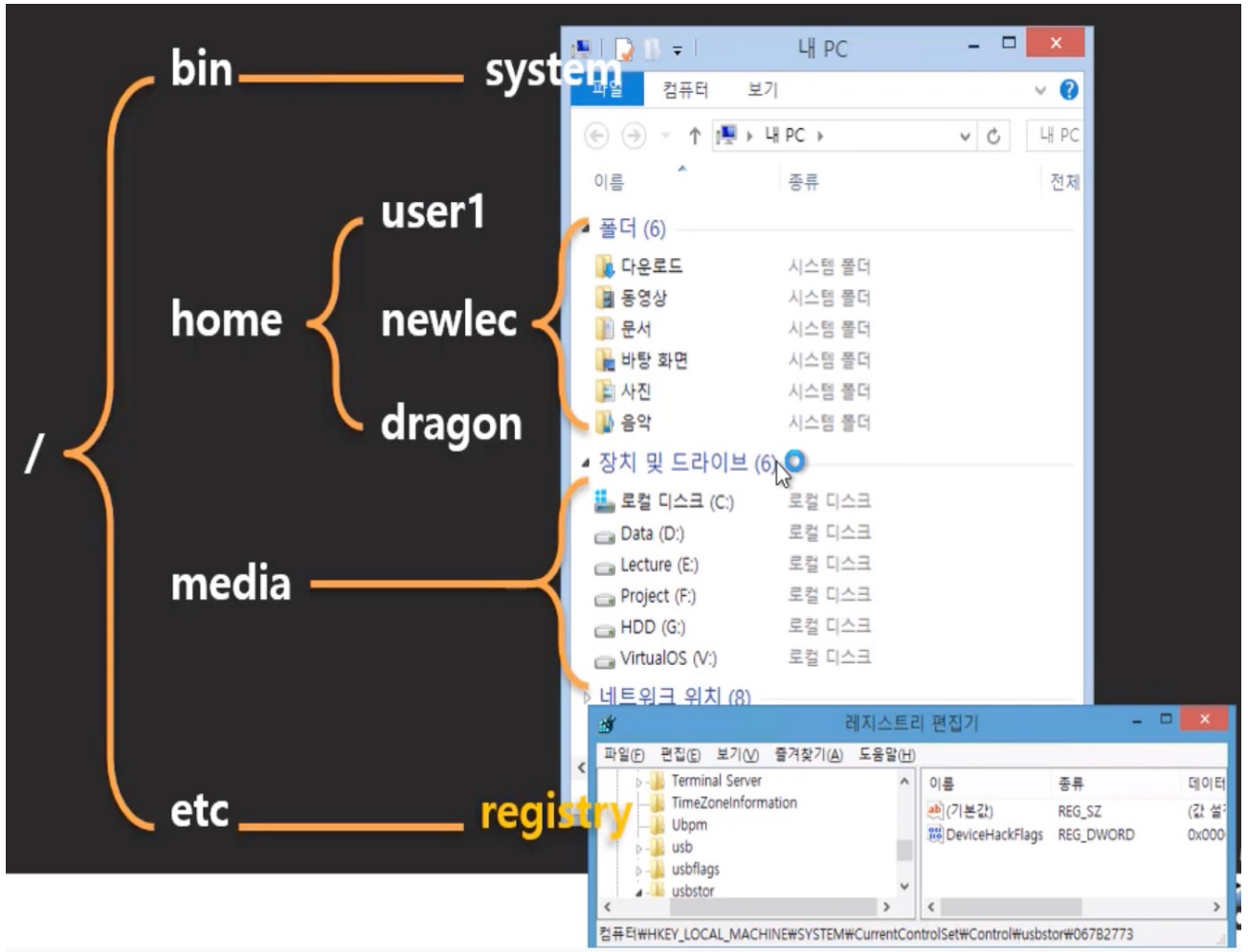
- Start a terminal: Ctrl+Alt+T

- ~$: regular user authority / ~#: root user authority

- When you want to execute something with root authority
  - ~$: sudo <<command>>
- When you switch user to the root
  - ~$: sudo su – root
  - ~#: exit (when you want to come back to a regular user)
  - ~$:

Linux File System

- /: root directory
- ~: user home directory
- /bin: execution files
- /etc: setting files
- /home: user files
- /media: drive like C: / D:



Linux File System

- Shell commands for navigation
  - pwd: print working directory
  - cd xx: change directory to xx
  - ls: list (ls -l: list with detailed information)

- Shell commands for file management
  - mkdir xx: make a directory xx
  - rmdir xx: remove a directory xx
  - touch xx: make an empty file xx
  - mv xx yy: move (or rename) file xx to yy
  - rm xx: remove file xx (rm –r xx)
  - cp xx yy: copy file xx to yy

Linux File System

- Shell commands for finding and looking into files
  - find <location> -name xx: find file with the name xx under <location>
  - cat xx: show content of file xx
  - head –n2 xx: show the first 2 lines of file xx
  - tail –n2 xx: show the last 2 lines of file xx
  - grep xx yy: show the lines of file yy that contains xx
  - diff xx yy: show the difference between files xx and yy

Software Package Install and Removal

- There are lots of software packages in Ubuntu package store

- Shell commands related to APT (Advanced Packaging Tool)
  - /etc/apt/sources.list: list of ubuntu package stores
  - apt-get update: update package list in the ubuntu package store
  - apt-get dist-update
  - apt-cache search X: find package X in the package list
  - apt-cache show X: show package X in the package list
  - apt-get install X:
  - apt-get remove X: Remove X
  - apt-get purge X: Remove X and its setting files

Summary

Summary

- Why Linux/Ubuntu?
  - Application softwares for data science support Linux better than Windows
  - Ubuntu: the most popular desktop Linux distribution
- History of Linux
  - Stack of users → Stack of jobs → Multi-user and Multi-tasking
  - Unix → Linux
- Quick overview of Ubuntu
  - shell
  - file system

Q&A

Any questions?

Thanks!