

# 430.211 Programming Methodology (프로그래밍 방법론)

## Linux part I

Lab2 #01

Fall 2025

# Lab2

- Monday (lab2)
  - Linux, Git, sorting algorithm, STL
- Wednesday (lecture)
- Friday (lab I)
  - Programming exercise after the lecture

# Lab2 TAs

- Heewoong Choi ([chw0501@snu.ac.kr](mailto:chw0501@snu.ac.kr))
- Hyungwoong Bae ([hwbae0326@snu.ac.kr](mailto:hwbae0326@snu.ac.kr))
- Sumin Yu ([ysmsoomin@snu.ac.kr](mailto:ysmsoomin@snu.ac.kr))



# Tentative Schedule for Lab2

Week	Contents	Week	Contents
1 (9/1)	Linux– Part. 1	9 (10/27)	No class
2 (9/8)	Sorting Algorithm– Part. 1	10 (11/03)	Standard Template Library – Part. 3
3 (9/15)	Sorting Algorithm– Part. 2	11 (11/10)	Standard Template Library – Part. 4
4 (9/22)	Version control system (Git) – Part. 1	12 (11/17)	<b>Project 2</b>
5 (9/29)	Version control system (Git) – Part. 2	13 (11/24)	Linux– Part. 2
6 (10/06) Chuseok	Standard Template Library – Part. 1 (No class. recorded video)	15 (12/01)	Problem Solving
7 (10/13)	Standard Template Library – Part. 2	15 (12/08)	Problem Solving
8 (10/20)	<b>Project 1</b>	15 (12/15)	No class

\*The project is planned as a team project  
(2~3 people)

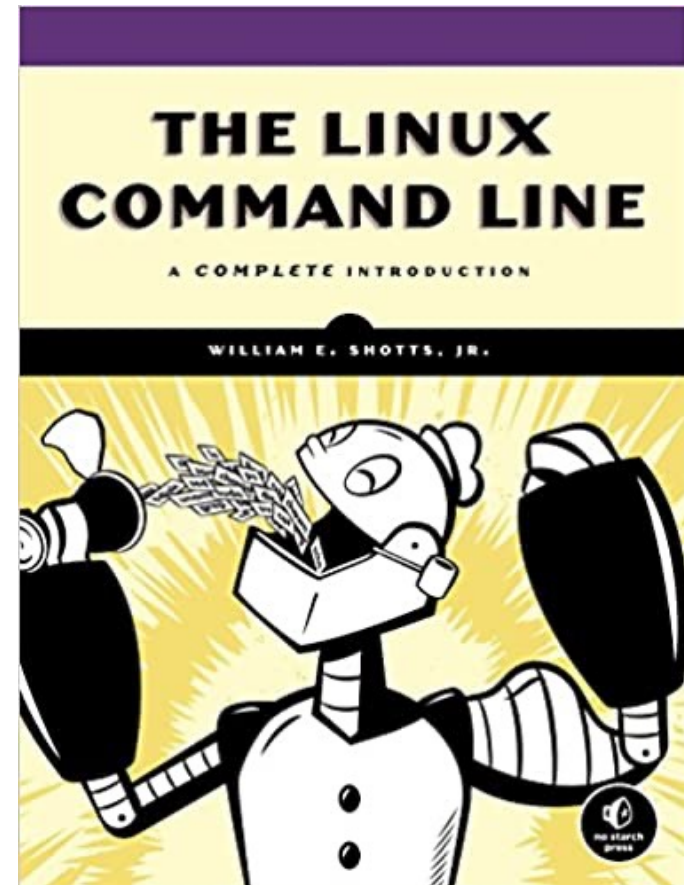
- Heewoong Choi: Linux & Sorting
- Hyungwoong Bae: STL
- Sumin Yu: Git

# Outline

- **Linux**
- **Shell tools**
- **Vim**

# Reference

- The Linux Command Line: A complete Introduction
- [MIT missing semester](#)



# Brief History

- UNIX (1970s)
  - Bell Labs developed operating system
  - Dennis Ritchie who also developed the C language
- GNU project (1983)
  - GNU's not Unix (i.e., replacement for UNIX)
  - The free software project started by Richard Stallman
- Linux (1991)
  - A UNIX-like operating system developed by Linus Torvalds
- GNU project adopted Linux kernel
  - GNU/LINUX OS (commonly referred to as "Linux")

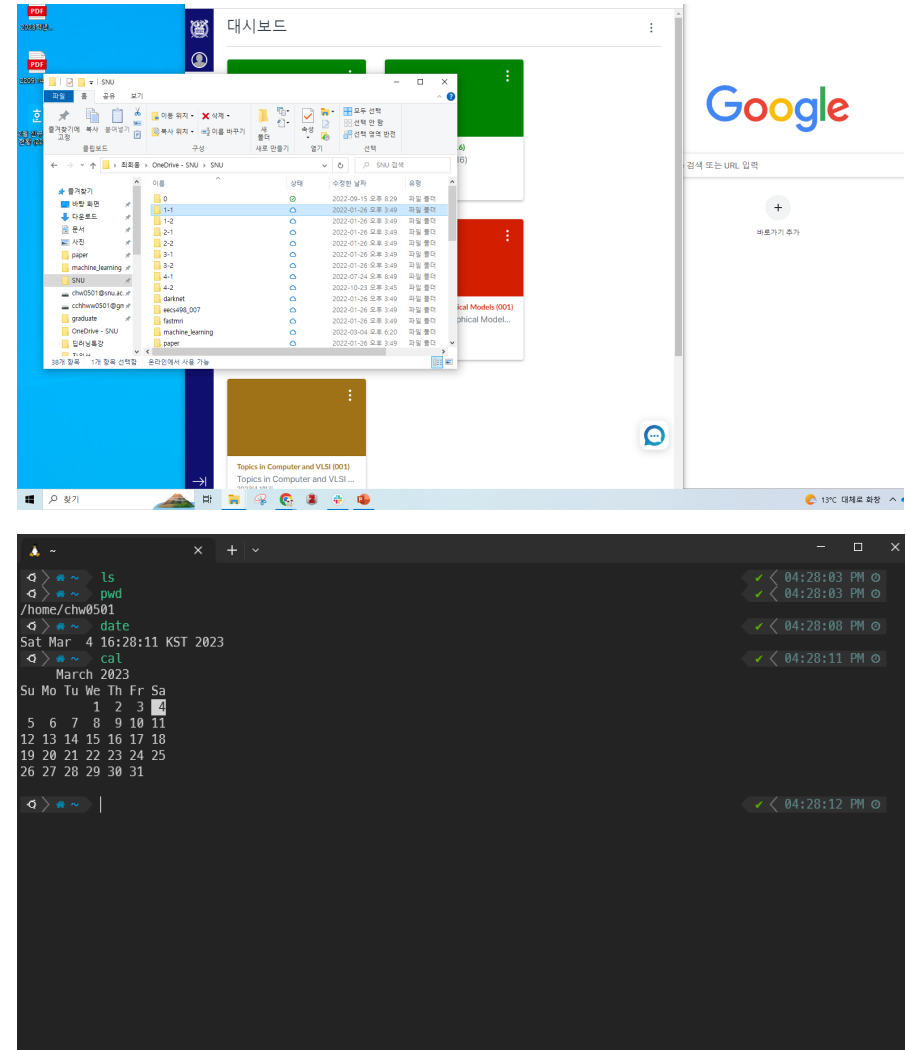
# OS

- Linux distribution
  - Ubuntu, CentOS, Fedora, Debian, Kali Linux,...
- macOS
  - BSD (Berkeley Software Distribution) Unix-based OS
- Windows
  - Developed by Microsoft Corporation
  - Different from Unix-based OS



# GUI vs CLI

- GUI (graphical user interface)
- CLI (command-line interface)
  - Many powerful commands and tools!



# Install Linux on windows

- Dual Booting
- Virtual machine (VMware)
- WSL2 (Windows Subsystem for Linux 2)
  - Highly recommend for Windows users
    - If you want to get used to the Linux
    - Easy to install


# Without Installation

- Using 코딩연습장 in etl

- <https://etl.snu.ac.kr/>







# Without Installation

**코딩연습장**

3 / 4

[이어서하기](#)

학습 수업 자료

 자유 코드 연습장 (CLI)	0
 자유 코드 연습장 (Jupyter_Notebook)	100
 자유 코드 연습장 (VS Code)	100
 자유 코드 연습장 (GUI)	100

하위 수업 목록

현재 하위 수업이 없습니다.

# Without Installation

The screenshot displays a web-based development environment interface. On the left, a sidebar titled '코딩연습장' (Coding Practice Room) lists four environments: 01 자유 코드 연습장 (CLI), 02 자유 코드 연습장 (Jupyter\_Notebook), 03 자유 코드 연습장 (VS Code) (highlighted with a red box), and 04 자유 코드 연습장 (GUI). The main content area is divided into two sections. The top section, titled '자유 코드 연습장 (VS Code)', provides an overview of the environment and lists supported features: 웹 서버(Nginx), Node.js, and 데이터베이스(PostgreSQL). The bottom section, titled '사용 방법' (Usage Method), includes instructions on how to use the environment, such as setting up the folder path and using the terminal. The right side of the interface shows the RunBox logo and a 'Start' button, along with a 'Next Up' section for deploying code-server and a 'Walkthroughs' section for getting started with VS Code for the web.

코딩연습장

코딩연습장 > 자유 코드 연습장 (VS Code)

자유 코드 연습장 (VS Code)

자유 코드 연습장(VS Code)은 다양한 프로그래밍 언어 지원, 디버깅, 터미널, 익스텐션(Extension) 등 실제 VS Code의 대부분의 기능을 이용할 수 있는 환경을 지원합니다.

자유 코드 연습장은 아래의 프론트엔드와 백엔드 개발 환경을 중심으로 세팅되어 있으므로 바로 이용할 수 있습니다.

- 웹 서버(Nginx)
- Node.js
- 데이터베이스(PostgreSQL)

VS Code 환경에서 자유롭게 코드를 작성해 보세요!

사용 방법

- 초기 세팅을 진행하고 싶다면, 화면 좌측의 Open Folder 버튼을 클릭합니다.
- 폴더 경로를 `/home/elicer/code_practice` 로 설정하고 OK 버튼을 클릭합니다.

터미널(Terminal)을 사용하고 싶다면 좌측 메뉴에서

RunBox

남은 시간 : 제한없음

Start

- New File...
- Open File...
- Clone Git Repository...

Recent

You have no recent folders, [open a folder](#) to start.

Next Up

Deploy code-server for your team

Provision software development environments on your infrastructure with Coder.

Coder is a self-service portal which provisions via Terraform—Linux, macOS, Windows, x86, ARM, and, of course, Kubernetes based infrastructure.

Get started →

Templates

Name	Used By
Kubernetes	128 developers
Windows	21 developers

Walkthroughs

Get Started with VS Code for the Web

Customize your editor, learn the basics, and start coding.

# Without Installation

코딩연습장

코딩연습장 1/1

- 01 자유 코드 연습장 (CLI)
- 02 자유 코드 연습장 (Jupyter\_Notebook)
- 03 자유 코드 연습장 (VS Code)**
- 04 자유 코드 연습장 (GUI)

자유 코드 연습장 (VS Code)

자유 코드 연습장(VS Code)은 다양한 프로그래밍 언어 지원, 디버깅, 터미널, 익스텐션(Extension) 등 실제 VS Code의 대부분의 기능을 이용할 수 있는 환경을 지원합니다.

자유 코드 연습장은 아래의 프론트엔드와 백엔드 개발 환경을 중심으로 세팅되어 있으므로 바로 이용할 수 있습니다.

- 웹 서버(Nginx)
- Node.js
- 데이터베이스(PostgreSQL)

VS Code 환경에서 자유롭게 코드를 작성해 보세요!

사용 방법

- 초기 세팅을 진행하고 싶다면, 화면 좌측의 Open Folder 버튼을 클릭합니다.
- 폴더 경로를 `/home/elicer/code_practice` 로 설정하고 OK 버튼을 클릭합니다.

터미널(Terminal)을 사용하고 싶다면 좌측 메뉴에서

RunBox

남은 시간 : 제한없음

항상 켜두기

Welcome

Start

- New File...
- Open File...
- Clone Git Repository...

Recent

You have no recent folders, [open a folder](#) to start.

Next Up

Deploy code-server for your team

Provision software development environments on your infrastructure with Coder.

Coder is a self-service portal which provisions via Terraform—Linux, macOS, Windows, x86, ARM, and, of course, Kubernetes based infrastructure.

Get started →

Templates

Name	Used By
Kubernetes	128 developers
Windows	21 developers

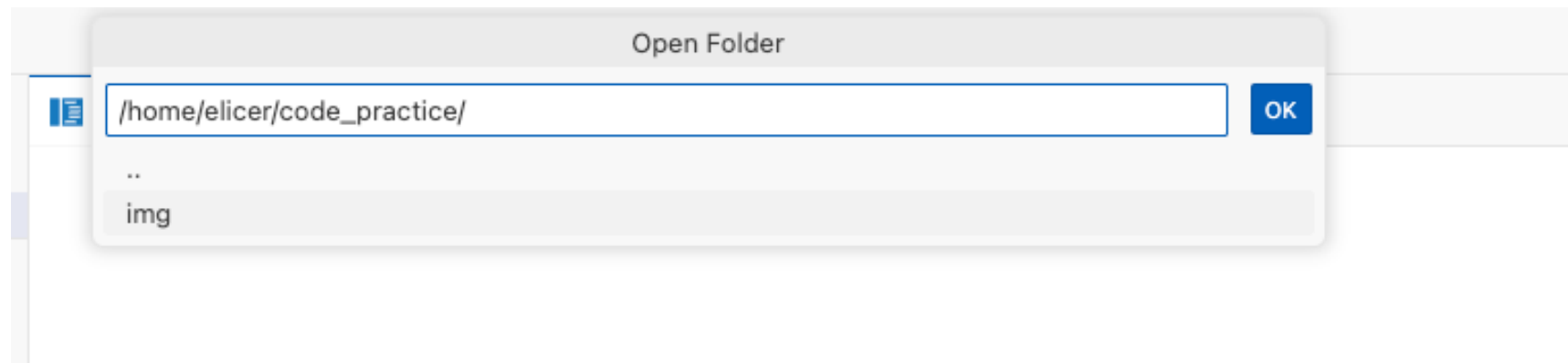
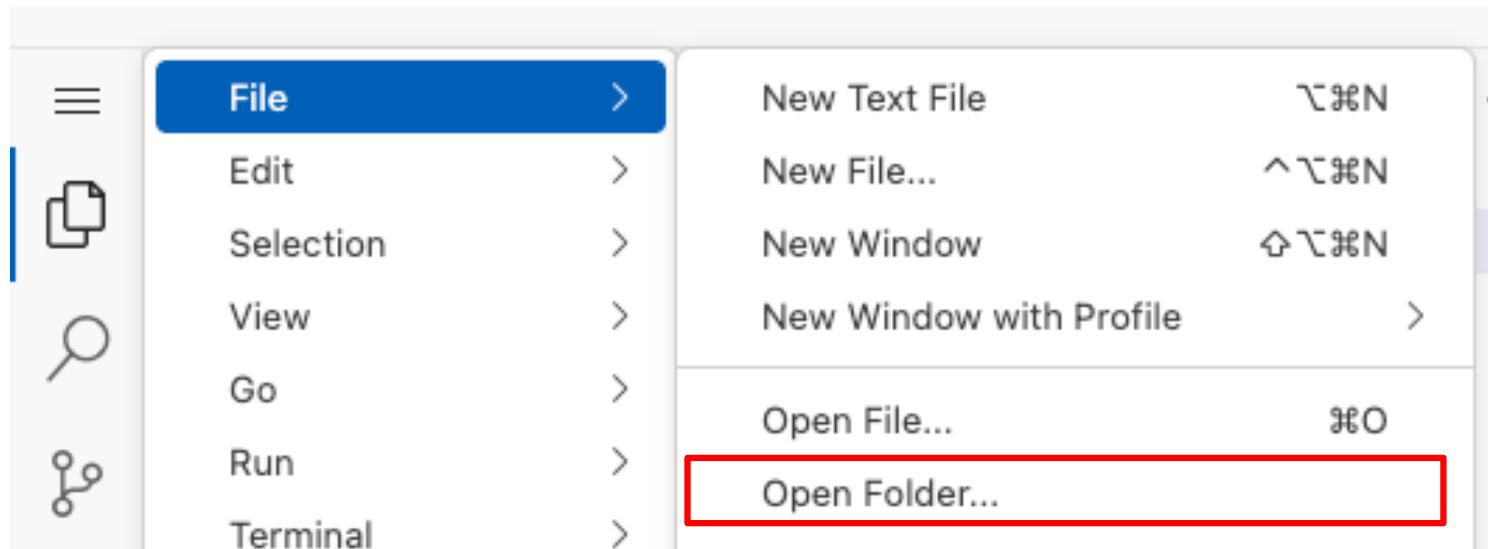
Walkthroughs

Get Started with VS Code for the Web

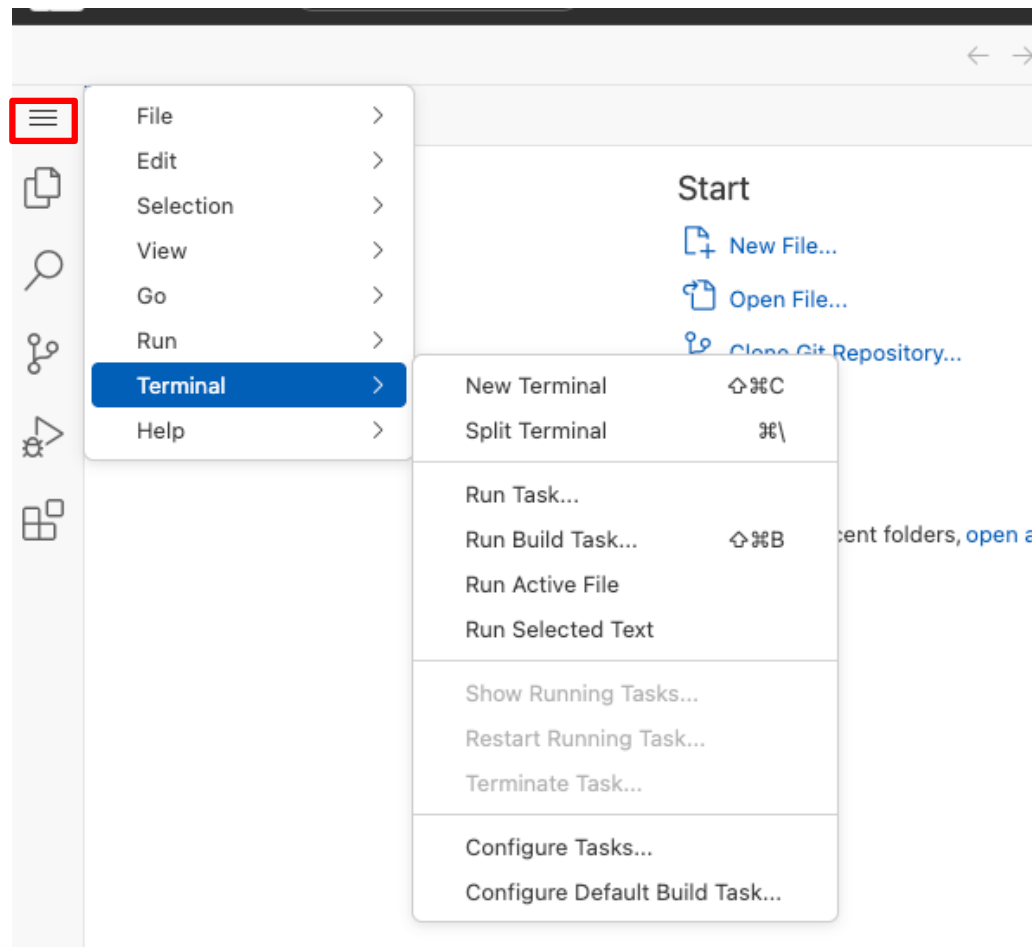
Customize your editor, learn the basics, and start coding.

bash

# Without Installation

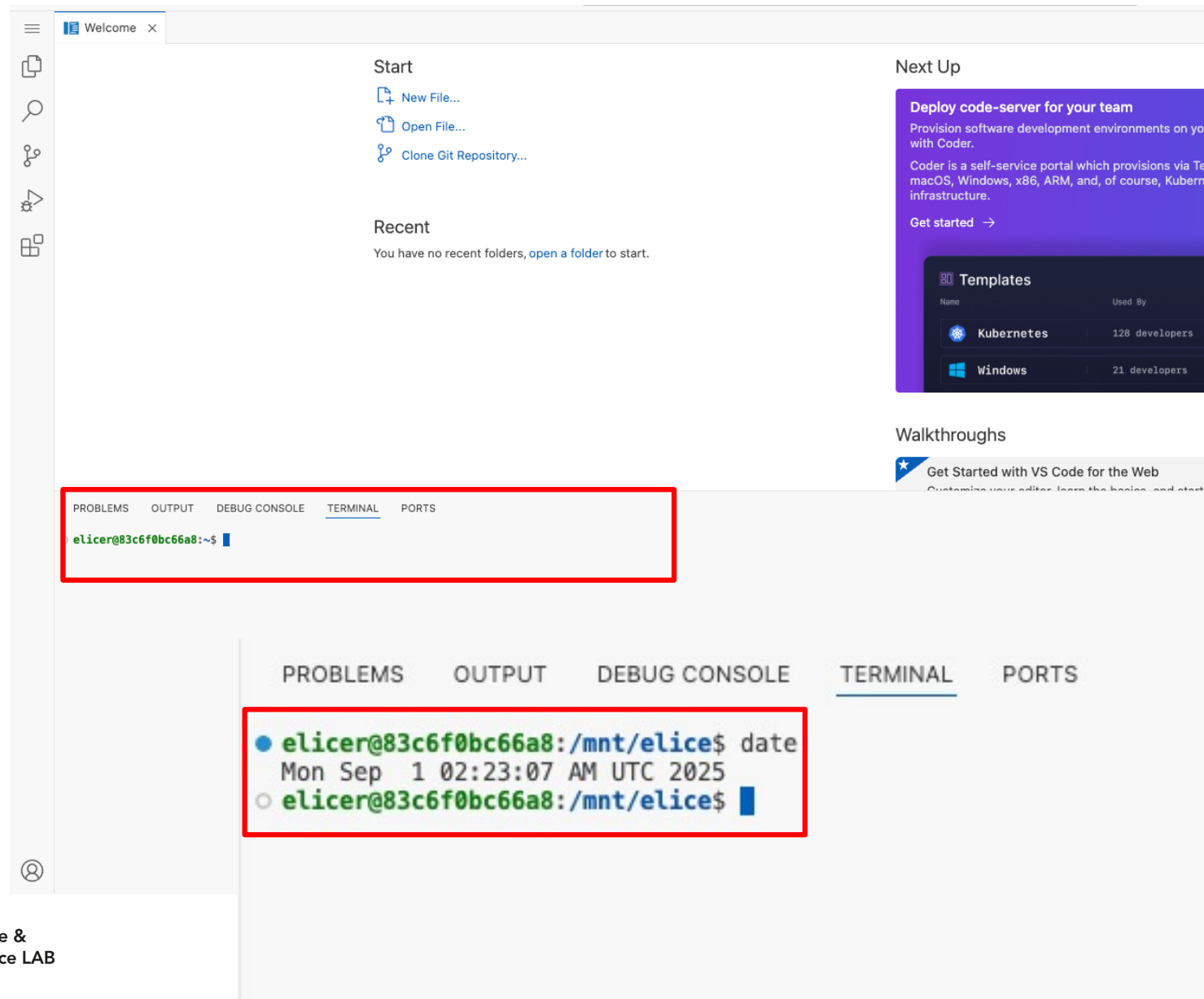


# Without Installation





# Without Installation

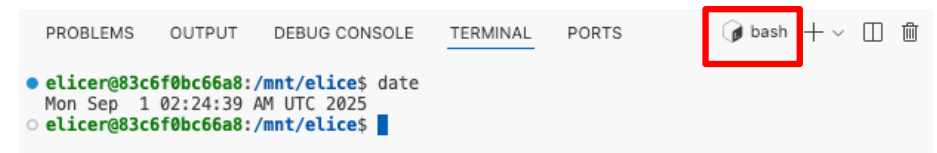


# Outline

- **Linux**
- **Shell tools**
  - cd, ls, pwd, mkdir, cp, rm, mv, ...
- **Vim**

# Shell tools

- Terminal
  - The program that provides the user interface for entering commands and viewing output
- Shell
  - A program that interprets the user's commands and executes them
  - **bash**, zsh, fish, PowerShell...
- Shell prompt
  - Text shown in a terminal that indicates the shell is ready to accept commands



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS bash + - [x] [trash]  
• elicer@83c6f0bc66a8:/mnt/elice$ date  
Mon Sep 1 02:24:39 AM UTC 2025  
○ elicer@83c6f0bc66a8:/mnt/elice$
```

# Shell tools

root@Linux:~# adfasdf

```
❌ test-38004386:~/test{main}$ adfasdf
bash: adfasdf: command not found
○ test-38004386:~/test{main}$
```

root@Linux:~# date

```
PROBLEMS 1 OUTPUT TERMINAL bash + - 
● test-38004386:~/test{main}$ date
  Sun Aug 31 08:02:07 AM UTC 2025
○ test-38004386:~/test{main}$
```

# Shell tools

root@Linux:~# df

root@Linux:~# df -h

- option/flag

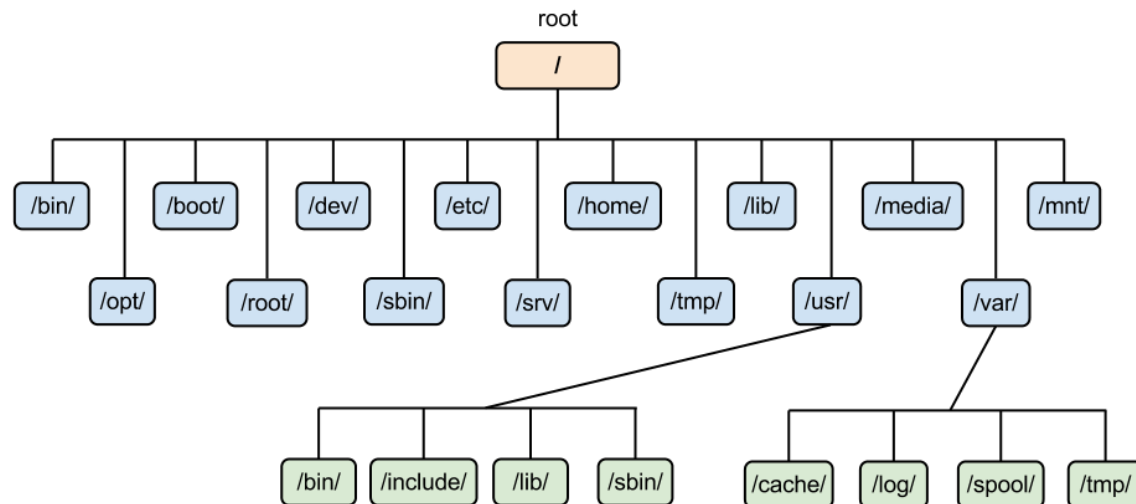
```
● test-38004386:~/test{main}$ df -h
Filesystem      Size  Used Avail Use% Mounted on
tmpfs            3.9G   12K   3.9G   1% /
tmpfs            64M    0    64M   0% /dev
shm             64M   4.0K   64M   1% /dev/shm
overlay         25G   23M   24G   1% /nix
tmpfs            3.9G    0   3.9G   0% /nix/store/pf5avvvL4ssd6kylcvg2g23hcjp71h19-glibc-2.39-52/etc
overlay         25G   23M   24G   1% /etc/bashrc
overlay         25G   23M   24G   1% /etc/login.defs
overlay         46G   33G   13G   74% /mnt
/dev/sdb1        46G   33G   13G   74% /etc/hosts
/dev/sda         23G   23G    0 100% /ephemeral
/dev/disk/by-id/google-home 9.8G 655M  8.6G   7% /home
overlay         9.8G   36K   9.3G   1% /mnt/ephemeral
tmpfs            3.9G    0   3.9G   0% /run
tmpfs            3.9G    0   3.9G   0% /var
```

# Shell tools

\$ pwd

- Print Working Directory
- Hierarchical tree structure

```
● test-38004386:~/test{main}$ pwd  
/home/user/test
```



# Shell tools

\$ cd /

- Change Directory
- '/': root directory  
(in Linux and macOS whereas 'C:\' in Windows)

# Shell tools

\$ cd /

\$ ls

- Show files and directories

```
● test-38004386:/$ ls
  aida bin boot code-oss dev ephemeral etc google home lib lib64 media mnt nix opt proc root run
  sbin srv sys tmp usr var
○ test-38004386:/$
```

\$ cd /usr/include

\$ ls

```
● elicer@58e7b08cbe6b:/usr/include$ ls
aio.h          ctype.h        eti.h          freetype2      iconv.h        libdeflate.h  lzma           netatalk
aliases.h      cursesapp.h    etip.h         fstab.h        ifaddrs.h     libdjbvu      lzma.h         netax25
alloca.h       cursesf.h      evdns.h        fts.h          iftypes.h     libexif       malloc.h       netdb.h
argp.h         curses.h       event2         ftw.h          inttypes.h    libexslt      math.h         neteconet
argz.h         cursesm.h      evhttp.h       gconv.h        jbig85.h      libgen.h      mcheck.h      netinet
ar.h           cursesw.h      evrpc.h        gdbm.h         jbig_ar.h     libintl.h     memory.h      netipx
arpa           cursslk.h      evutil.h       gdk-pixbuf-2.0 jbig.h        libltdl       menu.h         netiucv
asm-generic    cursesw.h      expat.h        glib-2.0       jerror.h      libmount      misc           netpacket
assert.h       db_185.h       expat_external.h glib-2.0       jmorecfg.h    libpng        mit-krb5      netrom
blkid          db.h           exectinfo.h    GL             jpegint.h     libpng16      mntent.h     netrose
bluetooth     dirent.h       fcctl.h        glib-2.0       jpeglib.h     libwmf        monetary.h    nfs
brotli        dlfcn.h        features.h      glob.h          jpegtint.h    libxml2       mqueue.h     nl_types.h
byteswap.h     drm            features-time64.h gmpxx.h        jpeglib.h     libxslt       mtd           nss.h
bzlib.h        elf.h          fincude        gnumake.h      kadm5         limits.h      mysql         obstack.h
c++            endian.h       fenv.h         gnu-versions.h kdb.h         link.h        nc_tparm.h   OpenEXR
cairo          err.h          fmtmsg.h       grp.h           krb5           linux         ncurses_dll.h openjpeg-2.1
com_err.h      errno.h        fontconfig     gssapi         krb5.h        locale.h     ncursesw     openjpeg-2.4
complex.h      error.h        form.h         gssapi         langinfo.h    lqr-1        net           openssl
cpio.h         et             fontconfig     gssapi         lastlog.h     lqr-1        net           panel.h
crypt.h        et             fontconfig     gssapi         lcms2.h       lqr-1        net           paths.h
et             et             fontconfig     gssapi         lcms2_plugin.h ltdl.h        netash
et             et             fontconfig     gssapi         lcms2_plugin.h ltdl.h        netash
```





# Shell tools

## Absolute path

- /home, /home/elicer, ...

## Relative path (. or ..)

- . means current path and .. means parent path
- './' is optional

```
test-38004386:~/test{main}$ cd /
test-38004386:/$ cd /home/user/test
test-38004386:~/test{main}$ cd ..
test-38004386:~$ ls
test
test-38004386:~$ cd ./test
test-38004386:~/test{main}$
```

```
test-38004386:~/test{main}$ cd ../../
test-38004386:/home$
```

# Shell tools

- up/down key to navigate previous commands
- Tab key for auto completing
- Ctrl+R (searching)
- Ctrl+C (interrupt signal)

\$ clear

\$ cd /home/user/te{tab}

\$ history

\$ !{number}

# Shell tools

\$ ls /lib

\$ ls /lib /dev

```
elicer@58e7b08cbe6b:~/code_practice$ ls /lib
apt      compat-ld  file      gnupg    init      mime      os-release  postgresql  python3.10  sftp-server  sysstat
srmerge  x86_64-linux-gnu
bfd-plugins  cpp      gcc      gnupg2   locale   nginx    pkgconfig   python2.7   python3.11  ssl          systemd
algrind
binfmt.d  dpkg      git-core  gold-ld  lsb      openssh   pkg-config.multiarch  python3     sasl2       sysctl.d    tc
ll
elicer@58e7b08cbe6b:~/code_practice$
```

```
elicer@58e7b08cbe6b:~/code_practice$ ls /lib /dev
/dev:
core fd full init mqueue null ptmx pts random shm stderr stdin stdout tty urandom zero

/lib:
apt      compat-ld  file      gnupg    init      mime      os-release  postgresql  python3.10  sftp-server  sysstat  tcl8.6  tcltk  tkConfig.sh  usrmerge  x86_64-linux-gnu
bfd-plugins  cpp      gcc      gnupg2   locale   nginx    pkgconfig   python2.7   python3.11  ssl          systemd  tclConfig.sh  terminfo  tmpfiles.d  valgrind
binfmt.d  dpkg      git-core  gold-ld  lsb      openssh   pkg-config.multiarch  python3     sasl2       sysctl.d    tc        tclloConfig.sh  tk8.6    udev        X11
elicer@58e7b08cbe6b:~/code_practice$
```

\$ cd /mnt/elice; ls -a

```
elicer@58e7b08cbe6b:~/code_practice$ cd /mnt/elice/; ls -a
.  .. .main_disk project
elicer@58e7b08cbe6b:/mnt/elice$
```

# Shell tools

\$ cd /home/elicer/code\_practice

\$ mkdir playground

\$ rm -r playground

- Deleting a file with rm makes the file unrecoverable
- '-r' option is for deleting directories

```
● elicer@58e7b08cbe6b:~/code_practice$ cd /home/elicer/code_practice/
● elicer@58e7b08cbe6b:~/code_practice$ mkdir playground
● elicer@58e7b08cbe6b:~/code_practice$ ls
img  index.html  playground
⊗ elicer@58e7b08cbe6b:~/code_practice$ rm playground/
rm: cannot remove 'playground/': Is a directory
● elicer@58e7b08cbe6b:~/code_practice$ rm -r playground/
● elicer@58e7b08cbe6b:~/code_practice$ ls
img  index.html
○ elicer@58e7b08cbe6b:~/code_practice$
```



# Shell tools

- Brace expansion

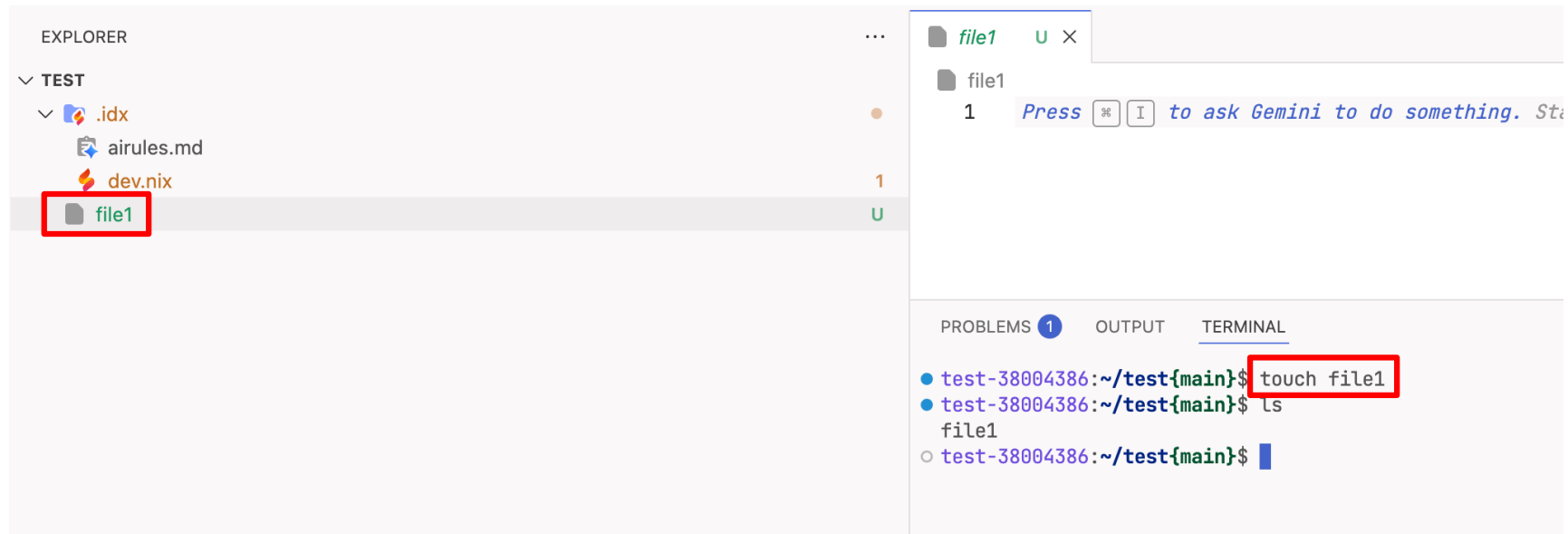
```
$ mkdir dir{1..9}
```

```
$ rm -r *
```

```
● test-38004386:~/test{main}$ mkdir dir{1..9}
● test-38004386:~/test{main}$ ls
  dir1  dir2  dir3  dir4  dir5  dir6  dir7  dir8  dir9  README.md
● test-38004386:~/test{main}$ rm -r *
● test-38004386:~/test{main}$ ls
○ test-38004386:~/test{main}$
```

# Shell tools

\$ touch file1



\$ mv file1 file2

\$ mv file2 ../

```
● test-38004386:~/test{main}$ mv file1 file2
● test-38004386:~/test{main}$ ls
file2
● test-38004386:~/test{main}$ mv file2 ../
● test-38004386:~/test{main}$ ls
● test-38004386:~/test{main}$ cd ..
● test-38004386:~$ ls
file2 test
○ test-38004386:~$
```

# Shell tools

Linux commands options

```
$ ls -a
```

```
$ ls -l
```

```
$ df -h
```

# Shell tools

\$ cp file1 file2

\$ touch foo

\$ rm file\*



# Shell tools

```
$ echo hello
```

- *echo* command simply prints out its arguments

```
$ echo this is a test
```

```
$ echo $((2+2))
```

```
$ echo $((2+2*3))
```

```
$ echo $((($((5**2)) * 3))
```

```
$ echo number_{A..G}
```

```
$ echo a{A{1,2},B{3,4}}b
```

# Shell tools

- 'echo', 'ls' and 'date' are programs
- Shell searches for programs in the list of directories called \$PATH (environment variable)

\$ echo \$PATH

\$ which asdasdf

\$ which echo

\$ /usr/bin/echo \$PATH

\$ which ls

\$ /usr/bin/ls

```
● test-38004386:~/test{main}$ which echo
/usr/bin/echo
● test-38004386:~/test{main}$ /usr/bin/echo $PATH
/nix/store/a5y3kk1gxvh4vv6x377y3lygjxgmbg04-code-oss/
oogle/idx/builtins/bin:node_modules/.bin:/home/user/f
-tools
○ test-38004386:~/test{main}$
```

# Shell tools

- I/O redirection ('>' and '<')
- 'ls' program outputs the result of the program to the stdout (standard output)

```
$ ls -l /usr/bin > ls-output.txt
```

```
$ less ls-output.txt
```

```
$ > file3
```

```
$ ls /usr/bin >> ls-output.txt
```

(=append)

```
$ echo hello > hello.txt
```

```
$ cat hello.txt
```

```
$ cat < hello.txt > hello2.txt
```

```
$ cat hello2.txt
```

# Space in file name

```
$ mkdir te\ st
```

```
$ mkdir “te st”
```

# Shell tools summary

cd, mkdir, mv, rm, cp, cat, echo, touch, etc

# Outline

- **Linux**
- **Shell tools**
  - cd, ls, pwd, mkdir, cp, rm, mv, ...
- **Vim**

# Vim

- Text editor
- Originated from Vi editor(1976)
- Vim is light and fast
- Vim can be used anywhere, and it is especially useful when a graphical environment is not supported

# Vim



**I Am Devloper**

@iamdevloper

Following



I've been using Vim for about 2 years now,  
mostly because I can't figure out how to exit it.

RETWEETS

14,083

LIKES

8,154



3:26 PM - 17 Feb 2014



314



14K



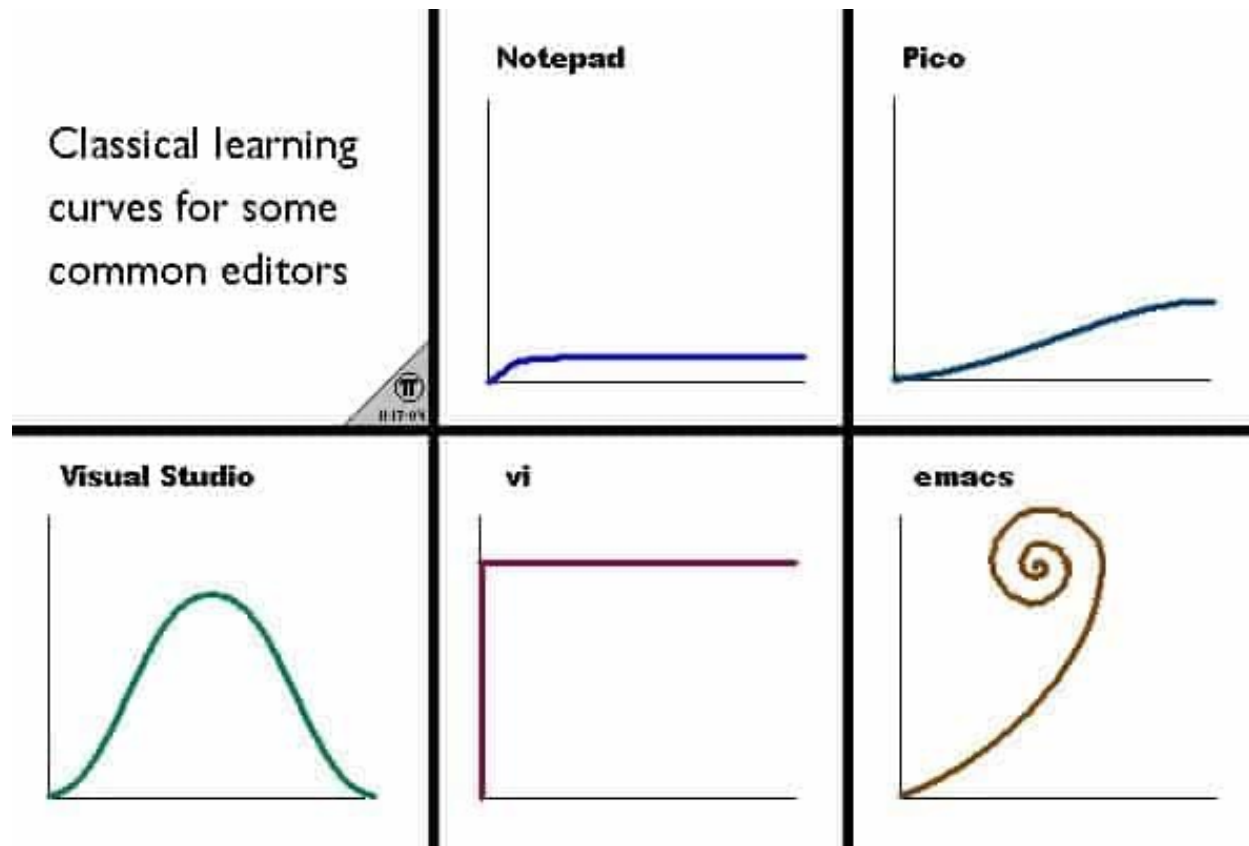
8.2K



Machine  
Intelligence &  
Data science LAB



# Vim



# Vim

```
$ ls -l /usr/bin > foo.txt
```

```
$ vi foo.txt
```

```
:q ← quit
```

```
:q! ← force quit (do not save the file)
```

---

```
$ vi foo.txt
```

- Press 'i' and enter insert mode, then write some words
- Next, press 'esc'
- :wq ← save and quit

# Vim

- Vim is **modal** editor!
- Multiple operating modes
  - Normal
  - Insert
  - Replace
  - Visual
  - Command line

# Vim

- Insert mode  $\xrightleftharpoons[\text{<i>}]{\text{<ESC>}}$  normal mode
- Command mode  $\xrightleftharpoons[\text{<:>}]{\text{<ESC>}}$  normal mode
  - :q = quit
  - :w = write (save)
  - :wq = save and quit
- Replace mode  $\xrightleftharpoons[\text{<R>}]{\text{<ESC>}}$  normal mode
- v: visual mode / V: visual line / Ctrl-v: visual block

# Vim

- Normal mode

- Movement: hjkl

- Words

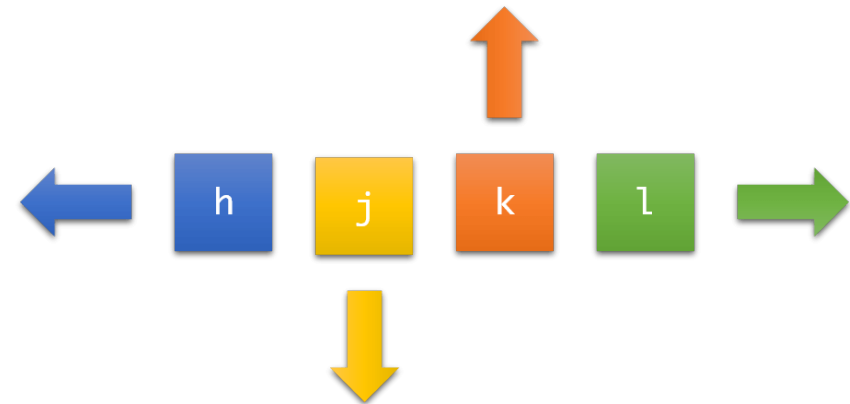
- w: next word
    - b: beginning of word
    - e: end of word

- Lines

- 0: beginning of line
    - \$: end of line

- File

- gg: beginning of file
    - G: end of file



# Vim

- Normal mode
  - f{character}: find {character} on the current line
    - , and ; for navigating matches
  - :%s/root/ROOT/g : change all 'root' to 'ROOT' in files
    - in command mode
    - 'g' stand for 'global'

# Vim

- Normal mode

- Edits

- d{motion}
      - dw: delete word
      - d\$: delete to end of line
      - d0: delete to beginning of line
    - c{motion}
      - cw: change word
      - (ex) ci( change contents inside of parentheses
    - Visual mode + manipulation
      - d: delete
      - c: change
      - (ex) visual block mode and delete the block

# Vim

- Normal mode
  - Edits
    - y: copy
    - p: paste
      - (ex) yy: copy current line
      - (ex) in visual mode, copy and paste
    - u: undo
    - ctrl-r: redo



# Vim

- Normal mode
  - Counts
    - 3w: move 3 words forward
    - 5j: move 5 lines down
    - 2dw: delete 2 words

# Assignment

## I. C++ Development Environment Setup

(For window user)

- Install WSL2
- Install VS code (<https://code.visualstudio.com/>)
- Integrating WSL2 with VS Code

(For mac user)

- Install VS code (<https://code.visualstudio.com/>)

# Assignment

## 2. Compile and execute c++ program

- Write any simple c++ program (\*.cpp)
- Compile using g++ (c++17 or higher)
  - Use linux commands (ex. g++ -std=c++17 main.cpp -o main)
- Run the executable to verify it works

3. Submit a PDF containing minimal screenshots showing the successful completion of each step.

4. Also submit Code of Ethics (수강생 윤리강령) with signing

~ 09.08 14:30 elice 과제 제출함

Format: 2025-xxxxx\_name.pdf