ITP20004 – Open-Source Software Labs

Computer Organization & vim

Charmgil Hong

charmgil@handong.edu

Spring, 2023

Handong Global University

Announcements

- Weekly schedule

Week	Mon		Week	Thur		
1	Course overview, motivation, administrivia		1	CPR: C Programming Reinforcement - Functions		
2	Computer organization and Linux environment (1)		2	CPR: C Programming Reinforcement - Strings		
3	Computer organization and Linux environment (2)		3	CPR: C Programming Reinforcement - User-defined types, and memory allocation		
4	Basic Linux commands + Writing code on Linux (vim)		4	Getting started with Linux / Hands-on Linux command-line tools		
5	More Linux commands		5	CPR: C Programming Reinforcement - Understanding compilation and build process		
6	Project management (1)	**Proj 1 출제**	6	Project management (2)		
7	-		7	Project: BASIC interpreter (2 periods)		Project 1
8	Midterm exam		8	Proj 1 due		
9	CPR: C Programming Reinforcement - Accessing files and (		9	Debugging with GDB + Unit testing with gtest		
10	Code review GNU utilities		12	Writing an application in C		
11	Computer network basics		10	Linux network commands	**AWS 가입 - lightsail**	
12	Linux machine as a server + Web services		11	Service launching	**lab problem + AWS 가입해지**	
13	Project: Text-based Game		13	Github and open-source community		Project 2
14	Using Github		14	Socket programming		
15	Project: Multi-user game		15	Project: Multi-user game		Project 3
16	Final exam		16			

Announcements

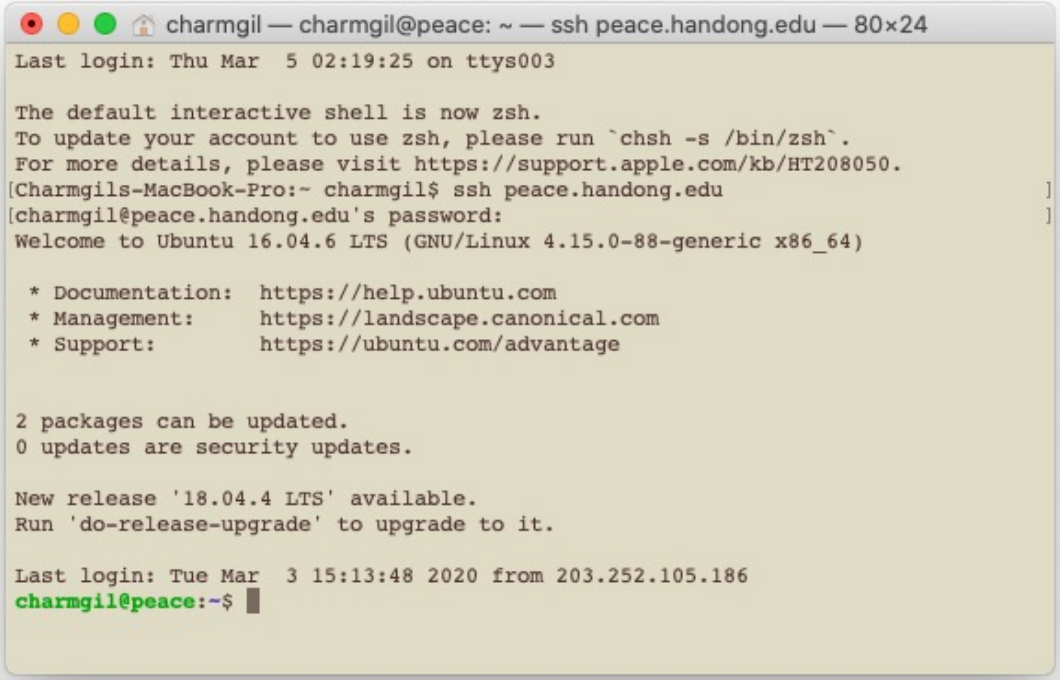
- There will be two re-shuffles of the teams
  - On Weeks 6 and 12
  - There will be a peer evaluation at the end of each cycle

Announcements

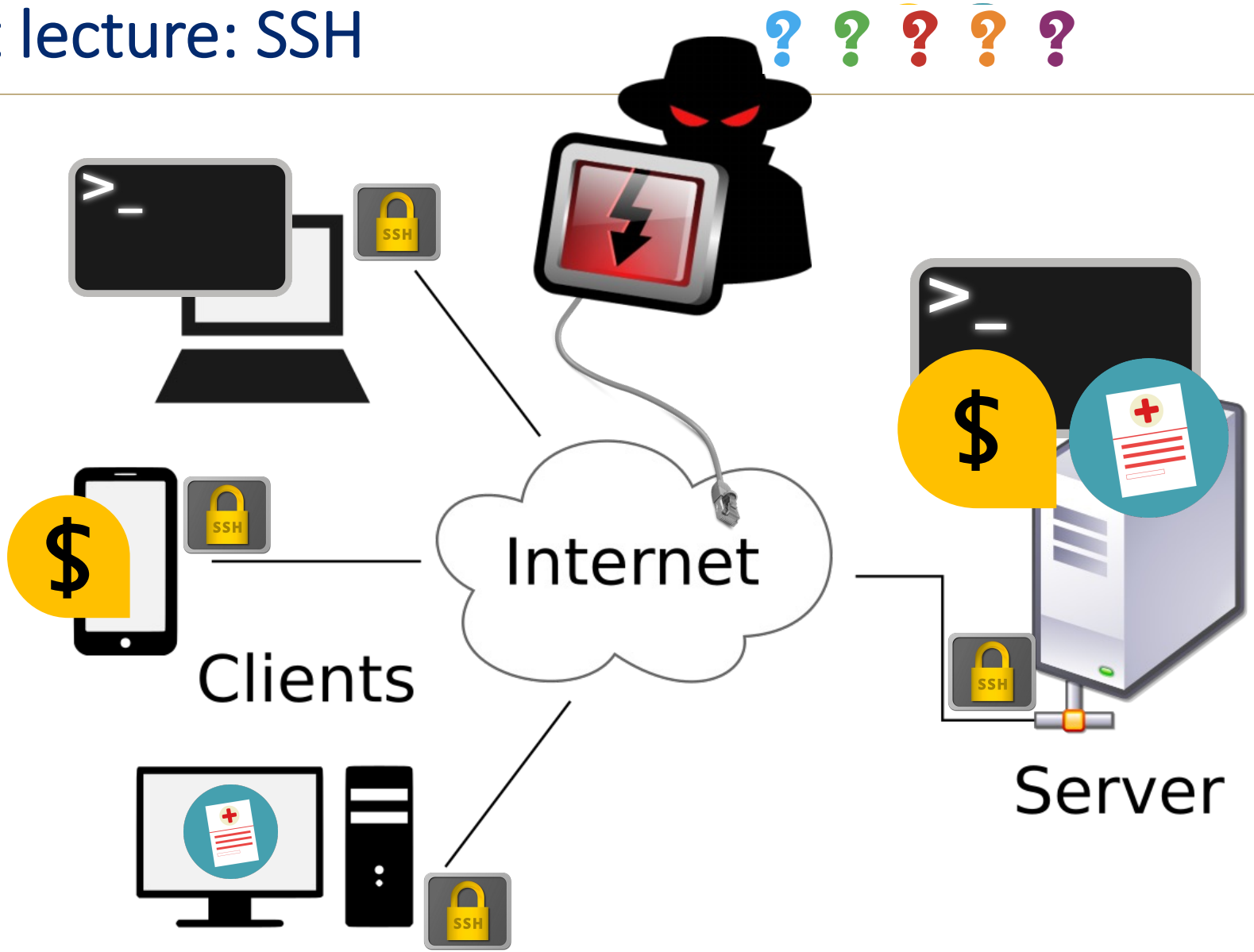
- For each lab
  - Before a lab, every student submits a pre-lab report (worksheet-type assignment) – **individual work**
  - After a lab, each team sees and reports to the TA with the results – **team work**

Last lecture: Connecting to peace.handong.edu

- You are in!
  - A prompt (`YOUR_ID@peace:~$`) will be displayed on the terminal
  - When you want to close, enter exit
    - Pressing *Control+d* does the same (sending an EOF character)



Last lecture: SSH



Last lecture: First Keystrokes

- Task 1 – Handshaking with Linux
  - `ls, cat, more, less, top`
- Task 2 – Working with Files and Directories
  - `cd/mkdir/rmdir, cp, mv, rm`

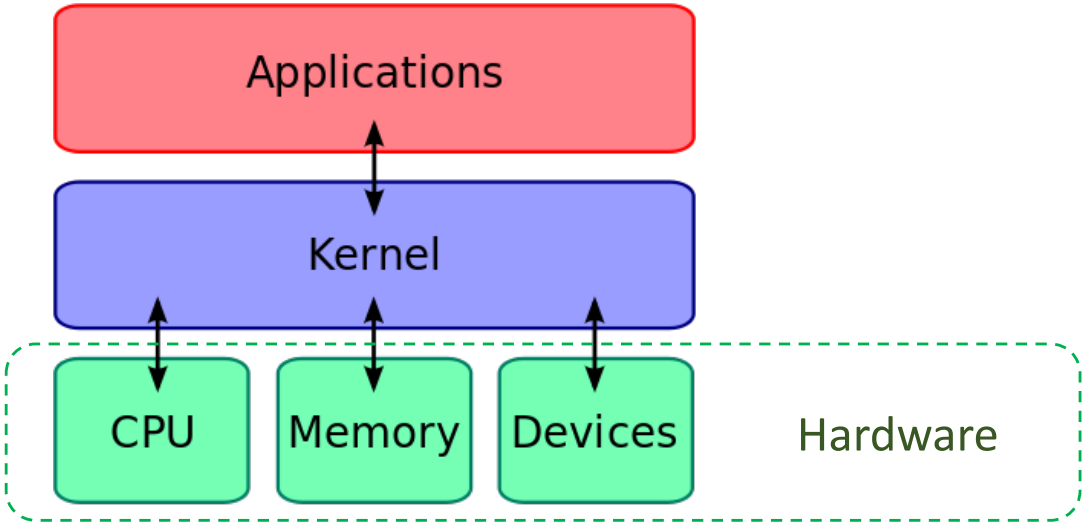
Agenda

- Computer hardware organization
- Text editing with *vim*

Computer System

- A computer system consists of hardware and system software that work together to run application programs

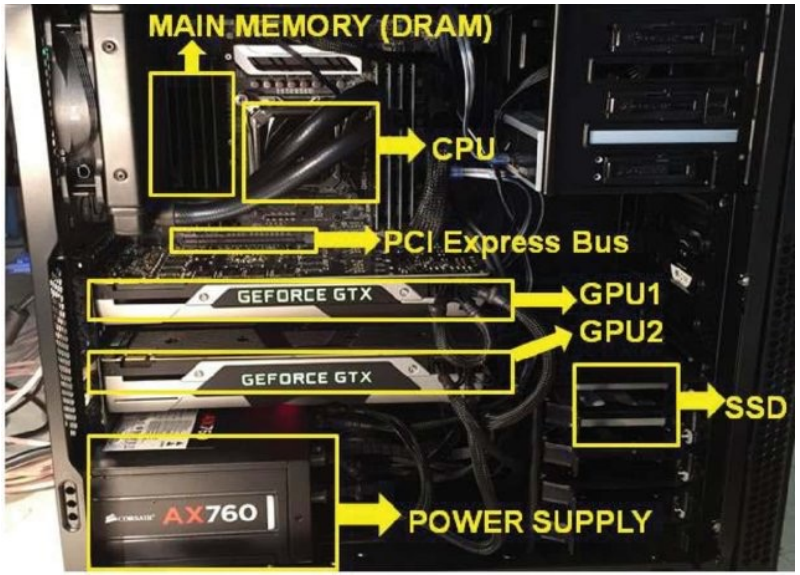
- Layers of logical abstraction



* Image src: https://en.wikipedia.org/wiki/Kernel_(operating_system)

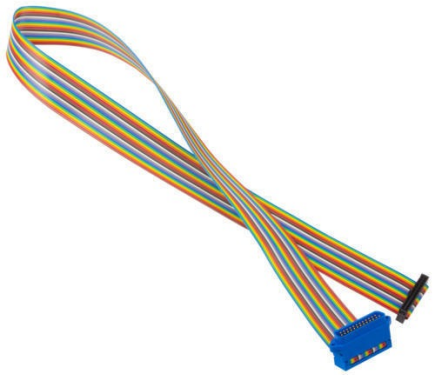
Hardware Organization

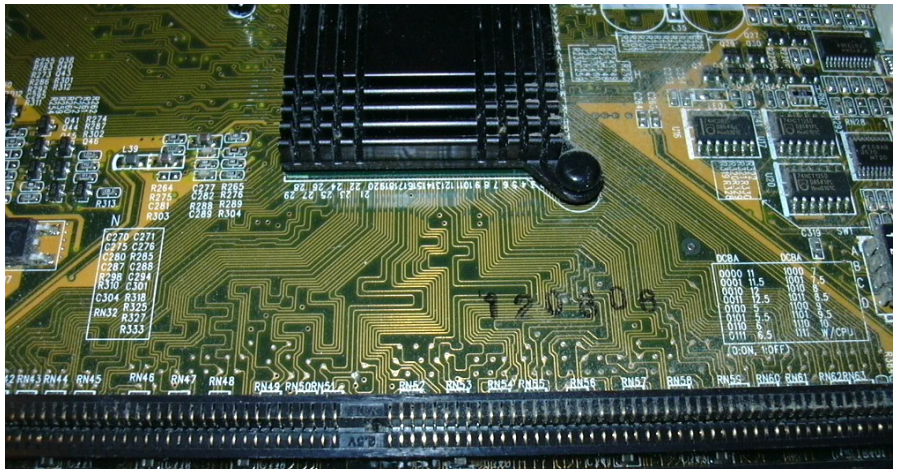
- Main components: CPU (central processing unit), Memory, Input/Output Devices, GPU (graphics processing unit)
  - Communicate over buses
    - **Bus**: A communication system that transfers data between components



Hardware Organization

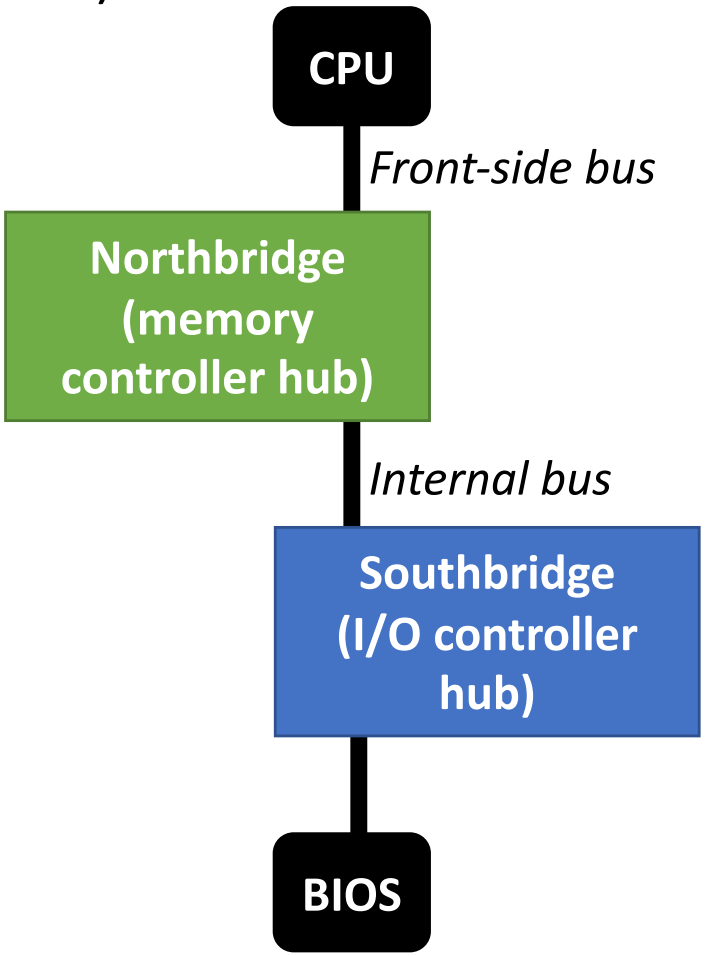
- Main components: CPU (central processing unit), Memory, Input/Output Devices, GPU (graphics processing unit)
  - Communicate over buses
    - **Bus**: A communication system that transfers data between components





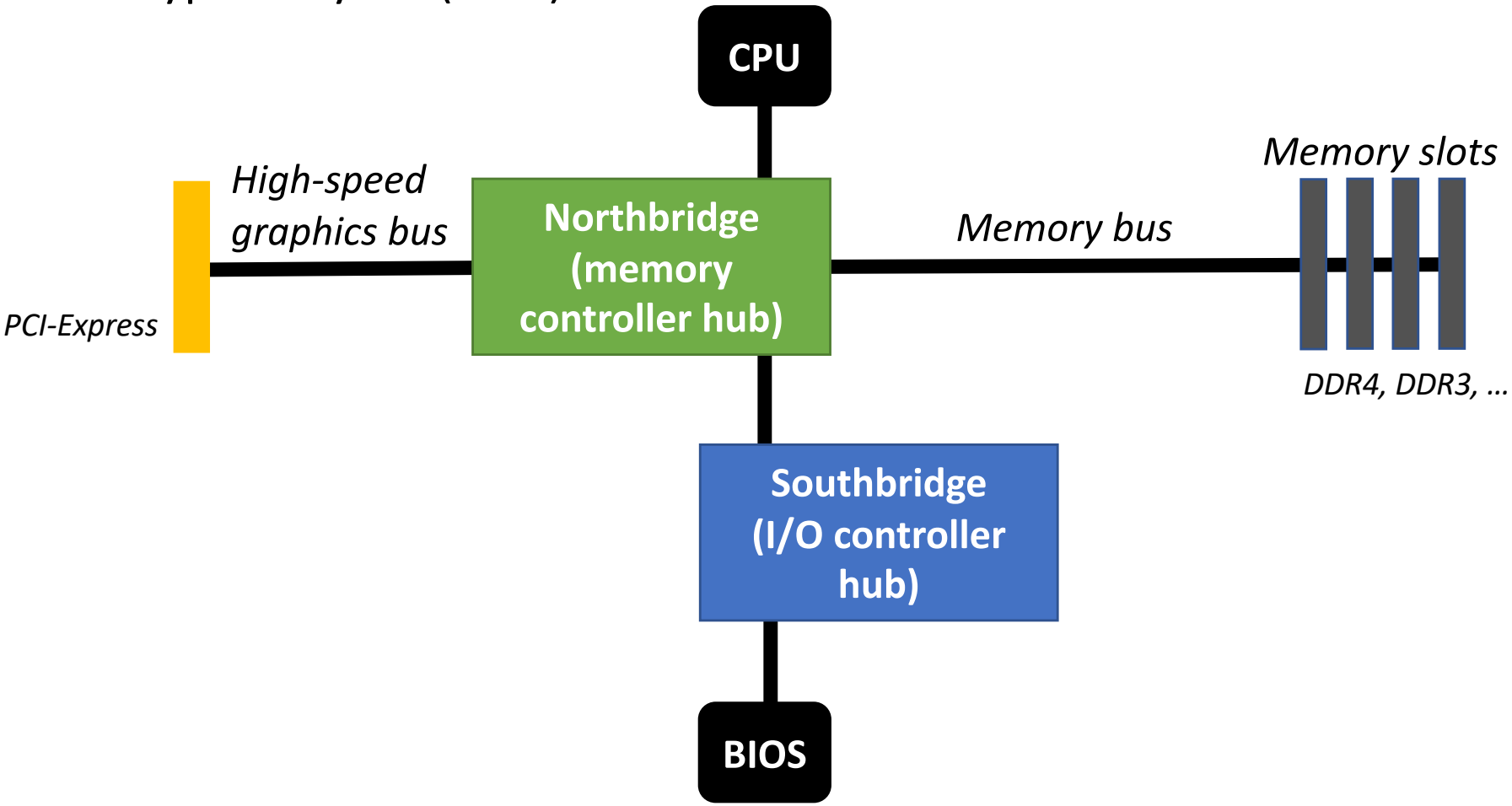
Hardware Organization

- A typical layout (Intel)



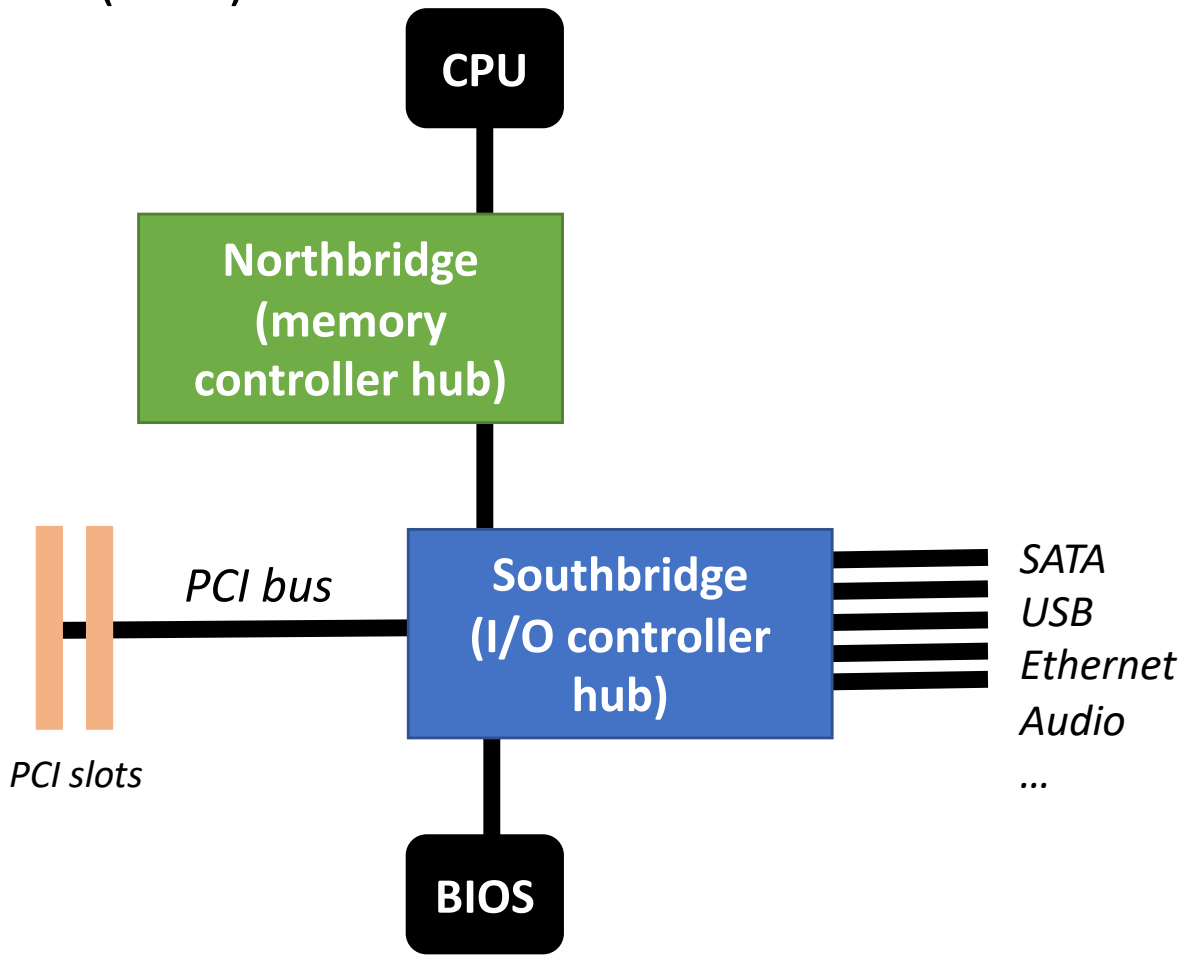
Hardware Organization

- A typical layout (Intel)



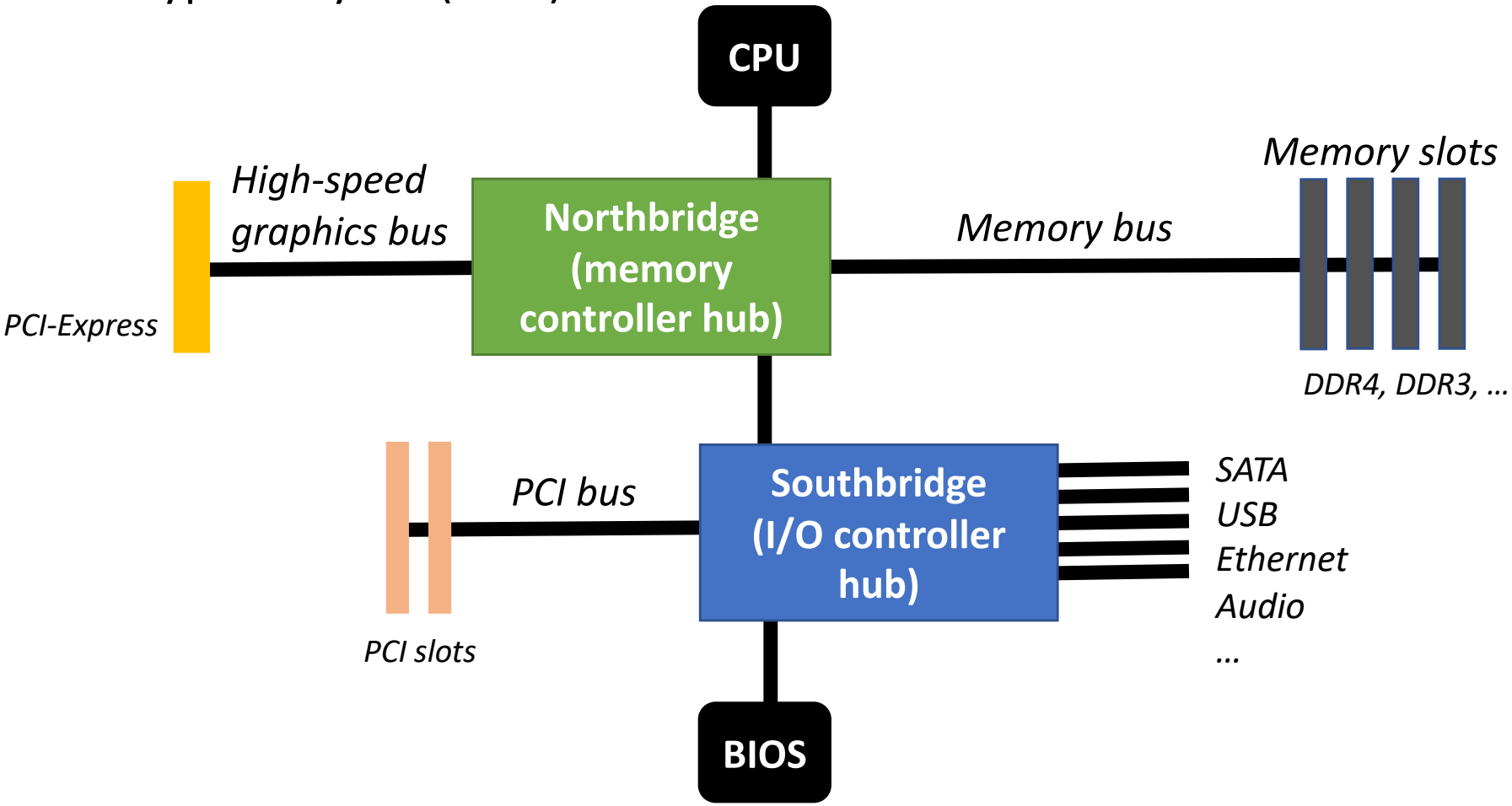
Hardware Organization

- A typical layout (Intel)



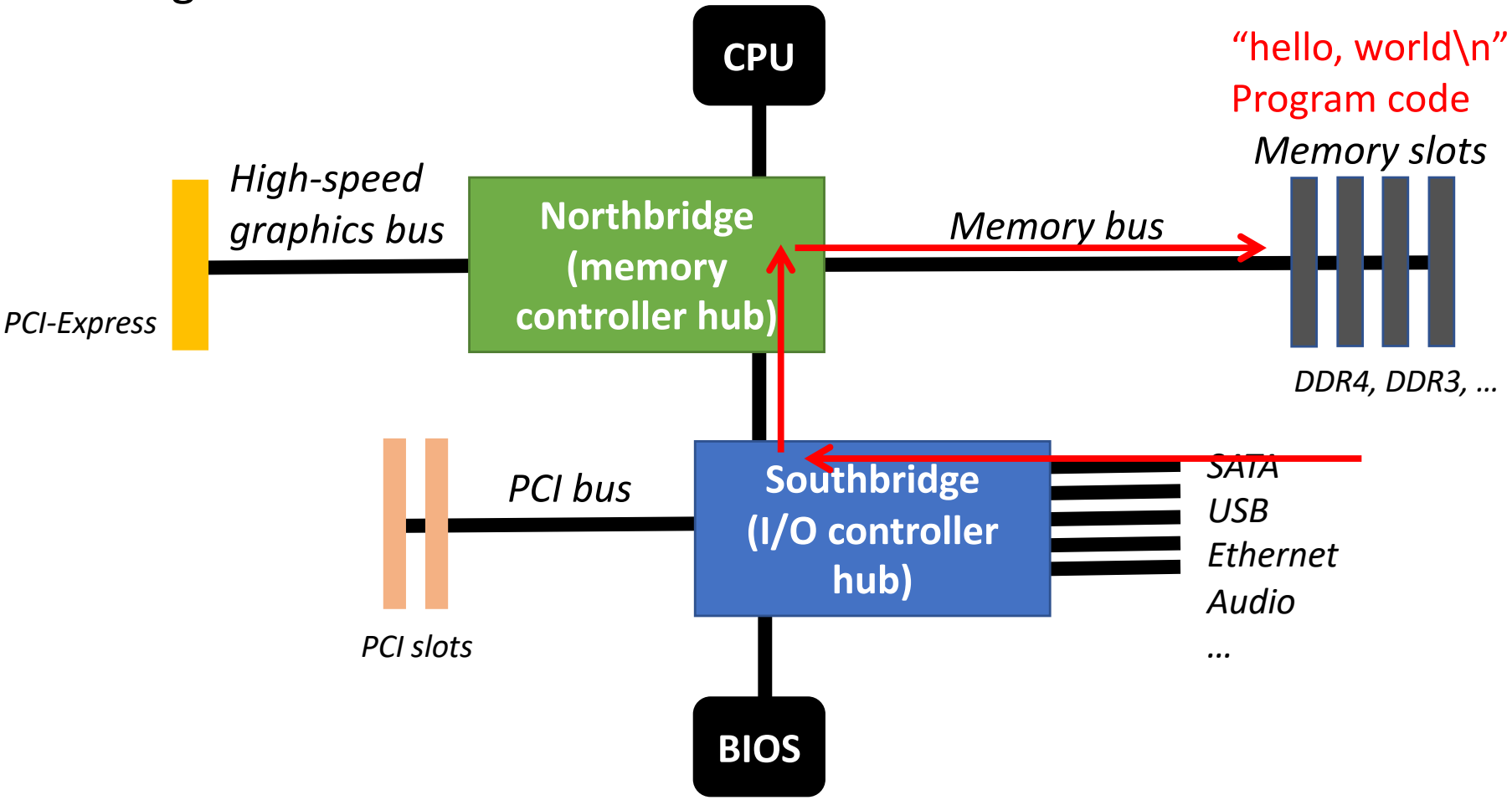
Hardware Organization

- A typical layout (Intel)



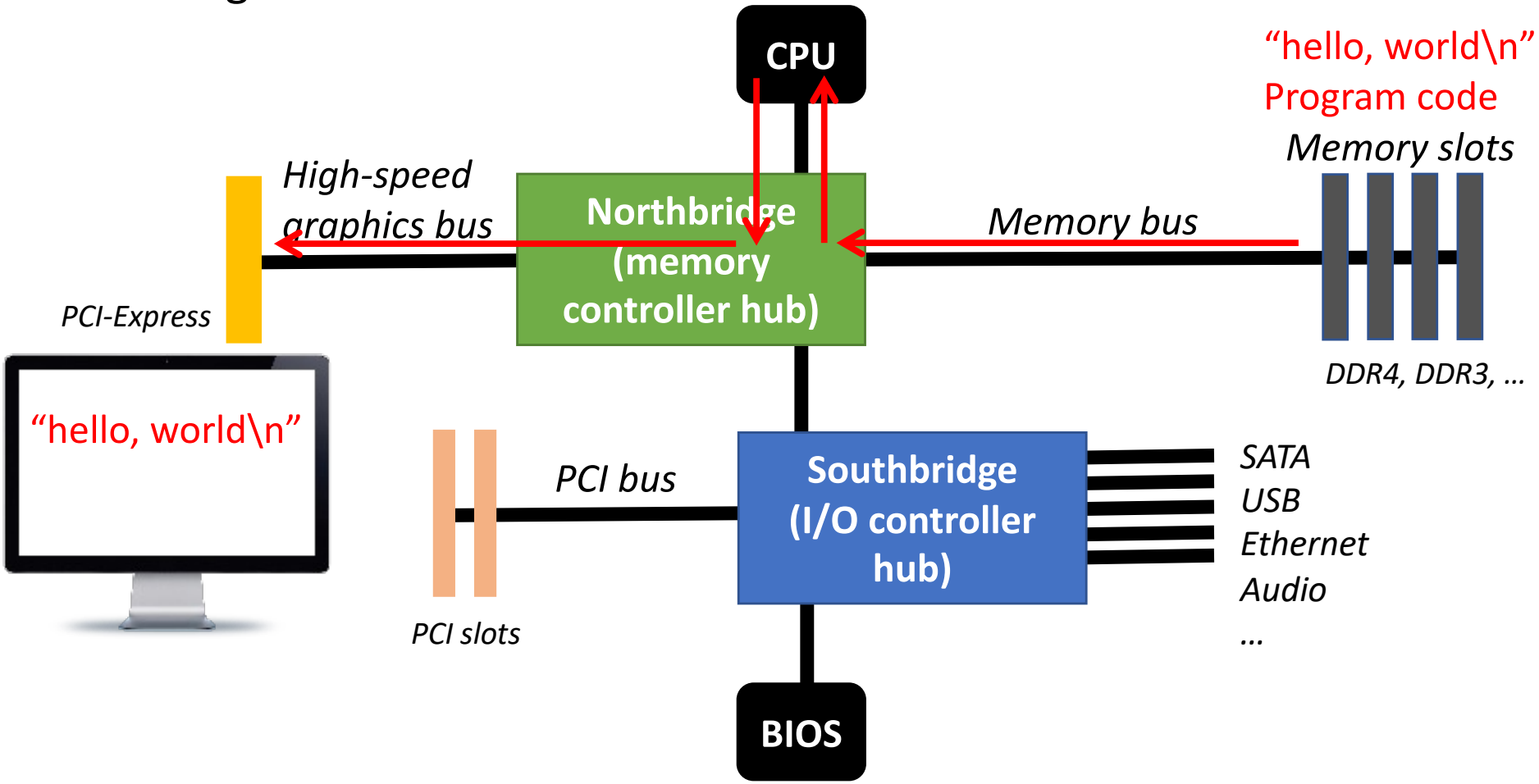
Hardware Organization

- Program execution



Hardware Organization

- Program execution



Comparisons

S.NO	North Bridge	South Bridge
1.	North bridge is located towards the north of motherboard.	South bridge is located towards South of PCI bus of the motherboard.
2.	North bridge is directly connected to the CPU.	South bridge is connected via North bridge to the CPU.
3.	It manages the communications between the CPU and other parts of the motherboard.	It manages the Input and Output functions.
4.	The North bridge is placed near to processor for easy access.	The South bridge is placed near PCI.
5.	North bridge communicates faster.	South bridge communicates slower.
6.	Other names for North bridge include Memory Controller Hub, host bridge.	IO Controller Hub is the other name for South bridge.
7.	The North bridge is hub for memory control.	The South bridge is a hub for input and output functioning.
8.	The North bridge connects the buses that work faster like the AGP bus.	The South bridge connects the buses that work slower like ISA.
9.	North bridge looks bigger.	South bridge looks smaller.
10.	North bridge connects components like RAM, AGP.	South bridge connects components like PCI, USB.

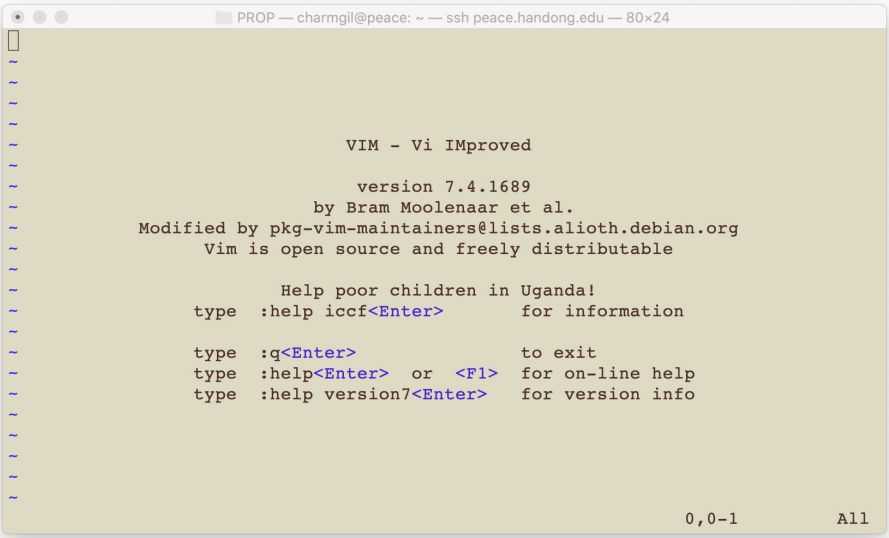
* Source: https://www.geeksforgeeks.org/difference-between-north-bridge-and-south-bridge/

Agenda

- Computer hardware organization
- **Text editing with *vim***

vim

- Vi Improved, a programmer's text editor
  - "vi" is a text editor from the early days of Unix
    - As the name suggests, "vim" adds a lot of functionalities to the original vi interface
  - Virtually every Linux machine has vim – *wherever you go, you can edit files in the same environment!*



vim - History

- Earlier name: **V**i **Im**itation
  - Developed 1988, released 1991 by *Bram Moolenaar* on Amiga
  - 1992 Ported to Unix (version 1.22)
  - 1996 Started to support GUI (graphical user interface)
  - URL: https://www.vim.org/

* Image src: https://usesthis.com/interviews/bram.moolenaar/; https://www.vim.org/; https://collection.maas.museum/object/424348

vi

- Vi by Bill Joy (William Nelson Joy)

The Traditional Vi

Source Code for Modern Unix Systems

The *vi* editor is one of the most common text editors on Unix. It was developed starting around 1976 by Bill Joy at UCB, who was tired of the *ed* editor. But since he used *ed* as a code base, access to the original sources has required a commercial Unix Source Code License for more than twenty years. In January 2002, Caldera was so kind to remove usage restrictions to the Ancient Unix Code by a BSD-style license (see the announcement at Slashdot) and thus *vi* is now finally free.

Compared to most of its many clones, the traditional *vi* is a rather small program (the binary size is approximately 160 kBytes on i386) just with its extremely powerful editing interface, but lacking fancy features like multiple undo, multiple screens, or syntax highlighting.

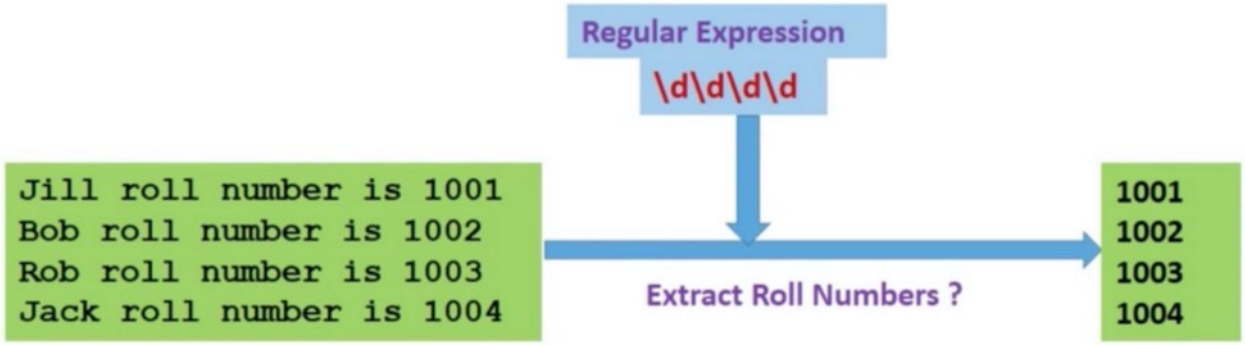
This port of *vi* has generally preserved the original style, terminal control, and feature set. It adds support for international character sets, including multibyte encodings such as UTF-8, and some minor enhancements that were not present in BSD *vi* 3.7, but had been included in later *vi* versions for System V or in POSIX.2.

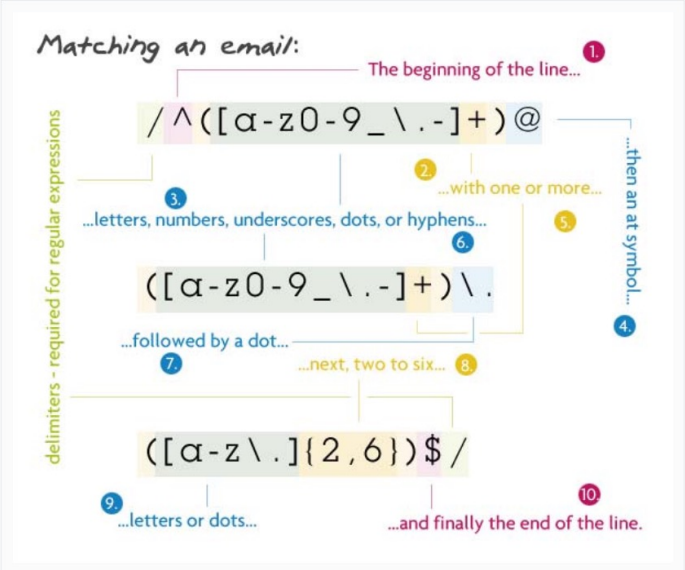
vim

- Key features
  - Very low memory footprint
  - Syntax highlighting

vim

- Key features
  - Very low memory footprint
  - Syntax highlighting
  - Find using regular expression
    - Regular expression: string defines a text matching pattern





* Image src: https://brentmarquez.com/in-a-nutshell-series/regular-expressions-in-a-nutshell/

vim

- Key features
  - Very low memory footprint
  - Syntax highlighting
  - Find using regular expression
    - Regular expression: string defines a text matching pattern
  - Highly configurable; many plug-ins are available

* Image src: https://medium.com/@huntie/10-essential-vim-plugins-for-2018-39957190b7a9

vim

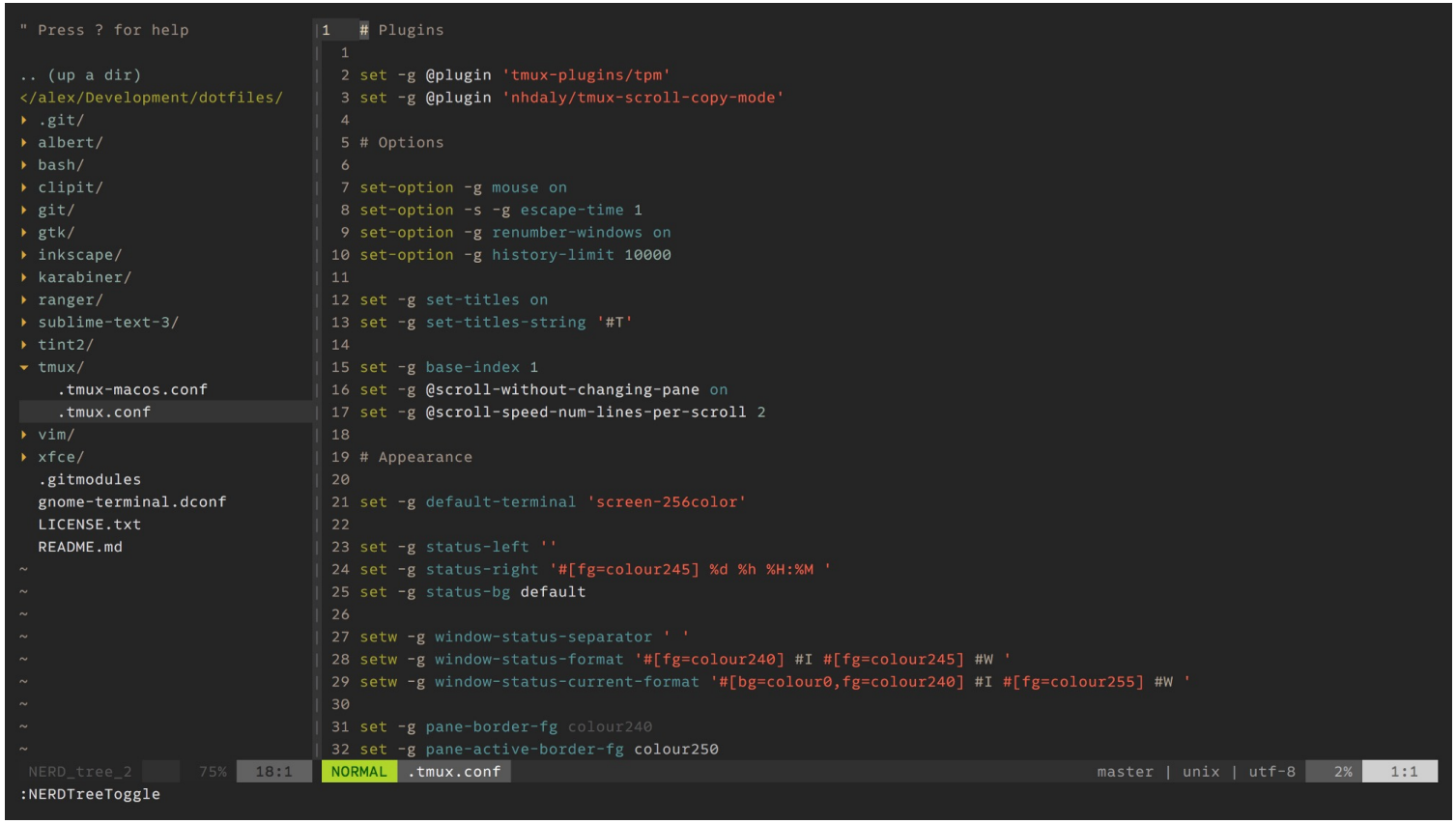
- Key features
  - Very low memory footprint
  - Syntax highlighting
  - Find using regular expression
    - Regular expression: string defines a text matching pattern
  - Highly configurable; many plug-ins are available

```
import os
import re
import subprocess
import sys

def get_setting(key, default=None):
    settings = sublime.load_settings('tmux.sublime-settings')
    os_specific_settings = {}

    if sys.platform == 'darwin':
        os_specific_settings = sublime.load_settings('tmux (OSX).sublime-settings')
    else:
        os_specific_settings = sublime.load_settings('tmux (Linux).sublime-settings')

    return os_specific_settings.get(key, settings.get(key, default))

class TmuxCommand():
    def resolve_file_path(self):
```

* Image src: https://medium.com/@huntie/10-essential-vim-plugins-for-2018-39957190b7a9

vim

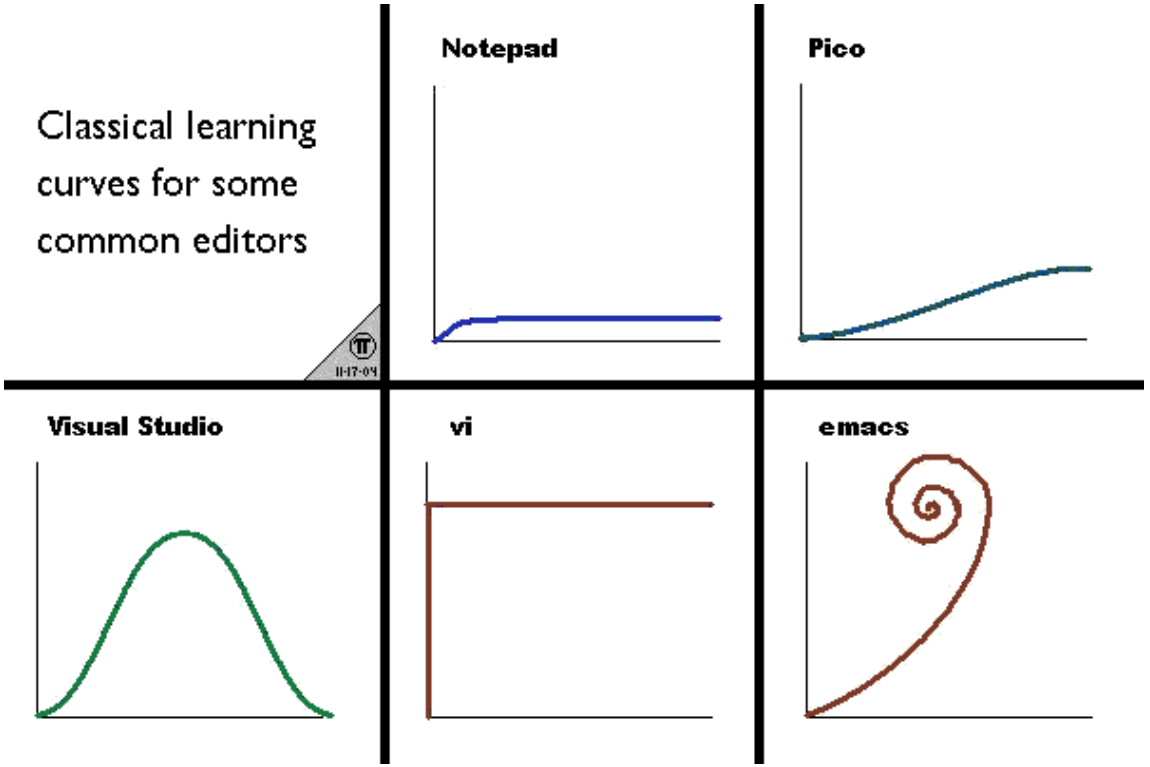
- Key features
  - Highly configurable; many plug-ins are available



* Image src: https://medium.com/@huntie/10-essential-vim-plugins-for-2018-39957190b7a9

vim – Modal Editor

- vim is complicated



* Image src: https://www.reddit.com/r/ProgrammerHumor/comments/9d3j49/text_editor_learning_curves/

vim – Modal Editor

- Two mode?
  - Command mode
  - Insert mode

vim – Modal Editor

- Four mode (*there were two more!*)
  - Command mode
  - Command line mode
  - Insert mode
  - Visual mode

vim – Modal Editor

- Four mode
  - Command mode: All keystrokes are interpreted as commands
  - Command line mode: Providing a command prompt
  - Insert mode: Regular typing as you expect from an editor
  - Visual mode: Highlighting

vim – Modal Editor

- Command mode (normal model)
  - Default when vim is opened
  - Keystrokes are commands
  - From other modes, use ‘esc’ to switch to normal mode

vim – Modal Editor

- Command line mode
  - From normal mode, press ‘:’ to trigger command line mode
  - Similar to the ”File” menu on common text editors
  - Used for opening, closing files; finding and replacing text; *etc.*

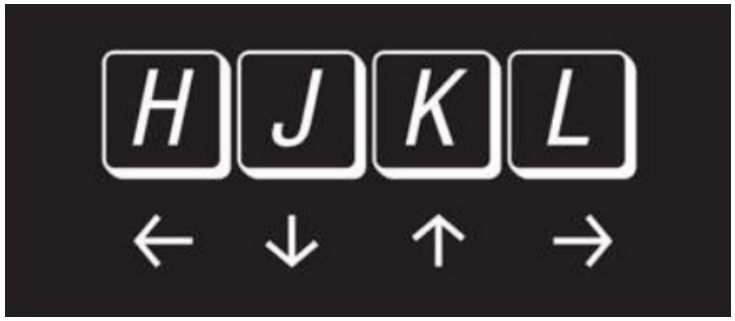
vim – Modal Editor

- Insert mode
  - More normal for modern editor users
  - Keystrokes insert text
  - Commands are possible with key combinations
  - From command mode, use ‘i’ to switch to insert mode

vim – Modal Editor

- Visual mode
  - For highlighting/text selection
  - Keystrokes are commands
  - From command mode, use ‘v’ to switch to visual mode

vim – Basic Operations

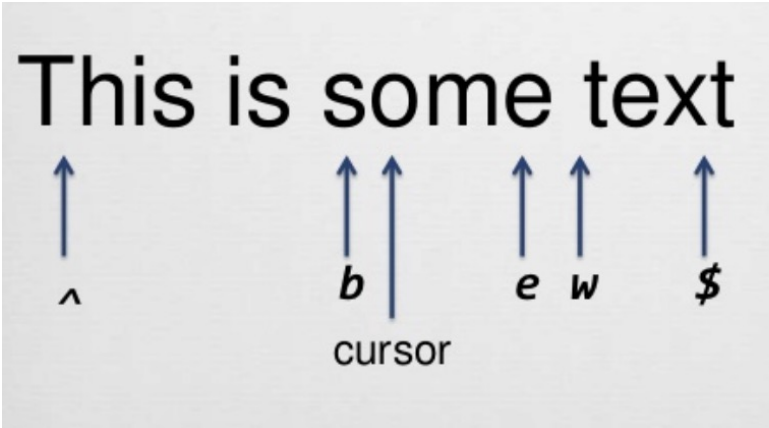


- H, J, K, L Maneuvering the cursor (the arrow keys work too)
- :wq Save and quit
- :q! Quit without save

vim – Basic Operations

- Shortcuts for cursor movement
  - ^ to the beginning of the line
  - $ to the end of the line

  - b beginning of the current word
  - e end of the current word
  - w to the next word



vim – Basic Operations

- Switching to insert mode
  - i Insert at the current position
  - a Insert at the next position
  - o Insert at the next line
  - A Insert at the end of the current line

  - R Replace

- Undo
  - u Undo

vim – Basic Operations

- Deletions (cut)
  - x Delete a character
  - *n*x Delete *n* characters

  - dd Delete a line
  - d*n* Delete *n* lines

  - dw Delete a word
  - d*n*w Delete *n* words

vim – Basic Operations

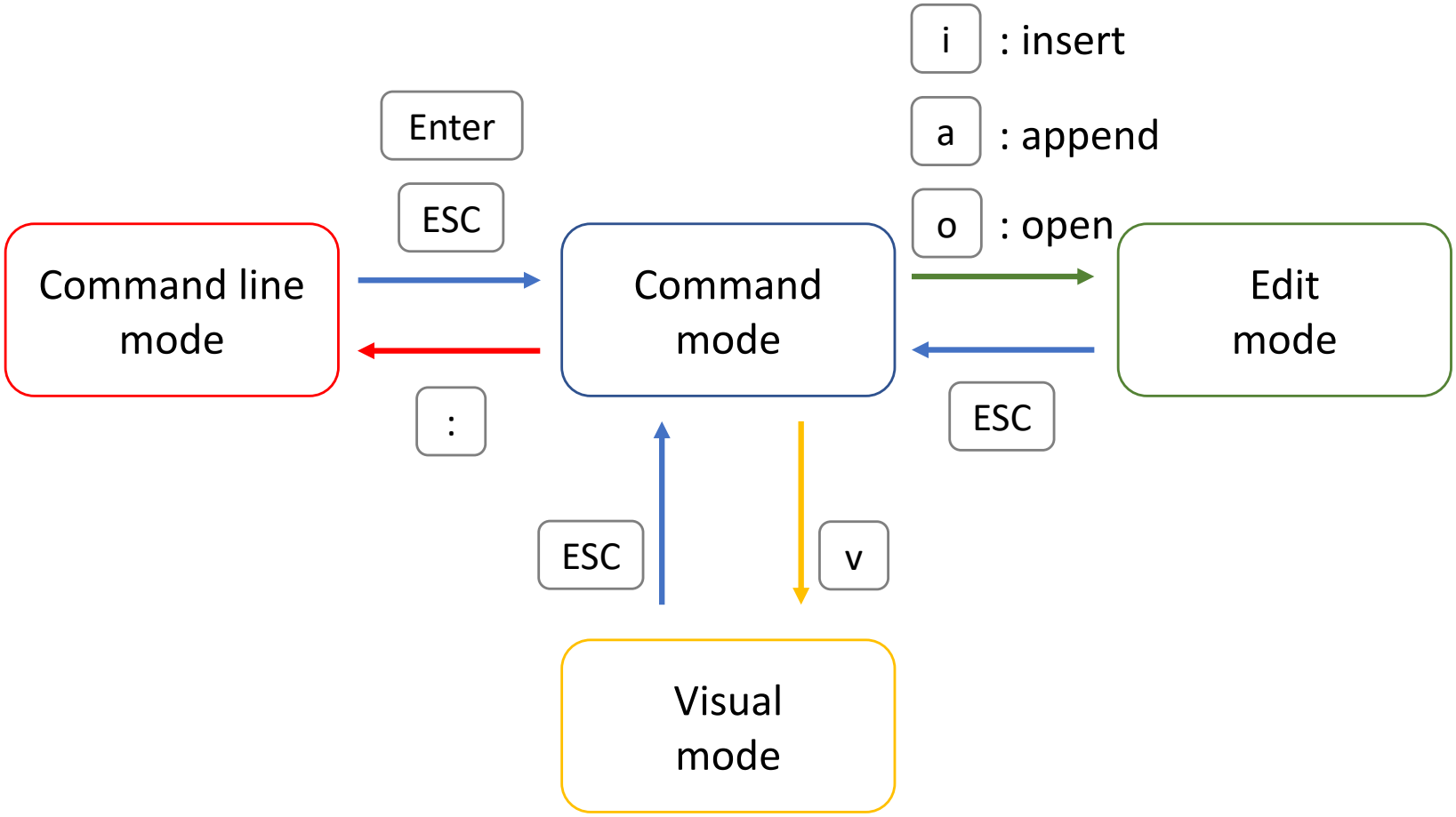
- Copy and Paste
  - Copy (yank)
    - yy Copy a line
    - y*n* Copy *n* lines
    - yw Copy a word
    - y*n*w Copy *n* words

    - p Paste

    - v Select text (visual mode)

vim

- Switching between modes



vim – Find and Replace

- / Searches in the document for a string (can take regular expression)
- n Goes to next occurrence of the search string
- N Goes to previous occurrence of the search string
- :%s/foo/bar Replaces all occurrences of ‘foo’ with ‘bar’

vim – Links

- You can download vim on your Windows too
  - https://www.vim.org/
- vim plug-ins
  - https://vimawesome.com/
- vim tutorials/manuals
  - https://www.csie.ntu.edu.tw/~piaip/vim/vimbook-OPL.pdf

- References:
  - https://www.slideshare.net/BenMcCormick/vim-survival-guide-71763917
  - https://www.slideshare.net/brandonliu/introduction-to-vim