# Introduction to Operating System

Tran Giang Son, tran-giang.son@usth.edu.vn

ICT Department, USTH

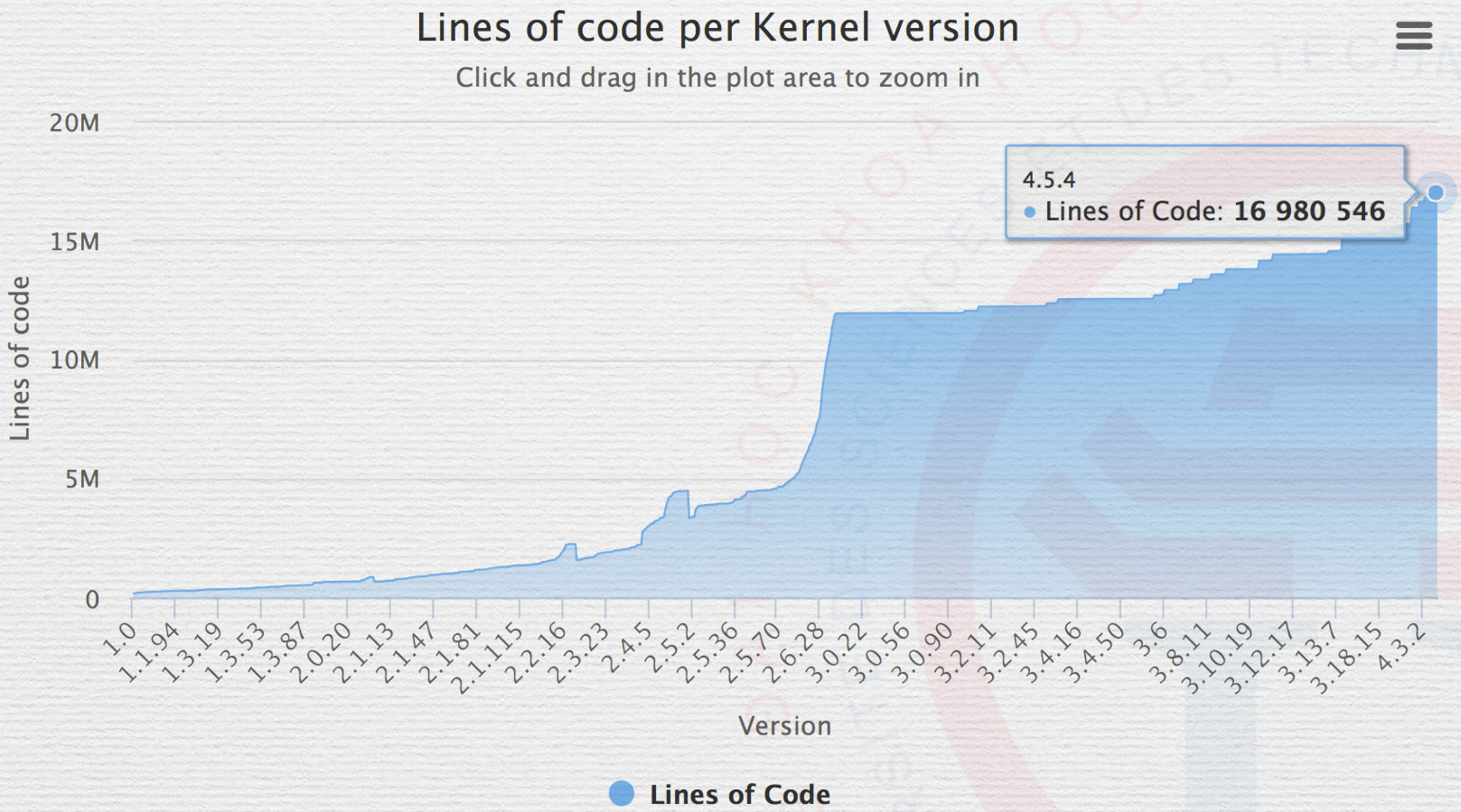
# Contents

# Definition

# What?

- An intermediary between a user of a computer and the computer hardware
- Allocate resources
  - CPU time,
  - Memory space
  - File storage space
  - I/O device
- Provides a set of system services to users

# History




Linux source code line count[1]

[1] Source: LinuxCounter.net

# History: Unix Philosophy (1994)

- Write programs that do one thing and do it well.
- Write programs to work together.
- Write programs to handle text streams, because that is a universal interface.

# History: The UNIX Programming Environment

*Even though the UNIX system introduces a number of innovative programs and techniques, no single program or idea makes it work well.* ***Instead, what makes it effective is the approach to programming****, a philosophy of using the computer. [...] at its heart is the idea that* ***the power of a system comes more from the relationships among programs*** *than from the programs themselves. Many UNIX programs do quite trivial things in isolation, but, combined with other programs, become general and useful tools.*

– Brian Kernighan and Rob Pike

# Practical Work 0: git/github

- Create a new directory in your home, name it «sa2020»
- Fork the course's git repository to your github account
  - `https://github.com/SonTG/sa2020.git`
- Clone your forked repository to your newly created directory
  - `git@github.com:<YourAccount>/sa2020.git`
- Edit «README.md», write your name as instructed.
- Make a new commit with a message "First student commit"
- Push your new commit to your forked github repository

# Advantages

# Advantages

- Separation
- Intermediary
- Efficiency
- Reuse

# Separation

- HW: physical device. provide electrical mechanisms. expensive.
- SW: provide functionality
- Need something in between
  - No OS → No communication between HW & SW
  - No OS → Single purpose

# Intermediary

- Applications do not need to support different HWs
  - HW compatiblity is responsibility of OS
- HW don't need to take care of how (well) it is used
  - How app uses HW is responsibility of OS
- More abstraction
- Like a translator

# Efficiency

- Hardware is best when maximize utilization
  - Processor: how to use processor effectively?
  - Memory: how to share memory between apps without conflict?
  - Storage: how to organize data in an efficient manner?
  - Network: how to facilitate network access to apps?

# Efficiency

- Hardware is best when maximize utilization
  - Processor: how to use processor effectively?
  - Memory: how to share memory between apps without conflict?
  - Storage: how to organize data in an efficient manner?
  - Network: how to facilitate network access to apps?

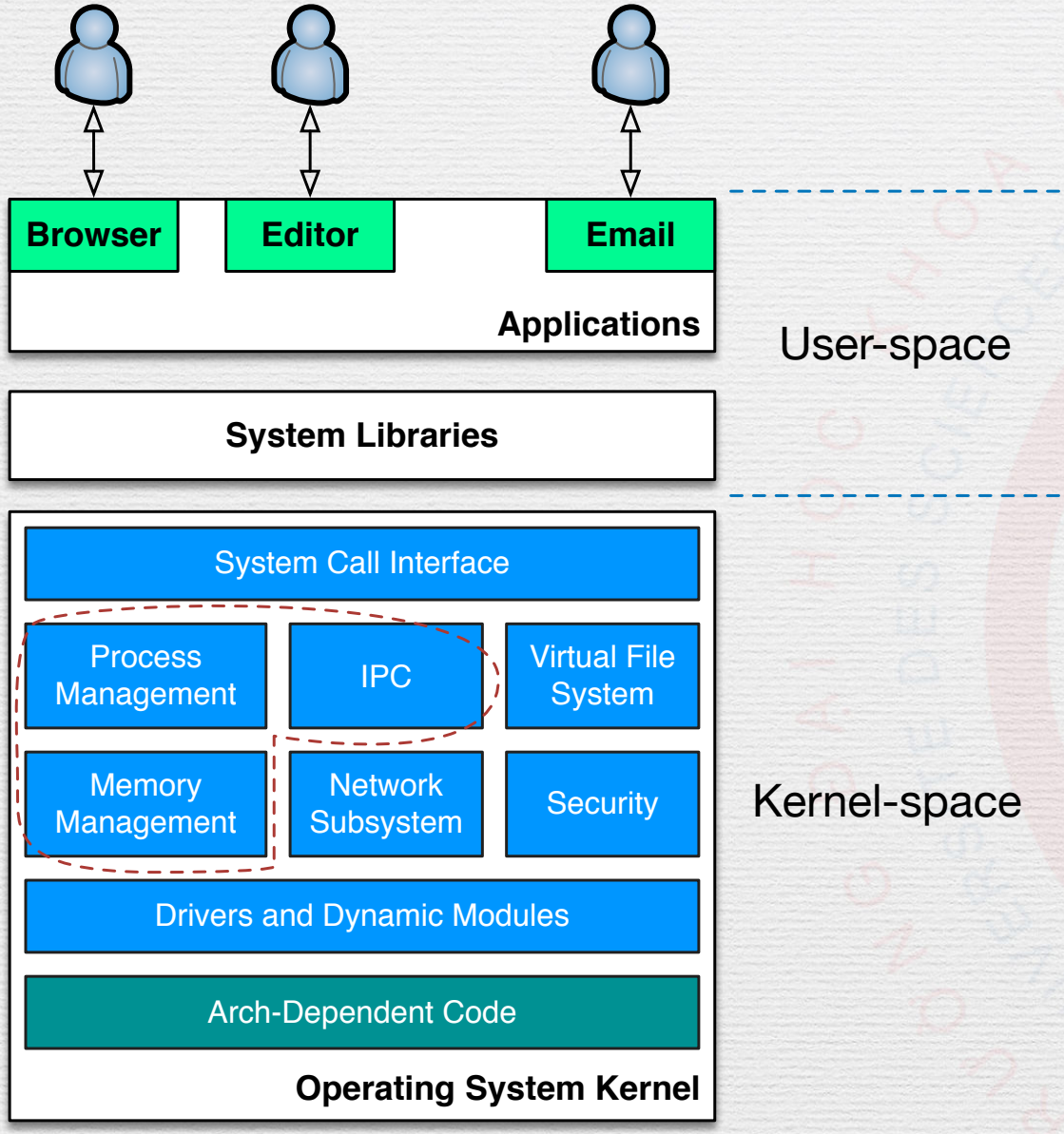
→ Resource management

# Reuse Common Services

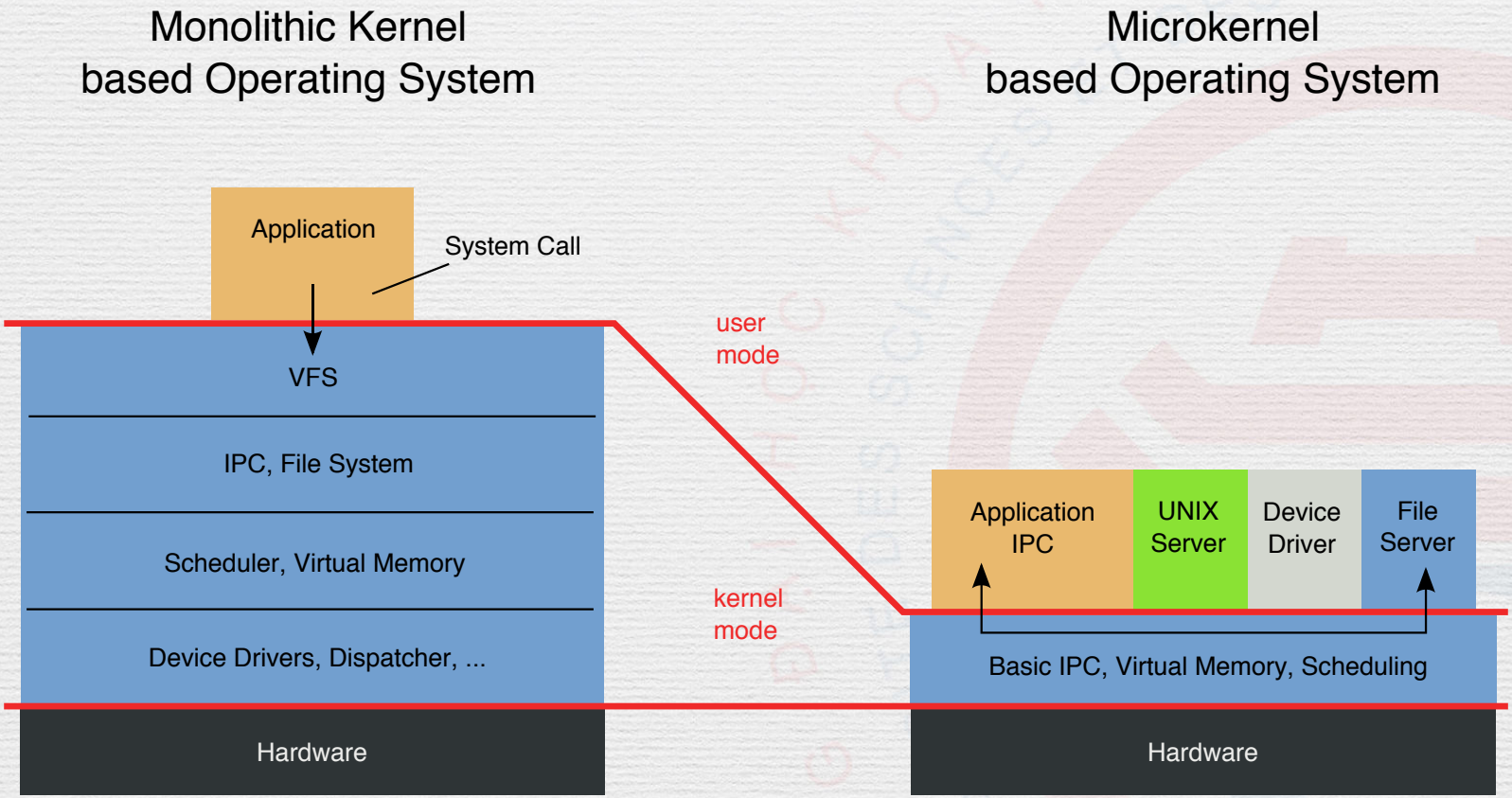
- System calls
- Avoid people reinventing the wheel
  - Standard organization of data (file systems)
  - Common mechanisms to interact with input/output devices
    - Input from keyboard
    - Output to console
    - Network devices (sockets)
  - Common user interface components
    - Windows
    - Icons
    - Menus

# Components

# Parts


Browser
Editor
Email
Applications
User-space
System Libraries
System Call Interface
Process Management
IPC
Virtual File System
Memory Management
Network Subsystem
Security
Kernel-space
Drivers and Dynamic Modules
Arch-Dependent Code
Operating System Kernel

# Monolithic vs Micro Kernels




Source: wikipedia

# Monolithic vs Micro Kernels

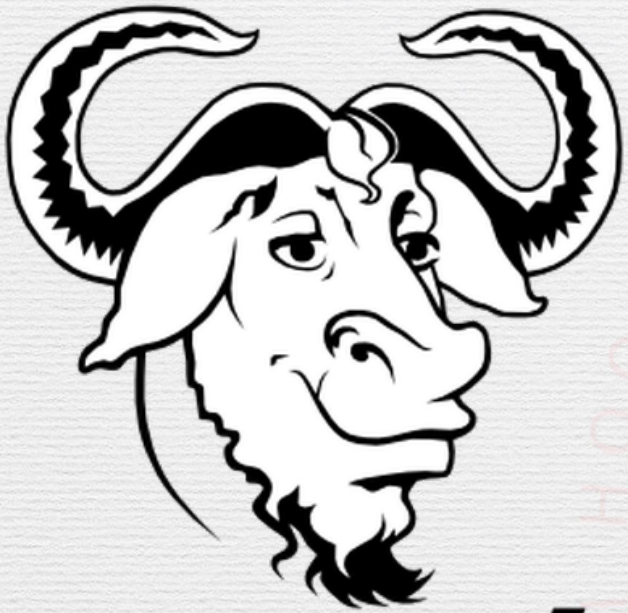
- Monolithic Kernel
  - “All in one"
  - All kernel components work in kernel space
  - Single System Call interface to interact with user space programs
- Micro Kernels
  - Minimalism
  - Only critical components work in kernel space
  - Most components work in user space
- Hybrid Kernels
  - Only some kernel components are moved to user space

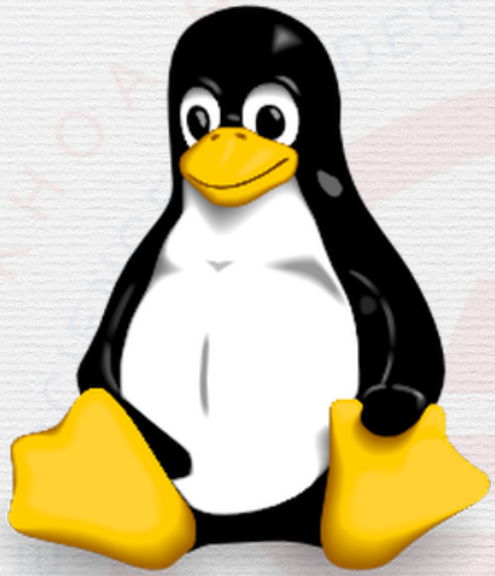
# Monolithic vs Micro Kernels

- Monolithic Kernels
  - Windows (Microsoft)
  - Linux (Linus)
  - Unix (Bell Labs, BSD)
  - DOS
- Micro Kernels
  - MINIX (only 12k LoC, Tanenbaum)
  - GNU Hurd (GNU Project)
  - Mach (Carnegie Mellon University)
- Hybrid Kernels
  - XNU (**X** is **N**ot **U**nix)
    - Combination of BSD Unix and Mach

# GNU/Linux

# What?





GNU/Linux

# What?

- Linux
  - The OS kernel
  - Linus Torvalds
  - 1991
  - GPL

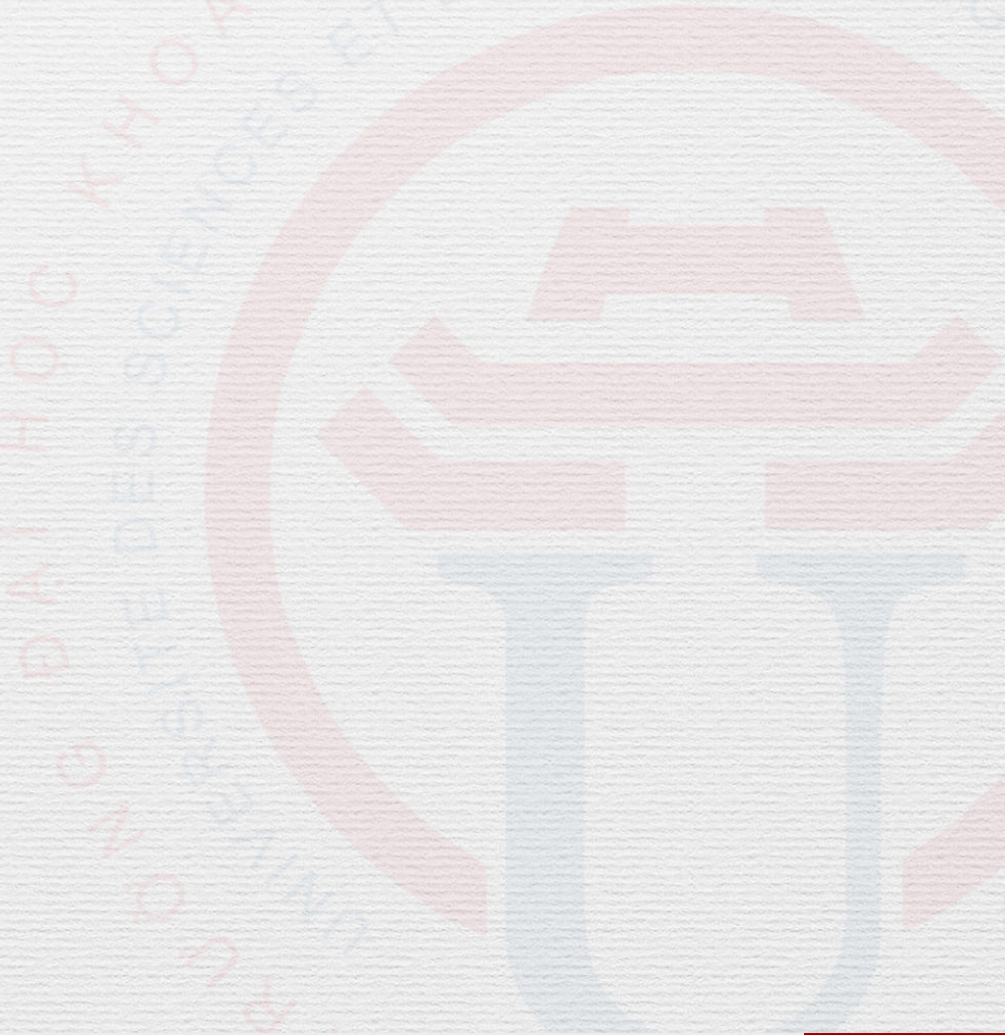
# What?

- Linux
  - The OS kernel
  - Linus Torvalds
  - 1991
  - GPL
- GNU
  - **G**NU is **N**ot **U**nix
  - Richard Stallman
  - 1983
  - User-space libraries and utilities
  - Hurd: kernel
  - Copyleft (mostly GPL)

# What?

- Very popular
- Different distros
  - Ubuntu/Debian
  - Fedora/CentOS/RHEL
  - ArchLinux
  - Gentoo
  - ...

# Why?



# Why?

- Free and open source
- Much fewer malware
- Minimalism: install and use what you need
- Compatibility: latest machines as well as 10-year-old ones
- Ease of use
  - GUI: for normal end users
  - Terminal: for geeks, sysad, netad...
- Ease of maintenance
  - Update whole system with one command
  - Update apps as well
- Customization

# How?

- Choose and install one distro
  - Replace your Windows
  - Dual boot with Windows on another partition
  - Install «inside» a directory in Windows (**WUBI**)
  - Live USB/CD
  - Use virtual machines

# How: Alternative GUI applications

| Windows/macOS | Linux |
|---|---|
| Safari/Microsoft Edge | FireFox/Chrome/Konqueror |
| Microsoft Office | LibreOffce/OpenOffice |
| Photoshop | GIMP |
| Lightroom | Darktable |
| Illustrator | Inkscape |
| Outlook | Thunderbird/Evolution |
| Notepad/Notepad++ | gedit/geany/kate |
| Other simple Linux apps | Wine |

# How: Cross platform applications

Cross platform == supported by major OS
(Windows/macOS/Linux)

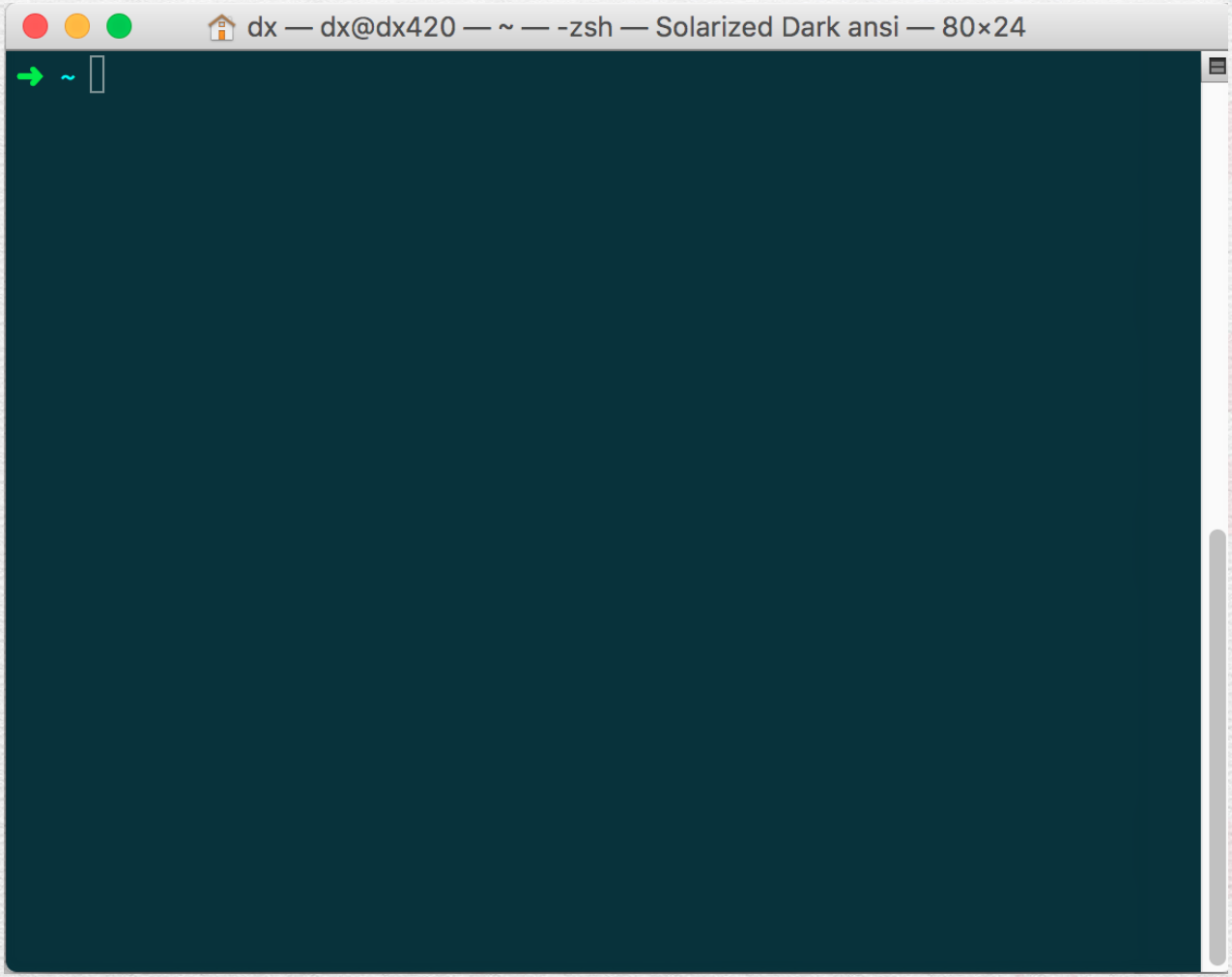
# How: Cross platform applications

Cross platform == supported by major OS
(Windows/macOS/Linux)

- Browsers
  - Chrome, Firefox, Opera
- Media players
  - VLC, Kodi, Plex
- Productivity
  - OpenOffice / LibreOffice
  - Dropbox, Google Drive
- Networking tools
  - uTorrent
  - Wireshark
- Development
  - Intellij IDEA
  - Eclipse
  - Android Studio
  - cmake
  - git/svn/...
- Utility
  - Sublime Text
- Game
  - Counter Strike
  - Dota 2

# How?

- Console / Terminal in this course



# How: Basic Commands

- Windows: big, dedicated programs
- GNU coreutils: Unix Philosophy
  - A set of small, simple tools
  - Basic building blocks of scripts
  - Use in combination

# How: Basic Commands[2]

| Commands | Description |
|---|---|
| `man [command]` | Display user **man**ual for the specified command. |
| `cd /directorypath` | **C**hange **d**irectory. |
| `ls [opts]` | **L**i**s**t directory contents. |
| `cat [files]` | Display file's contents after cont**cat**enation. |
| `mkdir [opts] dir` | **M**a**k**e a new **dir**ectory. |
| `cp [opts] src dest` | **C**o**p**y files and directories. |
| `mv [opts] src dest` | Rename or **m**o**v**e file(s) or directories. |
| `rm [opts] dir` | **R**e**m**ove files and/or directories. |
| `chmod [opts] mode file` | **Ch**ange a file's **mod**es (permissions). |
| `chown [opts] file` | **Ch**ange **own**er of a file. |
| `df [opts]` | Display **d**isk's **f**ree and used space. |
| `du [opts]` | Show **d**isk **u**sage that each file takes up. |
| `find [pathname] [expr]` | **Find** for files matching a provided pattern. |
| `grep [opts] pattern [file]` | Search files or output for a particular pattern. |
| `nano [file]` | **N**ano's **ano**ther editor |

[2] Source: Linux For Dummies

# How: Basic Commands[3]

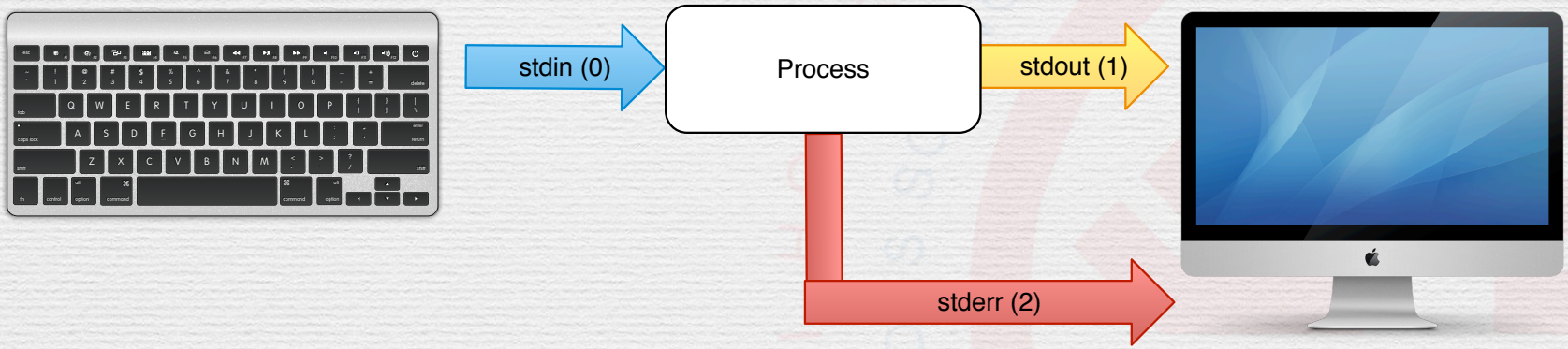
| Commands | Description |
|---|---|
| `kill [opts] pid` | **Kill** a process. |
| `less [opts] [file]` | View the contents of a file one page at a time. |
| `ln [opts] src [dest]` | Create a shortcut. (**li**n**k**s) |
| `passwd` | Change your **passwor**d |
| `ps [opts]` | List **p**rocess **s**tatus. |
| `pwd` | **P**rint **w**orking **d**irectory |
| `ssh [opts] user@host [cmd]` | Remotely log in to another machine with **s**ecured **sh**ell |
| `su [opts] [user]` | **S**witch to another **u**ser account. |
| `head [opts] [file]` | Display the first n **head**ing lines of a file. |
| `tail [opts] [file]` | Display the last n **tail**ing lines of a file. |
| `tar [opts] file` | Store/Extract (and compress/decompress) **t**ape **ar**chives |
| `top` | Displays resources being used on your system. |
| `touch file` | Create an empty file with the specified name. |
| `who [opts]` | Display **who** is logged on. |
| `wget url` | Non-interactive network downloader |

[3] Source: Linux For Dummies

# Command Pipe and I/O Redirection

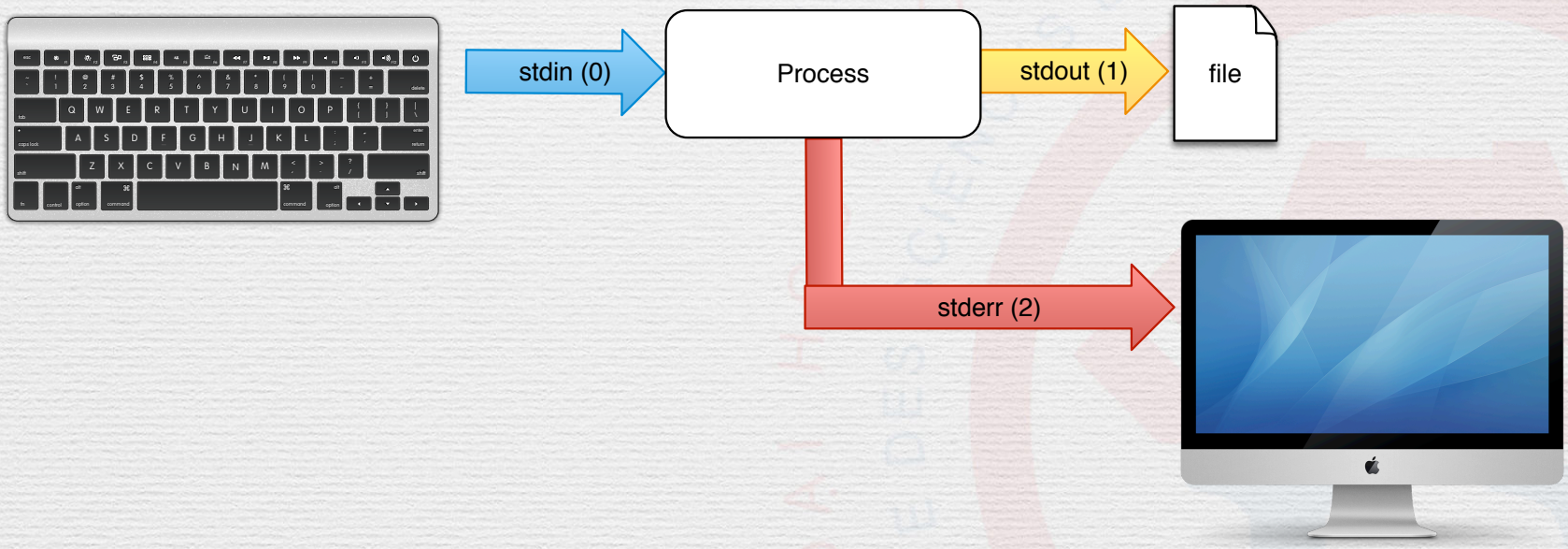
- **The** method for combining command line tools
- Stream in Java? Remind.
- Everything in Linux is file
- Each process has 3 streams
  - Output (`System.out.println();`)
  - Input (`new Scanner(System.in);`)
  - Error (`System.err.println();`)

# Command Pipe and I/O Redirection




Default: input from keyboard and output to terminal

# Command Pipe and I/O Redirection



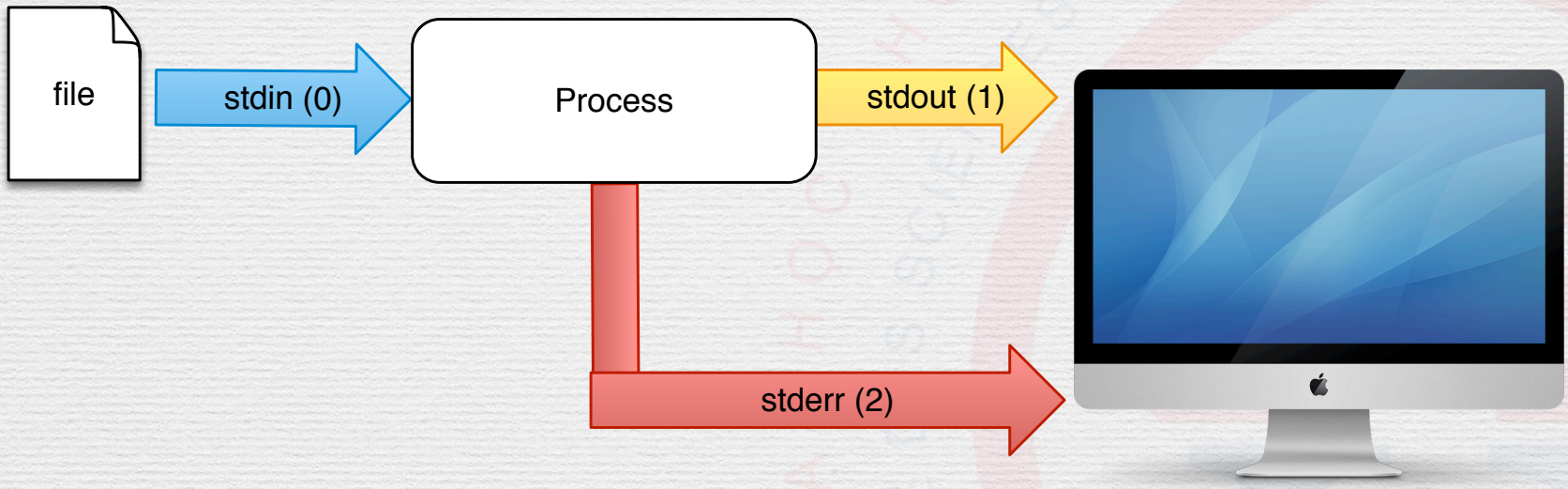

Input from keyboard and output to file

# Command Pipe and I/O Redirection

- Redirect output to file: use `> filename`
- Everything written to stdout (`printf`) will be written to file. Terminal will not have output.

```
$ ls /tmp > filelist
$ ls -la
total 12
drwxr-xr-x  2 dx dx 4096 Dec 13 15:05 .
drwxr-xr-- 18 dx dx 4096 Dec 13 09:45 ..
-rw-r--r--  1 dx dx   39 Dec 13 15:05 filelist
$ cat filelist
file.tsv
hsperfdata_dx
hsperfdata_root
```

# Command Pipe and I/O Redirection



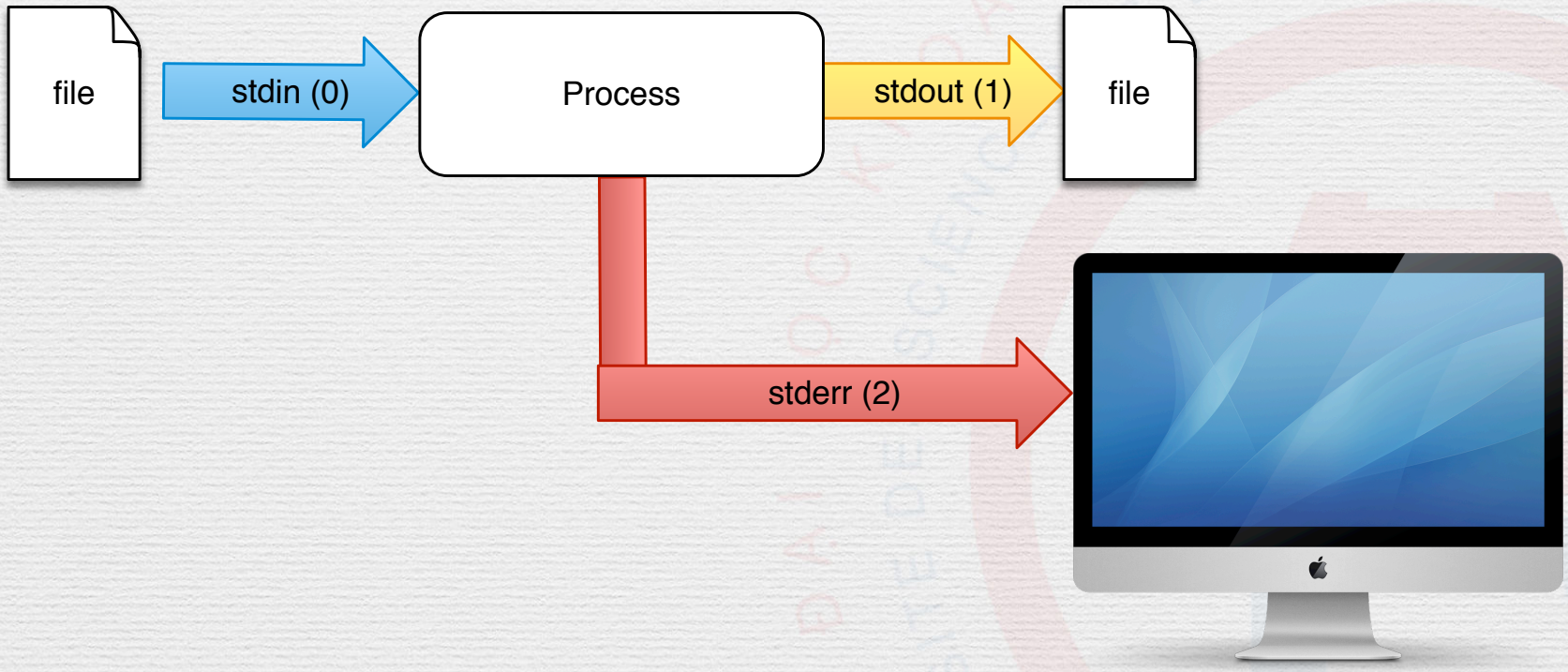

Input from file and output to terminal

# Command Pipe and I/O Redirection

- Redirect file to input : use `< filename`
- Everything read from stdin (default: Keyboard) will be read from file.

```
$ echo "1024 * 2 / 100 + 7" > formula
$ cat formula
1024 * 2 / 100 + 7
$ bc < formula
27
```

# Command Pipe and I/O Redirection

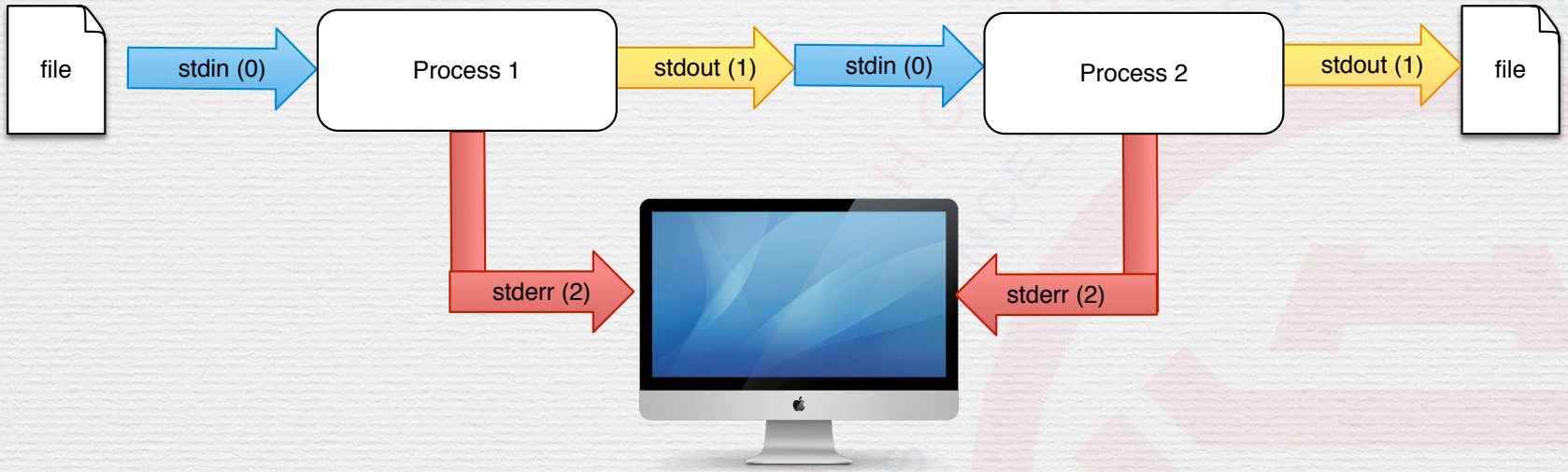



Input from file and output to another file

# Command Pipe and I/O Redirection

```
$ echo "1024 * 2 / 100 + 7" > formula
$ bc < formula > result
$ cat result
27
```

# Command Pipe and I/O Redirection




Input from file, pipe output of Process 1 to Process 2, output to another file

- stdout of the first process will be redirected to stdin of the second process.

# Command Pipe and I/O Redirection

- Make a pipe with |

```
$ ls /tmp > filelist
$ cat < filelist | gzip > filelist.gz
$ ls -la
total 28
drwxr-xr-x  2 dx dx 4096 Dec 13 15:39 .
drwxr-xr-- 18 dx dx 4096 Dec 13 09:45 ..
-rw-r--r--  1 dx dx   39 Dec 13 15:05 filelist
-rw-r--r--  1 dx dx   49 Dec 13 15:40 filelist.gz
$ gunzip < filelist.gz
file.tsv
hsperfdata_dx
hsperfdata_root
```

# Practical Work 1: Bash, Pipe and I/O Redirection

- Perform the following tasks in your Linux shell (next slide)
- Write a short report (text file, LaTeX or MarkDown is preferred, **NOT** Word document)
  - Name it « 01.report.bash.tex »
  - Write your commands or scripts
  - Save output of each command (last 10 lines if too long)
- Push your report to corresponding forked Github repository

# Practical Work 1: Bash, Pipe and I/O Redirection

Tasks

1. List everyone logged in and save the list in a file called "users" in your own home directory.
2. List all processes that are running and add this list to the end of the "users" file.
3. List everyone who is logged on sorted by their username
4. Count number of session each logged in user, sorted by this number in descending order

# Practical Work 1: Bash, Pipe and I/O Redirection

5. Show content of the first and last 3 lines of the file "/etc/fstab"
6. Retrieve line number 5 to 10 from the file "/etc/fstab" and write these lines into a new file "extract.txt"
7. List all files in current directory, recursively, to which the user has full permissions
8. Compare two files and show percentage of similarities between them

# Practical Work 1: Bash, Pipe and I/O Redirection

9. Find all files in current directory, recursively, that are at least 90% similar