# Introduction to Operating Systems

Daniel Hagimont (INPT)

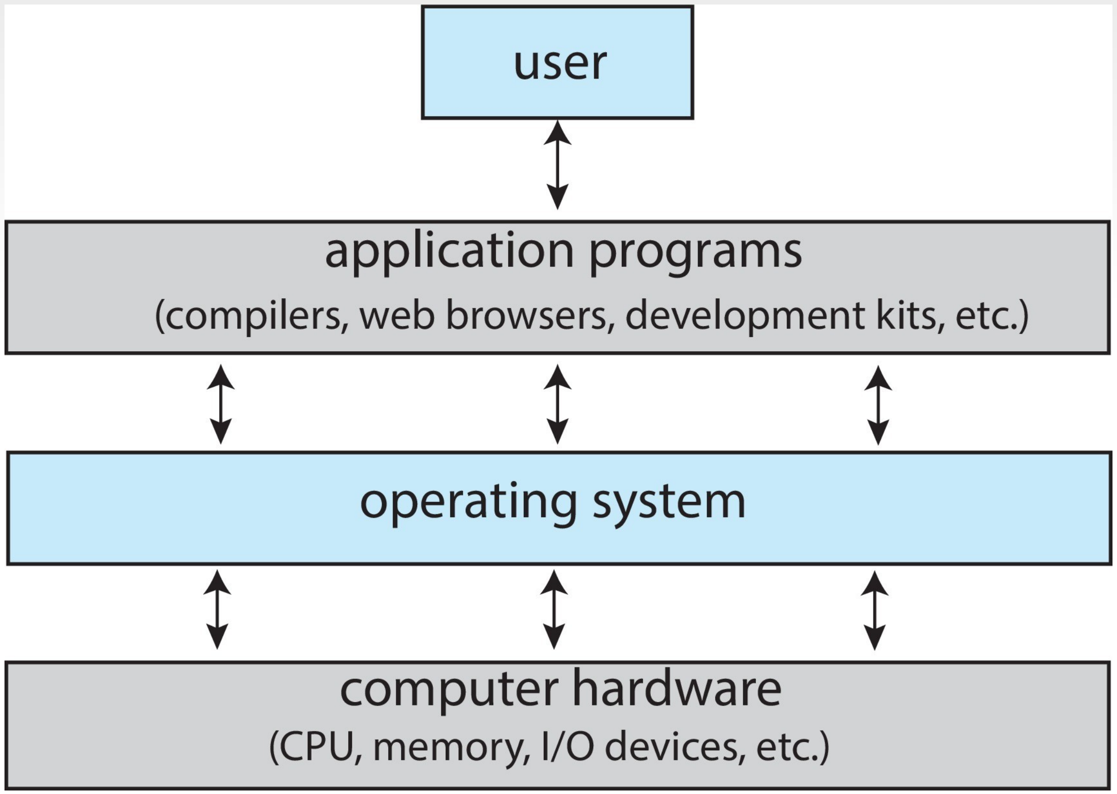
hagimont@enseeiht.fr

http://hagimont.perso.enseeiht.fr

## What is an Operating System?

- A program that acts as an intermediary between a user of a computer and the computer hardware
- Operating system goals:
  - Execute user programs and make solving user problems easier
  - Make the computer system convenient to use
  - Use the computer hardware in an efficient manner

# Abstract View of Components of Computer

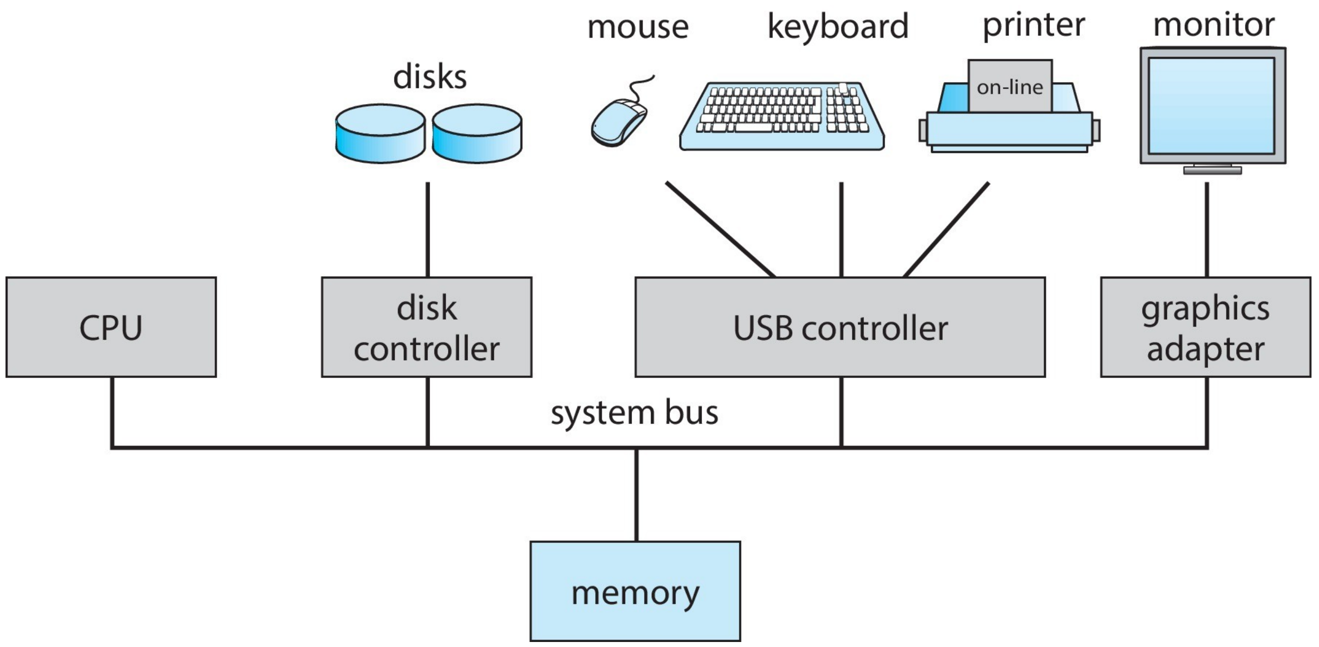

user
application programs
(compilers, web browsers, development kits, etc.)
operating system
computer hardware
(CPU, memory, I/O devices, etc.)

## Computer System Structure

- Computer system can be divided into four components:
  - Hardware – provides basic computing resources
    - CPU, memory, I/O devices
  - Operating system
    - Controls and coordinates use of hardware among various applications and users
  - Application programs – solve the computing problems of the users
    - Word processors, compilers, web browsers, database systems, video games
  - Users
    - People, machines, other computers

## Computer System Organization

- Computer-system operation
  - One or more CPUs, device controllers connect through common bus providing access to shared memory
  - Concurrent execution of CPUs and devices competing for memory cycles



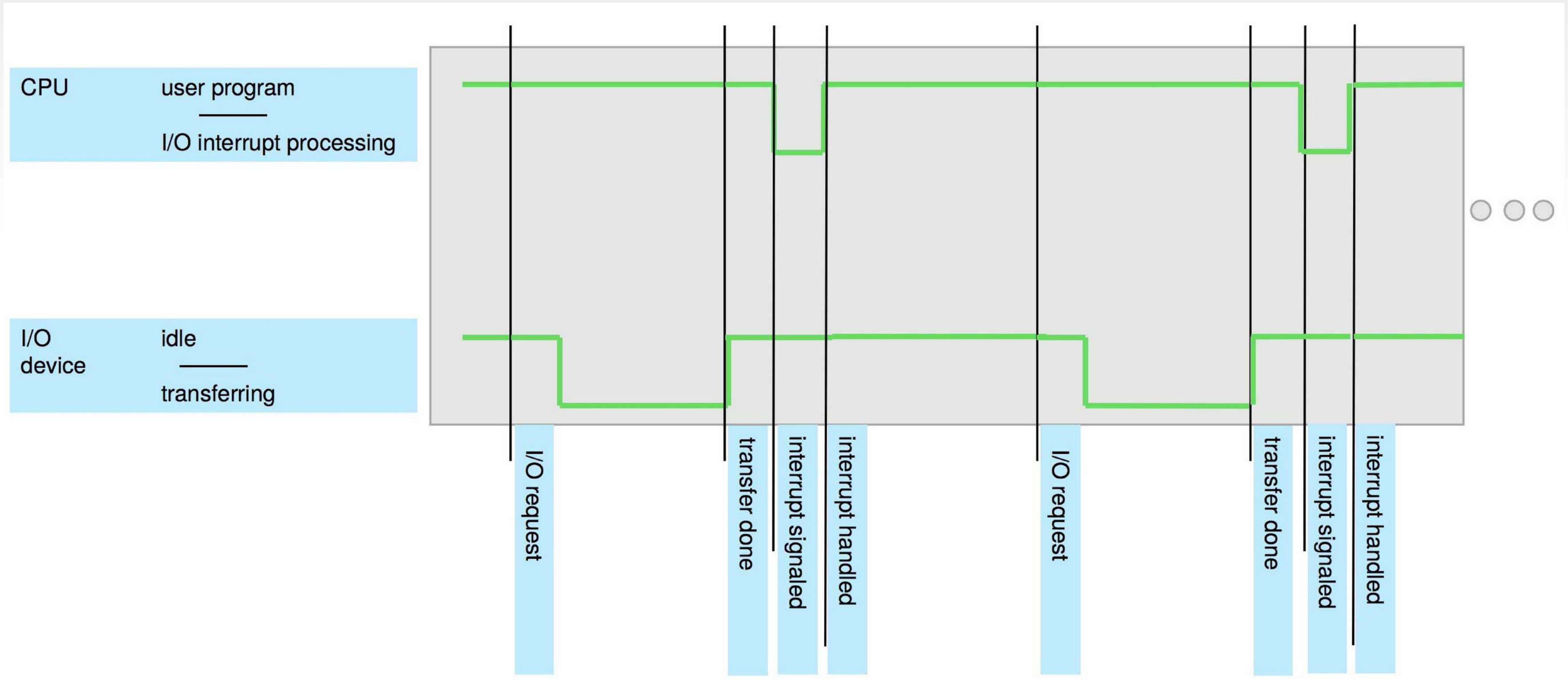
## Computer-System Operation

- I/O devices and the CPU can execute concurrently
- Each device controller is in charge of a particular device type
- Each device controller has a local buffer
- Each device controller type has an operating system device driver to manage it
- CPU moves data from/to main memory to/from local buffers (the buffer is generally mapped in memory)
- I/O is from the device to local buffer of controller
- Device controller informs CPU that it has finished its operation by causing an interrupt

## Common Functions of Interrupts

- Interrupt transfers control to the interrupt service routine generally, through the interrupt vector, which contains the addresses of all the service routines
- Interrupt architecture must save the address of the interrupted instruction (where it returns after handling)
- A trap or exception is a software-generated interrupt caused either by an error or a user request (system call)
- An operating system is interrupt driven

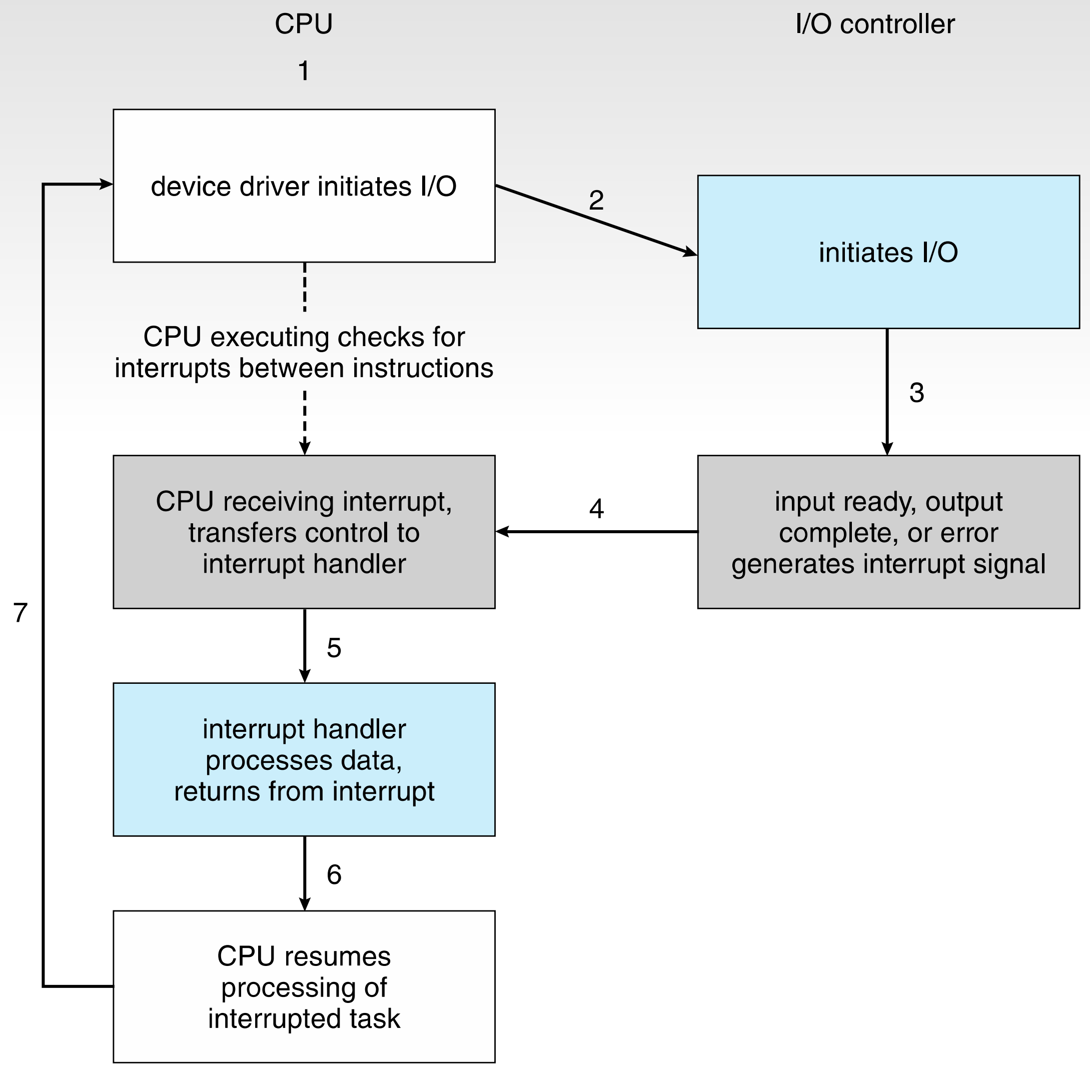
# Interrupt: Timeline



## Interrupt: Handling

- The operating system preserves the state of the CPU by storing the registers and the program counter
- Determines which type of interrupt has occurred
- Separate segments of code determine what action should be taken for each type of interrupt

## Interrupt: I/O Cycle



CPU
I/O controller
1
device driver initiates I/O
2
initiates I/O
CPU executing checks for interrupts between instructions
3
CPU receiving interrupt, transfers control to interrupt handler
4
input ready, output complete, or error generates interrupt signal
5
interrupt handler processes data, returns from interrupt
6
CPU resumes processing of interrupted task
7

## Two methods for handling I/O

- Non-blocking
    - After I/O starts, control returns to user program without waiting for I/O completion
- Blocking
    - After I/O starts, control returns to user program only upon I/O completion
- Process blocking is implemented by the OS

## Memory

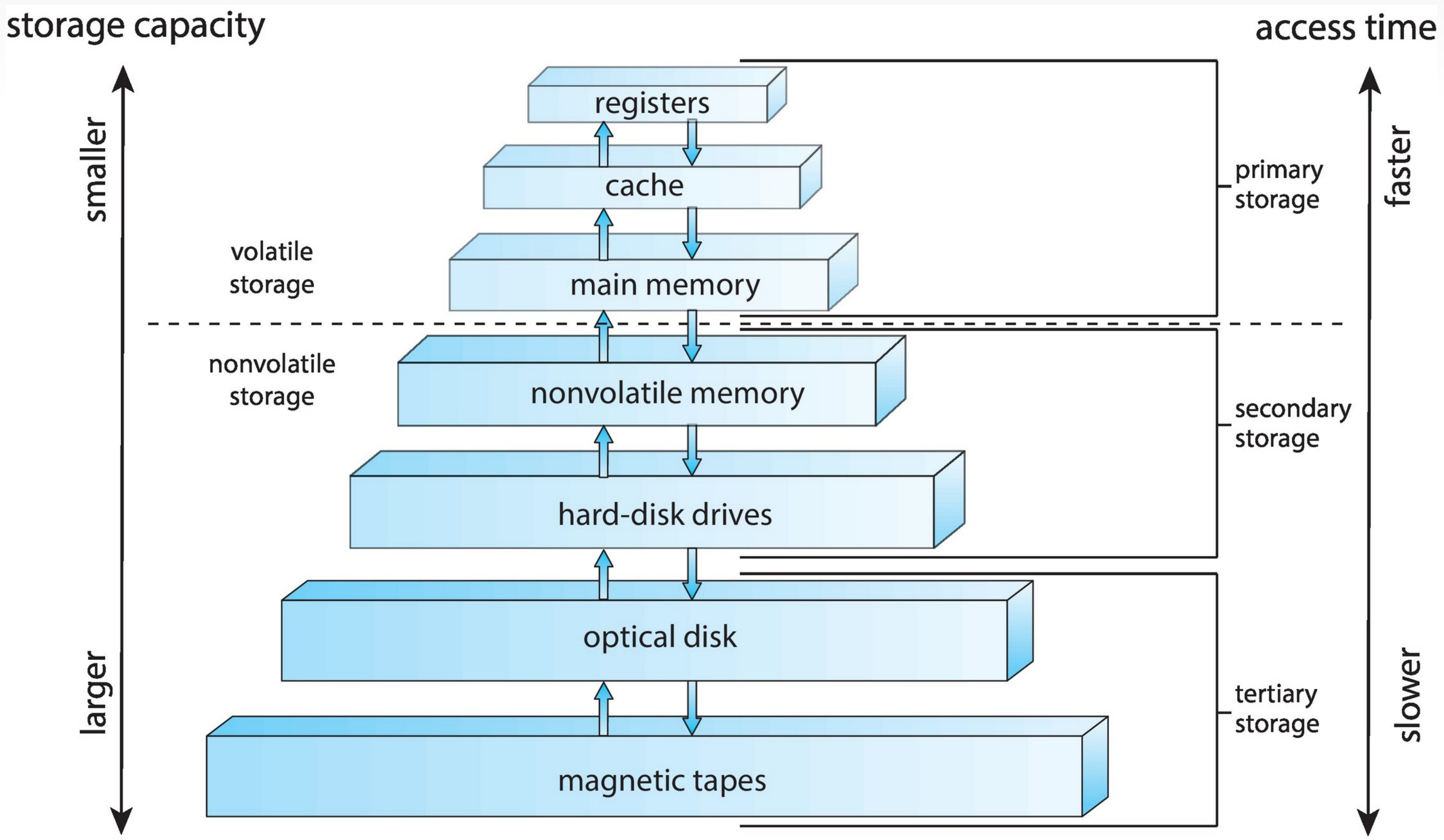
- Main memory – only large storage media that the CPU can access directly
  - Typically volatile
  - Typically random-access memory in the form of Dynamic Random-access Memory (DRAM)
- Secondary storage – extension of main memory that provides large non-volatile storage capacity

## Secondary storage

- Hard Disk Drives (HDD) – rigid metal or glass platters covered with magnetic recording material
    - Disk surface is logically divided into tracks, which are subdivided into sectors
    - The disk controller determines the logical interaction between the device and the computer
- Non-volatile memory (NVM) devices – faster than hard disks, nonvolatile
    - Various technologies
    - Becoming more popular as capacity and performance increases, price drops

# Memory Hierarchy

- Caching – copying information into faster memory system; main memory can be viewed as a cache for secondary storage



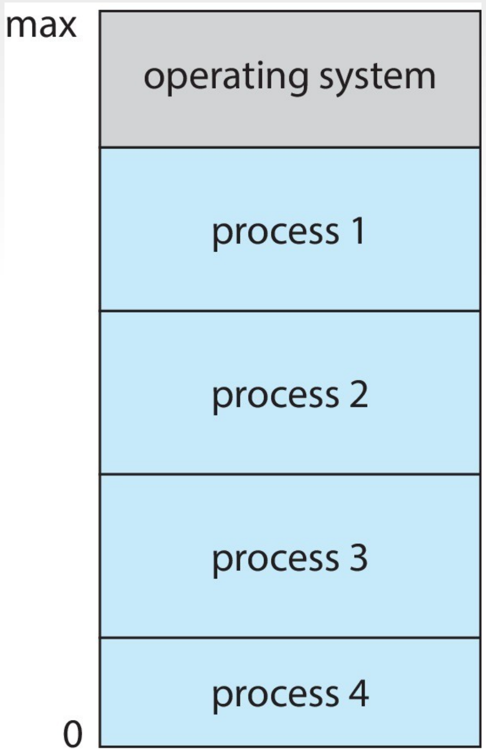
## Multiprogramming (Batch system)

- Single user cannot always keep CPU and I/O devices busy
- Multiprogramming organizes jobs (code and data) so CPU always has one to execute
- A subset of total jobs in system is kept in memory
- One job selected and run via job scheduling
- When job has to wait (for I/O for example), OS switches to another job

## Multitasking (Timesharing)

- A logical extension of Batch systems– the CPU switches jobs so frequently that users can interact with each job while it is running, creating interactive computing
- Response time should be < 1 second
- Each user has at least one program executing in memory - process
- If several jobs ready to run at the same time - CPU scheduling
- If processes don’t fit in memory, swapping (on disk) moves them in and out to run
- Virtual memory allows execution of processes not completely in memory

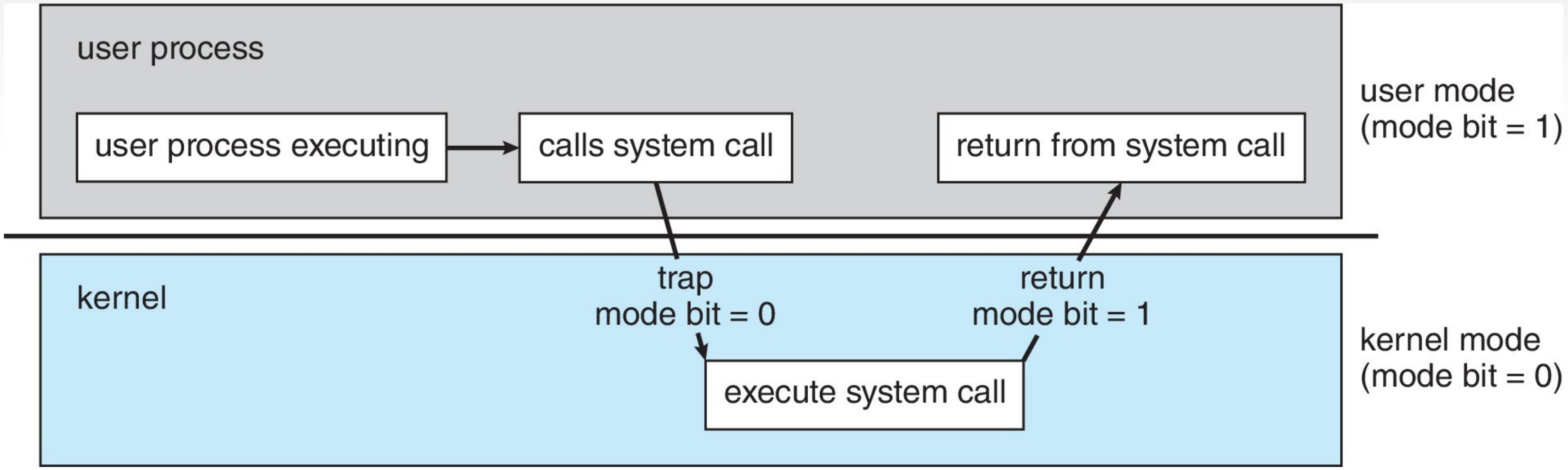
# Memory Layout for Multiprogrammed System



## OS protection: dual-mode operation

- Dual-mode operation allows OS to protect itself and other system components
    - User mode and kernel mode
- Mode bit provided by hardware
    - Provides ability to distinguish when system is running user code or kernel code.
    - When a user is running - mode bit is “user”
    - When kernel code is executing - mode bit is “kernel”
- How do we guarantee that user does not explicitly set the mode bit to “kernel”?
    - System call changes mode to kernel, return from call resets it to user
- Some instructions designated as privileged, only executable in kernel mode

# Transition from User to Kernel Mode



## Process Management

- A process is a program in execution. It is a unit of work within the system. Program is a passive entity; process is an active entity.
- Process needs resources to accomplish its task
  - CPU, memory, I/O, files
  - Initialization data
- Process termination requires reclaim of any reusable resources
- Each process has one program counter specifying location of next instruction to execute
  - Process executes instructions sequentially, one at a time, until completion
- Typically system has many processes, some user, some operating system running concurrently on one or more CPUs
  - Concurrency by multiplexing the CPUs among the processes

## Process Management Activities

- The operating system is responsible for the following activities in connection with process management
  - Creating and deleting both user and system processes
  - Suspending and resuming processes
  - Providing mechanisms for process synchronization
  - Providing mechanisms for process communication
  - Providing mechanisms for deadlock handling

## Memory Management

- To execute a program all (or part) of the instructions must be in memory
- All (or part) of the data that is needed by the program must be in memory
- Memory management determines what is in memory and when
    - Optimizing CPU utilization and computer response to users
- Memory management activities
    - Keeping track of which parts of memory are currently being used and by whom
    - Deciding which processes (or parts thereof) and data to move into and out of memory
    - Allocating and deallocating memory space as needed

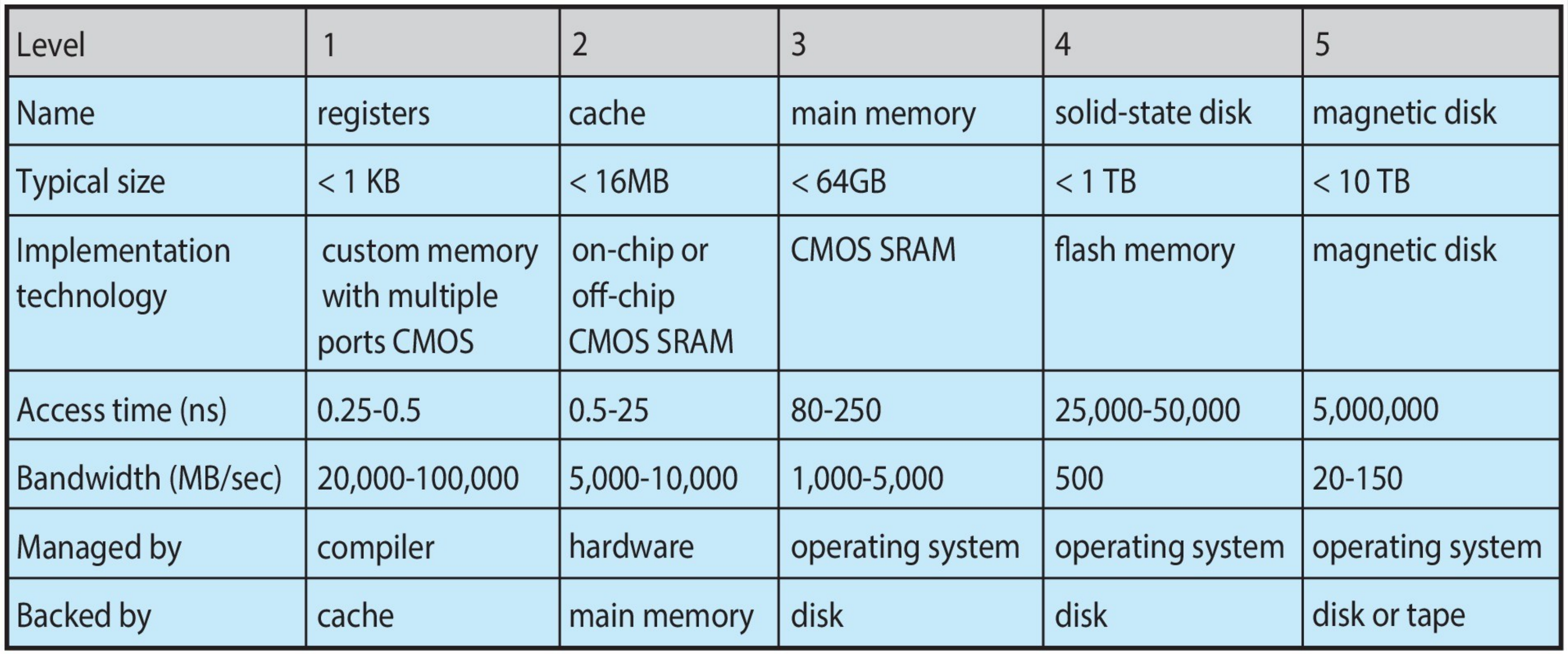
## File-system Management

- OS provides uniform, logical view of information storage
  - Abstracts from physical to logical storage unit - file
  - Each medium is controlled by device (i.e., disk drive, tape drive)
    - Varying properties include access speed, capacity, data-transfer rate, access method (sequential or random)
- File-System management
  - Files usually organized into directories
  - Access control on most systems to determine who can access what
  - OS activities include
    - Creating and deleting files and directories
    - Primitives to manipulate files and directories
    - Mapping files onto secondary storage
    - Backup files onto stable (non-volatile) storage media

## Caching

- Important principle, performed at many levels in a computer (in hardware, operating system, software)
- Information in use copied from slower to faster storage temporarily
- Faster storage (cache) checked first to determine if information is there
  - If it is, information used directly from the cache (fast)
  - If not, data copied to cache and used there
- Cache smaller than storage being cached
  - Cache management important design problem
  - Cache size and replacement policy

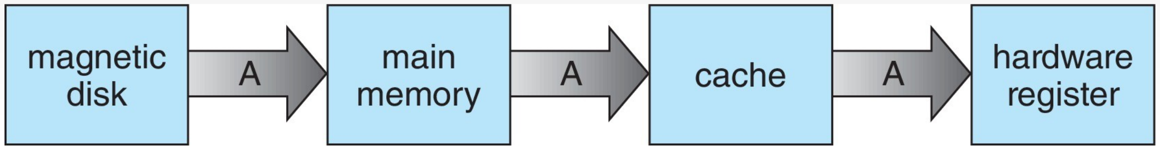
## Characteristics of Various Types of Storage



| Level | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| Name | registers | cache | main memory | solid-state disk | magnetic disk |
| Typical size | < 1 KB | < 16MB | < 64GB | < 1 TB | < 10 TB |
| Implementation technology | custom memory with multiple ports CMOS | on-chip or off-chip CMOS SRAM | CMOS SRAM | flash memory | magnetic disk |
| Access time (ns) | 0.25-0.5 | 0.5-25 | 80-250 | 25,000-50,000 | 5,000,000 |
| Bandwidth (MB/sec) | 20,000-100,000 | 5,000-10,000 | 1,000-5,000 | 500 | 20-150 |
| Managed by | compiler | hardware | operating system | operating system | operating system |
| Backed by | cache | main memory | disk | disk | disk or tape |

# Migration of data "A" from Disk to Register

- Multitasking environments must be careful to use most recent value, no matter where it is stored in the storage hierarchy



- Multiprocessor environment must provide cache coherency in hardware such that all CPUs have the most recent value in their cache
- Distributed environment situation even more complex
  - Several copies of a datum can exist

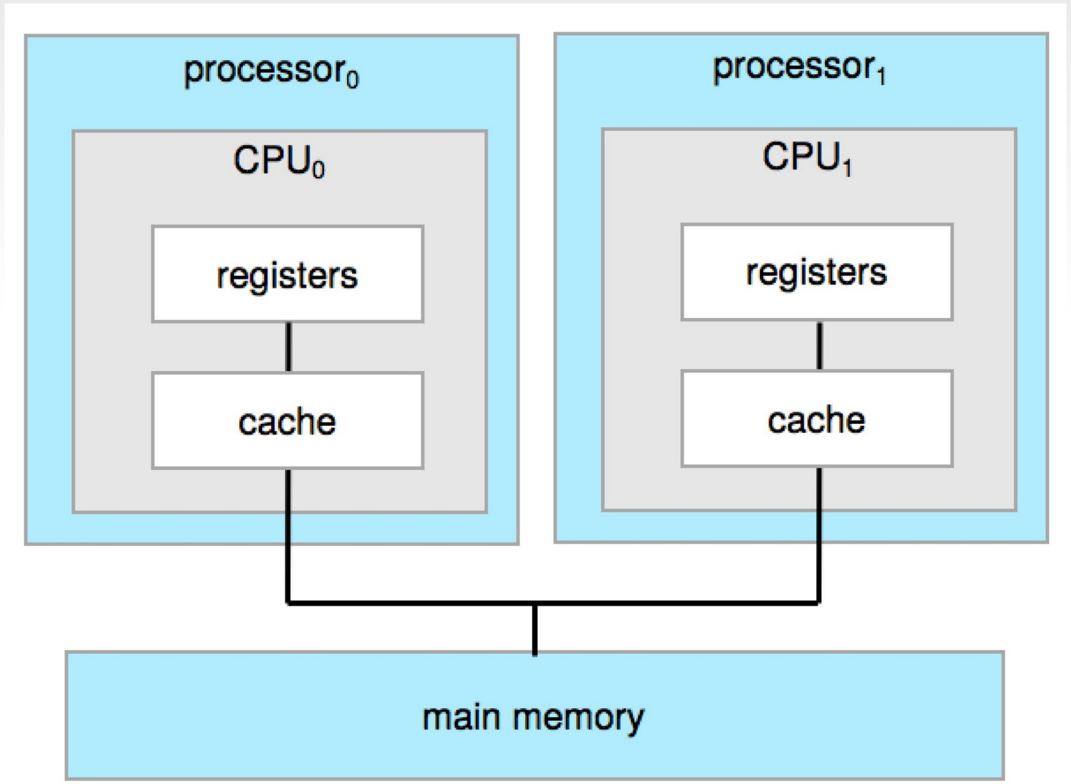
## Protection and Security

- Protection – any mechanism for controlling access of processes or users to resources defined by the OS
- Security – defense of the system against internal and external attacks
  - Huge range, including denial-of-service, worms, viruses, identity theft, theft of service
- Systems generally first distinguish among users, to determine who can do what
  - User identities (user IDs, security IDs) include name and associated number, one per user
  - User ID then associated with all files, processes of that user to determine access control
  - Group identifier (group ID) allows set of users to be defined and controls managed, then also associated with each process, file
  - Privilege escalation allows user to change to effective ID with more rights

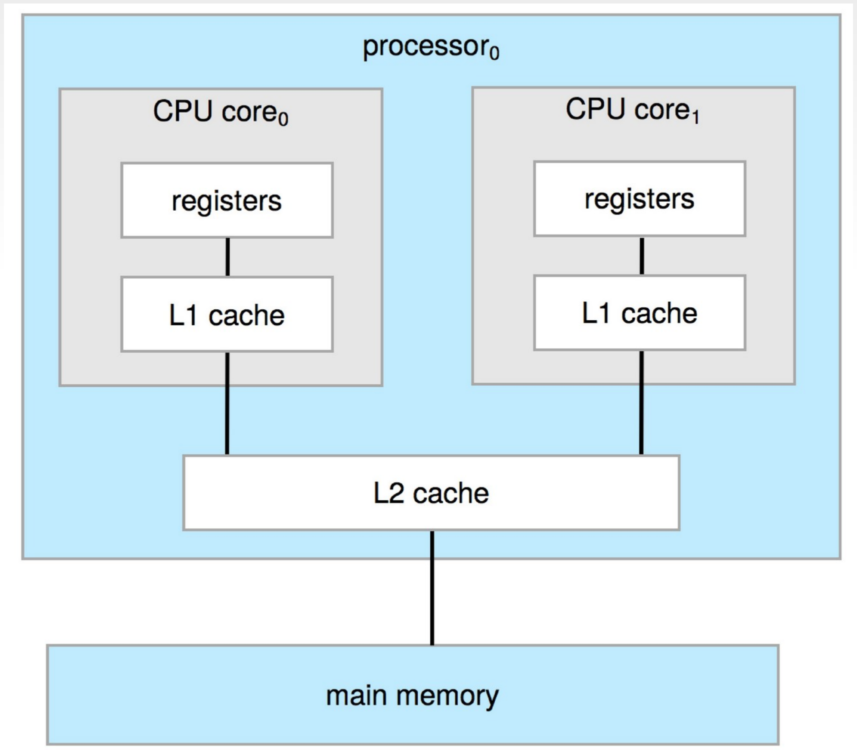
# Computer-System Architecture

- Most systems use a single general-purpose processor
    - This processor can be multi-core
    - Most systems have special-purpose processors as well
- Multiprocessors systems growing in use and importance
    - Also known as parallel systems, tightly-coupled systems
    - Advantages include:
        - Increased throughput
        - Economy of scale
        - Increased reliability – graceful degradation or fault tolerance
    - Two types:
        - Asymmetric Multiprocessing – each processor is assigned a specific task
        - Symmetric Multiprocessing – each processor performs all tasks

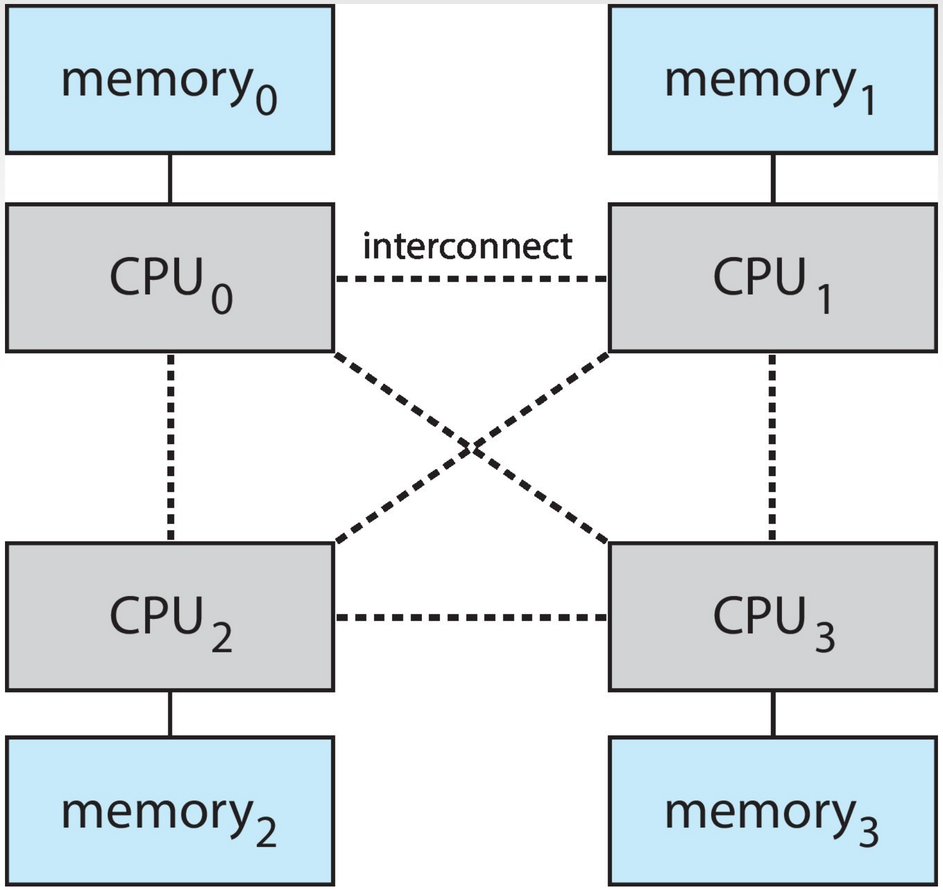
## Symmetric Multiprocessing Architecture



# Dual-Core Design



# Non-Uniform Memory Access System



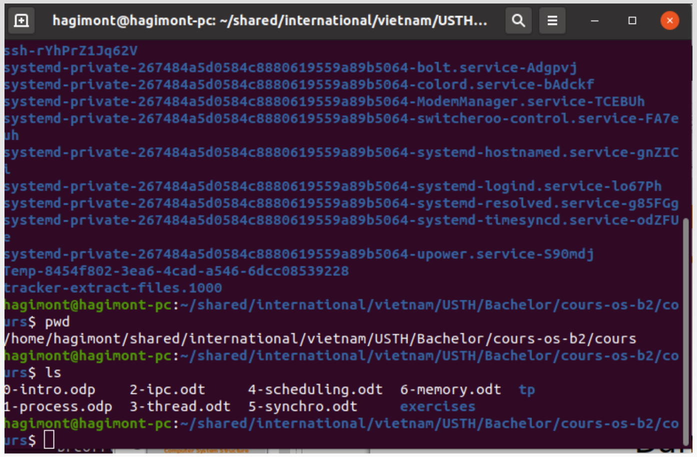
All memories are viewed as a unique memory range

## Clustered Systems

- Like multiprocessor systems, but multiple systems working together
    - Usually sharing storage via a storage-area network (SAN)
    - Provides a high-availability service which survives failures
        - Asymmetric clustering has one machine in hot-standby mode
        - Symmetric clustering has multiple nodes running applications, monitoring each other
    - Some type of clusters
        - High-performance computing (HPC)
        - Big data computing

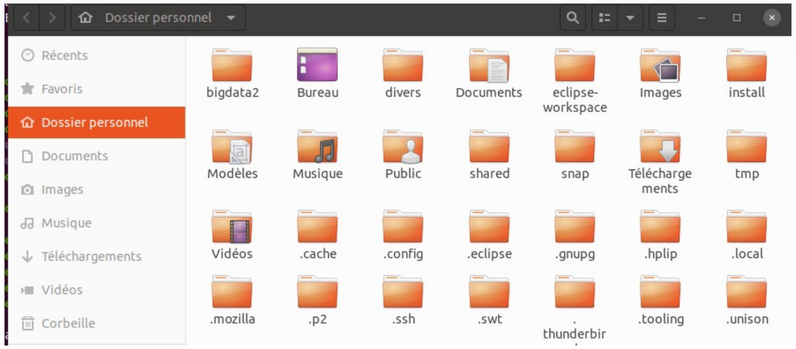
# System interface

- Command Line Interpreter

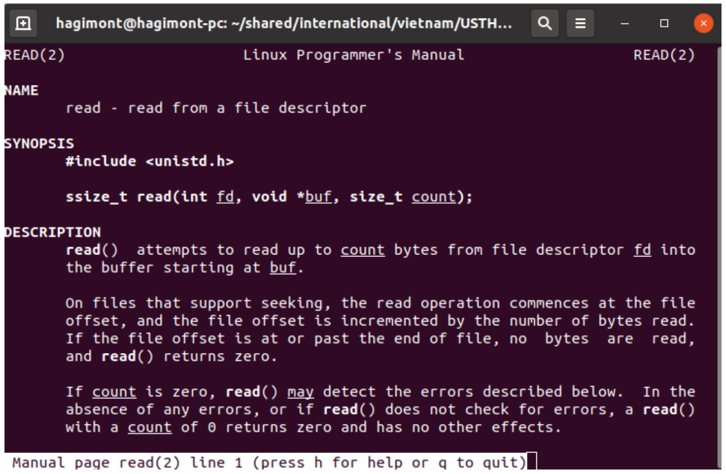


```
hagimont@hagimont-pc: ~/shared/international/vietnam/USTH...
ssh-rYhPrZ1Jq62V
systemd-private-267484a5d0584c8880619559a89b5064-bolt.service-Adgpvj
systemd-private-267484a5d0584c8880619559a89b5064-colord.service-bAdckf
systemd-private-267484a5d0584c8880619559a89b5064-ModemManager.service-TCEBUh
systemd-private-267484a5d0584c8880619559a89b5064-switcheroo-control.service-FA7e
uh
systemd-private-267484a5d0584c8880619559a89b5064-systemd-hostnamed.service-gnZIC
i
systemd-private-267484a5d0584c8880619559a89b5064-systemd-logind.service-lo67Ph
systemd-private-267484a5d0584c8880619559a89b5064-systemd-resolved.service-g85FGg
systemd-private-267484a5d0584c8880619559a89b5064-systemd-timesyncd.service-odZFU
e
systemd-private-267484a5d0584c8880619559a89b5064-upower.service-S90mdj
Temp-8454f802-3ea6-4cad-a546-6dcc08539228
tracker-extract-files.1000
hagimont@hagimont-pc:~/shared/international/vietnam/USTH/Bachelor/cours-os-b2/co
urs$ pwd
/home/hagimont/shared/international/vietnam/USTH/Bachelor/cours-os-b2/cours
hagimont@hagimont-pc:~/shared/international/vietnam/USTH/Bachelor/cours-os-b2/co
urs$ ls
0-intro.odp    2-ipc.odt     4-scheduling.odt  6-memory.odt  tp
1-process.odp  3-thread.odt  5-synchro.odt     exercises
hagimont@hagimont-pc:~/shared/international/vietnam/USTH/Bachelor/cours-os-b2/co
urs$ 
```

- Graphical User Interface

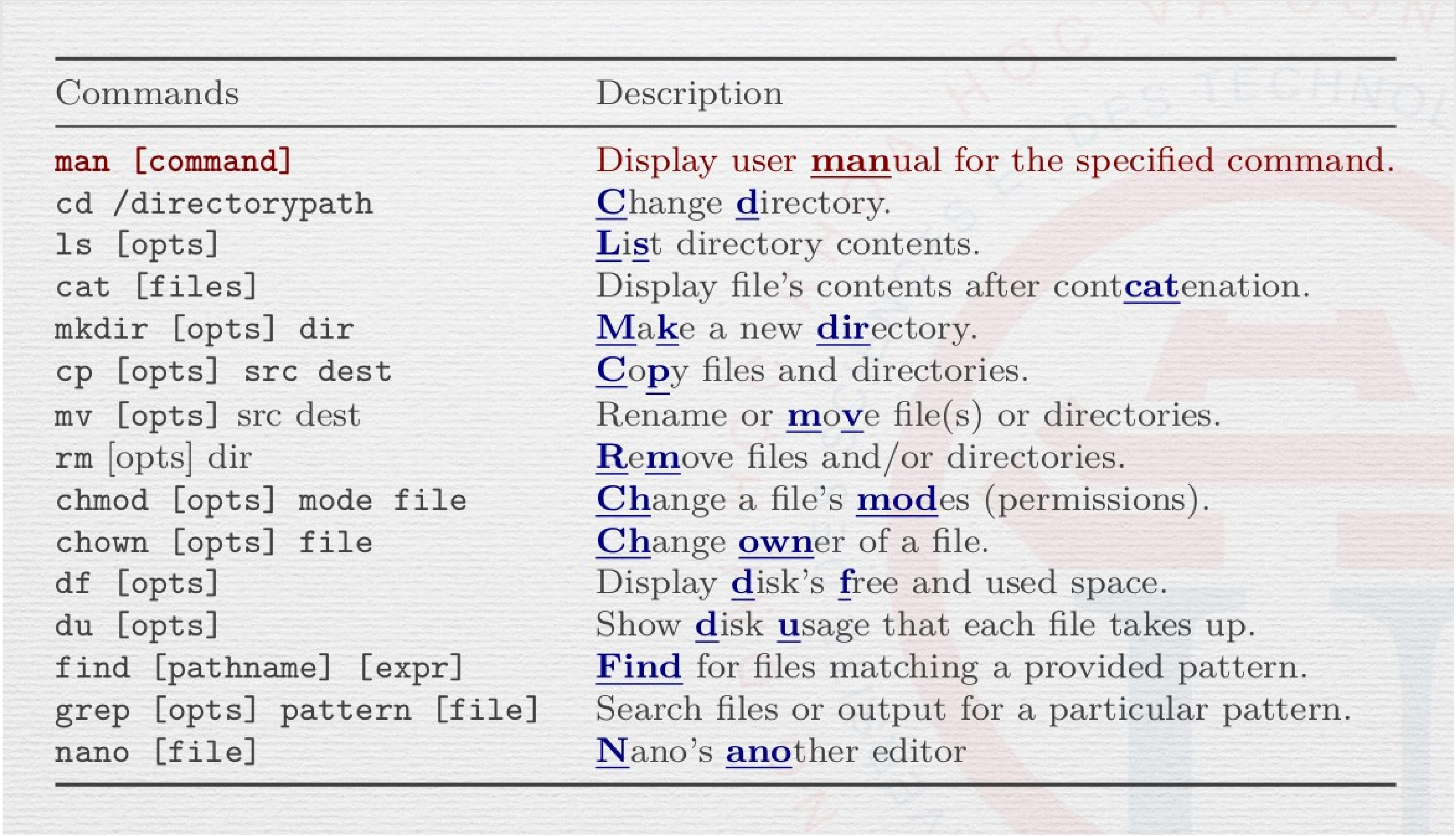


- System APIs



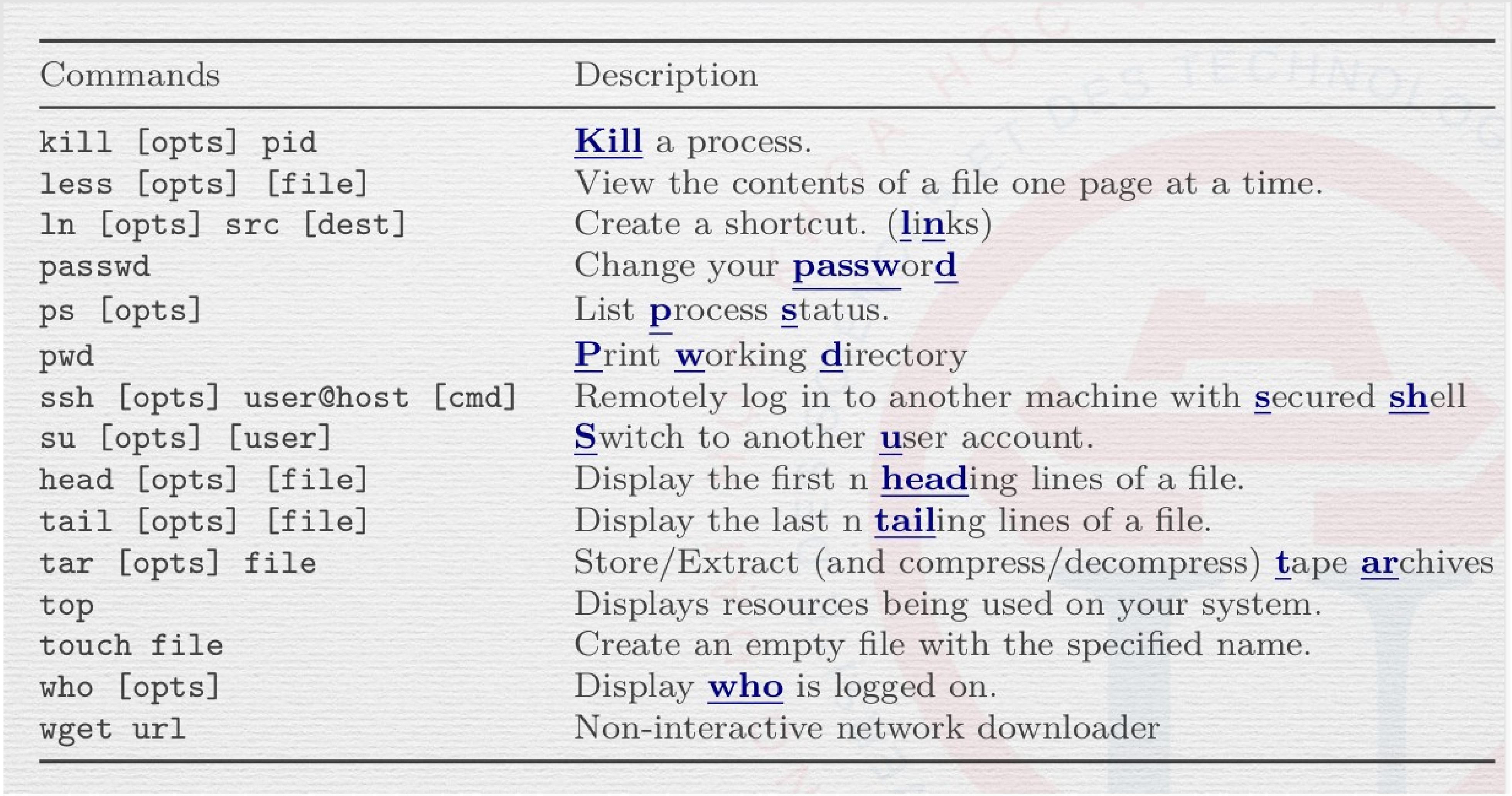
```
hagimont@hagimont-pc: ~/shared/international/vietnam/USTH...
READ(2)                    Linux Programmer's Manual                   READ(2)

NAME
       read - read from a file descriptor

SYNOPSIS
       #include <unistd.h>

       ssize_t read(int fd, void *buf, size_t count);

DESCRIPTION
       read()  attempts to read up to count bytes from file descriptor fd into
       the buffer starting at buf.

       On files that support seeking, the read operation commences at the file
       offset, and the file offset is incremented by the number of bytes read.
       If the file offset is at or past the end of file, no  bytes  are  read,
       and read() returns zero.

       If count is zero, read() may detect the errors described below.  In the
       absence of any errors, or if read() does not check for errors, a read()
       with a count of 0 returns zero and has no other effects.

 Manual page read(2) line 1 (press h for help or q to quit)
```

## Basic shell commands



| 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 con**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 |

# Basic shell commands



| 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. (**l**i**n**ks) |
| `passwd` | Change your **passw**or**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 |

## Organization

- To make it manageable, I propose that you constitute groups of 4
  - For exercises and/or practicle works
    - We will use discord
      - https://discord.gg/YPdVQ8uysT
    - We will use git/github (see below)
- Git
  - Create a new directory OS2020
  - Fork the course's git repository to your git account
    - https://github.com/hagimont/OS2020
  - Clone your forked repository to your OS2020 directory
    - git clone https://github.com/your-account/OS2020.git
  - Edit «README.md», add your names/emails
  - Commit and push

## git/github

- To make it manageable, I propose that you constitute groups of 4
- Create a new directory OS2020
- Fork the course’s git repository to your git account
    - https://github.com/hagimont/OS2020
- Clone your forked repository to your OS2020 directory
    - git clone https://github.com/your-account/OS2020.git
    - Edit «README.md», add your names/emails
    - Commit and push

## Exercise

- find
    - Find all the core file in a directory (recursively)
    - Remove them
- chmod
    - Make a binary executable or not
    - Make a directory accessible (with cd) or not
- grep
    - Find the occurences of a word in a file
- ps/kill
    - Create a process and kill it
- ln
    - Create a link to a file

## Exercise

- apt
  - Install a ssh server
- ssh
  - Connect to the ssh server of another student
- wget
  - Dowload a software from a web site
- tar
  - Uncompress an archive
  - Create an archive
- pipe
  - Count the number of firefox processes running