Operating Systems

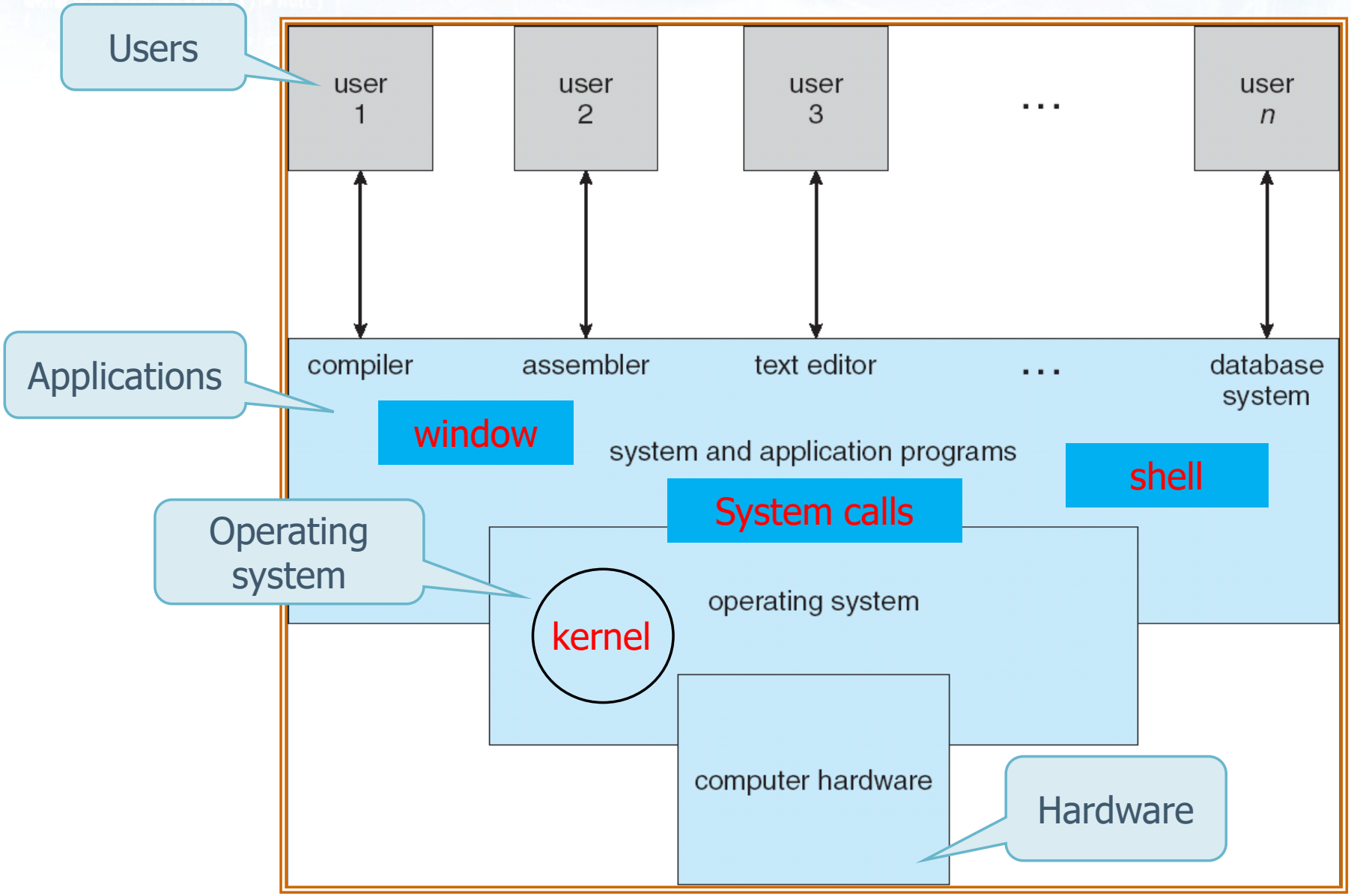
Introduction to Operating Systems

Stefano Quer, Pietro Laface, and Stefano Scanzio
Dipartimento di Automatica e Informatica
Politecnico di Torino
skenz.it/os stefano.scanzio@polito.it

Computer System Components

- An elaboration system is composed of the following components:
- Hardware
  - Provides basic computing resources (CPU, memory, I/O devices)
- Operating system
  - controls and coordinates the use of the hardware among the various application programs for the various users
- System and application programs
  - User services (compilers, databases, office automation programs, games, etc.)
- Users
  - People, machines, other computers

Computer System Components



Operating System

- What is?
  - A software interface between a user or an application program and the hardware
- Goal
  - Execute commands and programs (make easier problem solution)
  - Make system friendly
  - Use and share hardware efficiently
- Can be considered a
  - Virtual machine that manages and allocates available resources.
  - ”Program” that controls the execution of user programs, and operations of I/O devices

OS: Virtual Machine

- Hardware devices are really complex to be managed and programmed
- The OS can be considered using a top-down view as:
  - An abstract manager of resources and information
    - It abstract (and hide) information and resources
    - It allows to manage them in a simplified way
  - A software interface between system users and the hardware

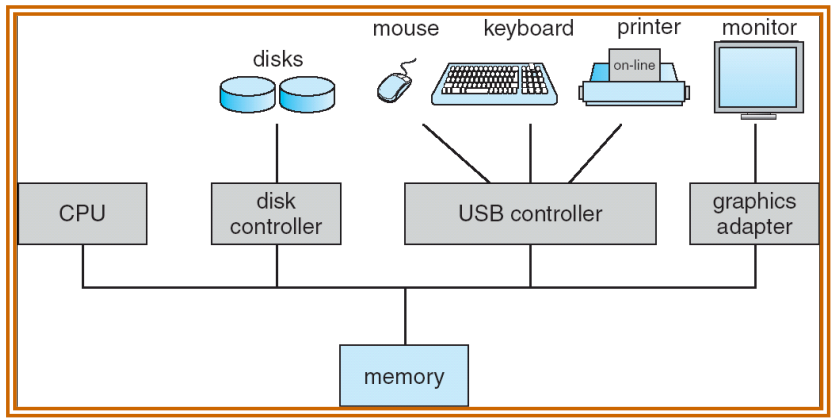
OS: Resource Manager

- With a bottom-up view, the OS can be categorized as a resource manager
- A program that controls:
  - Devices and operations on devices
  - The execution of users programs
- From this point of view, it can be considered as a set of **modules**, each of which provides some **services** to users

Modules and Services

- Modules and services of an Operating System
  - **Command interpreter**
  - **Process management**
  - Main Memory Management
  - Secondary Memory Management
  - Management of I/O devices
  - **File, and file system management**
  - Implementation of protection mechanisms
  - Network management, and distributed systems

Analyzed on this course



Modules and Services

- Modules and services of an Operating System
- Command interpreter
  - The user and OS communicate through an textual or graphical interface
  - The user performs its tasks through a command interpreter (shell)
  - The OS allows the user to
    - Manage processes
    - Manage main and secondary memory
    - Establish protection policies
    - Manage the network and external connections

Modules and Services

- Modules and services of an Operating System
- Process management
  - A process (active unit) is a program (passive unit) in execution
  - To run it requires resources
    - CPU, memory, devices, etc.
  - The OS offers support for
    - Creating, suspending and deleting processes
    - Establishing communication mechanisms and synchronization among processes

Modules and Services

- Modules and services of an Operating System
- Main Memory Management
  - The data and instructions of a program must be in a region of main memory to allows a process to be executed
  - Logically, main memory is a vector of elements (words)
  - The OS must
    - Manage the use of memory (which regions are used and which are free)
    - Decide which processes to allocate in memory, and which can be deallocated
    - Optimize CPU access to memory

Modules and Services

- Modules and services of an Operating System
- Secondary Memory Management
  - Since main memory is volatile and small, data are contained permanently on mass storage
  - The OS must
    - Organize information in the available space
    - Allocate/deallocate the required space
    - Manage the free space
    - Optimize R/W operation scheduling

Modules and Services

- Modules and services of an Operating System
- I/O devices management
  - I/O devices cannot be managed directly by the users (complexity, driver, sharing, etc.)
  - The OS must
    - Hide the details of a device to users by providing a uniform interface to the user
    - Providing read, write, control operations on devices

Modules and Services

- Modules and services of an Operating System
- File, and file system management
  - Data on secondary memory are organized into one or more file systems, which contain directories and files
  - The OS must
    - Create, read, write, remove files and directories
    - Establish appropriate access protection mechanisms for data privacy and sharing
    - Optimize R/W operations

Modules and Services

- Modules and services of an Operating System
- Implementation of protection mechanisms
  - **Protection** indicates access control for users and processes to system resources
  - The OS must
    - Define the access rights associated to users and resources
    - Distinguish between authorized and unauthorized use
    - Keep track of which users is using system resources

Modules and Services

- Modules and services of an Operating System
- Network and distributed systems management
  - A network is a collection of processors that do not share memory and clock
  - The nodes of the network are interconnected by communication paths
  - The OS must
    - Grant access to system resources
    - Increase the performance and reliability of the computing system, and the amount of data that can be processed

Terminology and basic concepts

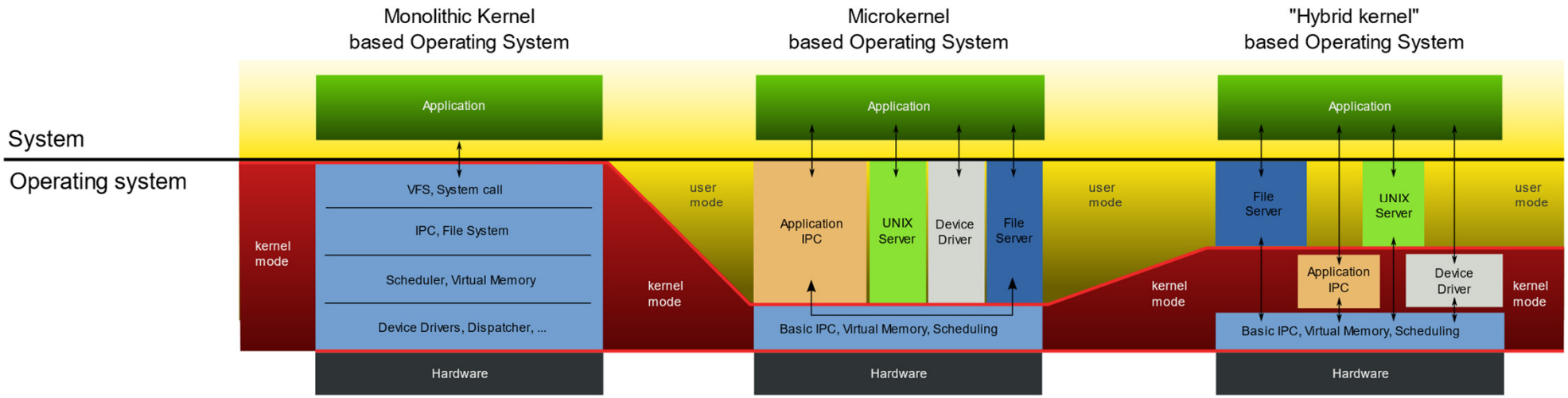
- Terminology and basic concepts of OS
  - Kernel, bootstrap, kernel protection, system call
  - Login, shell
  - Filesystem, filename, pathname, working directory, home directory, root directory
  - Program (sequential and concurrent), process, thread
  - Pipe
  - Deadlock, livelock, starvation, polling (busy waiting)

Terminology and basic concepts

- Terminology and basic concepts of OS:
- Kernel
  - Is the core of an OS
  - It manages all system resources
    - In particular, it manages memory and processors
    - A program (or better module) always in execution
    - All other programs are system programs or applications
  - There are different types of kernel
    - Modular kernel that is subdivided into levels
    - Microkernel that provides only basic functionalities
    - Monolithic kernel that provides functionalities through the device drivers (most common)

Kernel

- Several types of kernels
  - Direct access to hardware can also be very complex
  - Kernels implement one or more types of hardware abstraction



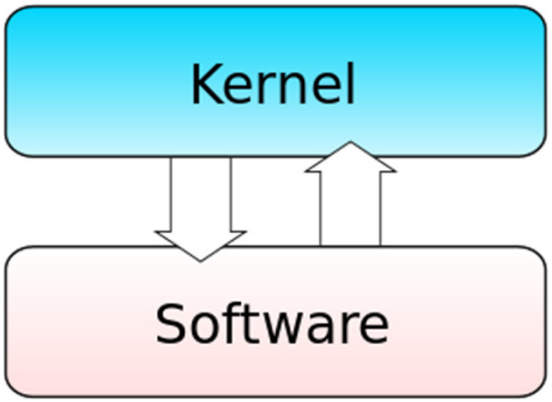
Terminology and basic concepts

Monolithic kernels

- The services are realized through a set of system calls made by separate modules but whose integration is very tight

System call

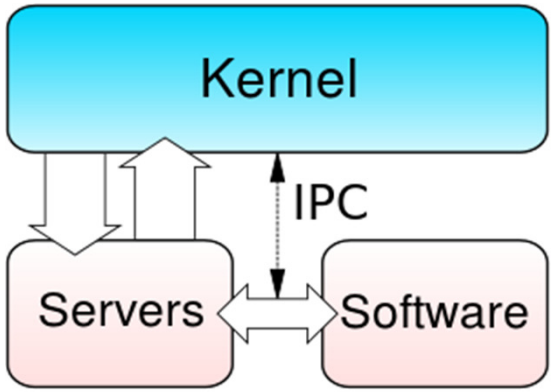
- Drawbacks
  - A problem on a module can block the entire system
  - Adding a new hardware device involves adding its module to the kernel and recompile all the kernel (in modern kernel modules can be loaded at runtime)
- Advantages
  - The tight internal integration of the components is extremely efficient
- Common solution
  - Unix, Linux, Open VMS, XTS-400



Terminology and basic concepts

Microkernel

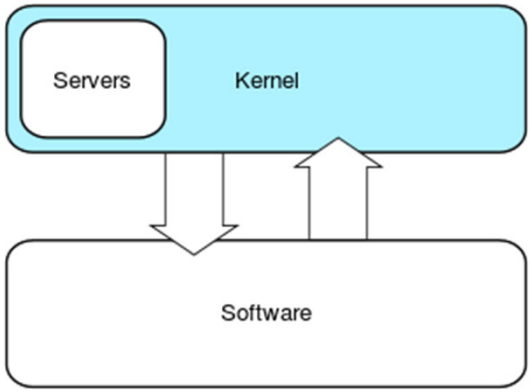
- It defines very small and simple software modules on top of the hardware that implement minimal services
- Separate core service implementations from operating system operational structures
- Drawbacks
  - Slow due to the high number of core services and context switching
- Advantages
  - Very stable
- Not common solution
  - March, L4, AmigaOS, Minix



Terminology and basic concepts

Hybrid approaches

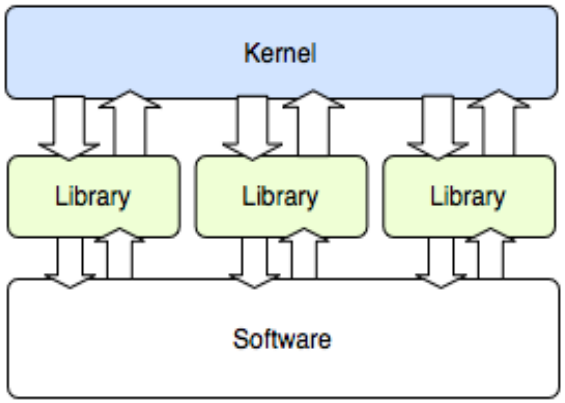
- Intermediate approach between the previous two
  - Microkernel with additional and "non-essential" kernel-level code, which can be executed very quickly
- Compromise adopted by many developers
- Drawbacks
  - Performance (slightly worse) but comparable with monolithic kernels
- Advantages
  - It integrates advantages of monolithic and microkernels
- Common solution
  - Windows NT, Netware, XNU Kernel di Mac OS X, Dragonfly BSD



Terminology and basic concepts

Esokernel

- Known as "vertical operating systems"
  - Radically different approach to operating system design, **more direct access to hardware**
  - They separate "protection from management"
- Extremely small and compact, as their functionality is arbitrarily limited to resource protection and multiplexing
- Drawbacks
  - They involve more work in application development (libraries decrease this effort)
- Advantages
  - Less hardware abstraction
- Examples
  - Nemesis, ExOS

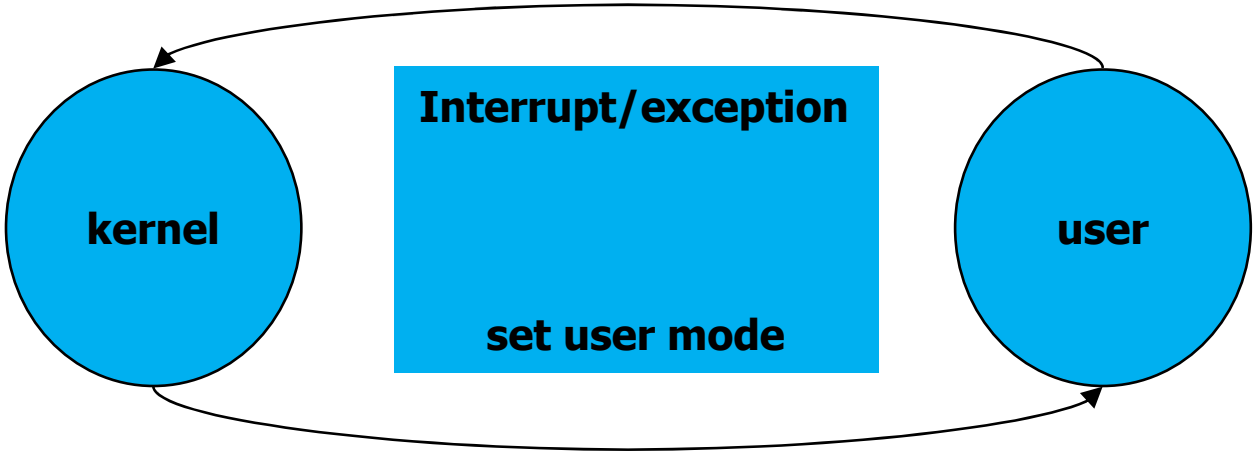


Terminology and basic concepts

- Terminology and basic concepts of OS:
- Bootstrap
  - Bootstrap (bootstrap or booting program)
    - Initialization program
    - Executes at power-on performing a proper check and initialization of the computer hardware, then it loads the kernel into main memory
  - The bootstrap program is usually
    - Stored in ROM and EEPROM (firmware)
    - Loaded at power-up or reboot

Terminology and basic concepts

- Terminology and basic concepts of OS:
- Kernel protection
  - **Mode bit** added to computer hardware to indicate the current mode: kernel (0) or user (1).
  - When an interrupt, exception, or fault occurs, hardware switches to kernel mode.



  - **Privileged instructions**, that can be issued only in kernel mode

Terminology and basic concepts

- Terminology and basic concepts of OS:
- Kernel protection
  - All I/O instructions are privileged instructions
  - Changing the content of a system register can only be done in kernel mode
    - Dual mode ensures that a user program cannot gain control of the computer in kernel mode
    - Memory protection does not allow a user to write in kernel memory, e.g., store a new address in the interrupt vector. Load the memory protection registers is a privileged instruction
    - Timer commonly used to implement time sharing
      - Load the timer is a privileged instruction

Terminology and basic concepts

- Given the I/O instructions are privileged, how does the user program perform I/O?
- More generally, how does a user program call a kernel function?

Terminology and basic concepts

- System call
- It is the interface with the services provided by the OS, i.e., it is the entry point of the OS
  - Often implemented in assembler
  - Often accessible with high level Application Program Interface (API)
    - Win32/64 API (for Windows)
    - POSIX API (for UNIX, Linux, MAC OS X)
    - Java API (for Java Virtual Machine)
  - How many system calls exist in an OS?
    - UNIX 4.4 BSD: about 110
    - Linux: between 240 and 260
    - UNIX FreeBSD: about 320

Terminology and basic concepts

- Terminology and basic concepts of OS:
- System call
  - Difference between system calls and functions
    - Both provide services to users
    - Any system call is typically coupled with one or more functions with the same name of the system call, and defined with a high-level programming language (e.g., C)
    - Functions can be substituted and modified, system calls are kept stable over the years
    - System calls provide basic functionalities (and they have a non-complex prototype), functions inside libraries provide more elaborate functionalities

Terminology and basic concepts

- Terminology and basic concepts of OS:
- System call: Example
  - POSIX versus Win32/64 API

UNIX

```
int read (int fd, void *buffer, size_t nbytes);
```

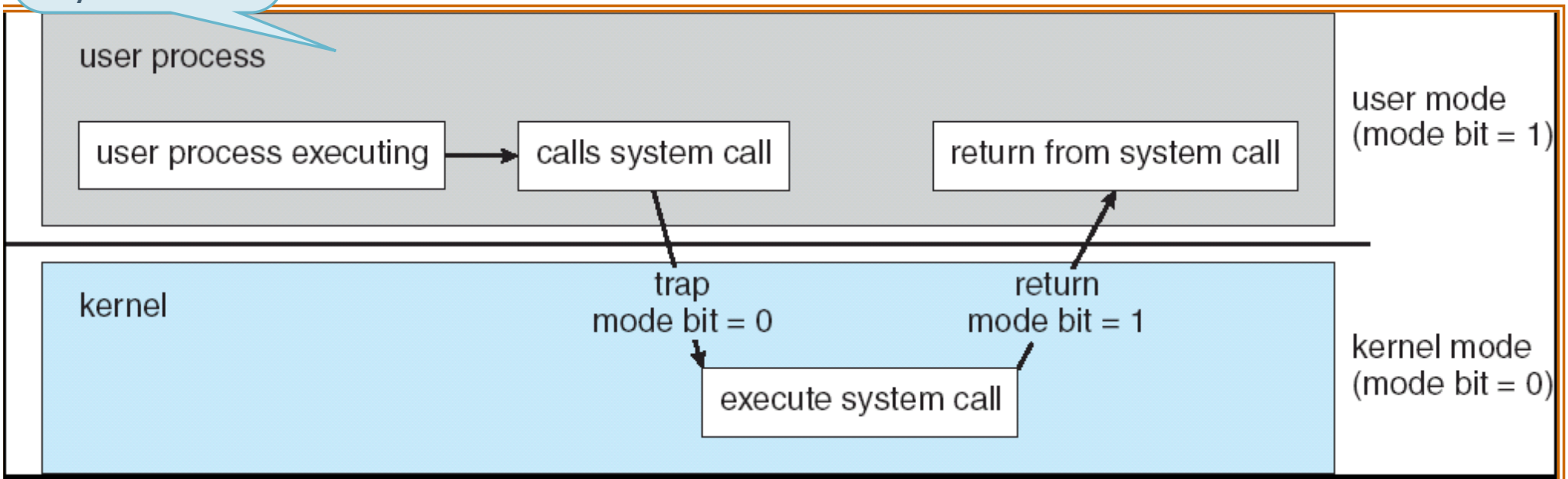
Windows

```
BOOL ReadFile (
   HANDLE fileHandle,
   LPVOID dataBuffer,
   DWORD numberOfByteToRead,
   LPDWORD numberOfByteRead,
   LPOVERLAPPED overlappedDataStructure
);
```

Terminology and basic concepts

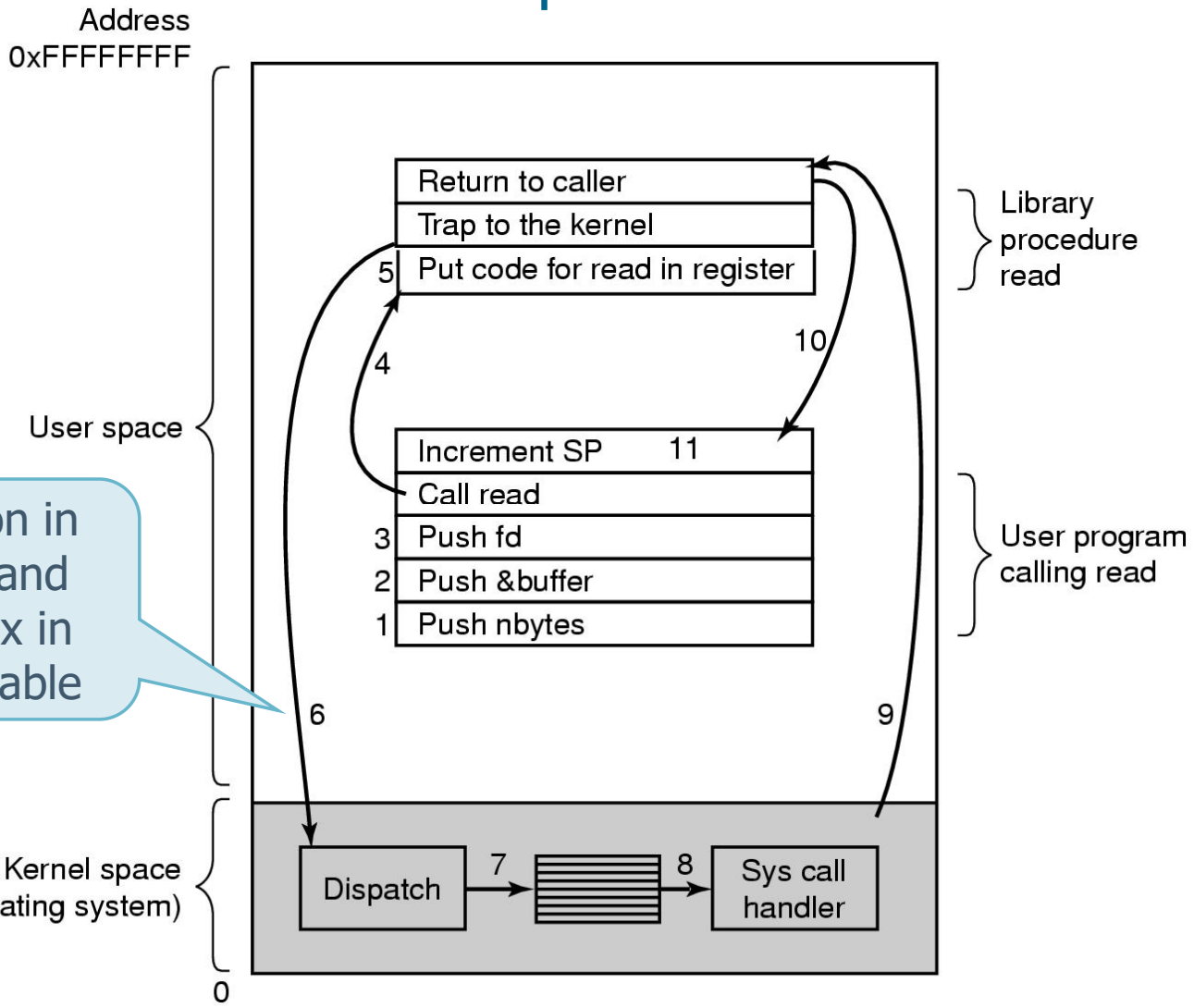
- System call
  - A **system call** causes an exception, and CPU switches to kernel mode (mode bit = 0)
  - The exception, a software interrupts (or **trap**), activates the corresponding service routine
    - It verifies that the parameters are correct and legal, executes the request, and returns control to the instruction following the system call

From user to kernel mode calling a system call



Terminology and basic concepts

- System call `read(fd, buffer, nbytes)` is performed in 11 steps



Trap: transition in kernel mode, and use of an index in an addresses table

Terminology and basic concepts

- Terminology and basic concepts of OS:
- System call: Examples of system calls versus library functions
  - **`printf` function** uses **system call `write`**
  - The allocation function **`malloc`** plausibly call the system call **`sbrk`**
  - Data/time management
    - Only one system call **`time`**
    - The system call **`time`** provides the number of seconds since **`01.01.1970`**
    - Date and time are provided by different functions that produce different result formats

Terminology and basic concepts

- Terminology and basic concepts of OS:
- System call: List of the most common system calls Linux system calls
  - Process management
    - `fork, wait, exec, exit, kill`
  - File management
    - `open, close, read, write, lseek, stat`
  - Directory management
    - `mkdir, rmdir, link, unlink, mount, umount, chdir, chmod`

Terminology and basic concepts

- Terminology and basic concepts of OS:
- Login
  - To login you must provide
    - Username
    - Password
      - Passwords were usually stored in `/etc/passwd`

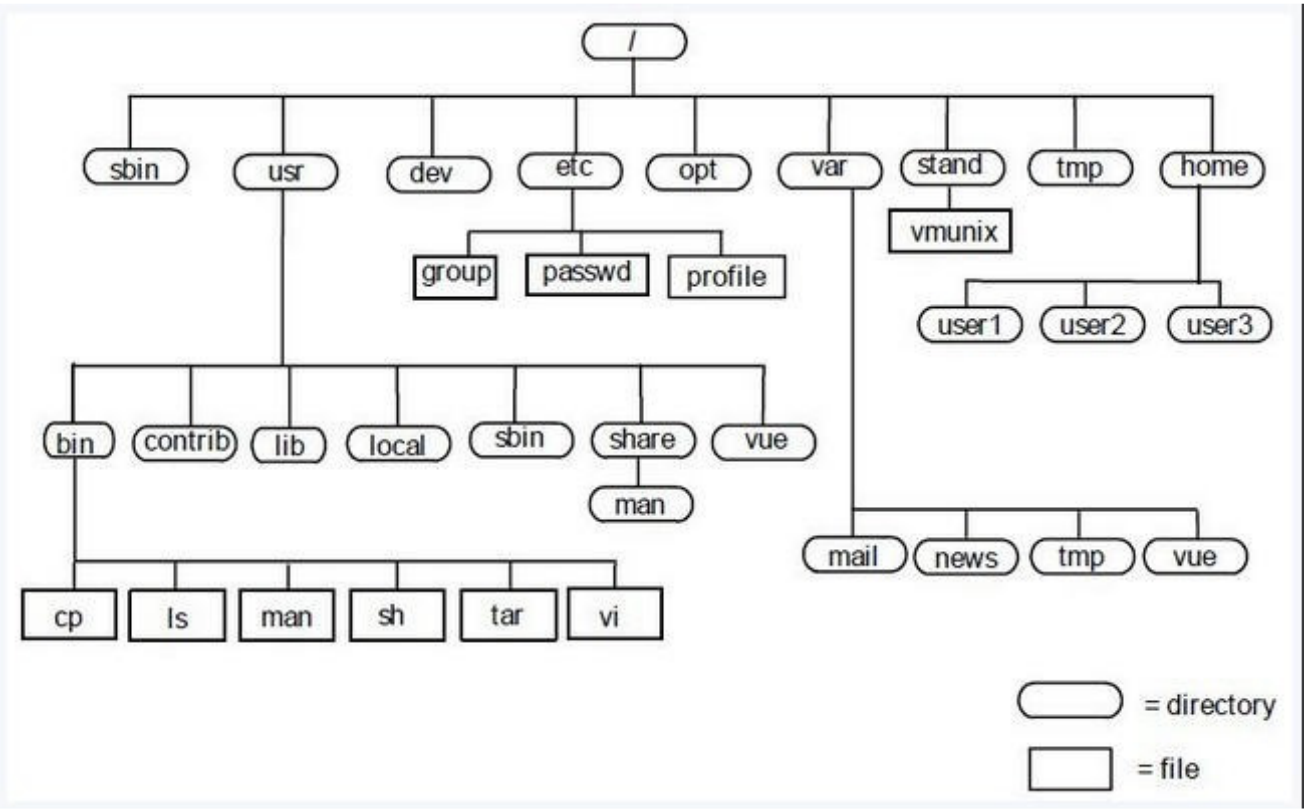
Terminology and basic concepts

- Terminology and basic concepts of OS:
- Shell
  - Command line interpreter (typical UNIX interface)
  - It is not directly part of the OS
    - It is a set of user space commands included in the OS distribution
  - Reads user commands and executes them
  - The commands are typed on the terminal or read from a "script file"
  - There are several shells
    - Bourne shell (**`sh`**)
    - Bourne againg shell (**`bash`**)
    - **`tcsh, ksh, etc.`**

Analyzed in this course

Terminology and basic concepts

- Terminology and basic concepts of OS:
- File System
  - Hierarchical structure of
    - Directories
    - Files



Terminology and basic concepts

- Terminology and basic concepts of OS:
- Filename
  - There are few composition rules and length limitations (i.e., typical 255 bytes)
  - In UNIX the only characters that cannot be used for a filename are
    - slash "/"
    - character "null"

Terminology and basic concepts

- Terminology and basic concepts of OS:
- Pathname
  - A sequence of names separated by slashes '/'
    - Examples: `/usr/bin, /home/scanzio, etc.`
    - `'.'` indicates the current directory
    - `'..'` indicates the parent directory
    - A pathname can be specified as
      - Absolute path (from the **Root directory**)
      - Relative path (from the current **Working directory**)

Terminology and basic concepts

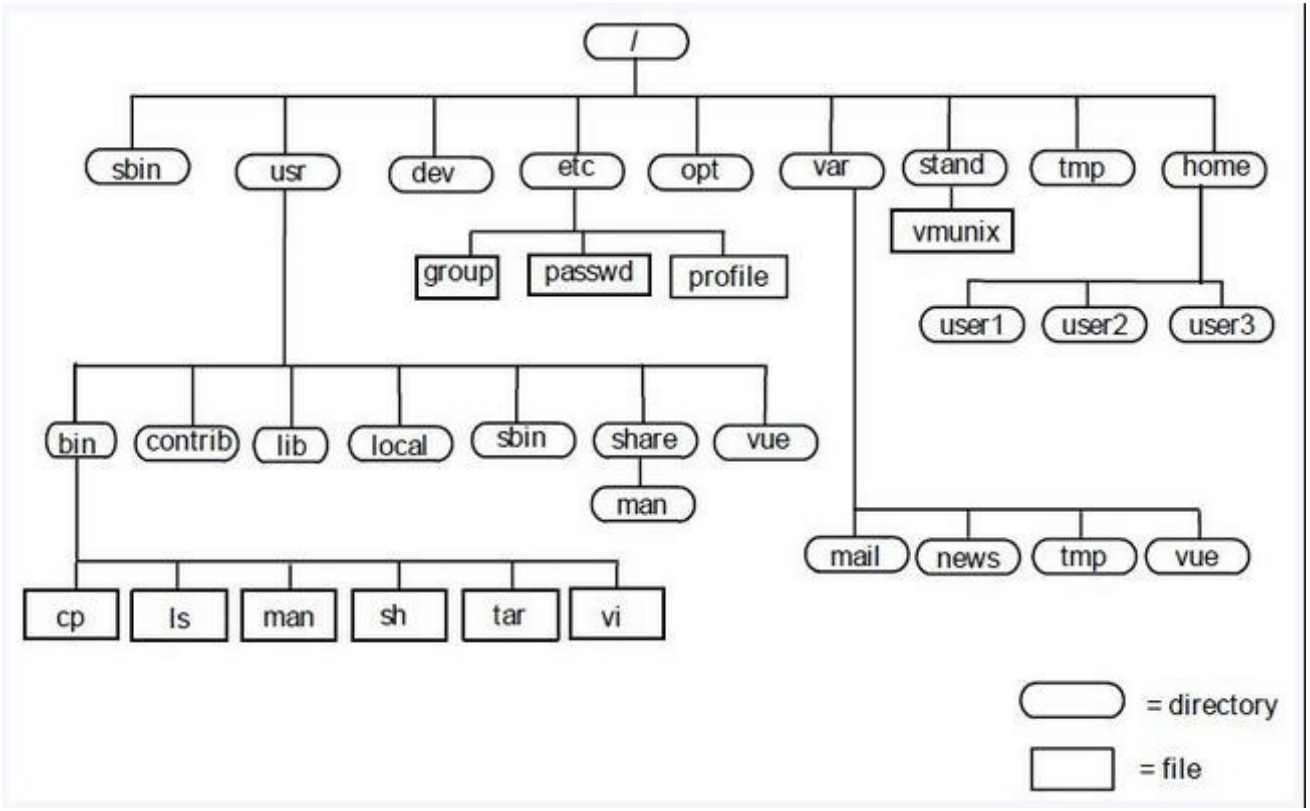
- Terminology and basic concepts of OS:
- Home directory
  - Directory that is accessed after **login** operation
  - It contains files and directories of the **user** that performed the login
  - Identified by tilde **~** in UNIX-like systems
    - The home directory of user **`foo`** is usually **`/home/foo`**, which corresponds to **~** for that user
- Root directory
  - Main directory
  - Root of the directories tree
  - Origin point to interpret absolute pathnames

Terminology and basic concepts

- Terminology and basic concepts of OS:
- Working directory
  - Origin point for interpreting relative pathnames
  - Initially equal to the Home directory (i.e., ~)
  - It can be changed by following the structure of the file system
  - Owned by each process
    - i.e., each process has its own Working directory
  - The reference to the Working directory is implicit, if any pathname is specified
    - directory/file1 is equal to ./directory/file1

Terminology and basic concepts

- Example of access
  - ls (list the content of a directory)
  - cd (for moving in the directory tree)



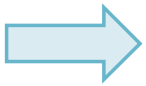
Terminology and basic concepts

- Terminology and basic concepts of OS:
- Program
  - executable file that resides on disk
    - Passive entity
    - Specifies a set of operations to execute a defined task
- Sequential program
    - Its operations are performed in sequence
    - A new instruction starts at the end of the previous one (fetch - decode - execute)
- Concurrent or parallel program
    - Several statements can be executed in parallel
    - An operation can be performed without waiting for the completion of the previous one

Terminology and basic concepts

- An operation is **atomic** if it cannot be interrupted (e.g., by another processors)
  - Example

```
x++;
```



```
add BYPE PTR [0x20], 1
```

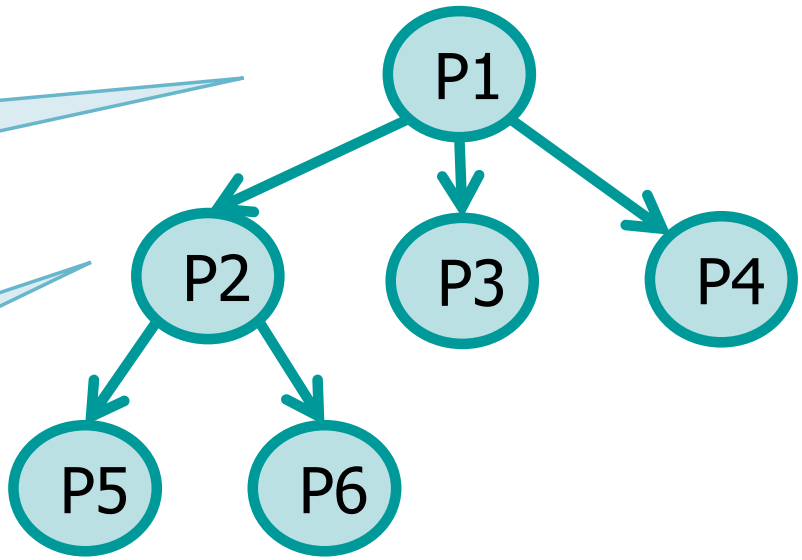
    - To add 1 in a memory location it is common to copy the value into a CPU register
    - If, in the meanwhile, another processor executes the same operation, one of the two increments can be lost
    - Atomicity guarantees that this cannot happen
    - Why not always guarantee atomicity?
      - Atomicity is expensive to guarantee and slows down the entire computer

Terminology and basic concepts

- Terminology and basic concepts of OS:
- Process
  - A running program (which includes a program counter, registers, variables, etc.)
    - Active entity
  - On UNIX systems, each process is characterized by a unique integer (positive) identifier
  - Process tree

P1 creates 3 children processes P2, P3, and P4

P2 creates 2 children processes P5 and P6



Terminology and basic concepts

- Terminology and basic concepts of OS:
- Threads (or light process)
  - A process uses a set of resources
  - A process can have one or more control streams running
  - Each of these streams is a thread
  - Each thread, belong to a process, and shares its resources, but it has its own identifier, and ”life”
    - Each thread is an entity that is scheduled separately

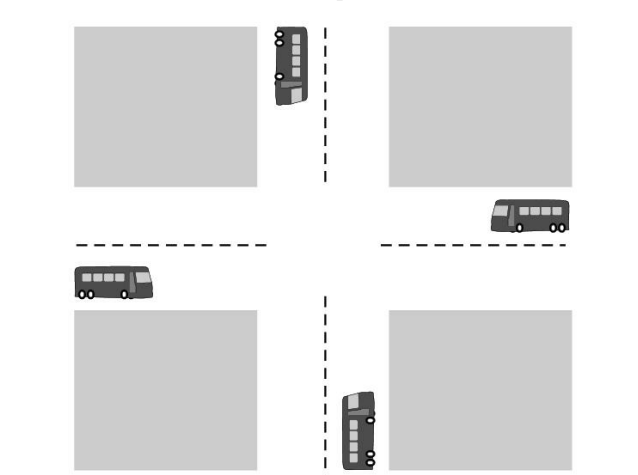
Terminology and basic concepts

- Terminology and basic concepts of OS:
- Pipe
  - A pipe allows a communication **data flow** to be established between two processes
  - Typically, the channel is **half-duplex** (mono-directional)
    - Communication in one direction from P1 to P2 or from P2 to P1

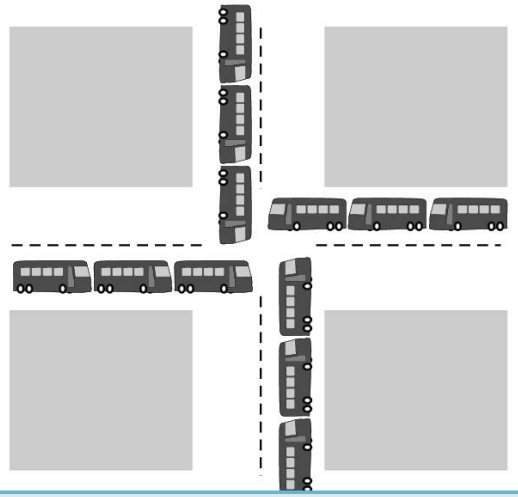
One direction -> half-duplex
Bi-directional -> full-duplex

Terminology and basic concepts

- Terminology and basic concepts of OS:
- Deadlock
  - A deadlock is a situation in which entities (processes) sharing the same resource wait indefinitely an event, which is caused by other entities, resulting that one or more entities are blocked forever.
    - Examples



Potential deadlock



Deadlock

Terminology and basic concepts

- Terminology and basic concepts of OS:
- Livelock (active deadlock)
  - Situation similar to the deadlock in which the entities are not actually blocked but do not make any progress because they are too busy responding to each other to resume work
  - Examples
    - Two people meeting in a corridor and trying to pass, repeatedly move from one side to the other of the corridor
    - Two units perform **polling** (**busy waiting**) to check the status of the other and do not show progress (mutual livelock), but they are not in deadlock because each is doing the poll operation

Terminology and basic concepts

- Terminology and basic concepts of OS:
- Starvation
  - Access to a resource needed for its progress is repeatedly refused to an entity
  - Starvation does not imply deadlock
    - While an entity may starve other can progress
  - Instead, deadlock imply starvation
    - No entity can progress, consequently all are in starvation