Operating Systems

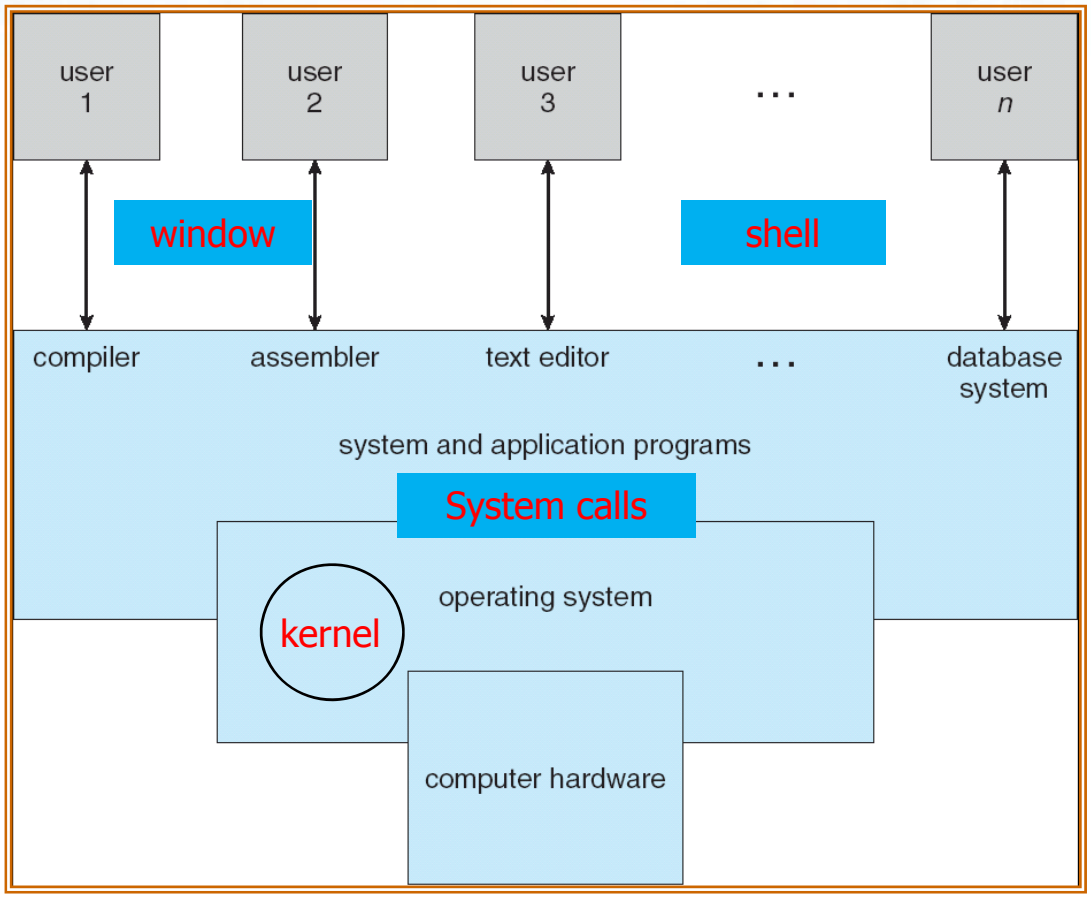
Introduction to Operating Systems (part A)

Stefano Quer and Pietro Laface
Dipartimento di Automatica e Informatica
Politecnico di Torino

Computer System 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

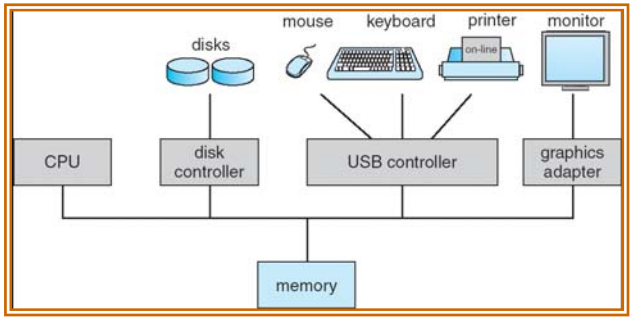
Operating System

- Can be considered a
  - Virtual machine that manages and allocates available resources.
    - Who, when, how much time, how many
  - ” Program” that controls the execution of user programs, and operations of I/O devices

The kernel **is not a program**, but a module that can be considered a big interrupt handler

Modules and Services

- Modules and services of an Operating Systems
  - 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



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

Terminology and basic concepts

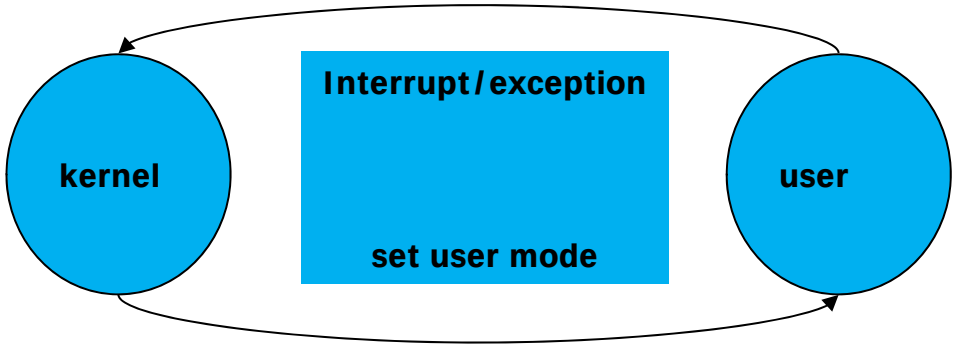
- Kernel
  - Is the core of an Operating System
  - It manages all system resources
  - In particular, it manages memory and processors
  - There are different types of kernel
    - Micro-kernel that provide only the basic functionality
    - Monolithic kernel that provide functionality through the device drivers

Terminology and basic concepts

- 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

- 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

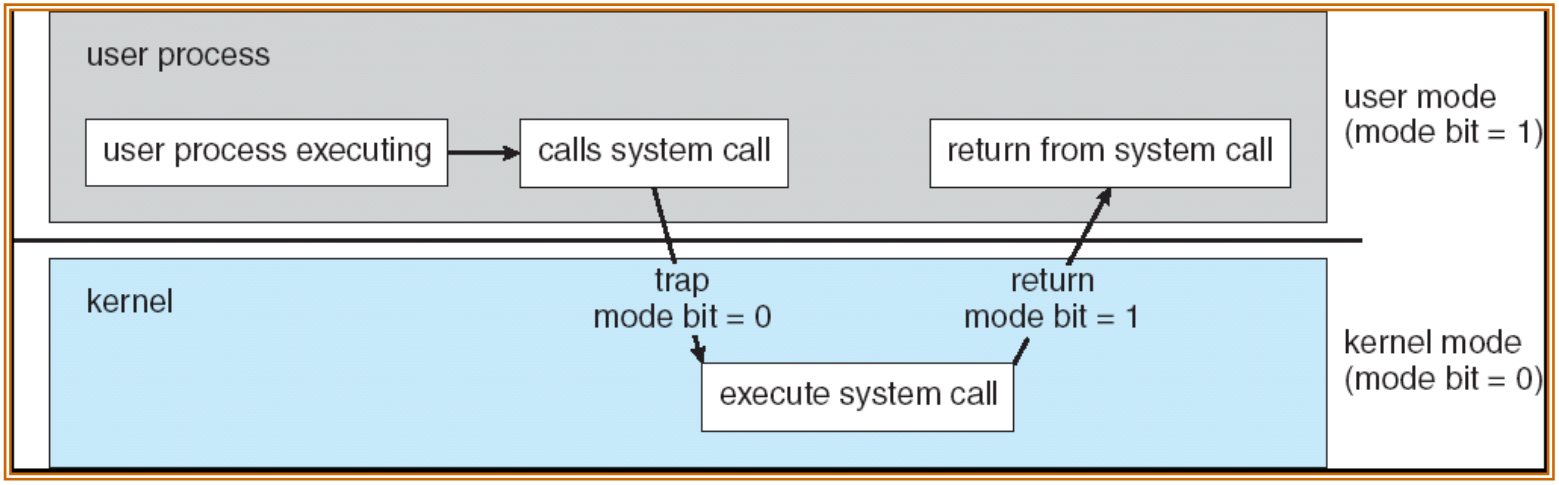
- Kernel protection
  - All I/O instructions are privileged instructions
  - Changing the content of a system register can only be done in kernel mode
    - Dual mode ensures that a user program cannot gain control of the computer in kernel mode
    - Memory protection does not allow a user to write 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

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



Terminology and basic concepts

System call example

- POSIX versus Win32/64 API

UNIX

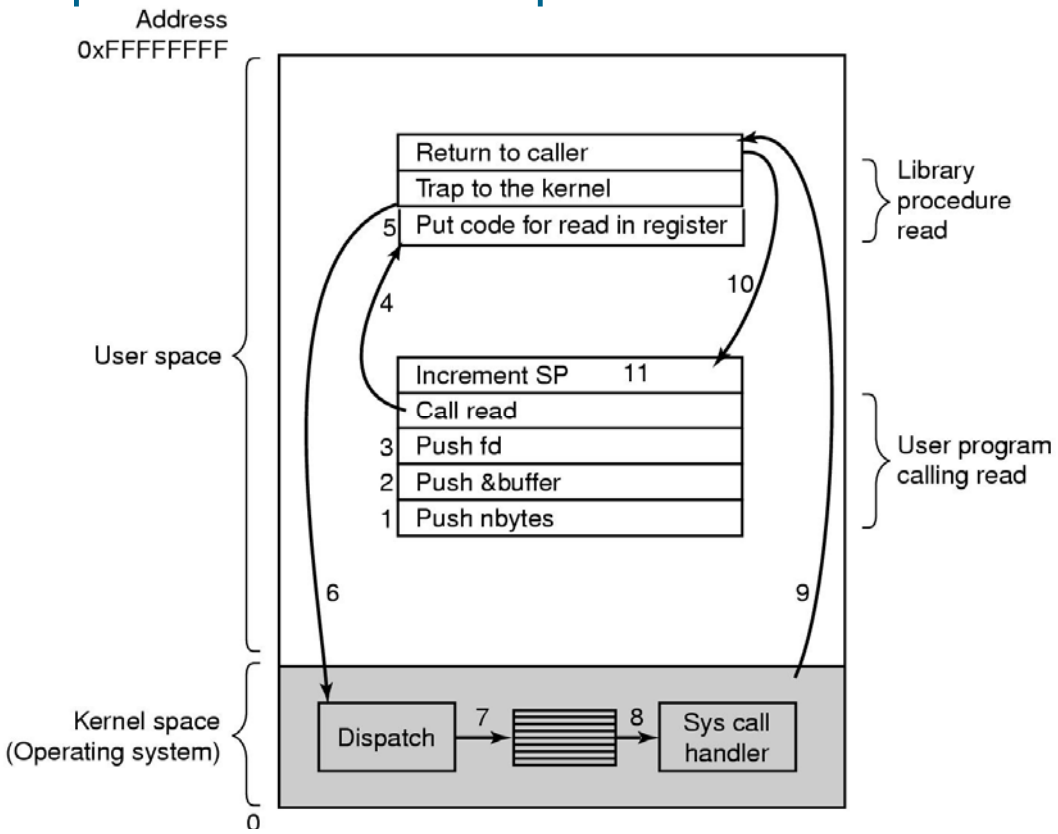
```
int read (int fd, void *buffer, size_t nbytes);
```

Windows

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

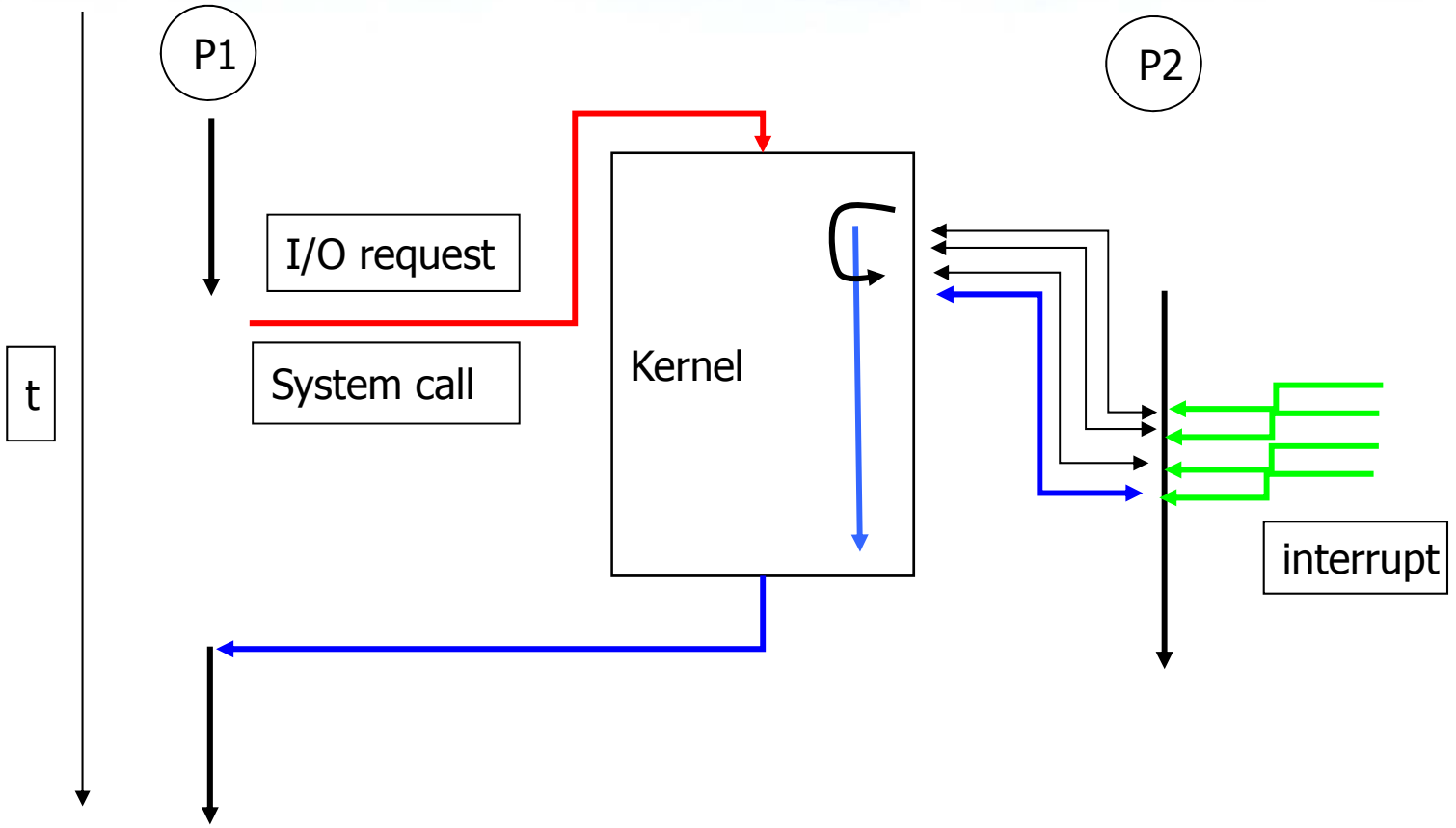
Terminology and basic concepts

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



Terminology and basic concepts

I/O and System Call management



Terminology and basic concepts

- System calls
  - Are the only possible entry-points to the kernel services (approximately 250 in Linux)
  - An important difference exists between functions and system calls
    - A user function call remains on the user space
    - A system call generates an exception to ask a service to the kernel

Terminology and basic concepts

- 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
    - 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

- 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

- Login
  - To login you must provide
    - Username
    - Password
      - Passwords were usually stored in **`/etc/passwd`**
      - An **`x`** character in **`/etc/passwd`** indicates that encrypted password is stored in **`/etc/shadow`**

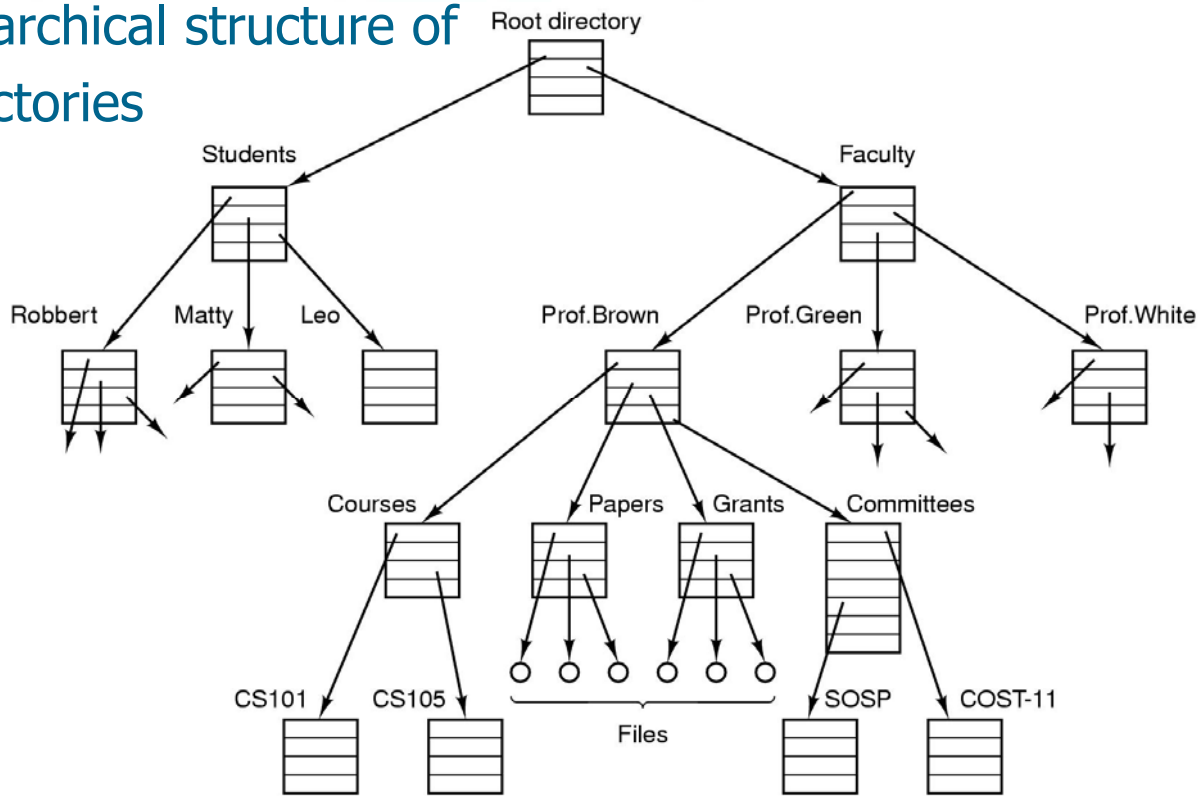
Terminology and basic concepts

- Shell
  - Command line interpreter
  - 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.**

Terminology and basic concepts

File System

- Hierarchical structure of
- Directories
- Files



Terminology and basic concepts

❖ Filename

- ➢ There are composition and length rules
- ➢ In Linux the only characters that cannot be used for a filename are
  - slash '/'
  - character null'

Terminology and basic concepts

- Pathname
  - A sequence of names separated by slashes '/'
    - '.' indicates the current directory
    - '..' indicates the parent directory
    - A pathname can be specified as
      - Absolute path
      - Relative (to the current directory) path

Terminology and basic concepts

- Home directory
  - Directory that is accessed once logged
  - Identified by tilde **~** in UNIX-like systems
  - The home directory of user **`foo`** is usually **`/home/foo`**, which corresponds to **~** for that user

Terminology and basic concepts

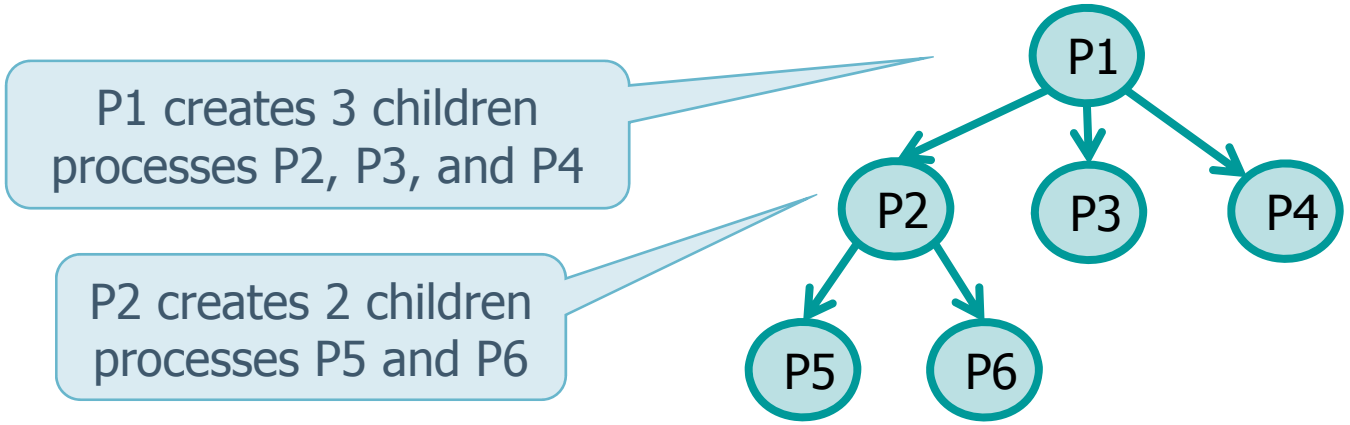
- Working directory
  - Initially equal to the home directory
  - It can be changed by following the structure of the file system
  - Owned by each process
  - It is the origin point for interpreting relative pathnames

Terminology and basic concepts

- Program
  - executable file that resides on disk
    - Passive entity
- 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

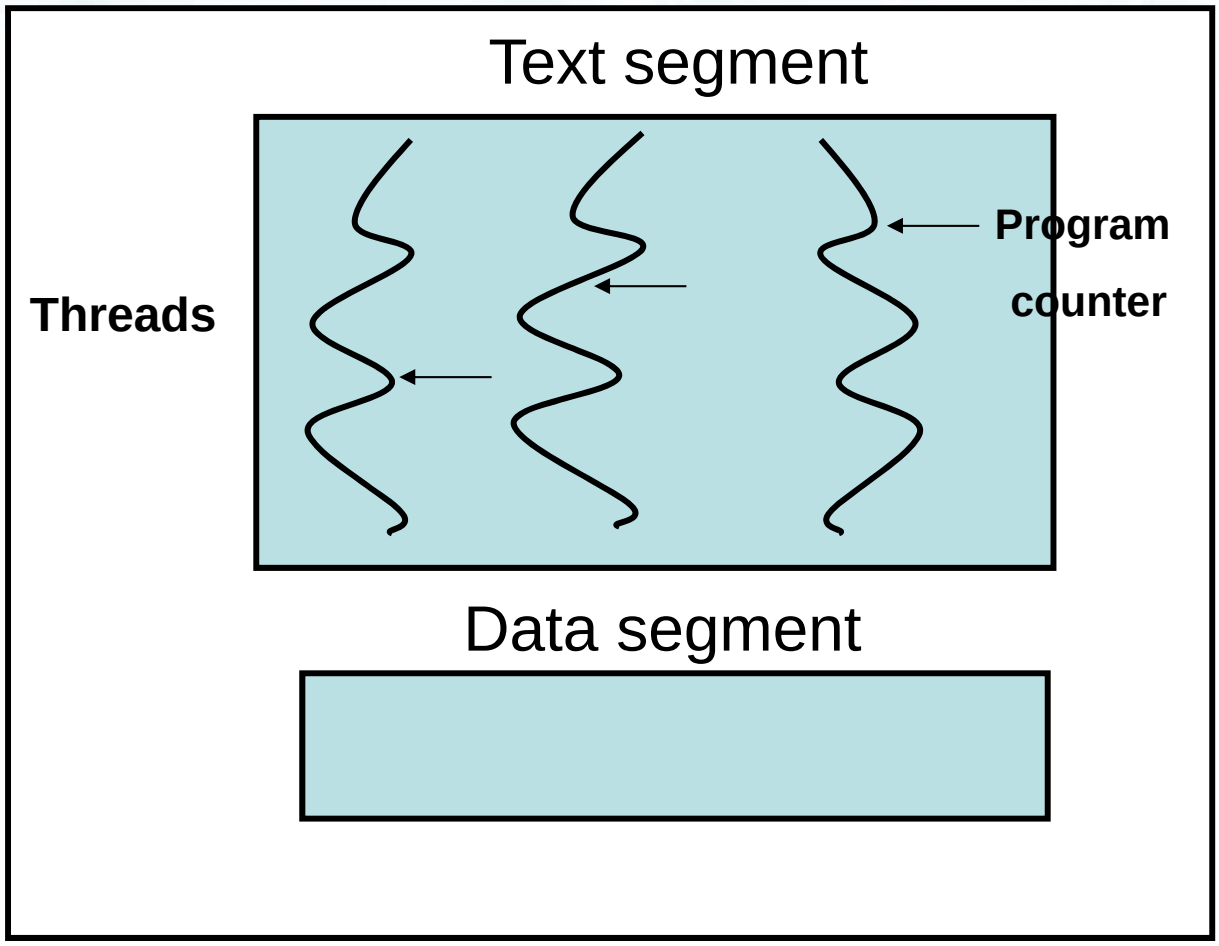
- Process
  - A running program
    - Active entity
  - On UNIX systems, each process is characterized by a unique integer (positive) identifier
  - Process tree



Terminology and basic concepts

- Threads
  - A process uses a set of resources
  - A process can have one or more control streams running
  - Each of these streams is a thread of execution
  - Each thread, belong to a process, and shares its resources, but it has its own identifier, and ”life”

Terminology and basic concepts

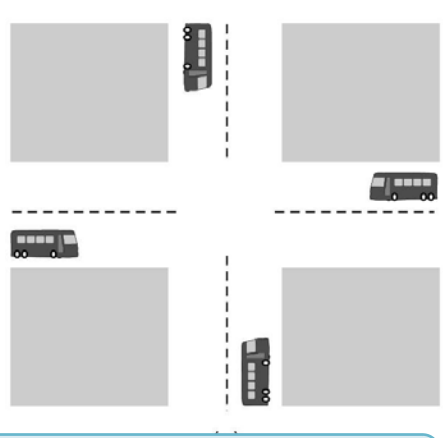


Terminology and basic concepts

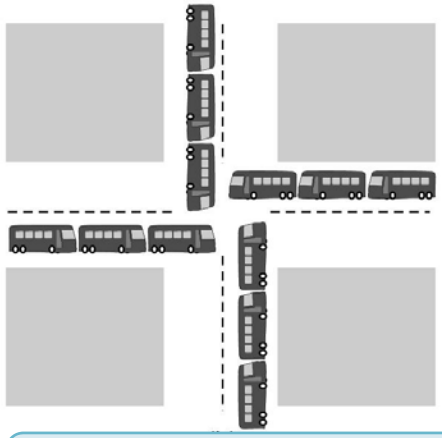
- Pipe
  - A pipe allows a communication **data flow** to be established between two processes
  - Typically the channel is **half-duplex** (mono-directional)
    - Communication in one direction from P1 to P2 or from P2 to P1

Terminology and basic concepts

- Deadlock
  - A deadlock is a situation in which entities (processes) sharing the same resource are preventing each other from accessing the resource, resulting in both programs blocking forever.



Potential deadlock



Deadlock

Terminology and basic concepts

- Livelock
  - 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 are not in deadlock because each is doing the poll operation

Terminology and basic concepts

- 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