Processes

Theoretical Aspects

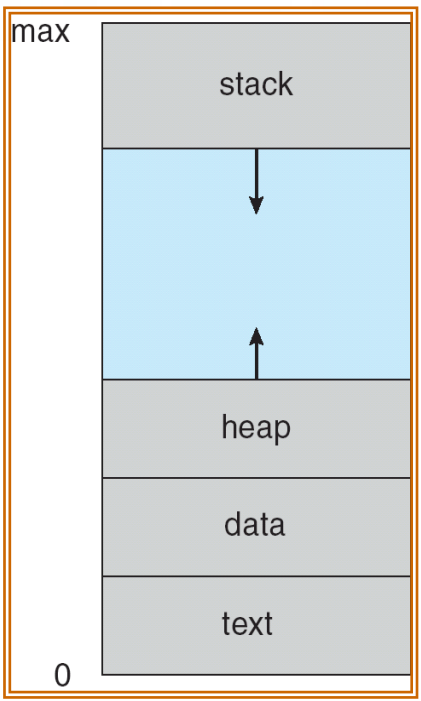
Stefano Quer, Pietro Laface, and Stefano Scanzio
Dipartimento di Automatica e Informatica
Politecnico di Torino
skenz.it/os stefano.scanzio@polito.it

Program

- ***Algorithm:*** a logical procedure that in a finite number of steps solves a problem
- ***Program:*** formal expression of an algorithm by means of a programming language
  - Sequence of code lines
  - Static entity

Process

- ***Process:*** a **sequence** of operations performed by a program in execution on a given set of input data.
  - Dynamic entity
  - Program in execution (running)
    - Text area (executable code)
    - Data area (global variables)
    - Stack (function parameters and local variables)
    - Heap (dynamic variables allocated during the process execution)
    - Registers (Program counter, stack pointer, etc.)

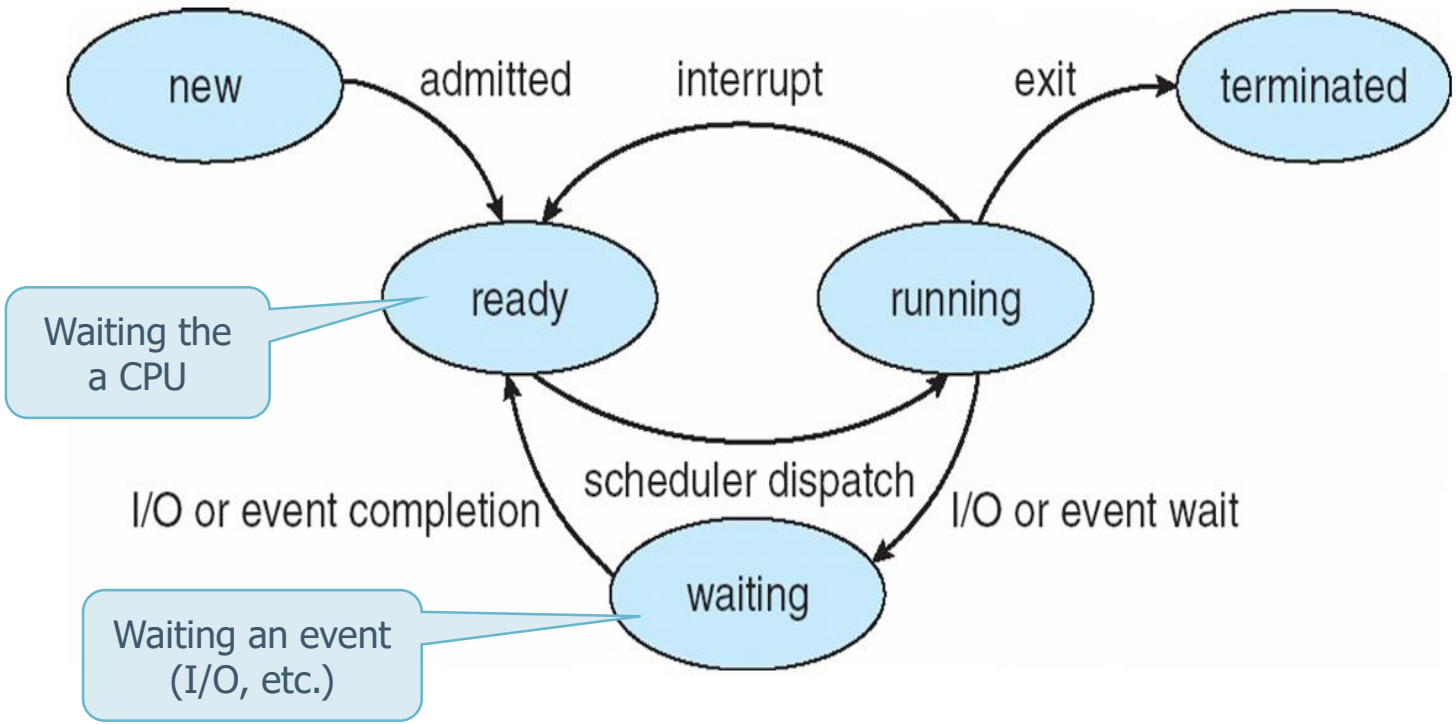


Process state

- During its execution a process change its state
  - **New**: process is created and submitted to the OS
  - **Running**: a CPU is allocated to the process (in execution)
  - **Ready**: logically ready to run, waiting that a CPU is available
  - **Waiting**: for an event or for resources
  - **Terminated**: releases the resource it is using

State diagram

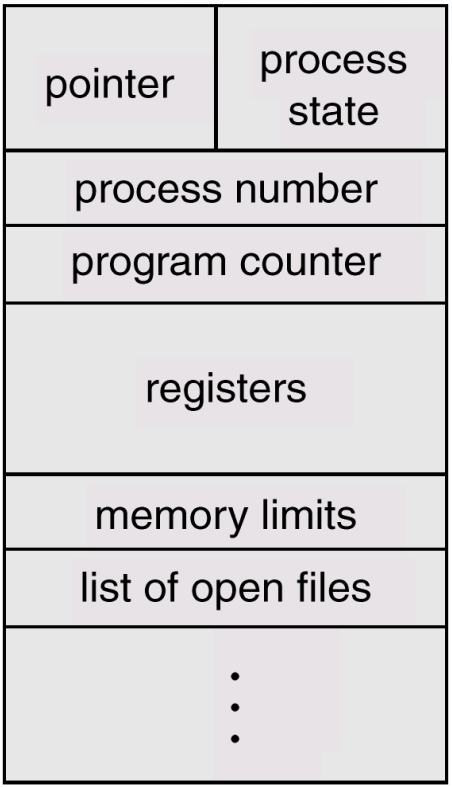
- The possible state evolution of a process is described by a state diagram



Process Control Block (PCB)

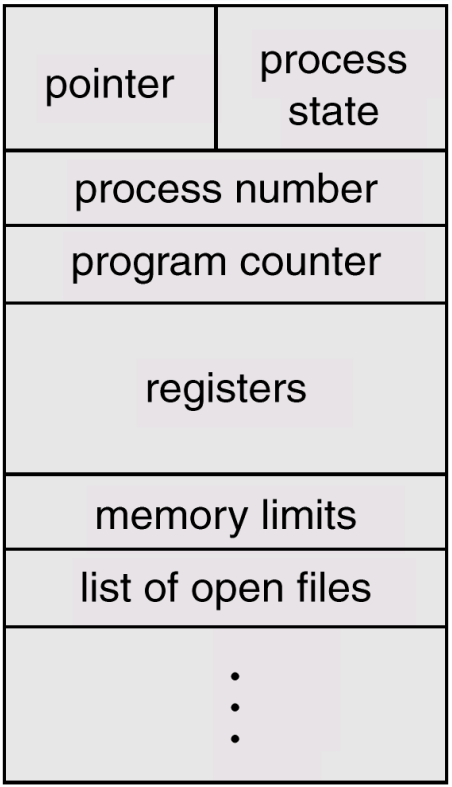
Process Control Block (PCB)

- The kernel stores for each process a set of data, e.g.,
  - The process state
    - New, Ready, Running, Waiting, Terminated
  - Copy of the CPU registers
    - Their number and type is hardware-dependent
  - The program counter
    - Address of the next instruction to be executed



Process Control Block (PCB)

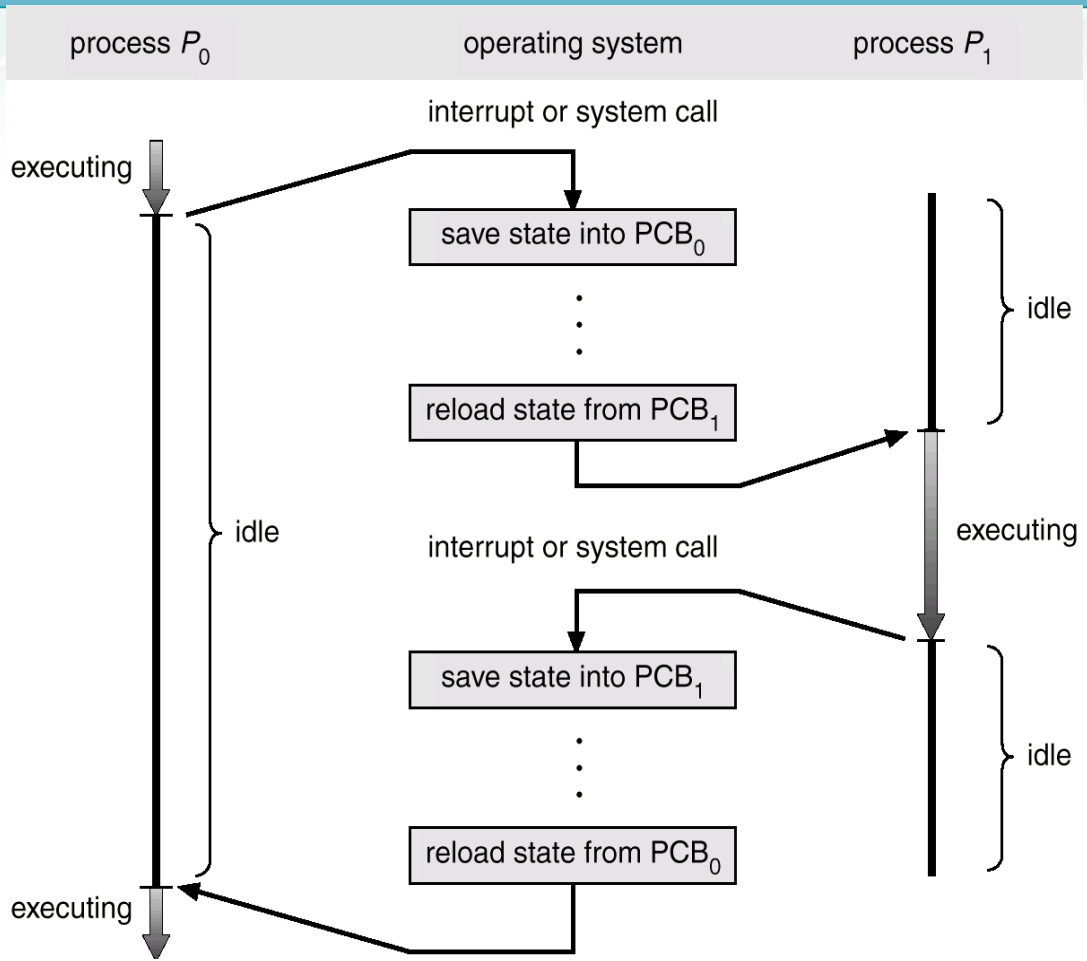
- Data useful for CPU scheduling
  - Priority, pointers to queues, etc.
- Data useful for memory management
  - Base register, Limit register, Segment and paging registers, etc.
- Signal table
  - signal handlers
- Various administration data
  - CPU usage, limits, etc.
- I/O status information
  - I/O device list, open files, etc.



Context switching

- When the CPU is assigned to another process, the kernel
  - Save the state of the running process
  - Load the state previously saved for the new process
- The time devoted to this **context switching** is overhead, i.e., time not directly useful for any process
- The amount of time for context switching is hardware-dependent

Context switching



Process scheduling

- Multiprogramming aims at maximizing the CPU usage by processes
- Processes can be classified as
  - I/O-bound
    - Spend more time for I/0 than for computation
    - Require short CPU service times
  - CPU-bound
    - Spend more time for computation than for I/0
    - Require long CPU service times

Process scheduling

- To maximize CPU usage, the kernel manages the sharing of the CPU among processes by means of a **scheduler**
  - A scheduler selects the next process to run, among the ready ones, according to a strategy that tries to maximize the CPU usage and to satisfy the response time for users
  - Examples
    - After a fork proceeds parent or child
    - When does a process end, which is the next one?
    - When does a process done I/O, which is the next one?

Process scheduling

- Different types of schedulers
  - Long-term scheduler
    - Run less frequently
    - Rescheduling time in the order of seconds/minutes
    - Selects which process image can be inserted in the ready list, and loaded in main memory (swapper)
    - Controls the degree multiprogramming
  - Short-term scheduler
    - Selects the next process to run (context-switching)
    - Run frequently
    - Rescheduling performed every 1 to 10 milliseconds
    - Must be extremely fast

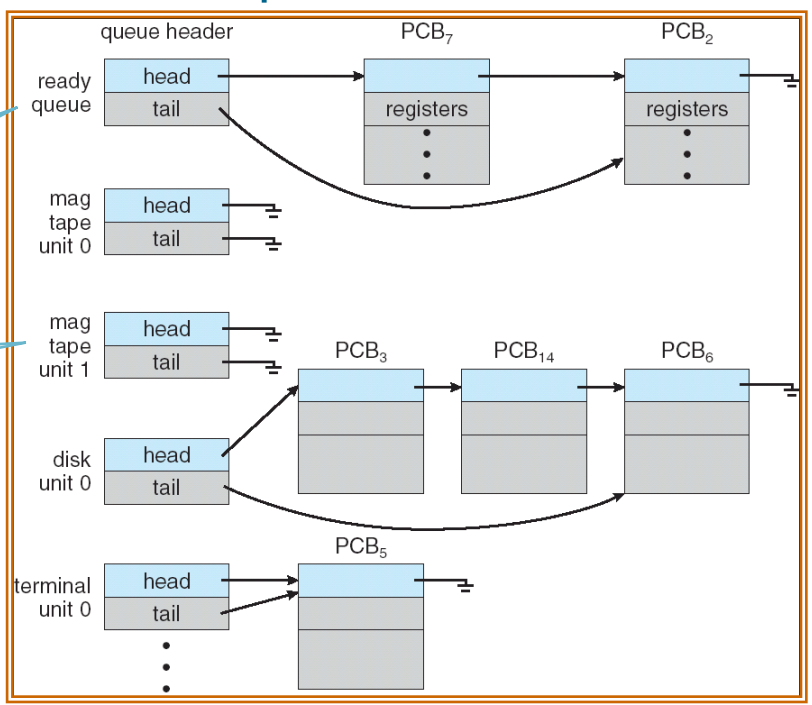
Process scheduling

- A scheduler manages waiting processes by means of **process queues**
  - There are several queues one per device
  - Each queue is a linked list

Ready process queue

I/O waiting process queue

To maximize the efficiency, each device has its own queue



Dynamic analysis of processes

Queuing diagram

- The queuing diagram shows the possible process transitions from one queue to another one
  - Each rectangle represents a queue

Process terminates

Process initially goes on the ready queue

A running process releases the CPU and goes on the ready state due to (I/O completion, interrupt, fork, etc.)