Operating Systems
“Process description and control”

Mathieu Delalandre
University of Tours, Tours city, France
mathieu.delalandre@univ-tours.fr

Lecture available at http://mathieu.delalandre.free.fr/teachings/operating1.html

Process description and control (1)

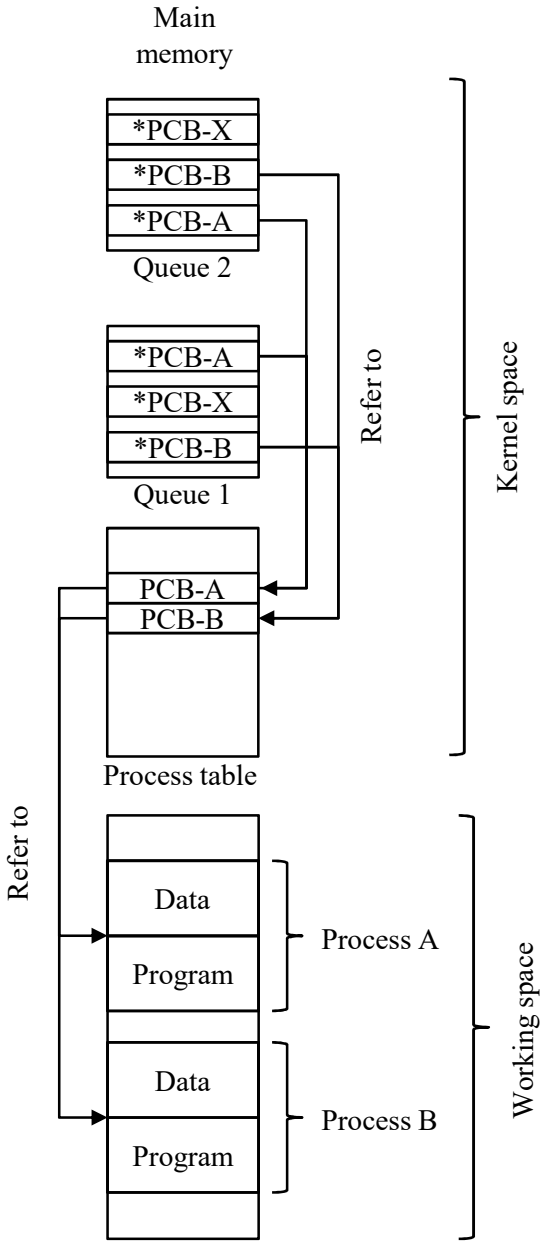
A process(us) is an instance of a computer program that is being executed. It contains the program code and the liked data.

PCB "Process Control Block" (i.e. Task Controlling Block or Task Structure) is a data structure in the operating system kernel containing the information needed to manage a particular process.

Processus Table is an area of memory protected from normal user access, to manage the PCBs, as they contain critical information for processes.

A thread takes part of a process but it has its own program counter, stack and registers. The threads belonging to a process share common code and data.

TCB "Thread Control Block" is a data structure in the operating system kernel containing the information needed to manage a particular thread (PCB look-like).



Process description and control (2)

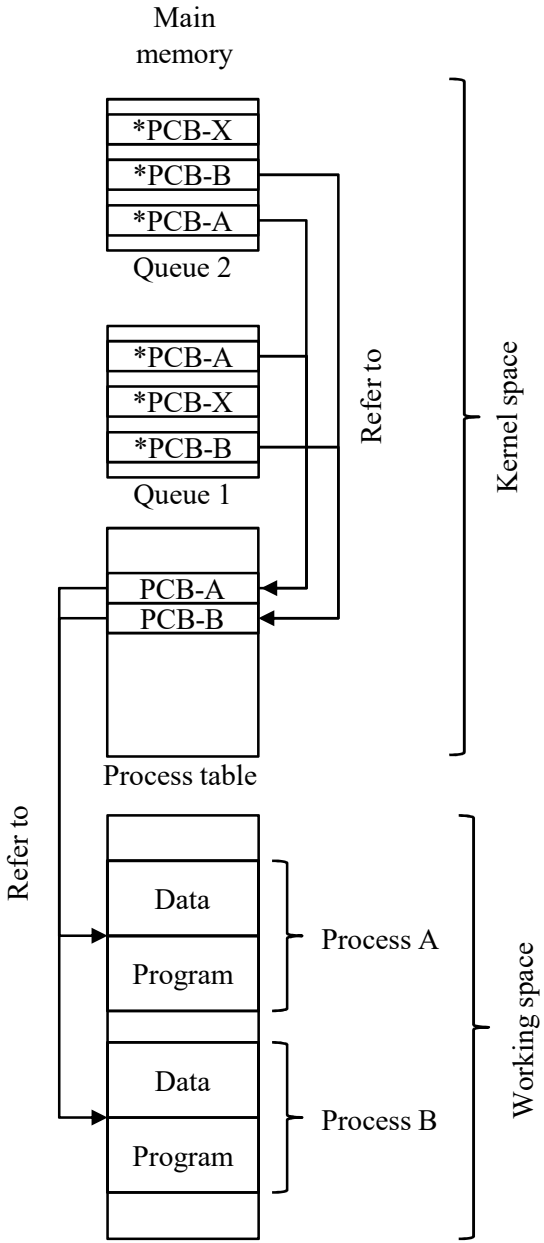
List of frequent data appearing in a PCB

- ✓**Process identifier (pid)** refers the process in the OS.
- ✓**Group data**, hierarchy information (e.g. parents and childs), type of process and group memberships.
- ✓**CPU-scheduling information** e.g. process priority, pointers to scheduling queues, etc.
- ✓**Process state** e.g. ready, running, waiting, terminated, etc.
- ✓**Program counter (PC)** refers the current execution state of the process.
- ✓**CPU registers** correspond to the current state of the CPU.
- ✓**Security attributes** refer the owner or set of permissions (allowable operations) of the process.
- ✓**Accounting information** e.g. start time, end time, amount of CPU used, real-time used, etc.
- ✓**Etc.**

related to process management

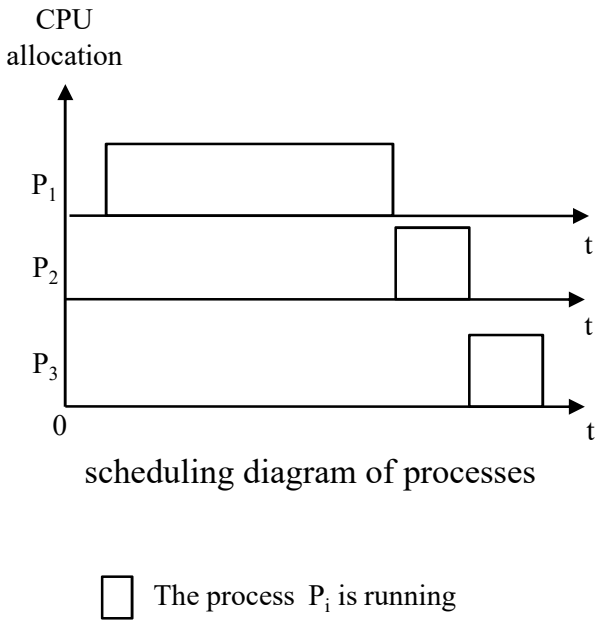
- ✓**Memory management information** includes page and segment tables on the executable code, call stack (to keep track of active subroutines and/or other events), etc.
- ✓**Operating system descriptors** refer to the resources that are allocated to the process, such as files, devices, other data sources.
- ✓**Etc.**

related to data management



Process description and control (3)

Multitasking (i.e. multiprogramming) is a method by which multiple tasks share common processing resources such as a CPU. With a single CPU, only one task can run at any time. Multitasking solves the problem by scheduling the tasks i.e. which task must run on the CPU, and which task must wait.



scheduling diagram of processes

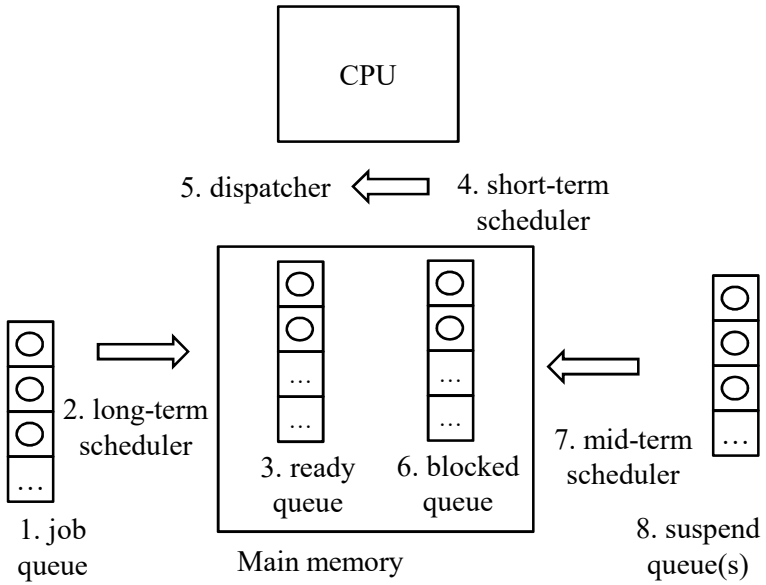
The process P_i is running

Process description and control (4)

Scheduling refers to the way processes are assigned to run on the CPU. The aim of scheduling is to assign processes to be executed by the processor over the time, in a way that meets objectives of the system, such as the response time, throughput and processor efficiency.

In many systems, the scheduling activity is broken into three separate functions: long, medium and short-term scheduling.

Scheduling affects the performance of the system because it determines which processes will wait and which will progress. Scheduling is a matter of managing queues to minimize queuing delay and to optimize performance in a queuing environment.



Process description and control (5)

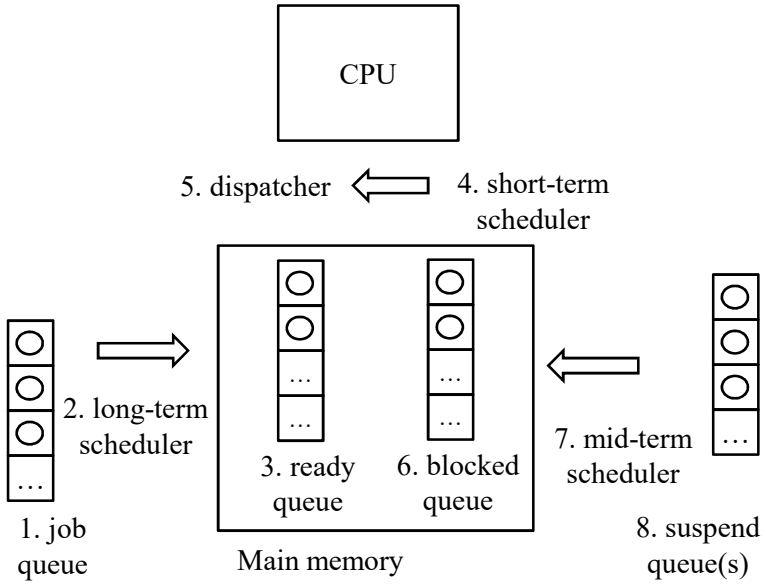
1. Job queue stores processes to enter in the system, they are put into the job queue. The job queue contains the list of processes to create.

2. Long term scheduler (admission scheduler) decides which processes are to be admitted to the ready queue, they are then created and loaded into the main memory.

3. Ready queue is a data structure to keep in the main memory the processes that are in a ready state.

4. Short-term scheduler (i.e. CPU scheduler) decides which of the ready, in-memory processes, are to be executed (allocated to the CPU) following a clock interrupt, an I/O interrupt, an operating system call or ayn other form of signal.

5. Dispatcher gives the control of the CPU to the process selected by the short-term scheduler.

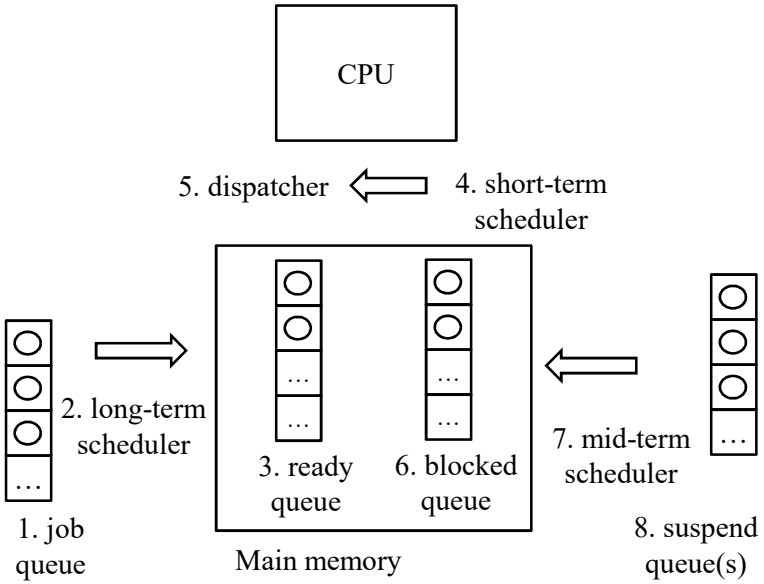


Process description and control (6)

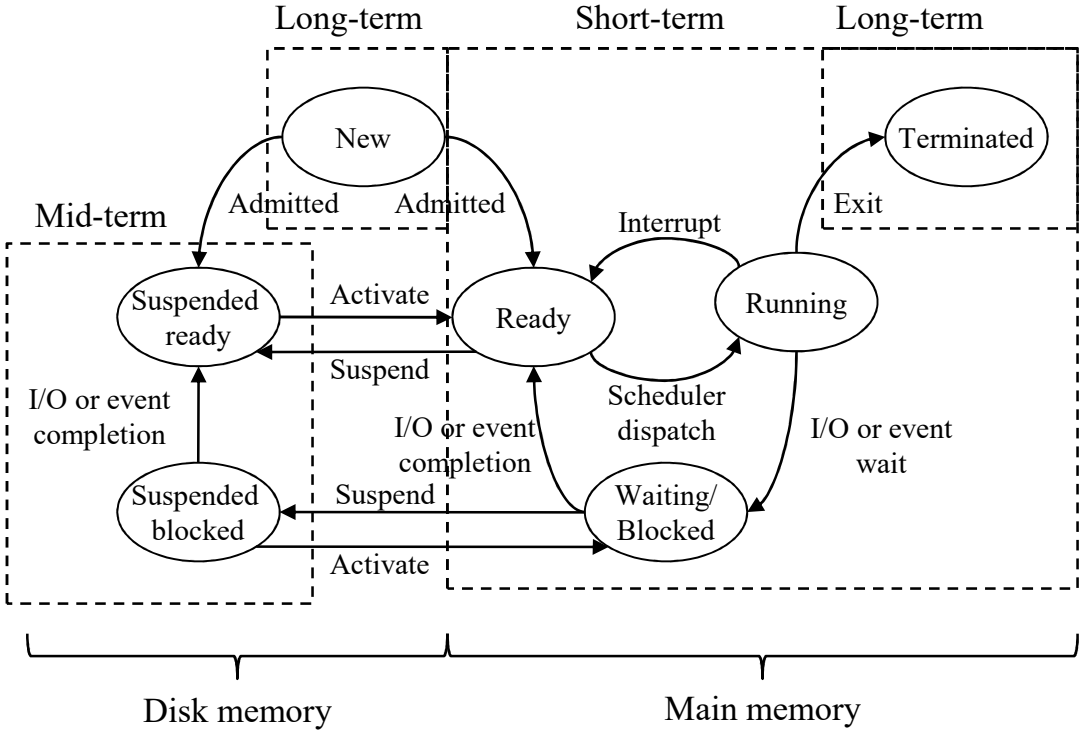
6. Waiting/Blocked queue is a data structure to keep in the main memory the processes in a blocked state.

7. Mid-term scheduler removes processes from the main memory (if full) and places them on a secondary memory (such as a disk drive) and vice-versa.

8. Blocked suspend queue(s) contain lists of processes moved to the disk (i.e. swapping). Two queues are usually managed, related to the processes in a suspended-blocked or a suspended-ready state.



Process description and control (7)



As a process executes, it changes its state. The state of a process is defined in part by the current activity of the process.

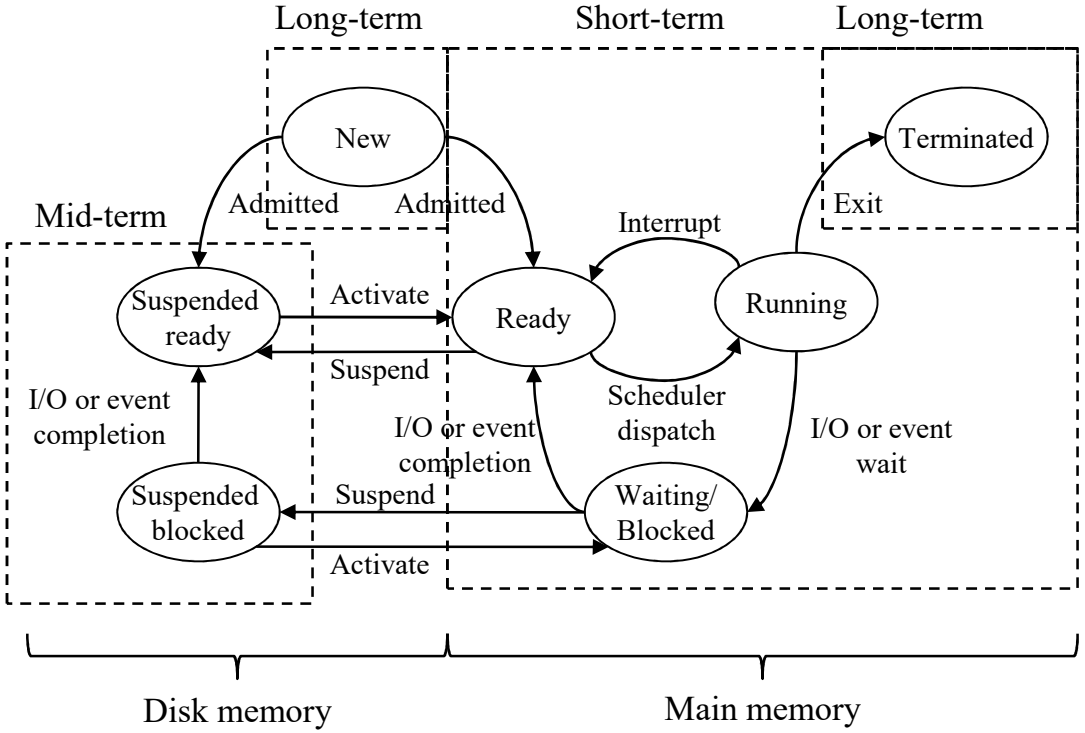
✓**New:** in this state, the process awaits for an admission to the ready state. This admission will be approved or delayed by a long-term, or admission, scheduler.

✓**Ready**: a ready process has been loaded into the main memory and the ready queue and is awaiting for an execution on the CPU (to be loaded into the CPU by the dispatcher following the decision of the short-term scheduler).

✓**Running:** process is being executed by CPU.

✓**Terminated:** a process may be terminated, either from the running state by completing its execution or by explicitly being killed. If a process is not removed from the memory, this state may also be called zombie.

Process description and control (8)



As a process executes, it changes its state. The state of a process is defined in part by the current activity of the process.

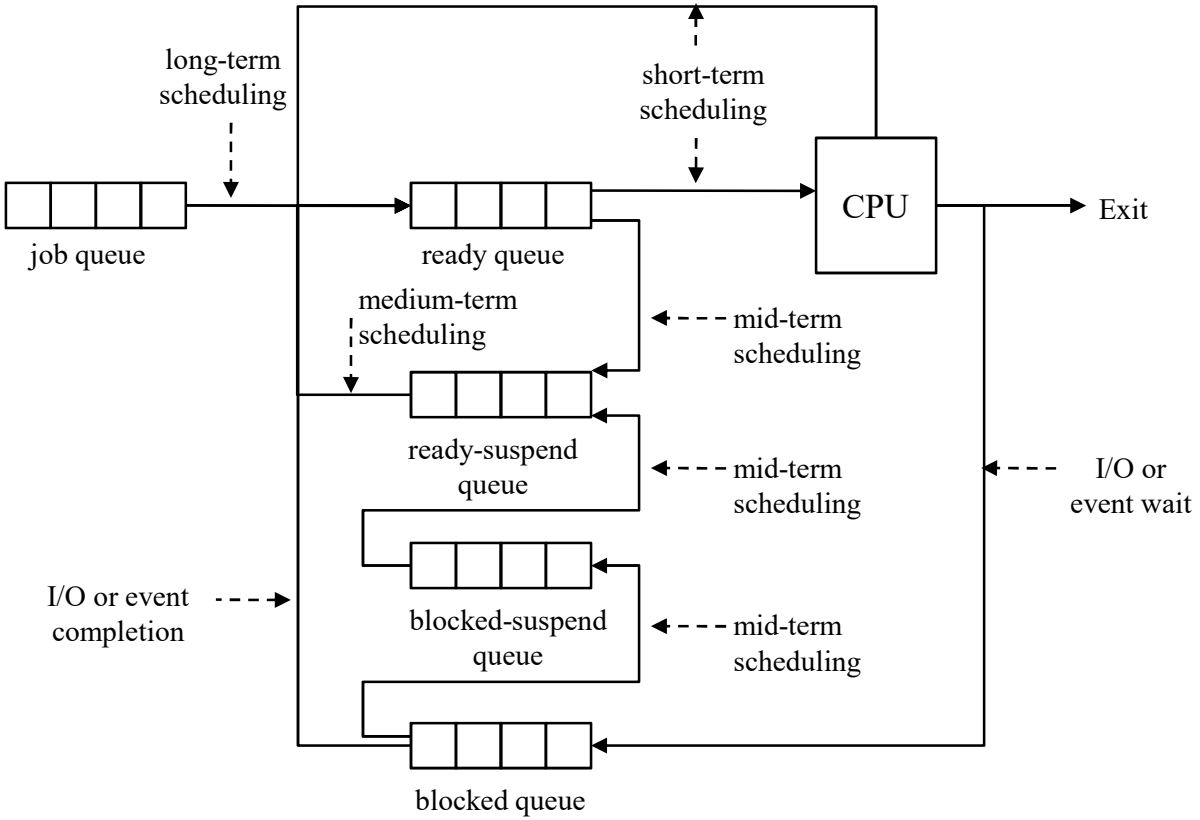
✓**Waiting/Blocked:** a process that cannot execute until some events occurs, such as the completion of an I/O operation or a signal. Every blocked process is moved to the blocked queue.

✓**Suspended blocked:** a process is put in the disk memory by the mid-term scheduler (i.e. swapping out).

✓**Suspended ready:** a process is ready to be loaded from the disk to the main memory (i.e. swapping in).

Process description and control (9)

Queuing diagram for scheduling shows the queues involved in the state transitions of processes.
Rq. For simplicity, this diagram shows new processes going directly to the ready state without the option of either the ready state or either the ready/suspend state.



Process description and control (10)

✓**New → Ready:** the OS will move a process from the new state to the ready state (i.e. from the job queue to the ready queue) when it is prepared to take an additional process. Most of the systems set some limits based on the number of existing processes in memory.

✓**Ready → Running:** when it is time to select a process to run, the OS chooses one of the processes in the ready state. This is the job of the scheduler.

✓**Running → Terminated:** the currently running process is terminated by the OS if the process indicates that it has completed, or if it aborts.

✓**Running → Ready:** the most common reasons for this transition are

(1) in the case of a preemptive scheduling, the OS assigns different levels of priority to different processes; thus a process A can preempt a process B and B will go to the ready state and shift to the ready queue.
(2) the running process has reached the maximum allowable time for an uninterrupted execution (all the multiprogramming OS impose this type of time discipline).
(3) a process may voluntarily release the control of the processor (e.g. a background process that performs some accounting or maintenance functions periodically).

Process description and control (11)

✓**Running → Blocked:** a process is put in the blocked state (and moves to the blocked queue) if
(1) it requests something (i.e. a resource) for which it must wait such as a file, a shared section, etc. that is not immediately available (e.g. a down operation on a Mutex).
(2) it requests a service to the OS that is not prepared to perform immediately. A request to the OS is usually in the form of a system service call; that is; a call from the running program to a procedure that is part of the OS.

Blocked → Running: a process in the blocked state is moved to the ready state (and moved to the ready queue) when the event for which it has been waiting occurs (e.g. up operation on a Mutex, system call return, etc.).

Ready → Terminated: this transition is not shown on the state and queuing diagrams, in some systems, a parent may terminate a child process at any time. Also, if a parent terminates, all child processes associated wit that parent may terminate.

Terminated → Ready: this transition has nosense.