Part II
Process Management
Chapter 3: Processes

Process Management

- **The Concept of a Process**
- **Process Scheduling**
- **Operations on Processes**
- **Cooperating Processes**
- **Interprocess Communication**

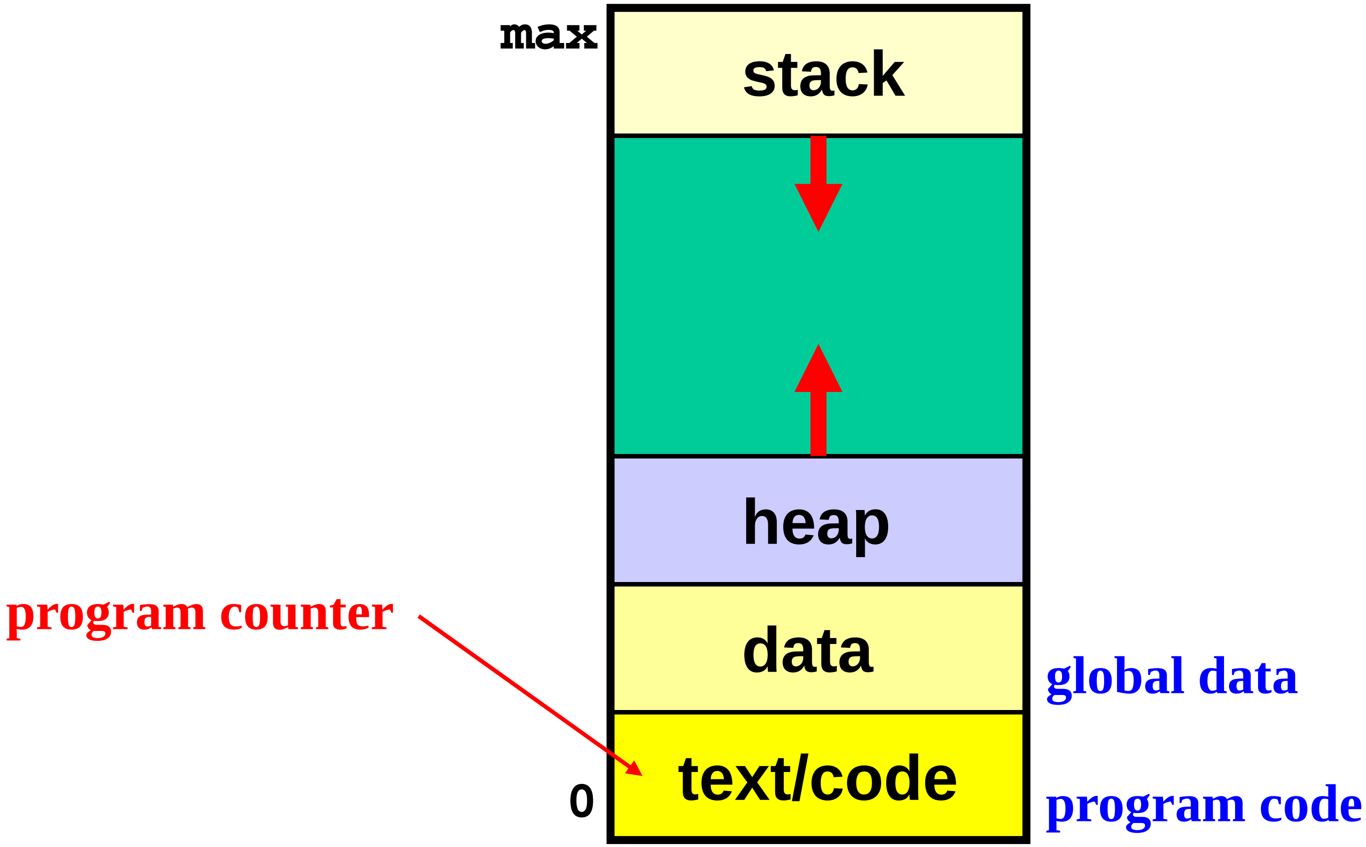
The Concept of a Process

- **What is a process?**
- **Process states**
- **Process control block**
- **Threads**

Process: Definition 1/2

- **When the OS runs a program (*i.e.*, a binary executable), this program is loaded into memory and the control is transferred to this program's first instruction. Then, the program starts to run.**
- ***A process is a program in execution*.**
- **A process is more than a program, because the former has a *program counter*, *stack*, *data section* and so on (*i.e.*, the runtime stuffs).**
- **Moreover, multiple processes may be associated with one program (*e.g.*, run the same program multiple times at the same time).**

Process: Definition 2/2

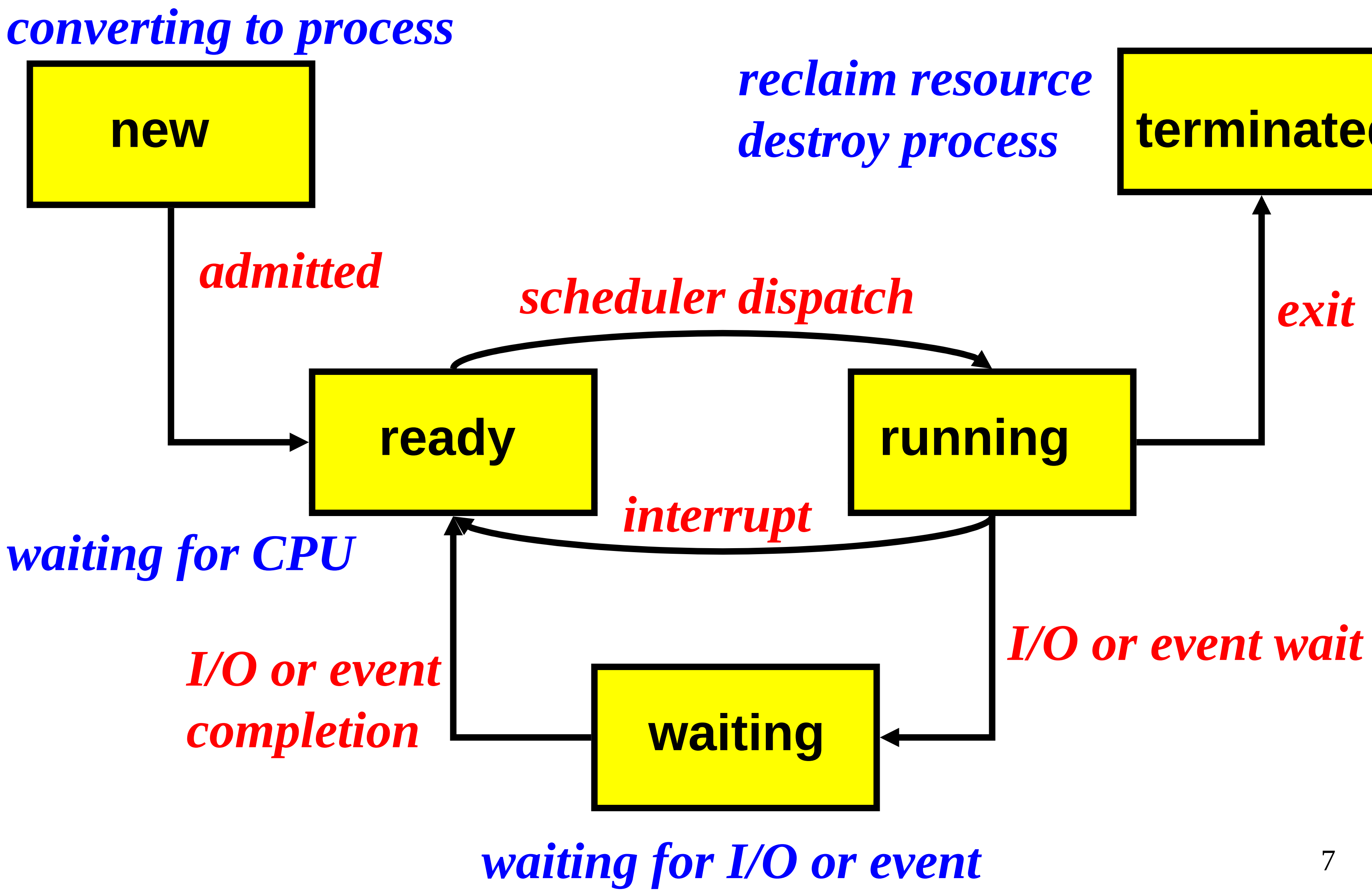


Process States

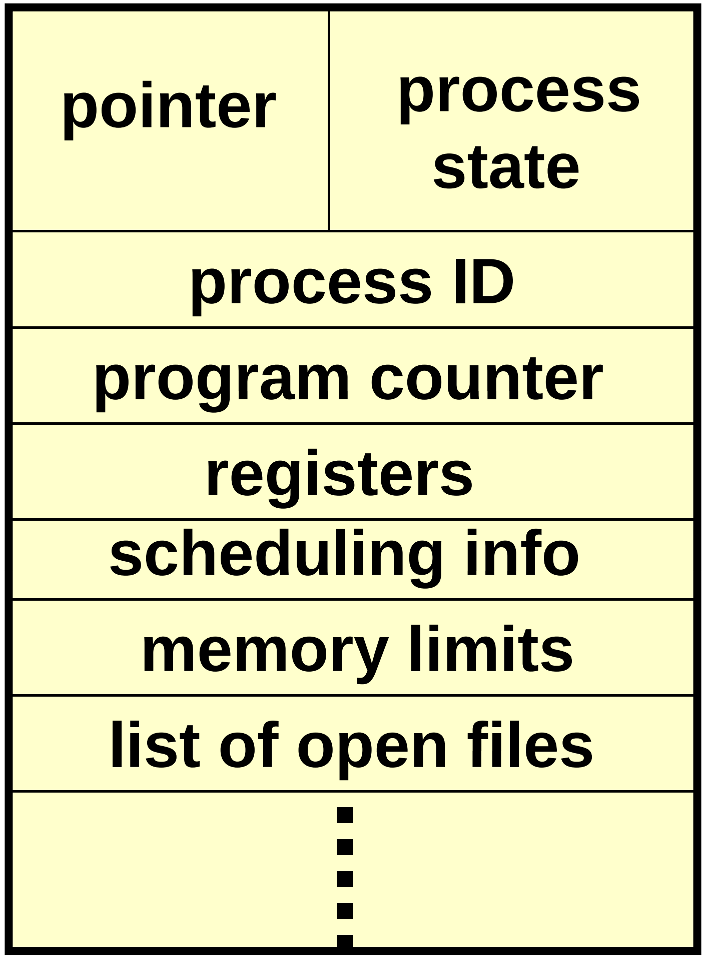
At any moment, a process can be in one of the five states: new, running, waiting, ready and terminated.

- ***New*: The process is being created**
- ***Running*: The process is executing on a CPU**
- ***Waiting*: The process is waiting for some event to occur (*e.g.*, waiting for I/O completion)**
- ***Ready*: The process is waiting to be assigned to a processor.**
- ***Terminated*: The process has finished execution.**

Process State Diagram



Process Control Block (PCB)

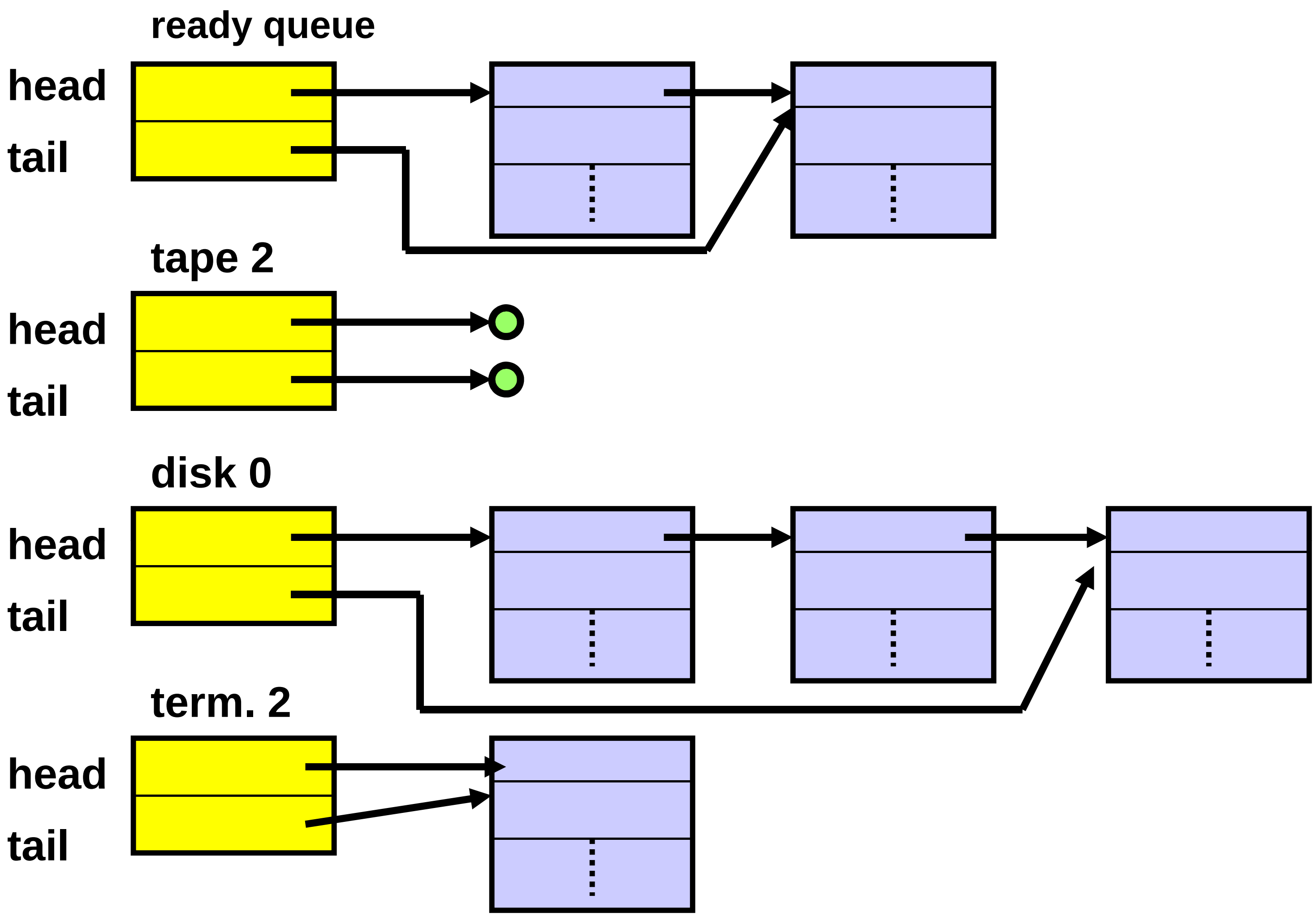


- **Each process has a number, the *process ID*.**
- **Process info are stored in a table, the *process control block* (PCB).**
- **These PCBs are chained into a number of lists. For example, all processes in the ready state are in the *ready queue*.**

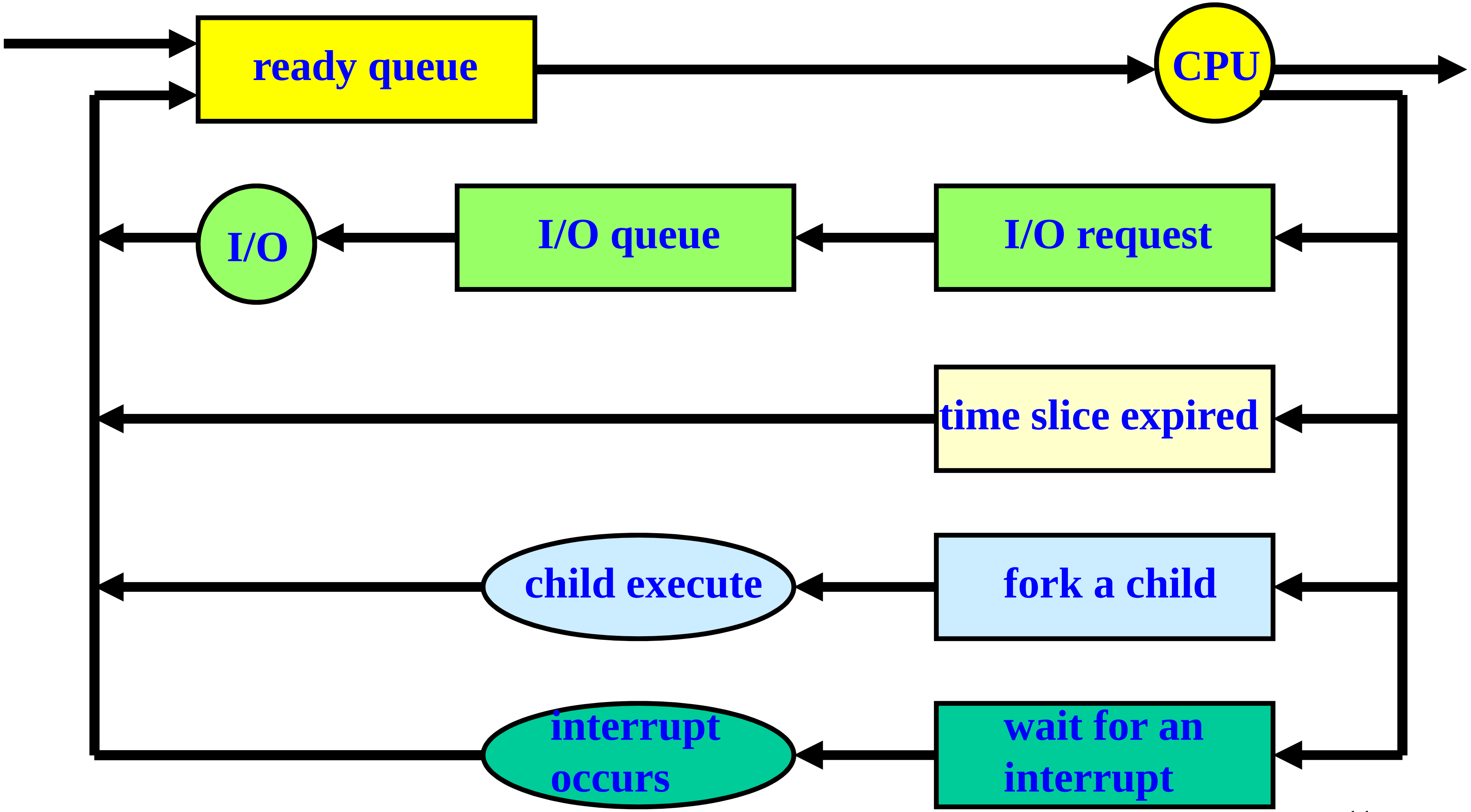
Process Scheduling

- **Since the number of processes is always larger than the number of available CPUs, the OS must maintain *maximum CPU utilization*.**
- **To determine which process can do what, processes are chained into a number of *scheduling queues*.**
- **For example, in addition to the ready queue, each event may have its own scheduling queue (*i.e.*, waiting queue).**

Various Scheduling Queues



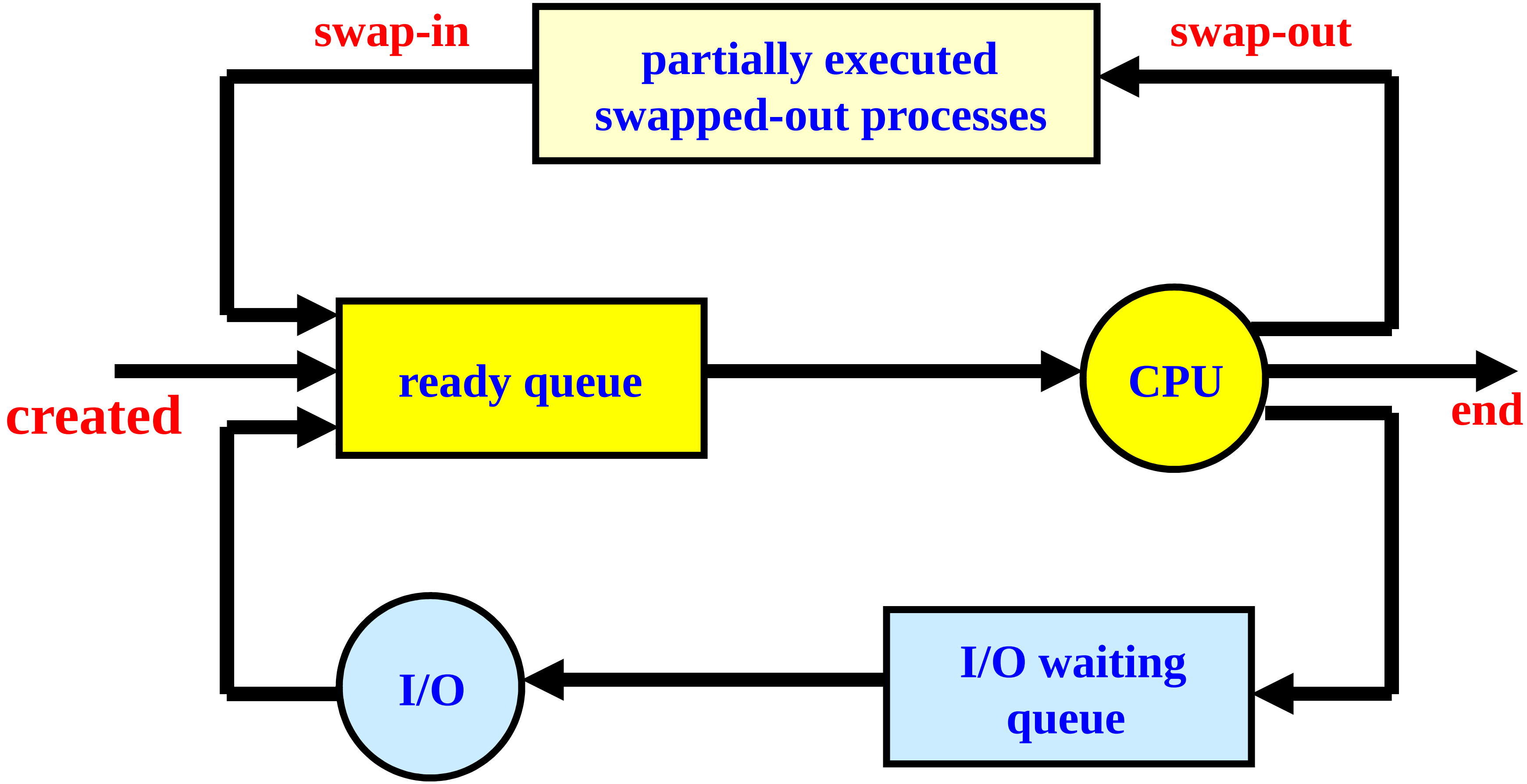
Scheduling Queuing Diagram



Schedulers

- **There are three types of schedulers**
  - ***Long-Term (Job) Scheduler*: selects jobs and loads them into the system for execution (the new state). Executes less frequently.**
  - ***Short-Term (CPU) scheduler*: selects from among the processes (in the ready queue), and allocates the CPU to one of them. Executes very frequently.**
  - ***Medium-Term Scheduler*: does swapping to balance system load.**

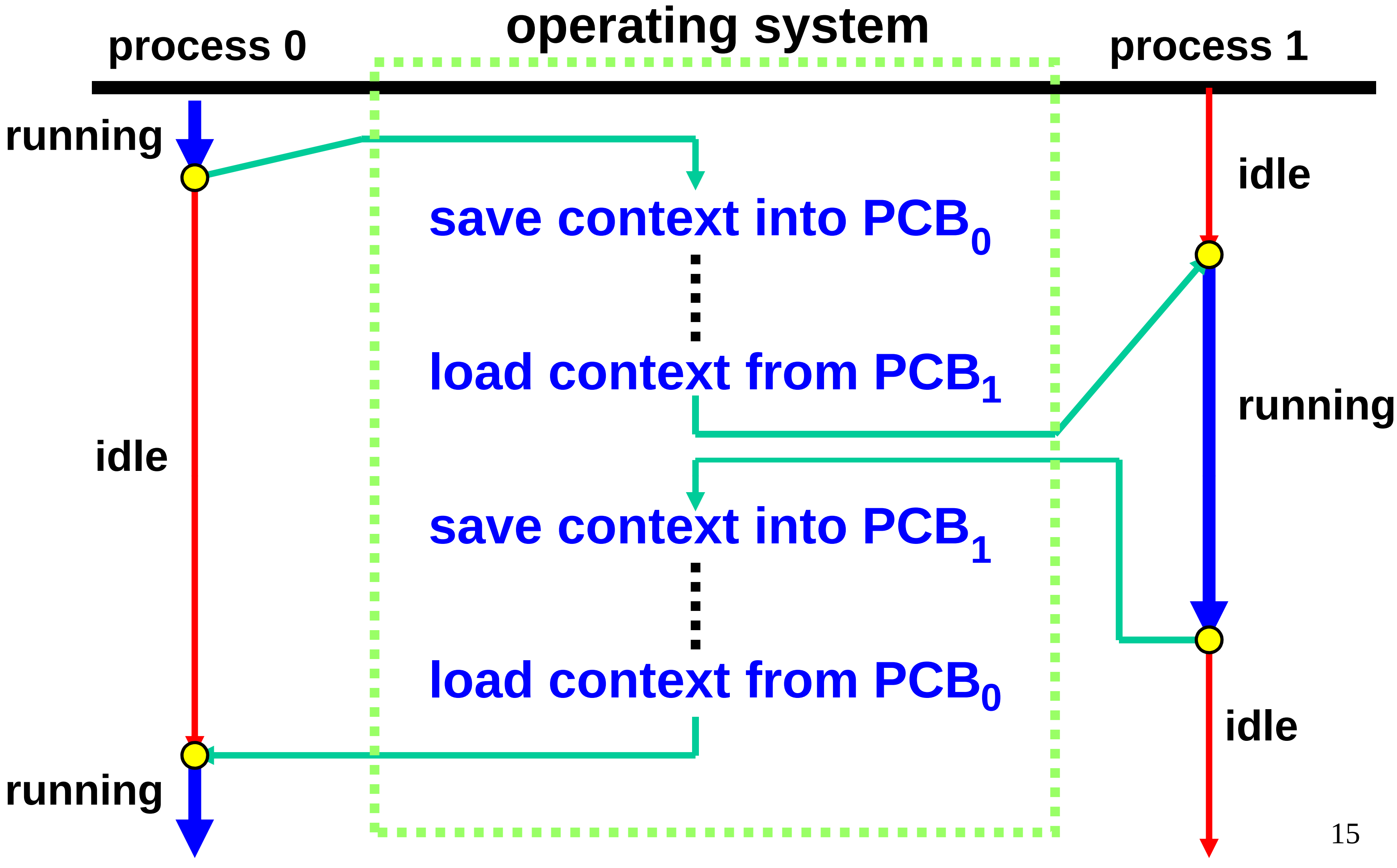
Medium-Term Scheduler



Context Switch

- **What is a process context?** **The *context* of a process includes the values of CPU registers, the process state, the program counter, and other memory/file management information (*i.e.*, execution environment).**
- **What is a context switch?** **After the CPU scheduler selects a process and before allocates CPU to it, the CPU scheduler must**
  - **save the *context* of the currently running process,**
  - **put it into a queue,**
  - **load the *context* of the selected process, and**
  - **let it run.**

Context Switch



Operations on Processes

- **There are three commonly seen operations:**
  - **Process Creation: Create a new process. The newly created is the child of the original. Use `fork()` or `vfork()` in Unix to create new processes.**
  - **Process Termination: Terminate the execution of a process. Under Unix, use `exit()`.**
  - **Process Join: Wait for the completion of a child process. Under Unix, use `wait()`.**
- **`fork()`, `vfork()`, `exit()` and `wait()` are system calls.**
- **Use "`ps -aux`" to see all running processes**
- **Will discuss this later in this semester.**

Cooperating Processes

- **A process is *independent* if it cannot affect or be affected by the other processes executing in the system.**
- **A process is *cooperating* if it can affect or be affected by the other processes executing in the system.**
- **Therefore, any process that *shares* resources (*e.g.*, files, memory blocks, etc) with other processes is a *cooperating* process.**

Why Is Cooperating Processes Important?

- **Information sharing: Multiple processes can use the same set of information (*e.g.*, files).**
- **Computation Speedup: One may split a process into multiple processes to use multiple processors.**
- **Modularity: A system can be divided into separate processes. Use the `ps` command to see how many processes are not yours!**
- **Convenience: A user may have multiple tasks to work at the same time (*e.g.*, editing, browsing, printing).**
- **However, handling cooperating processes is difficult. You will hate me very soon, ☺**

Interprocess Communications (IPC)

- **Cooperating processes must communicate to get the job done.**
- **There are two types of communications:**
  - **Processes that share the same memory: *locks*, *semaphores*, *monitors*, and others.**
  - **Processes that are running in a distributed environment: *message passing*, *sockets*, *remote procedure calls* (RPC).**

The End