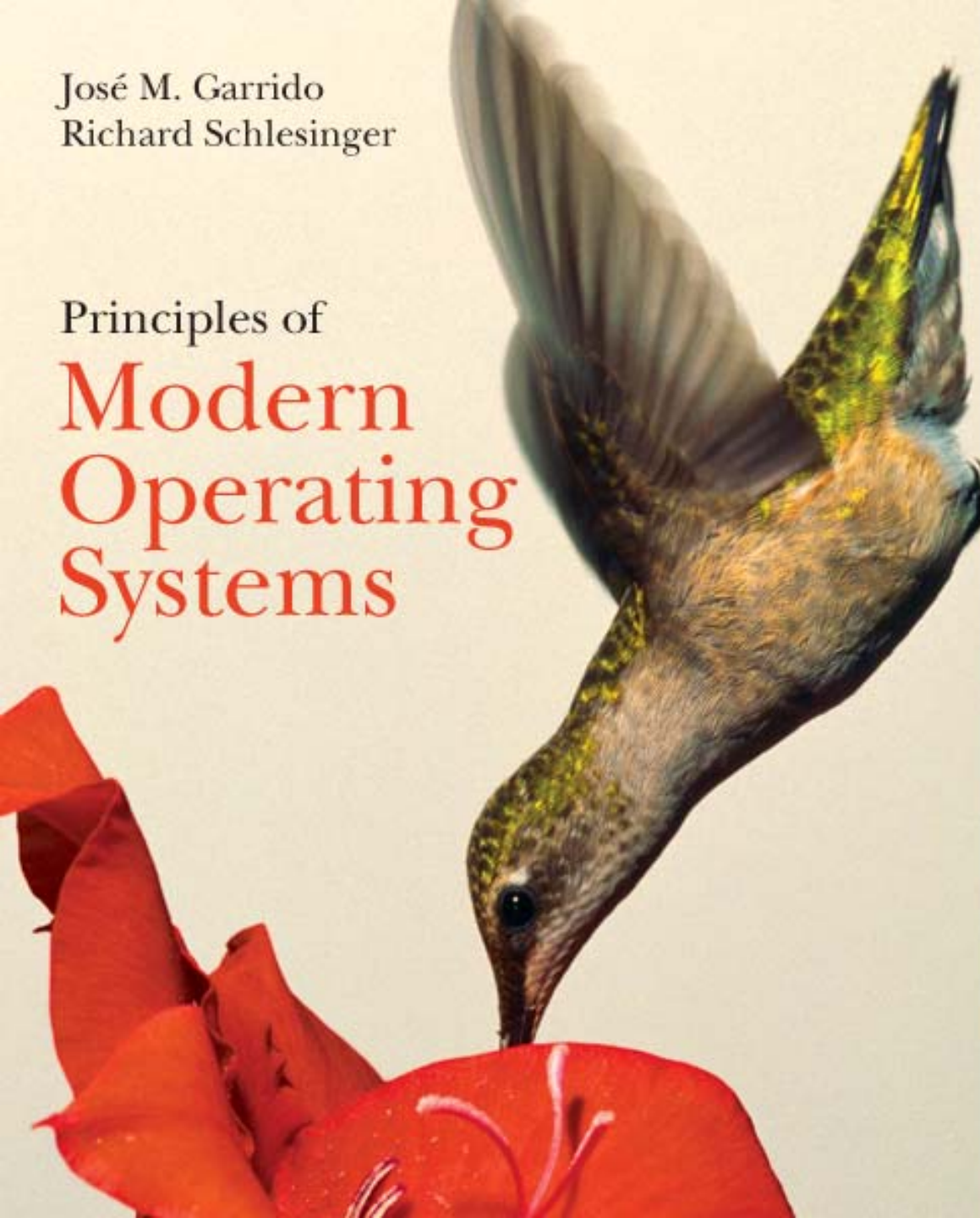


José M. Garrido
Richard Schlesinger

Principles of
**Modern
Operating
Systems**



Chapter 2

Processes and Threads

Processes

- A Process is the execution of a Program
- More specifically...
 - A process is a program in memory either
 - in execution
 - waiting for CPU, I/O, or other service.

Processes

- Two important types of dynamic entities in a computer system are processes and threads.
- Dynamic entities only exist at execution time, and their behavior exhibits state changes.
- A process is the fundamental computational unit that requests various types of services in the computer system.

Processes and Threads

- Processes are executing, ready to execute, or waiting to resume execution
- Several processes can be active at a time (O.S with multiprogramming)
- Only one process is actually running at any instant of time (in a single-CPU system)
- The OS switches the CPU from one process to another

Process Creation

- On a typical operating system, a process is created
 - In response to a new job arriving
 - A new user who carries out a login
 - A currently executing process that creates a new process.
- A process is created if there is sufficient memory available, and the process waits if other resources it needs are not available.

Process Behavior

- A process requests CPU and I/O processing at various times and usually will have to wait for these services.
- The OS maintains a waiting line or queue of processes for every one of these services.
- At some point in time, the process will be in a queue waiting for CPU service. At some other point in time, the process will be in a different queue waiting for I/O service.
- When the process completes all service requests, it terminates and exits the system.

System Resources

- Active resources: CPU, I/O devices
- Passive resources: memory

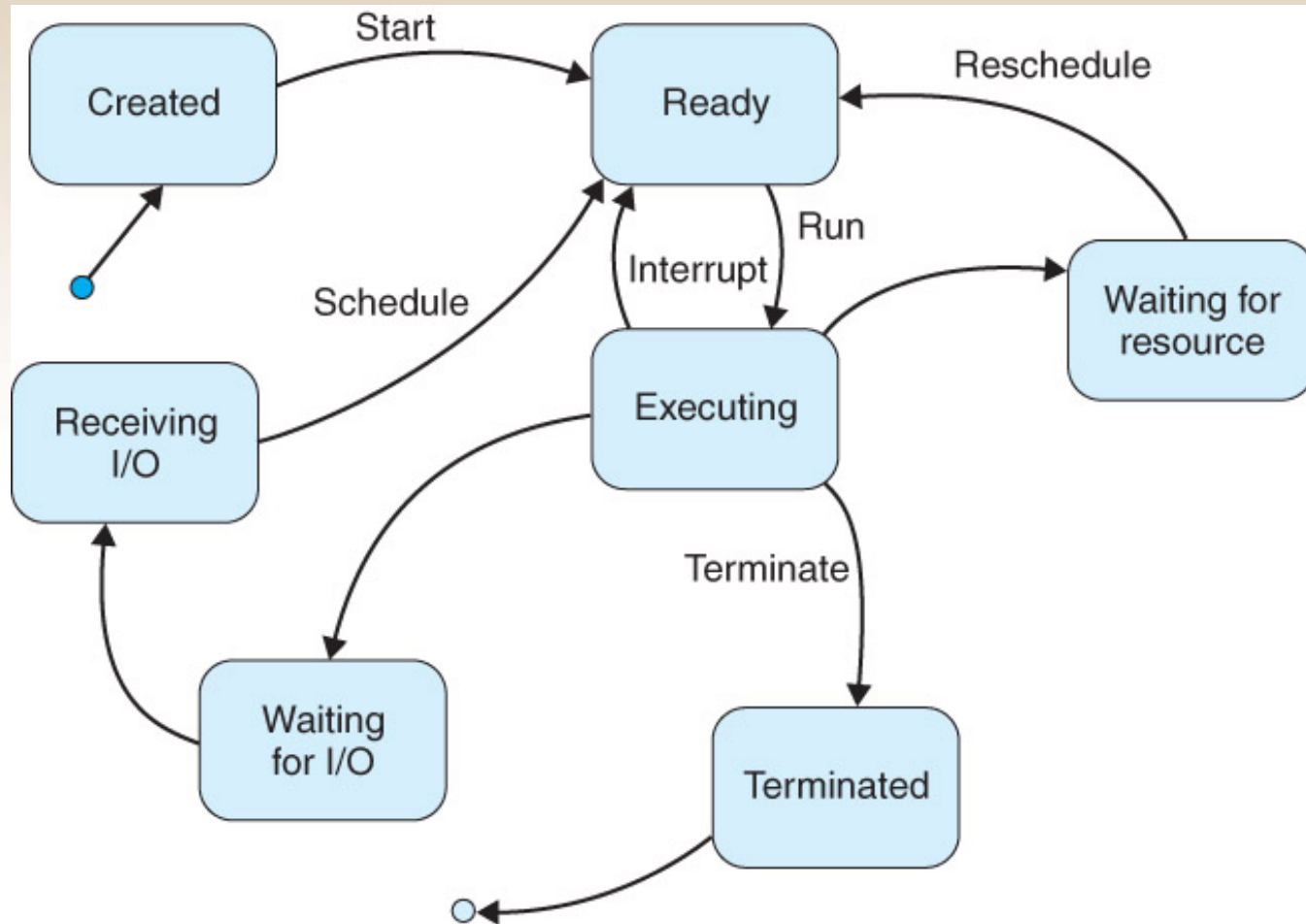
Process States

- Created
- Ready (waiting in the ready queue for the CPU)
- Running (executing)
- Blocked (suspended and waiting for I/O or other event)
- I/O processing
- Waiting for passive resources
- Interrupted by the OS
- Terminated

Process State Diagram

- Represents the various states of a process and the possible transitions in the behavior of a process.
- Each state is indicated by a rounded rectangle and the arrows connecting the states are the transitions from one state to the next.
- The small black circle denotes that the state it points to is the initial state.
- In a similar manner, the state before the hollow small circle indicates that it is the last state in the behavior of a process.
- Normally a process spends a short finite time in any of its states.

Process State-Transition Diagram



Process Descriptor

- Also called process control block (PCB)
- A data structure in which the OS maintains data about a process.

Data in a Process Descriptor

Each process is represented by its own PCB

- Process name
- Process state
- Program counter, CPU registers
- I/O status
- Memory information
- Accounting information
- Scheduler information (priority, queues, ...)
- List of resources

Process Descriptor

Process ID
State
User
Resources
Permissions
CPU registers
Parent pointer
Child list
Stack and code pointers

Threads

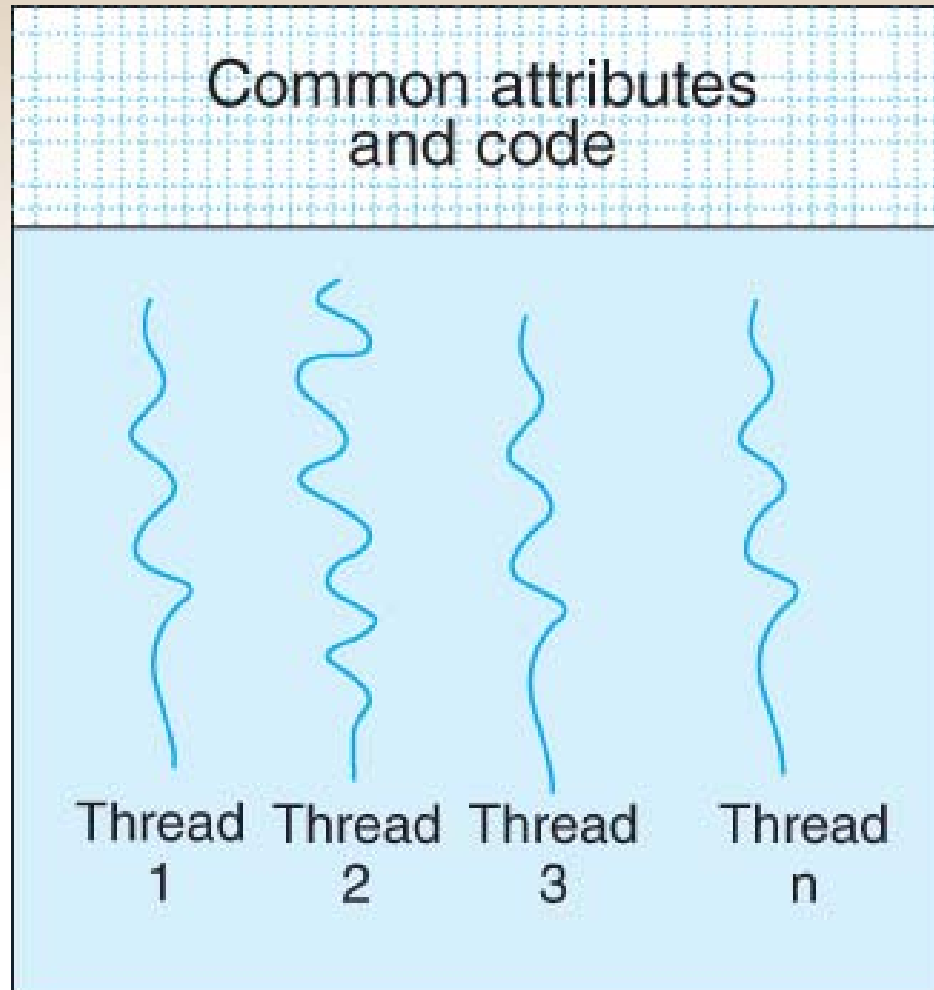
A Thread is:

- A light-weight process
- A single execution sequence
- A component of a process
- Shares with other threads code and resources allocated to the process
- Most OSs support multi-threading: several threads in a process

Multiple Threads

- In addition to multiple processes, newer operating systems support other computational units called ***threads***.
- A thread is called a lightweight process and is a dynamic component of a process.
- Several threads are usually created within a single process. These threads share part of the program code and the resources of the process.
- Most modern operating systems support multiple processes and multiple threads of execution within a single process, this feature is called multi-threading.

Threads Executing in a Process



Threads in a Process

- The operating system manages processes and threads in a multiprogramming environment
- The execution of threads are handled much more efficiently than the execution of processes.

Threads in a Process

- The modern view of threads represents them as the active elements of computation within a process.
- All threads that belong to a process share the state and resources of the process.

Thread Attributes

- Execution state
- Context, its own program counter within the process
- Execution stack
- Local memory block (for local variables)
- Reference to parent process to access the shared resources allocated to the process

Thread Descriptor

- A thread descriptor is a data structure used by the OS to store all the relevant data of a thread.
 - It contains the following fields:
 - thread identifier
 - execution state of the thread
 - the process owner of the thread
 - list of related threads
 - execution stack
 - thread priority
 - thread-specific resources
- Some of the fields of a thread descriptor also appear as part of the corresponding process descriptor.

Thread States

- Threads have their own execution state, context, execution stack, local memory (for local variables), in addition to the resources allocated to the process.
- When a process is created, only one thread is created called the base thread of the process; then other threads of the process may be created by the process or by the base thread.
- A process can have a single thread; in this case it is considered a simple process because it has only one thread of execution.

Types of Threads

There are two general types of threads:

- User-level threads (ULT)
- Kernel-level threads (KLT).

User-level Threads

- With the user-level threads, the thread management is carried out at the level of the application without kernel intervention.
- The application carries out threads management using a thread library, such as the POSIX Pthreads Library.
- Using this library, the application invokes the appropriate functions for the various thread management tasks such as: creating, suspending, and terminating threads.

Kernel-Level Threads

- The thread management tasks are carried out by the kernel.
- A process that needs these thread handling services has to use the API to the kernel thread facility.
- One of the advantages of kernel-level threads is that the process will not be blocked if one of its threads becomes blocked. Another advantage is that the possibility of scheduling multiple threads on multiple CPUs.

Process Management

- Process management is one of the major functions of the operating system; it involves creating processes and controlling their execution.
- In most operating systems, several processes are stored in memory at the same time and the operating system (OS) manages the sharing of the CPU and other resources among the various processes.
- This technique in the operating system is called multiprogramming.

Multi-programming

- One of the requirements of multiprogramming is that the OS must allocate the CPU and other resources to the various processes in such a way that the CPU and other active resources are maintained busy the longest period possible.
- If there is only one CPU in the computer system, then only one process can be in execution at any given time.
- The other processes are ready, waiting for CPU service. Processes also request access to passive resources, such as memory.

Context Switch

- When the CPU switches to another process, the OS must save the state of the current process and load the saved state of the new process
- Context switch time is overhead time
- Context switch time is dependent on hardware support

Multiprogramming

- An operating system can support several processes in memory.
- While one process receives service from the CPU, another process receives service from an I/O device and the other processes are waiting in some queues.
- The number of processes that the system supports is called the degree of multiprogramming.

Multiprogramming (2)

- In most computer systems, the I/O controllers enable a system to overlap device I/O operation with processor (CPU) operation.
- This results in a more efficient utilization of the system facilities
- Several processes can be in memory at a time; one receiving CPU service and another receiving I/O service, the others are waiting.

Processes and Multiprogramming

- Processes are execution instances of programs
- With multiprogramming, several processes can be active at a time
- Only one process is actually running at any instant of time (Only a single CPU)
- The OS switches rapidly the CPU from one process to another (context switching)
- The selection of which process to run next is made by the scheduler

Features Needed for Multiprogramming

- I/O routines supplied by the system
- Memory management
- CPU scheduling
- Allocation of devices

Service Demands

- A process will require several CPU and I/O requests (demands)
- The total CPU request of a process is divided into shorter service periods: each one is called a CPU burst
- The total I/O request is divided into shorter service periods called I/O bursts
- In normal behavior of a process, it alternates between a CPU and I/O burst

Total CPU Service Demand

- The total CPU service demand for a process is the sum of all its CPU bursts
- Each CPU burst has a different duration τ
- The total CPU service time requested by process P_i is:

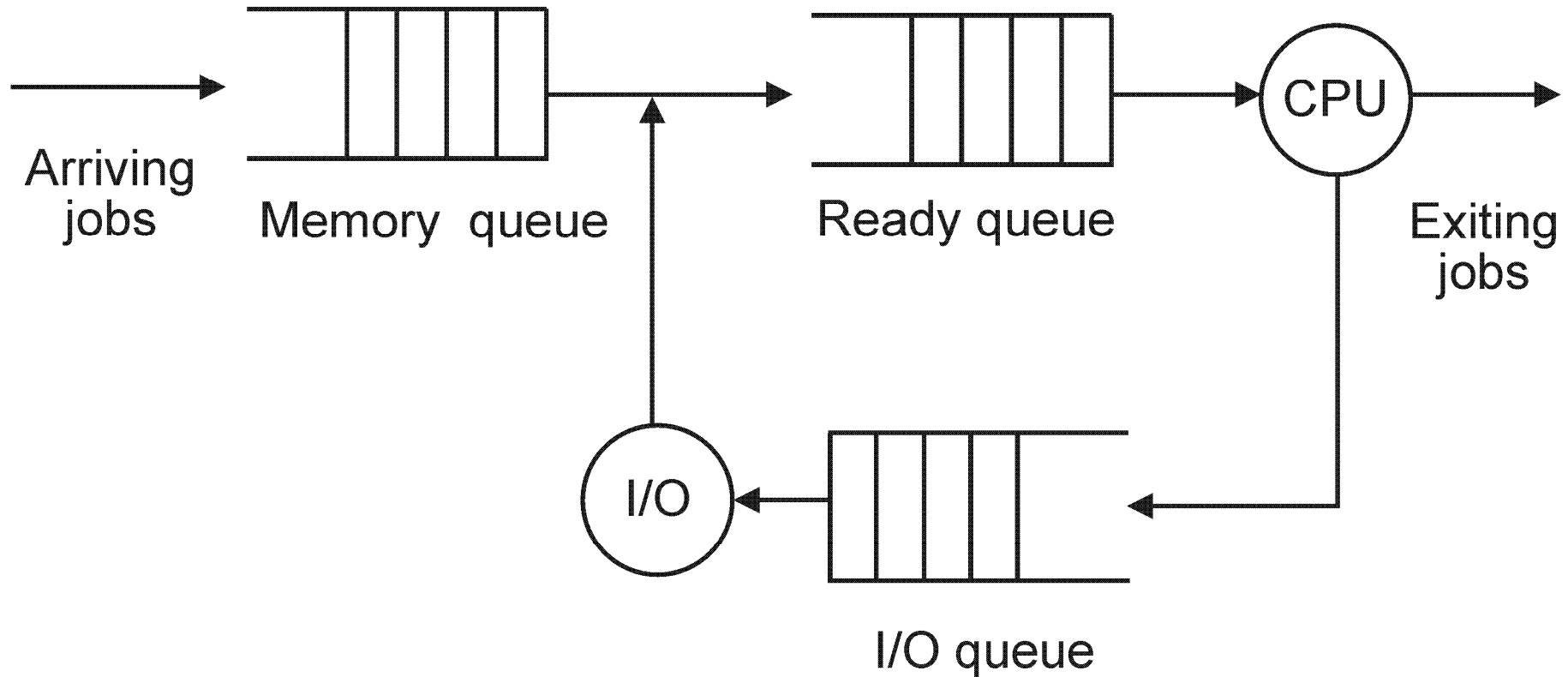
$$\tau_i = \tau_1 + \tau_2 + \dots + \tau_m$$

Total I/O Service Demand

- The total I/O service demand for a process is the sum of all its I/O bursts
- Each I/O burst has a different duration, δ
- The total I/O service time requested by process P_i is:

$$\delta_i = \delta_1 + \delta_2 + \dots + \delta_n$$

Batch OS with I/O

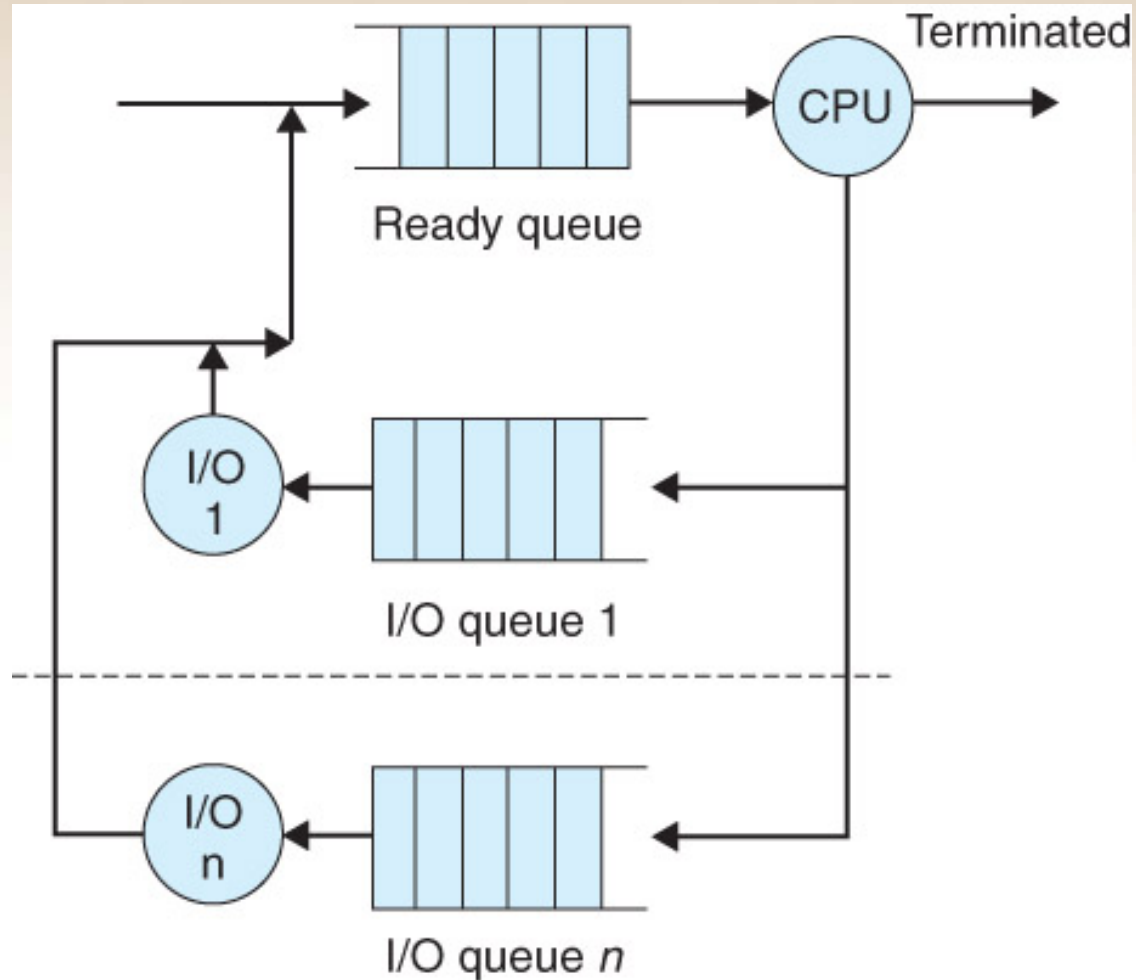


Interrupts and Context Switches

A context switch occurs when:

- The running process completes its current CPU burst and needs an I/O operation, such as reading data from a file.
- The running process is interrupted by a timer under OS control. The process returns to its ready state in the wait line for CPU service, the ready queue.
- A high priority process arrives or is made to transition to the ready state, the OS interrupts the running process if it has a lower priority.

CPU and I/O Devices



A Model with I/O Processing

- The C++ simulation model for the batch operating system with I/O is implemented in file `batchmio.cpp`
- Only one process is allowed to be stored in memory at a time. The degree of multiprogramming is set to 1 (no multiprogramming)
- When the degree of multi-programming is greater than 1, then several processes are allowed in memory