Process Description and Control

Chapter 3

Major Requirements of an Operating System

- ✓ Interleave the execution of many processes to maximize *processor utilization* while providing reasonable *response time*
- ✓ Allocate *resources* to processes
- ✓ Support *interprocess communication* and user creation of processes

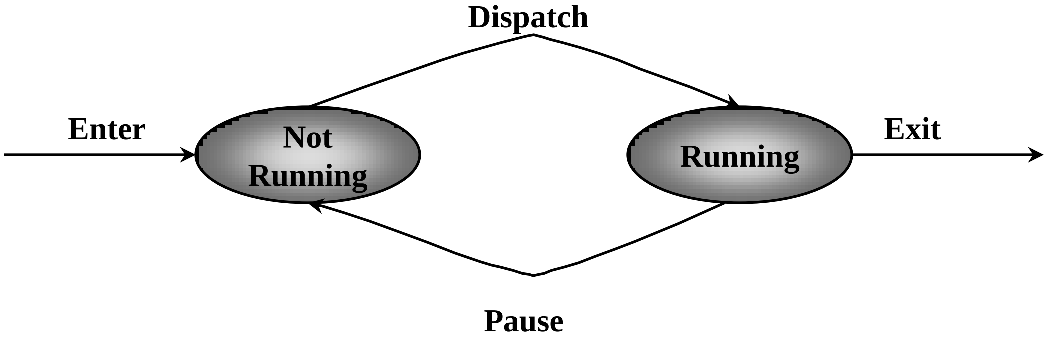
Process

- Also called a *task*
- Execution of an individual program
- Can be traced
    - list the sequence of instructions that execute

Dispatcher

- ✓ Program that assigns the processor from one process to another
- ✓ Prevents a single process from monopolizing processor time

Two-State Process Model



Dispatch

Enter

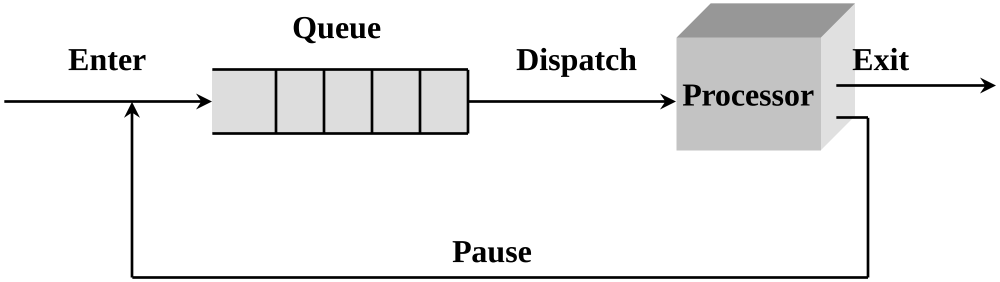
Not Running

Running

Exit

Pause

(a) State transition diagram



Queue

Enter

Dispatch

Processor

Exit

Pause

(a) Queuing diagram

How Processes are Created

- ✓ Submission of a batch job
- ✓ User logs on
- ✓ Created to provide a service such as printing
- ✓ Spawned by an existing process

How Processes Terminate

- ✓ Batch job issues *Halt* instruction
- ✓ User logs off
- ✓ Process executes a service request to terminate
- ✓ Error and fault conditions

Reasons for Process Termination

- ✓ Normal completion
- ✓ Time limit exceeded
- ✓ Memory unavailable
- ✓ Bounds violation
- ✓ Protection error
  - example write to read-only file
- ✓ Arithmetic error
- ✓ Timeout
  - process waited longer than a specified maximum for an event

Reasons for Process Termination (contd)

- I/O failure
- Invalid instruction
  - happens when try to execute data
- Privileged instruction executed in user mode
- Data misuse
- Operating system intervention
  - such as when deadlock occurs
- Parent terminates so child processes terminate
- Parent request

Process States

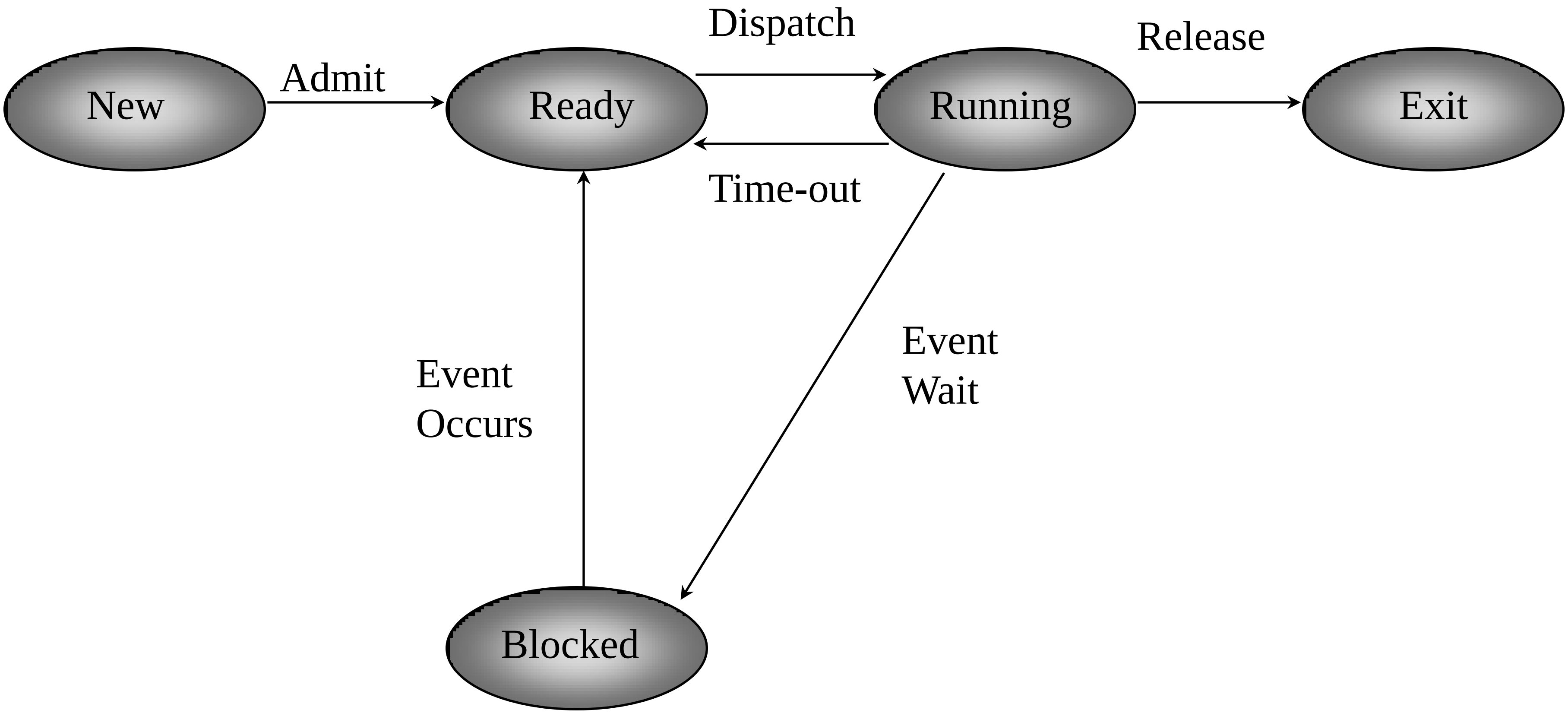
- ✓ Running
- ✓ Not-running
  - *ready* to execute
- ✓ Blocked
  - *waiting* for I/O

- ✓ *Dispatcher cannot just select the process that has been in the queue the longest because it may be blocked*

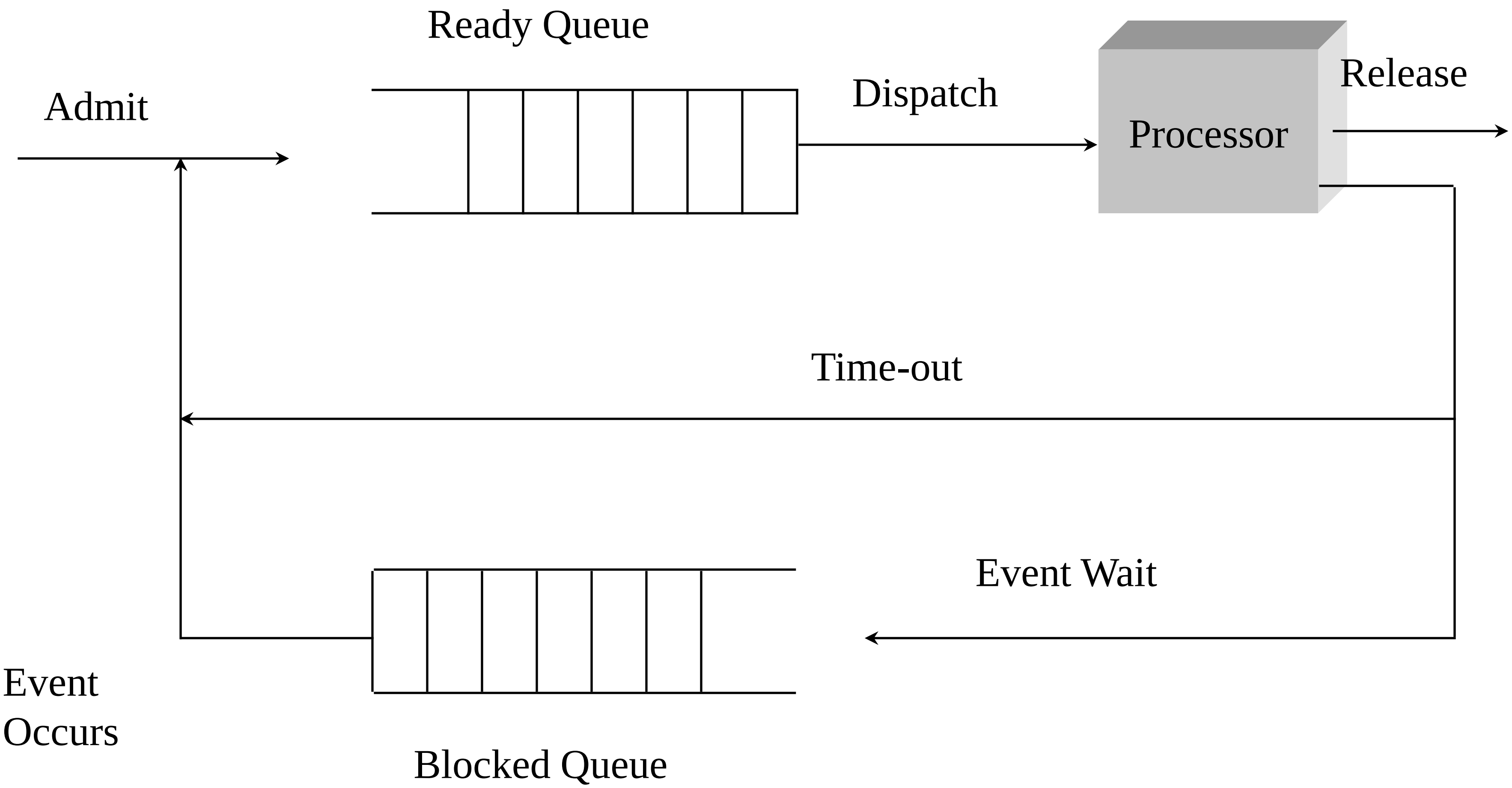
A Five-State Process Model

- Running
- Ready
- Blocked
- New
- Exit

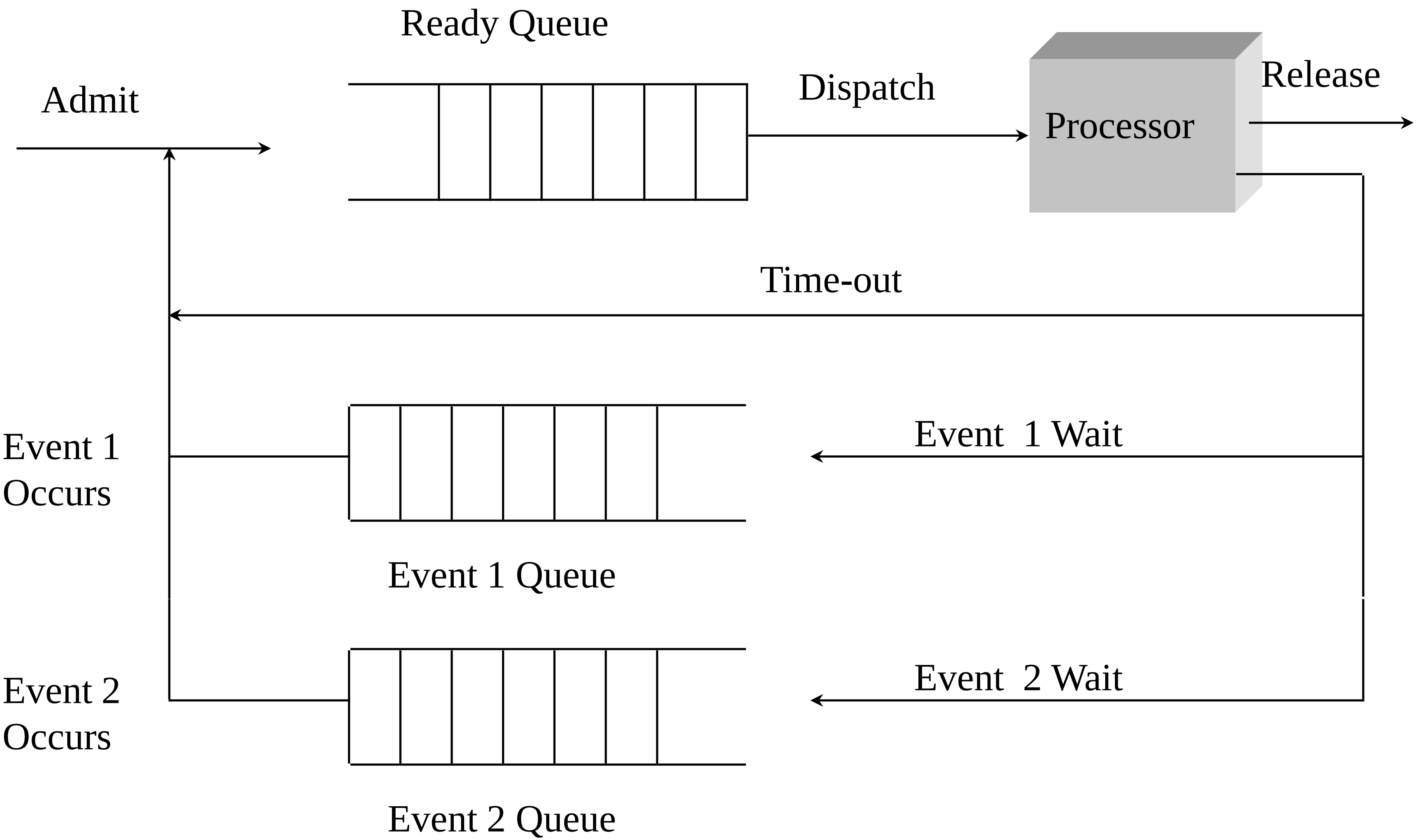
Five-State Process Model



Single Blocked Queue



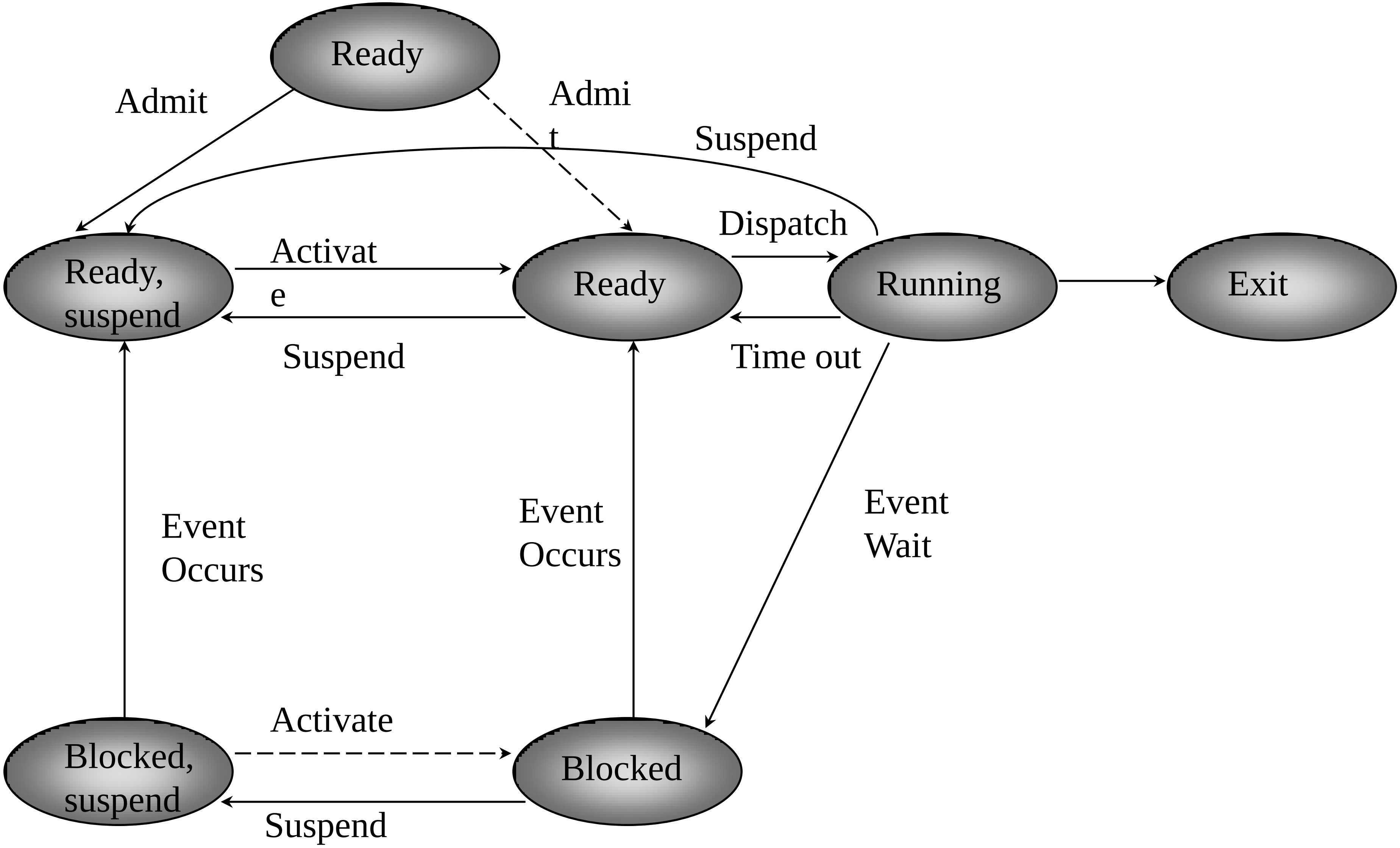
Multiple Blocked Queues



Suspended Processes

- ✓ CPU is faster than I/O so all processes could be waiting for I/O
- ✓ Swap these processes to disk to free up more memory
- ✓ Blocked state becomes suspend state when swapped to disk
- ✓ Two new states:
  - *Blocked-suspended*
  - *Ready-suspended*

Process State Transition Diagram with Two Suspend States



Operating System Control Structures

- ✓ An OS schedules and dispatches processes for execution by the CPU
- ✓ Allocates resources to processes
- ✓ Responds to requests by user programs. Therefore
  - ○ Tables (a.k.a. control blocks) are constructed for each entity managed by the OS

Memory Tables

- ✓ Allocation of main memory to processes
- ✓ Allocation of secondary memory to processes
- ✓ Protection attributes for access to shared memory regions
- ✓ Information needed to manage virtual memory

I/O Tables

- ✓ Whether an I/O device is available or assigned
- ✓ Status of I/O operation
- ✓ Location in main memory being used as the source or destination of the I/O transfer

File Tables

- ✓ Existence of files
- ✓ Location of files in secondary memory
- ✓ Current Status
- ✓ Attributes
- ✓ This information is maintained by a file-management subsystem (that runs on top of the OS kernel)

Process Table

- Process image consists of program, data, stack, and attributes
- Attributes
  - process control block

Process Control Block
Process Identification

- ✓ Unique numeric identifier
  - may be an index into the primary process table
- ✓ User identifier
  - who is responsible for the process

Processor State Information

- ✓ Contents of processor registers
  - User-visible registers
  - Control and status registers
  - Stack pointers
- ✓ Program status word (PSW)
  - contains status information
  - Example: the EFLAGS register -- Pentium machines version of PSW

Process Control Information

- ✓ Additional information needed by the operating system to control and coordinate the various active processes
  - scheduling and state information
  - data structuring (e.g., parent-child relationships; membership in wait/ready queues)
  - interprocess communication
  - process privileges
  - memory management
  - resource ownership and utilization

Typical Functions of an Operating-System Kernel

Process Management

- ✓ Process creation and termination
- ✓ Process scheduling and dispatching
- ✓ Process switching
- ✓ Process synchronization and support for inter-process communication
- ✓ Management of process control blocks

Typical Functions of an Operating-System Kernel

Memory Management

✓ Allocation of address space to processes

✓ Swapping in/out of memory blocks

✓ Page and segment management

Typical Functions of an Operating-System Kernel

I/O Management

- ✓ Buffer management
- ✓ Allocation of I/O channels and devices to processes

Support Functions

- ✓ Interrupt handling
- ✓ Accounting
- ✓ Monitoring

Process Creation

- ✓ Assign a unique process identifier
- ✓ Allocate space for the process
- ✓ Initialize process control block
- ✓ Set up appropriate linkages
  - Ex: add new process to linked list used as a scheduling queue
- ✓ Other
  - maintain an accounting file

When to Switch a Process

- Interrupts
  - Clock
    - process has executed for the maximum allowable time slice
  - I/O
- Memory fault
  - memory address is in virtual memory so it must be brought into main memory

When to Switch a Process

- ✓ Trap
  - error occurred
  - may cause process to be moved to Exit state
- ✓ Supervisor call
  - such as file open

Change of Process State

- Save context of processor including program counter and other registers
- Update the process control block with the new state and any accounting information
- Move process control block to appropriate queue - ready, blocked

Change of Process State

- ✓Select another process, **P**, for execution
- ✓Update the process control block of **P**
- ✓Update memory-management data structures (e.g., page table register should now point to the page table of process **P**)
- ✓Restore execution context of **P** (registers, program counter, etc.)

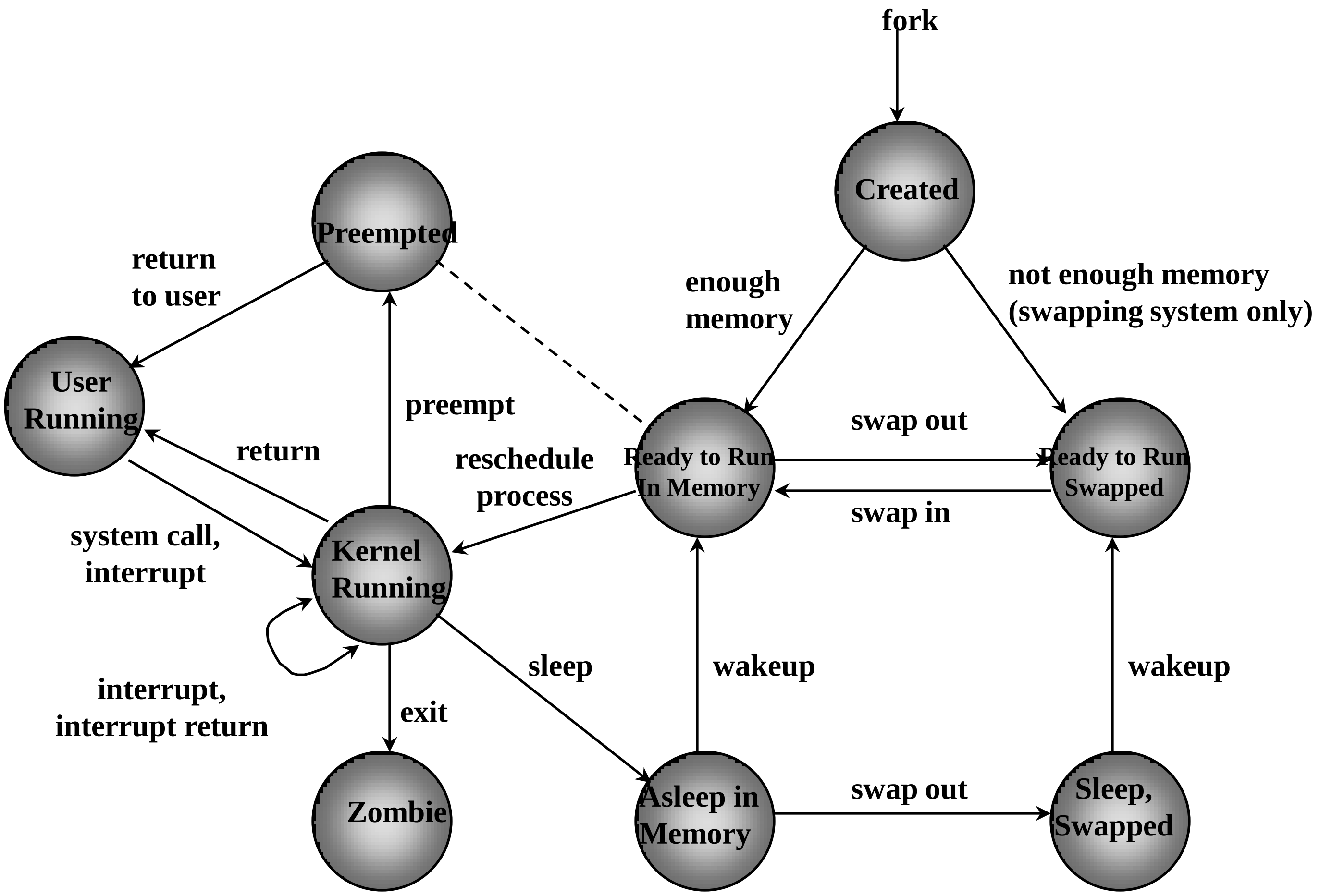
Execution of the Operating System

- ✓ Nonprocess Kernel
  - execute kernel outside of any process
  - operating system code is executed as a separate entity that operates in privileged mode
- ✓ Execution Within User Processes
  - operating system software within context of a user process
  - process executes in privileged mode when executing operating system code

Execution of the Operating System

- ✓ Process-Based Operating System
  - major kernel functions are separate processes
  - a process is invoked by the operating system

UNIX Process State Transition Diagram


fork
Created
enough memory
not enough memory (swapping system only)
Preempted
return to user
User Running
preempt
swap out
reschedule process
Ready to Run In Memory
Ready to Run Swapped
return
swap in
system call, interrupt
Kernel Running
interrupt, interrupt return
sleep
wakeup
wakeup
exit
Zombie
Asleep in Memory
swap out
Sleep, Swapped