OS Main Goals

- Interleave the execution of the number of processes to maximize processor utilization
- Provide reasonable response time
- Allocate resources to processes
- Support inter-process communication and user creation of processes

Process

- Abstraction of a running program
- Unit of work in the system
- Split into two abstractions in modern OS
  - Resource ownership (traditional process view)
  - Stream of instruction execution (thread)
- Pseudoparallelism, or interleaved instructions
- A process is traced by listing the sequence of instructions that execute for that process

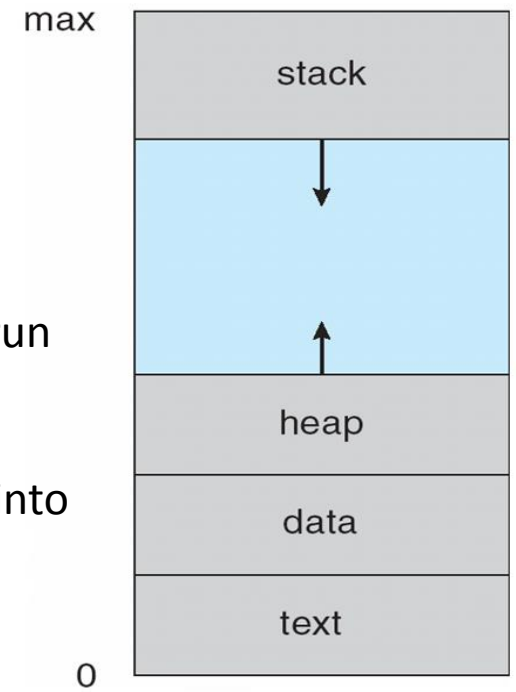
Process vs. Program

The difference between a **process** and a **program**:

- Baking analogy:
  - Recipe = Program
  - Baker = Processor
  - Baking the cake = Process
- Interrupt analogy
  - The baker's son runs in with a wounded hand
  - First aid guide = interrupt code

Modeling process/task

- Multiple parts
  - The program **code**, also called **text section**
  - Current activity including **program counter**, processor registers
  - **Stack** containing temporary data
    - Function parameters, return addresses, local variables
  - **Data section** containing global variables
  - **Heap** containing memory dynamically allocated during run time
- Program is passive entity, process is active
  - Program becomes process when executable file loaded into memory
- Execution of program started via GUI mouse clicks, command line entry of its name, etc
- One program can be several processes
  - Consider multiple users executing the same program

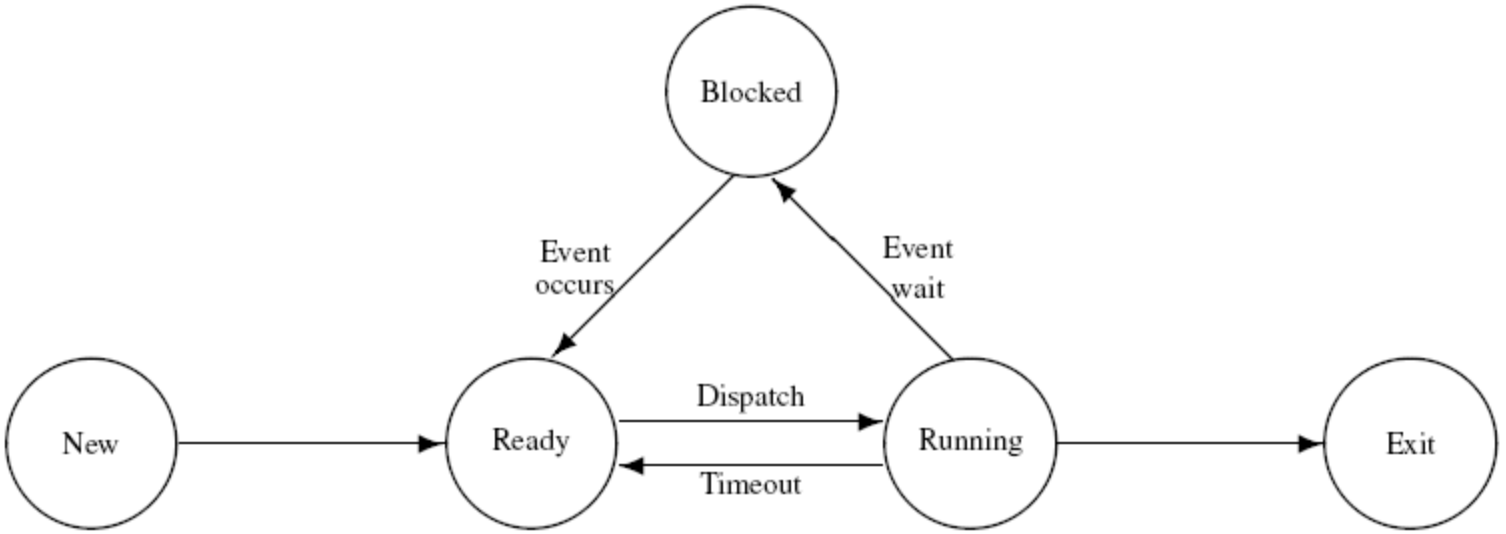


Concurrent Processes

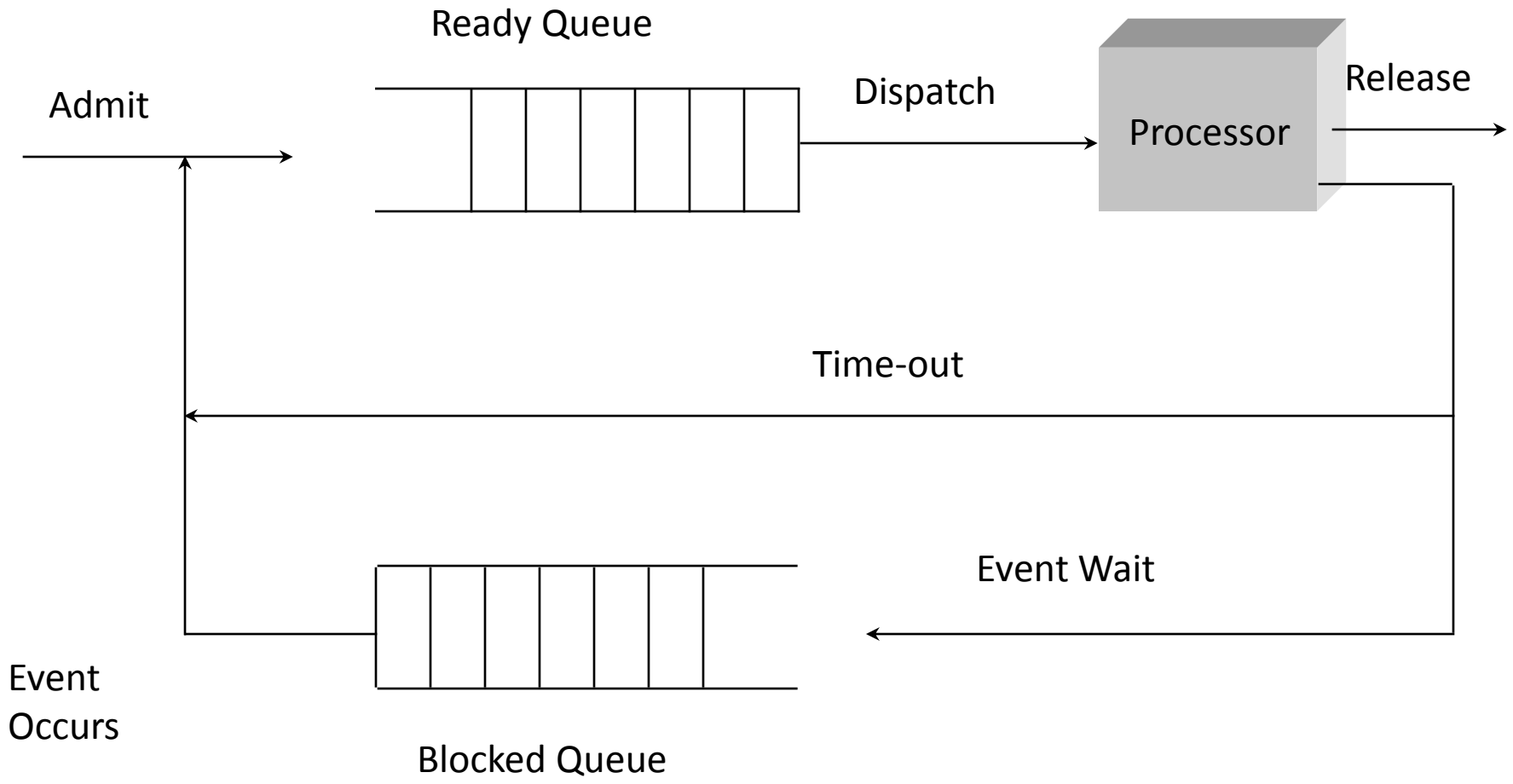
- Multiprogramming
- Computation speedup
  - Break each task into subtasks
  - Execute each subtask on separate processing element
- Modularity
  - Division of system functions into separate modules
- Convenience
  - Perform a number of tasks in parallel

Process State – 5-state model

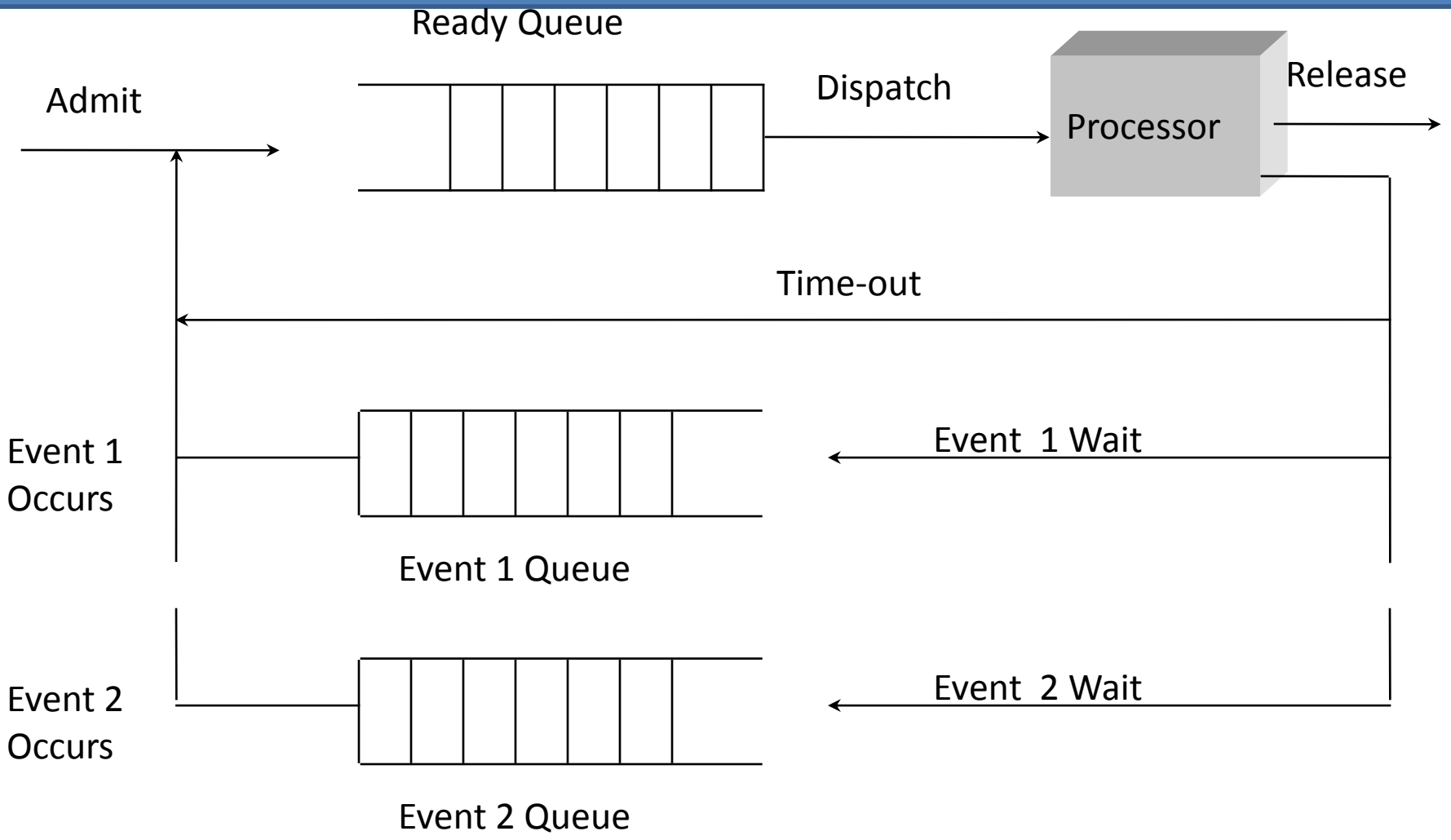
- As a process executes, it changes *state*
  - **New**: The process is being created
  - **Running**: Instructions are being executed
  - **Blocked/Waiting**: The process is waiting for some event to occur
  - **Ready**: The process is waiting to be assigned to a processor
  - **Exit/Terminated**: The process has finished execution



Single Blocked Queue



Multiple Blocked Queue



5-state model ?

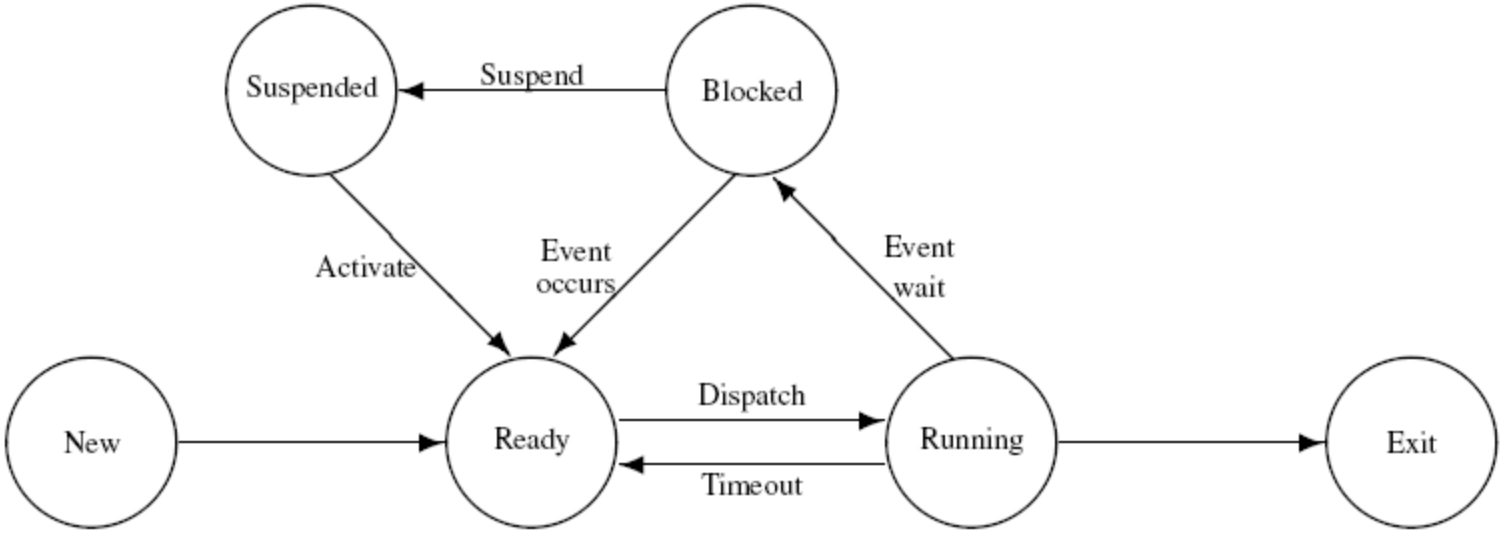
- 5-state model suffices for most of the requirements of process management; however,
  - what will happen when all the processes are resident in memory and they all are waiting for some event to happen?

E.g., Processor is faster than I/O so all processes could be waiting for I/O.

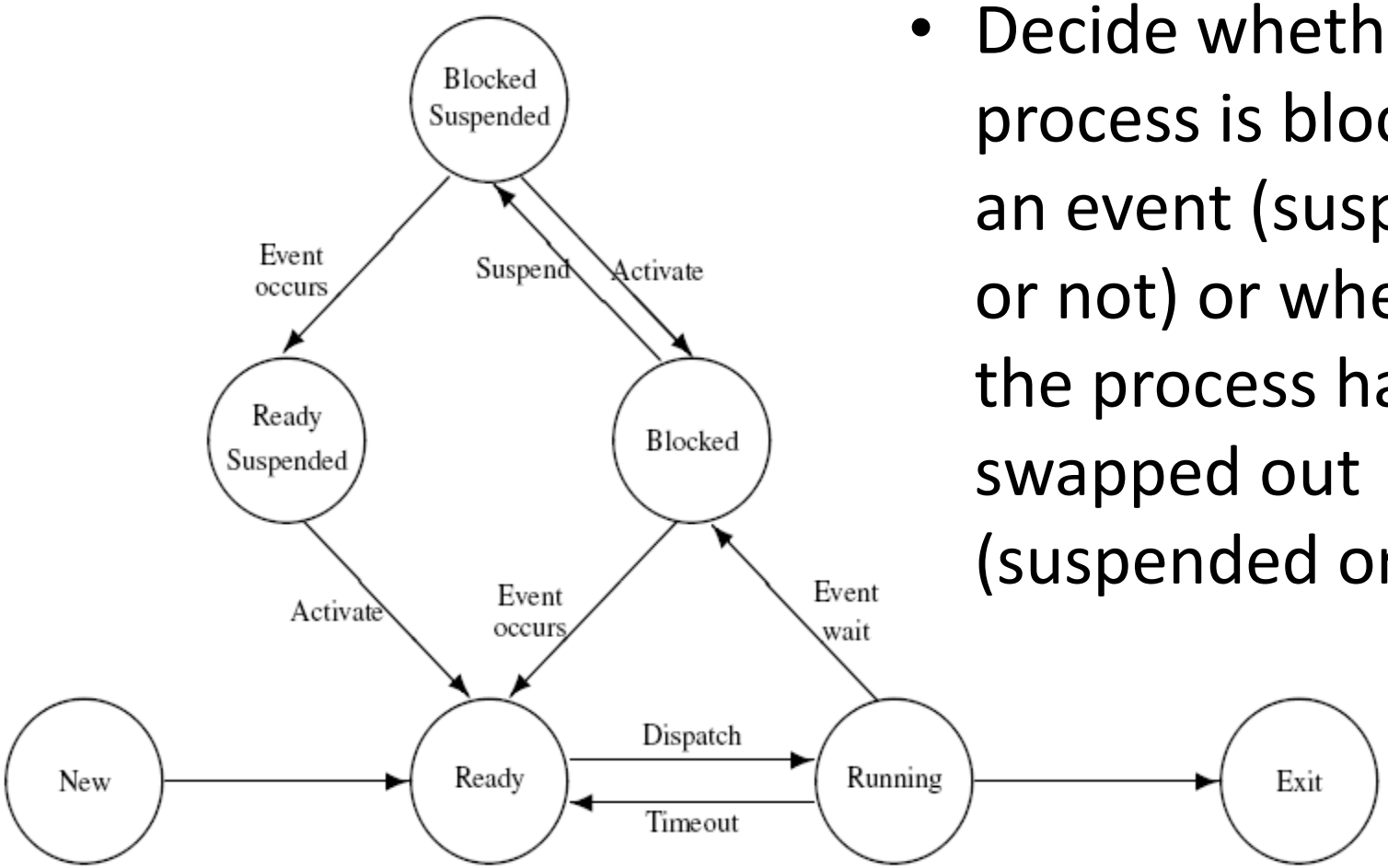
- Swap these processes to disk to free up more memory

5 + “suspend”

- Create a new state Suspend to keep track of blocked processes; and make room for new processes.
  - Blocked state becomes suspend state when swapped to disk



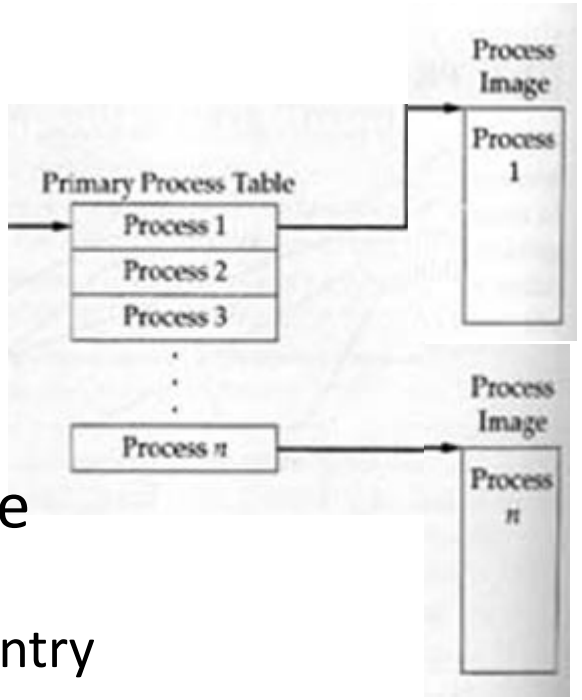
7-state model



- Decide whether the process is blocked on an event (suspended or not) or whether the process has been swapped out (suspended or not)

Implementation of Processes

- Process table
  - One entry for each process
    - program counter
    - stack pointer
    - memory allocation
    - open files
    - accounting and scheduling information
- Interrupt vector
  - Contains address of interrupt service procedure
    - saves all registers in the process table entry
    - services the interrupt



Process Creation

Principal events that cause process creation:

1. Execution of a process creation system call (e.g. fork() in Unix)
2. Initiation of a batch job (a sequence of commands to be executed by the operating system is listed in a file)
3. System initialization (deamons)

- Assign a unique process identifier to the new process; add this process to the system process table that contains one entry for each process
- Allocate space for all elements of process image – space for *code (text), data*, and *user stack*
- Build the data structures that are needed to manage the process, especially *process control block (PCB)*

Process Image

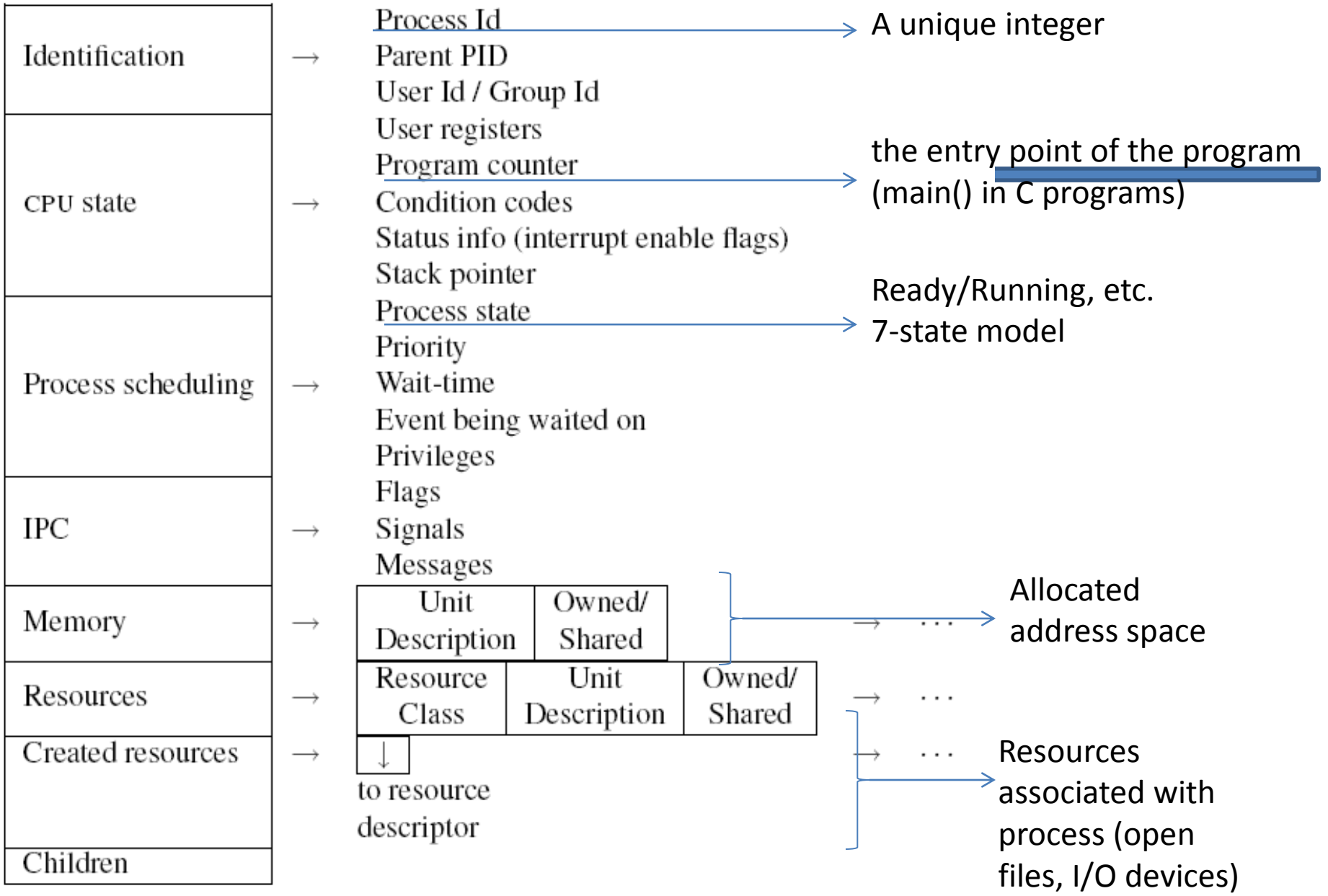
- Collection of programs, data, stack, and attributes that form the process
- User data
  - Modifiable part of the user space
  - Program data, user stack area, and modifiable code
- User program
  - Executable code
- System stack
  - Used to store parameters and calling addresses for procedure and system calls
- Process control block
  - Data needed by the OS to control the process
- Location and attributes of the process
  - Memory management aspects: contiguous or fragmented allocation

Data Structures for Processes

- Memory tables
  - Used to keep track of allocated and requested main and secondary memory
  - Protection attributes of blocks of main and secondary memory
  - Information to map main memory to secondary memory
- I/O tables
  - Used to manage I/O devices and channels
  - State of I/O operation and location in main memory as source/destination of operation
- File tables
  - Information on file existence, location in secondary memory, current status, and other attributes
  - Part of file management system
- Cross-referenced or linked in main memory for proper coordination

Process Control Block (PCB)

- Most important data structure in an OS
- Set of all process control blocks describes the state of the OS
- Read and modified by almost every subsystem in the OS, including scheduler, resource allocator, and performance monitor
- Constructed at process creation time
  - Physical manifestation of the process
  - Set of data locations for local and global variables and any defined constants
- Contains specific information associated with a specific process
  - The information can be broadly classified as process identification, processor state information, and process control information


Identification
CPU state
Process scheduling
IPC
Memory
Resources
Created resources
Children
Process Id
Parent PID
User Id / Group Id
User registers
Program counter
Condition codes
Status info (interrupt enable flags)
Stack pointer
Process state
Priority
Wait-time
Event being waited on
Privileges
Flags
Signals
Messages
Unit Description
Owned/ Shared
Resource Class
Unit Description
Owned/ Shared
to resource descriptor
A unique integer
the entry point of the program (main() in C programs)
Ready/Running, etc. 7-state model
Allocated address space
Resources associated with process (open files, I/O devices)