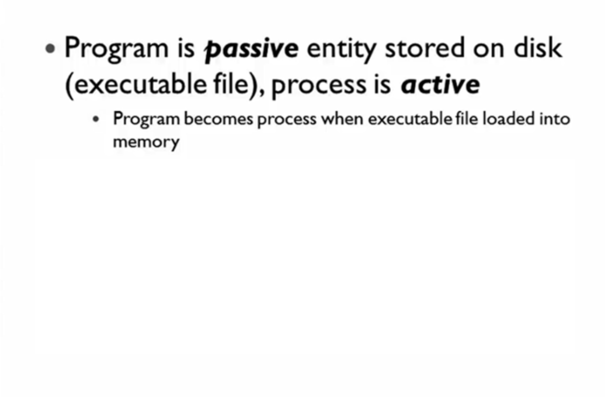
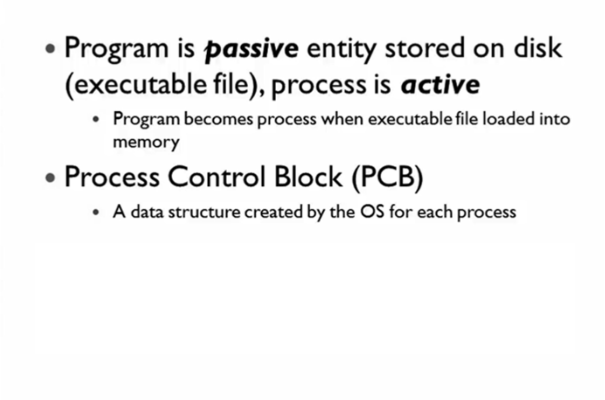
PROCESS

PROCESS CONCEPT



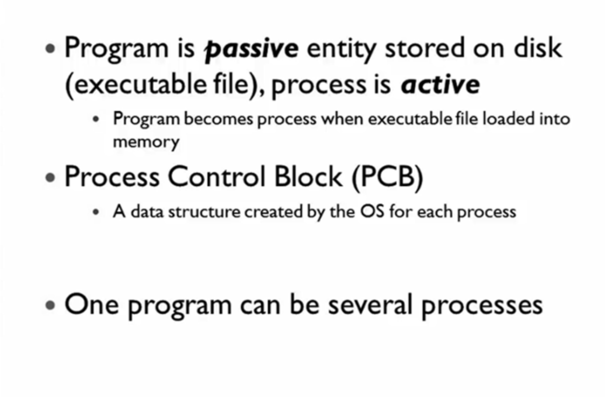
- Program is ***passive*** entity stored on disk (executable file), process is ***active***
  - Program becomes process when executable file loaded into memory

PROCESS CONCEPT



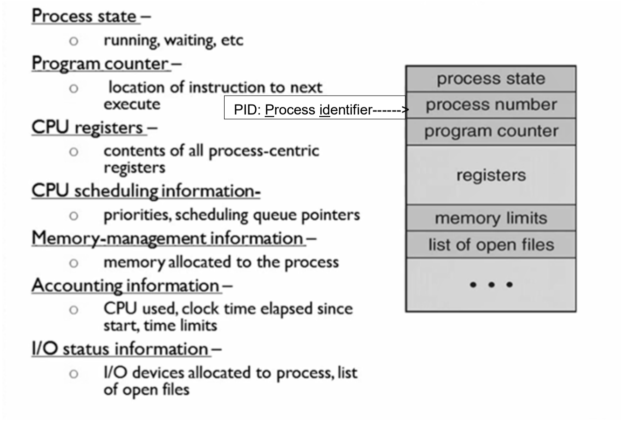
- Program is ***passive*** entity stored on disk (executable file), process is ***active***
  - Program becomes process when executable file loaded into memory
- Process Control Block (PCB)
  - A data structure created by the OS for each process

PROCESS CONCEPT



- Program is ***passive*** entity stored on disk (executable file), process is ***active***
  - Program becomes process when executable file loaded into memory
- Process Control Block (PCB)
  - A data structure created by the OS for each process
- One program can be several processes

Process **C**ontrol **B**lock (PCB)



Process state –
- running, waiting, etc

Program counter –
- location of instruction to next execute

CPU registers –
- contents of all process-centric registers

CPU scheduling information-
- priorities, scheduling queue pointers

Memory-management information –
- memory allocated to the process

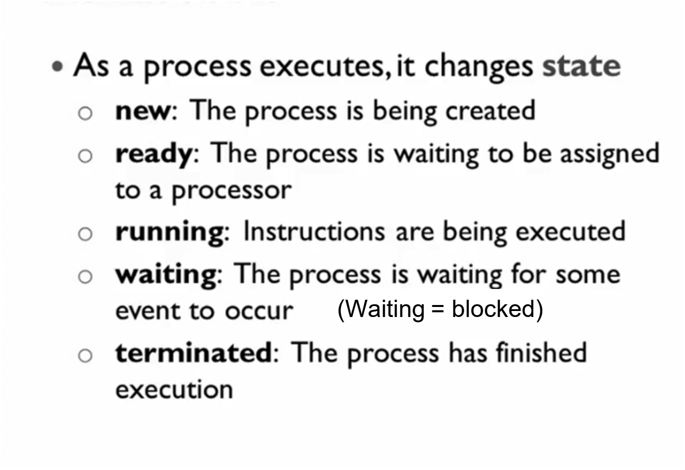
Accounting information –
- CPU used, clock time elapsed since start, time limits

I/O status information –
- I/O devices allocated to process, list of open files

PID: Process identifier------>

process state
process number
program counter
registers
memory limits
list of open files
• • •

Process States



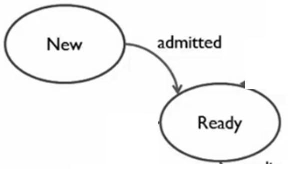
- As a process executes, it changes **state**
  - **new**: The process is being created
  - **ready**: The process is waiting to be assigned to a processor
  - **running**: Instructions are being executed
  - **waiting**: The process is waiting for some event to occur (Waiting = blocked)
  - **terminated**: The process has finished execution

Events causing a state transition

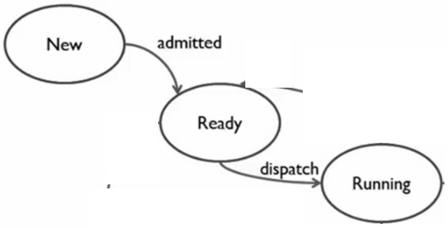
- A new process is created
- The process makes a resource request
- Resource is released
- The process requests an I/O device
- An I/O device is released after access
- The allocated time slice for a process is over. In this case, system timer sends a timer interrupt
- A higher-priority process appears in the ready queue. In this case, the running lower-priority job is pre-empted by a newly arrived higher-priority process
- The process reaches its end of execution or is aborted
- Any hardware interrupt is generated
- An error or exception condition is generated in the current running process

Process States

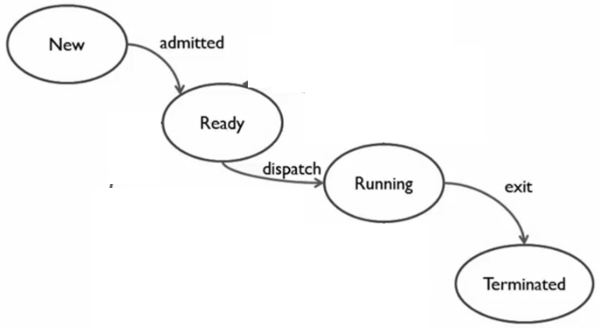
Process States



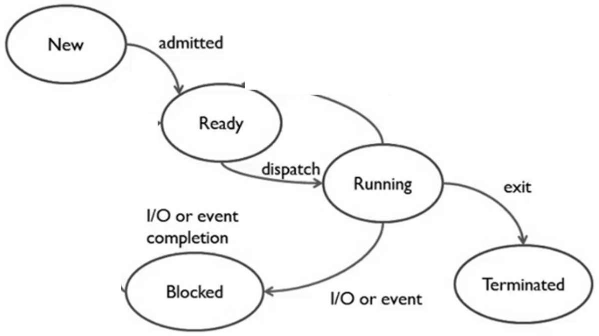
Process States



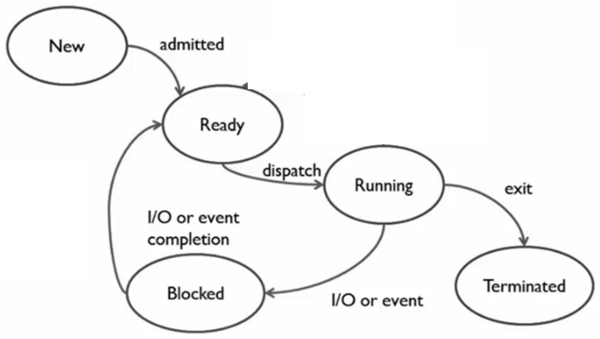
Process States



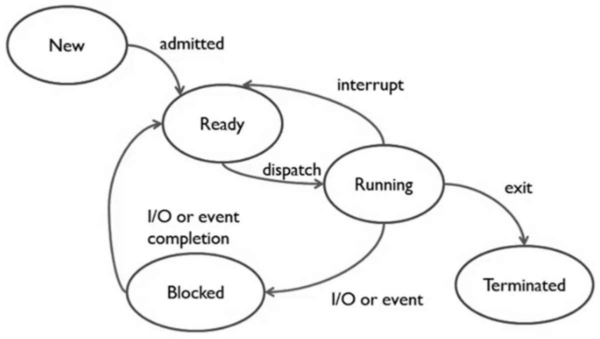
Process States



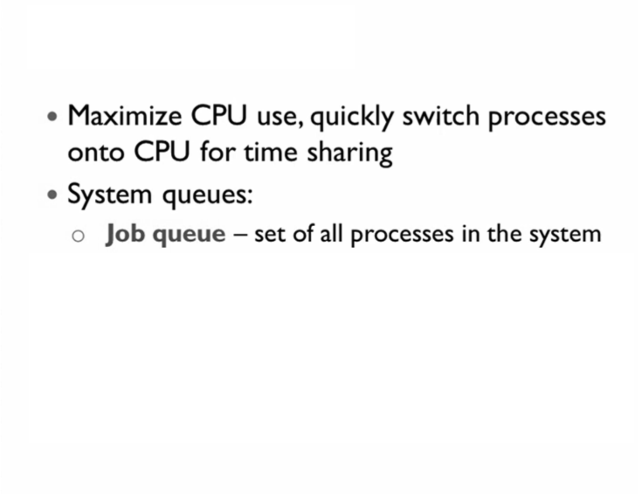
Process States



Process States

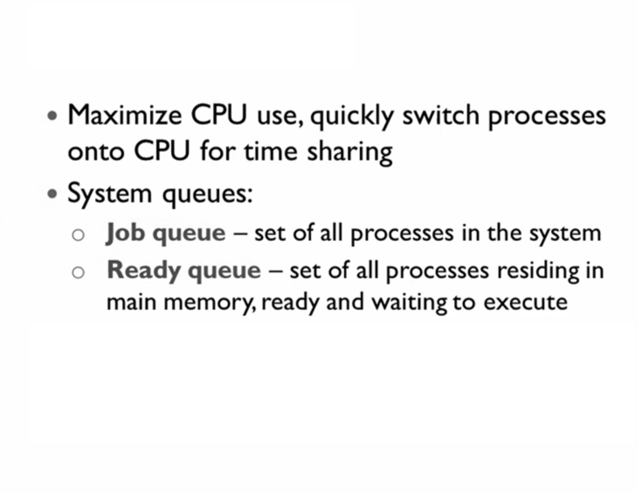


Process Scheduling



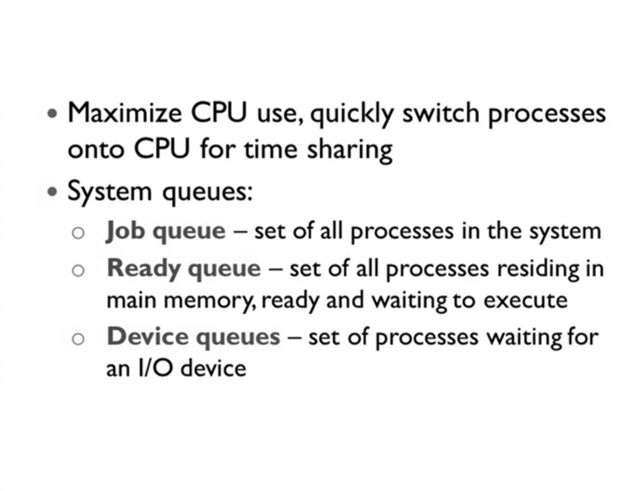
- Maximize CPU use, quickly switch processes onto CPU for time sharing
- System queues:
  - **Job queue** – set of all processes in the system

Process Scheduling



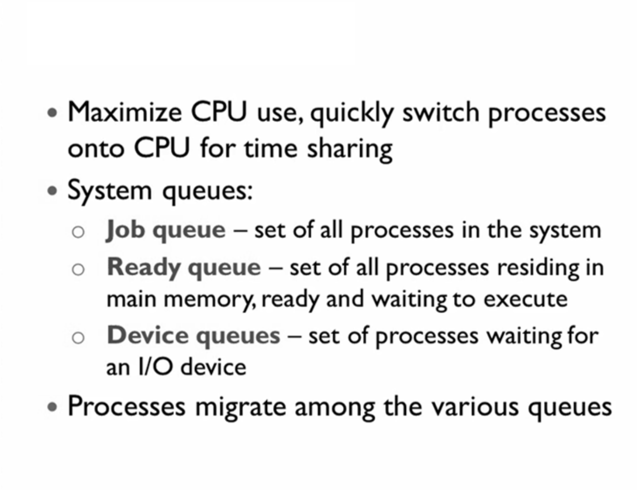
- Maximize CPU use, quickly switch processes onto CPU for time sharing
- System queues:
  - **Job queue** – set of all processes in the system
  - **Ready queue** – set of all processes residing in main memory, ready and waiting to execute

Process Scheduling



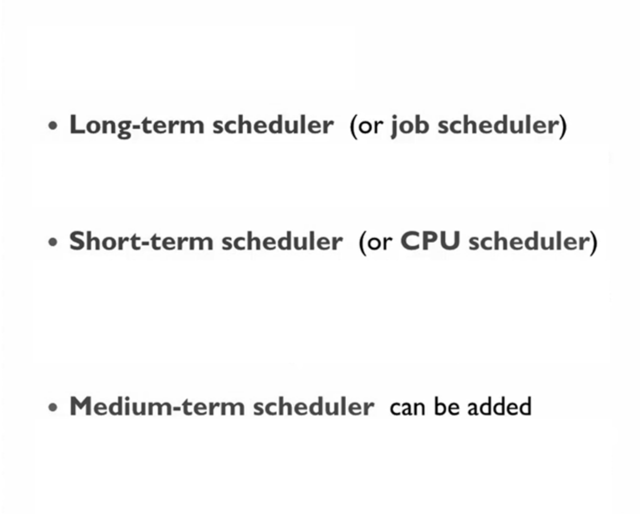
- Maximize CPU use, quickly switch processes onto CPU for time sharing
- System queues:
  - ○ **Job queue** – set of all processes in the system
  - ○ **Ready queue** – set of all processes residing in main memory, ready and waiting to execute
  - ○ **Device queues** – set of processes waiting for an I/O device

Process Scheduling



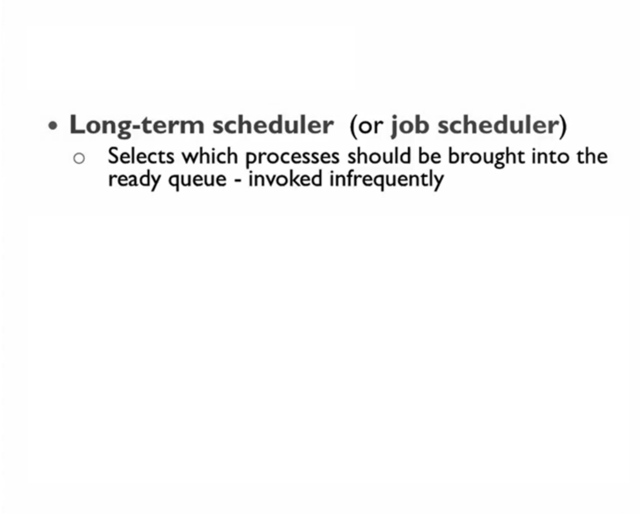
- Maximize CPU use, quickly switch processes onto CPU for time sharing
- System queues:
  - ○ **Job queue** – set of all processes in the system
  - ○ **Ready queue** – set of all processes residing in main memory, ready and waiting to execute
  - ○ **Device queues** – set of processes waiting for an I/O device
- Processes migrate among the various queues

Schedulers



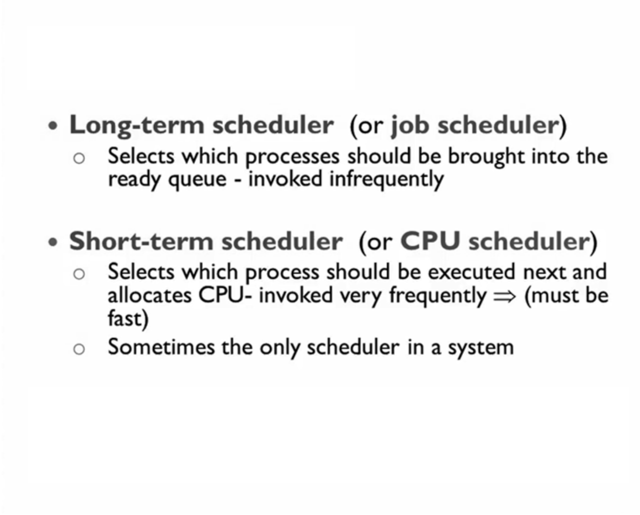
- **Long-term scheduler** (or **job scheduler**)
- **Short-term scheduler** (or **CPU scheduler**)
- **Medium-term scheduler** can be added

Schedulers



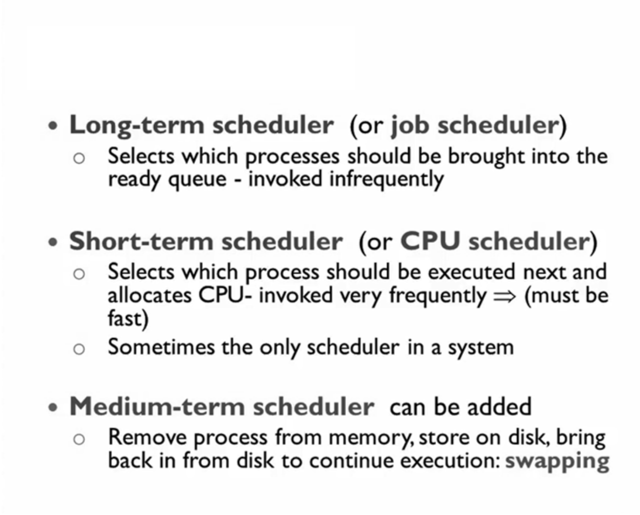
- **Long-term scheduler** (or **job scheduler**)
  - Selects which processes should be brought into the ready queue - invoked infrequently

Schedulers



- **Long-term scheduler** (or **job scheduler**)
  - Selects which processes should be brought into the ready queue - invoked infrequently
- **Short-term scheduler** (or **CPU scheduler**)
  - Selects which process should be executed next and allocates CPU- invoked very frequently ⇒ (must be fast)
  - Sometimes the only scheduler in a system

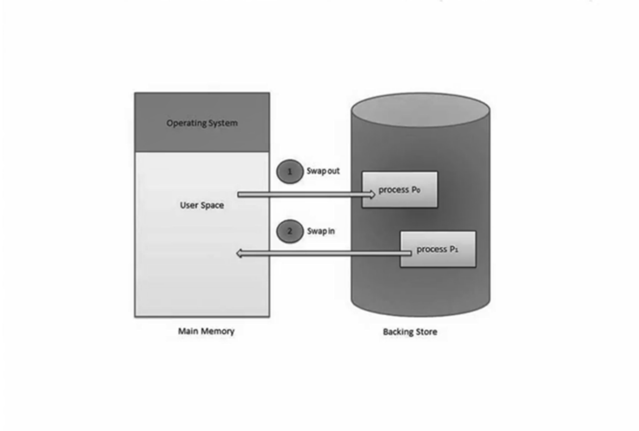
Schedulers



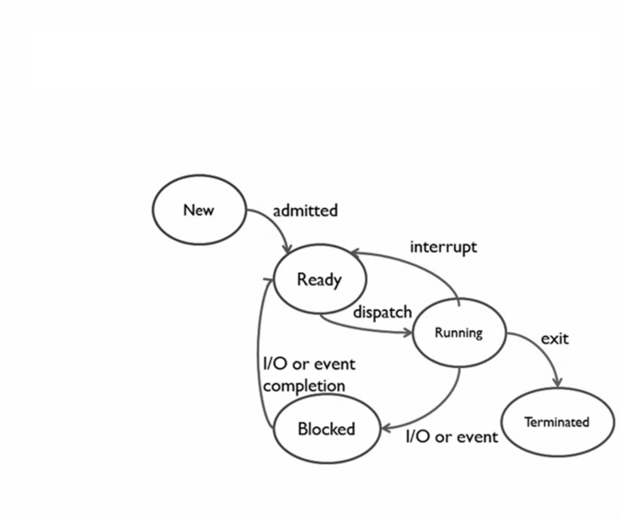
- **Long-term scheduler** (or **job scheduler**)
  - Selects which processes should be brought into the ready queue - invoked infrequently
- **Short-term scheduler** (or **CPU scheduler**)
  - Selects which process should be executed next and allocates CPU- invoked very frequently ⇒ (must be fast)
  - Sometimes the only scheduler in a system

- **Medium-term scheduler** can be added
  - Remove process from memory, store on disk, bring back in from disk to continue execution: **swapping**

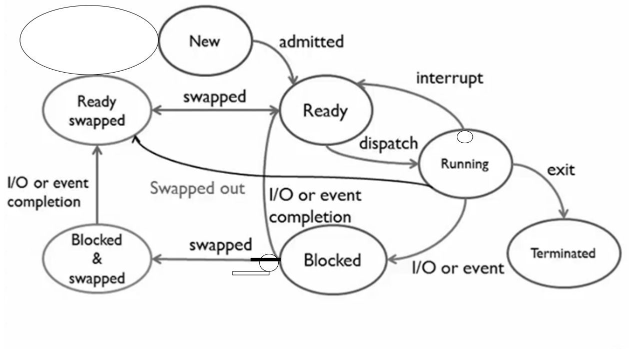
Medium Term Scheduling (Swapping)



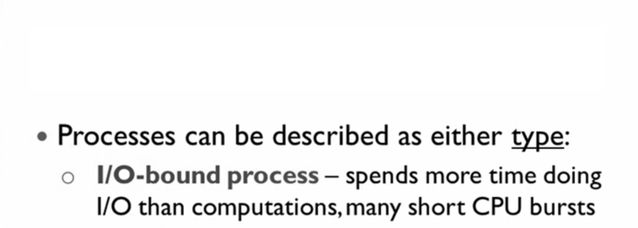
Medium Term Scheduling (Swapping)



Medium Term Scheduling (Swapping)

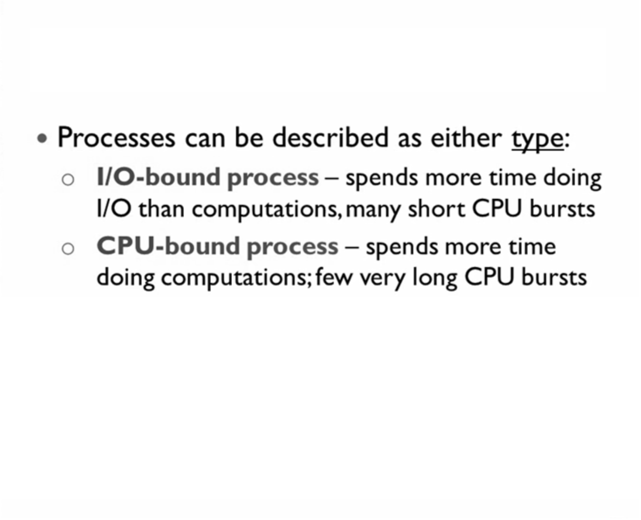


Process Definition



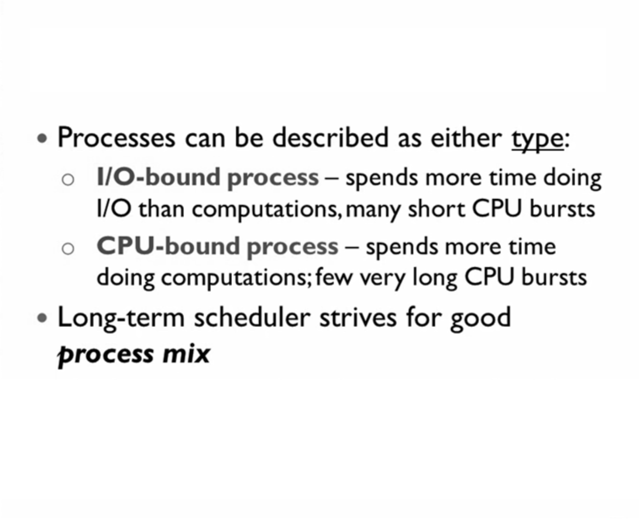
- Processes can be described as either <u>type</u>:
  - ○ **I/O-bound process** – spends more time doing I/O than computations, many short CPU bursts

Process Definition



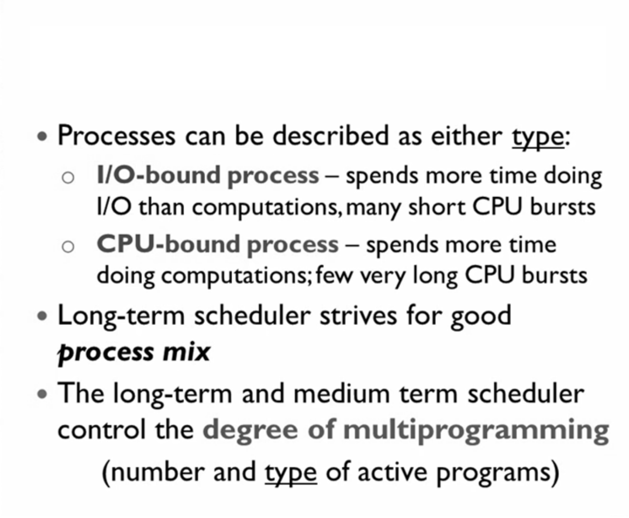
- Processes can be described as either <u>type</u>:
  - ○ **I/O-bound process** – spends more time doing I/O than computations, many short CPU bursts
  - ○ **CPU-bound process** – spends more time doing computations; few very long CPU bursts

Process Definition



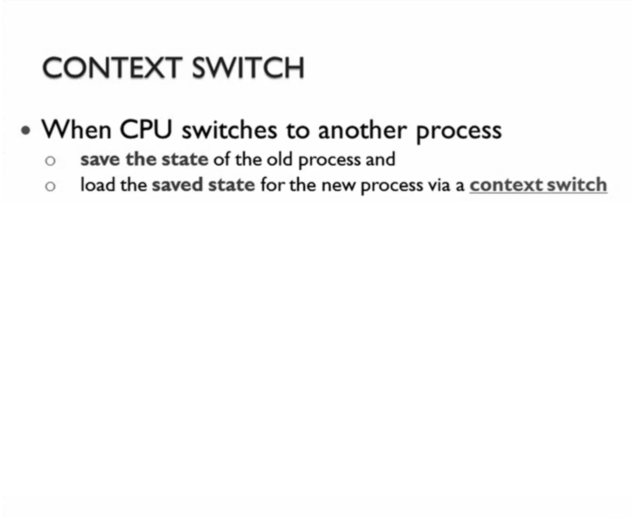
- Processes can be described as either <u>type</u>:
  - ○ **I/O-bound process** – spends more time doing I/O than computations, many short CPU bursts
  - ○ **CPU-bound process** – spends more time doing computations; few very long CPU bursts
- Long-term scheduler strives for good ***process mix***

Process Definition



- Processes can be described as either type:
  - **I/O-bound process** – spends more time doing I/O than computations, many short CPU bursts
  - **CPU-bound process** – spends more time doing computations; few very long CPU bursts
- Long-term scheduler strives for good ***process mix***
- The long-term and medium term scheduler control the **degree of multiprogramming**

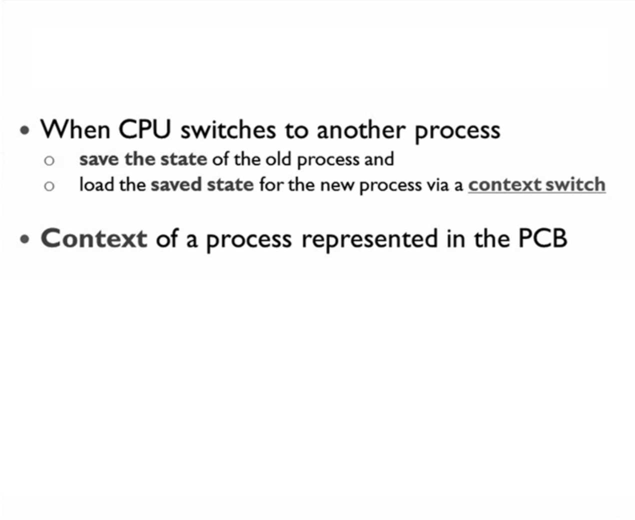
  (number and type of active programs)



CONTEXT SWITCH

- When CPU switches to another process
  - **save the state** of the old process and
  - load the **saved state** for the new process via a **context switch**

Context Switch

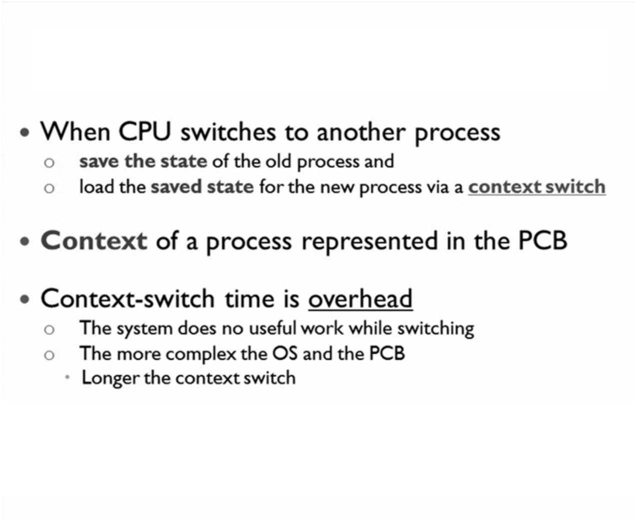


- When CPU switches to another process
  - **save the state** of the old process and
  - load the **saved state** for the new process via a **context switch**
- **Context** of a process represented in the PCB

31

31

Context Switch

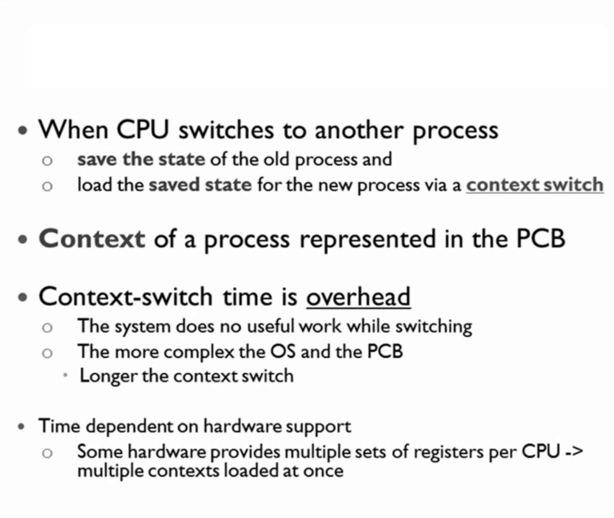


- When CPU switches to another process
  - **save the state** of the old process and
  - load the **saved state** for the new process via a **context switch**
- **Context** of a process represented in the PCB
- Context-switch time is overhead
  - The system does no useful work while switching
  - The more complex the OS and the PCB
    - Longer the context switch

32

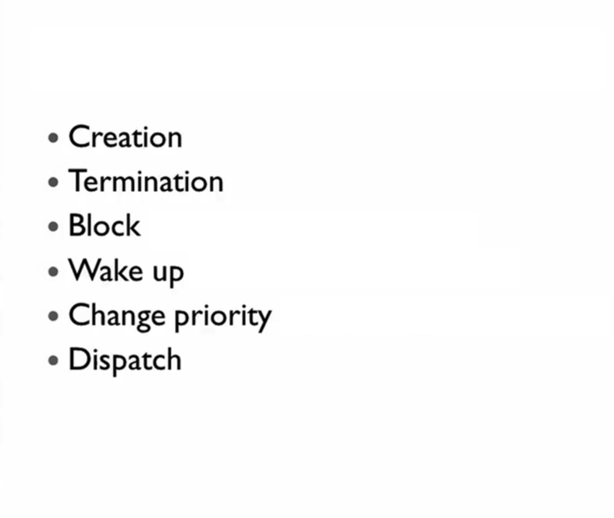
32

Context Switch



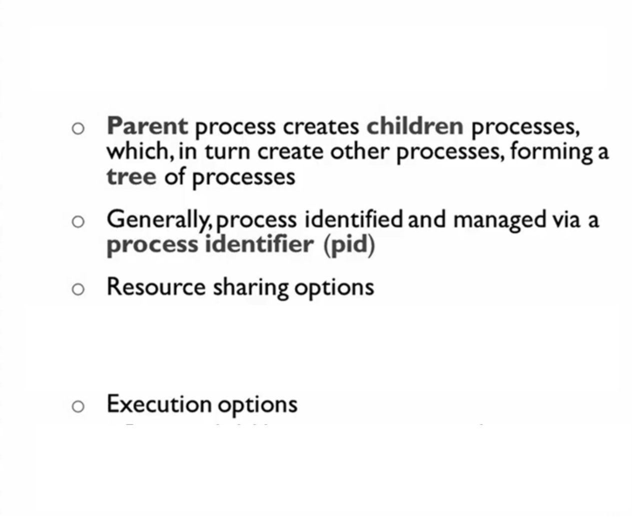
- When CPU switches to another process
  - **save the state** of the old process and
  - load the **saved state** for the new process via a **context switch**
- **Context** of a process represented in the PCB
- Context-switch time is overhead
  - The system does no useful work while switching
  - The more complex the OS and the PCB
    - Longer the context switch
- Time dependent on hardware support
  - Some hardware provides multiple sets of registers per CPU -> multiple contexts loaded at once

Operations on Processes



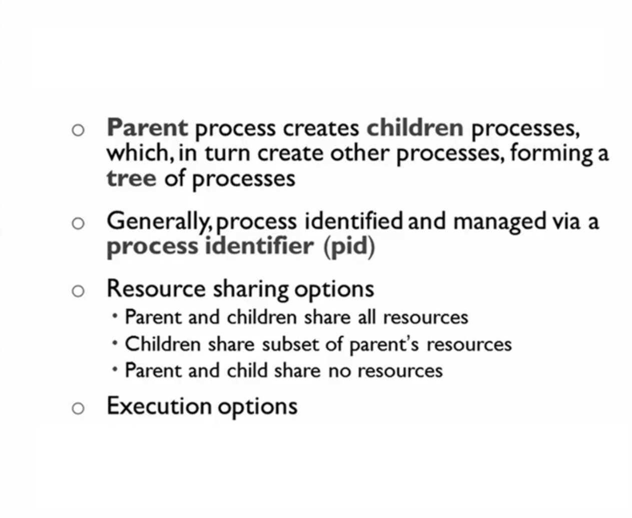
- Creation
- Termination
- Block
- Wake up
- Change priority
- Dispatch

Process Creation



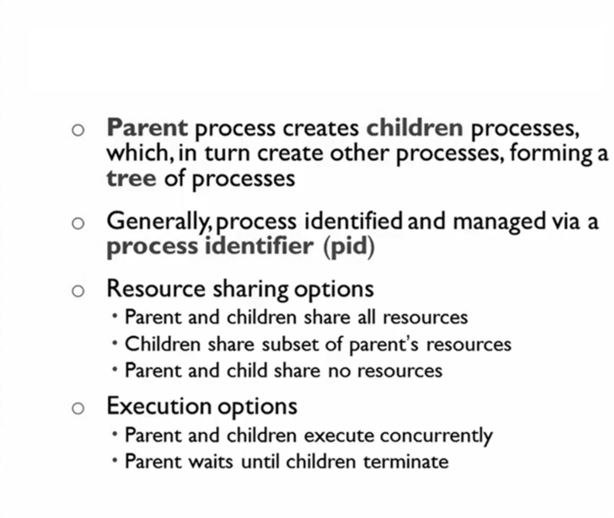
- **Parent** process creates **children** processes, which, in turn create other processes, forming a **tree** of processes
- Generally, process identified and managed via a **process identifier (pid)**
- Resource sharing options
- Execution options

Process Creation



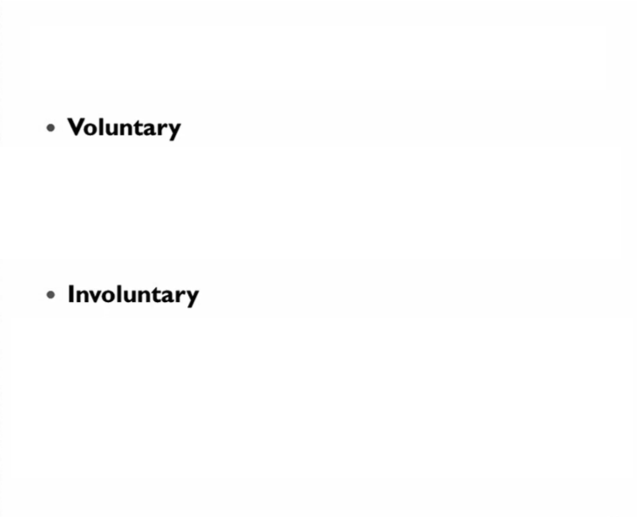
- **Parent** process creates **children** processes, which, in turn create other processes, forming a **tree** of processes
- Generally, process identified and managed via a **process identifier (pid)**
- Resource sharing options
  - Parent and children share all resources
  - Children share subset of parent's resources
  - Parent and child share no resources
- Execution options

Process Creation



- **Parent** process creates **children** processes, which, in turn create other processes, forming a **tree** of processes
- Generally, process identified and managed via a **process identifier (pid)**
- Resource sharing options
  - Parent and children share all resources
  - Children share subset of parent's resources
  - Parent and child share no resources
- Execution options
  - Parent and children execute concurrently
  - Parent waits until children terminate

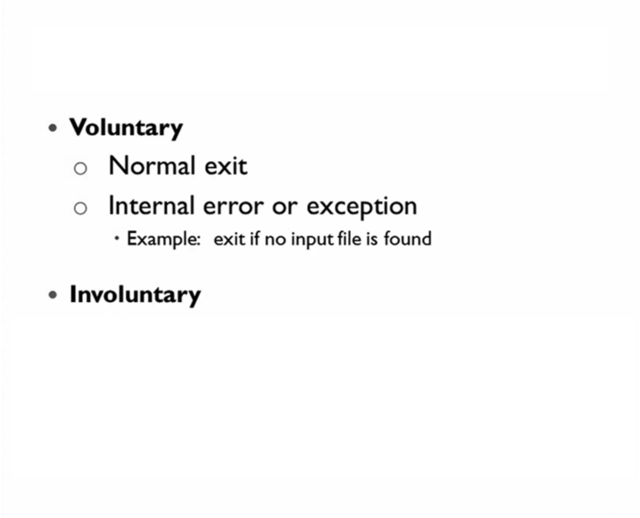
Process Termination



- **Voluntary**

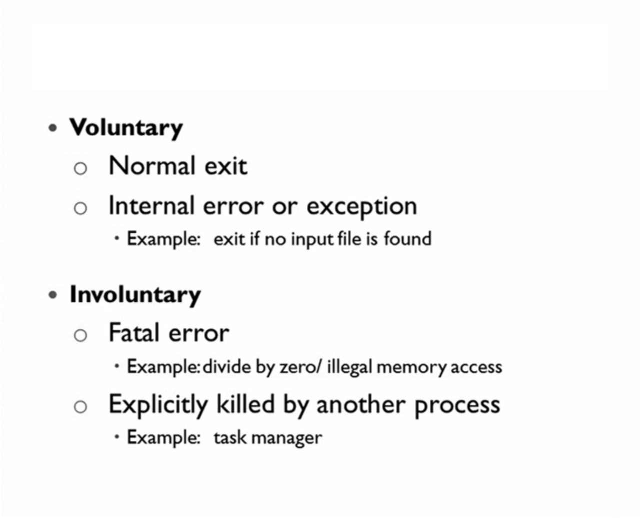
- **Involuntary**

Process Termination



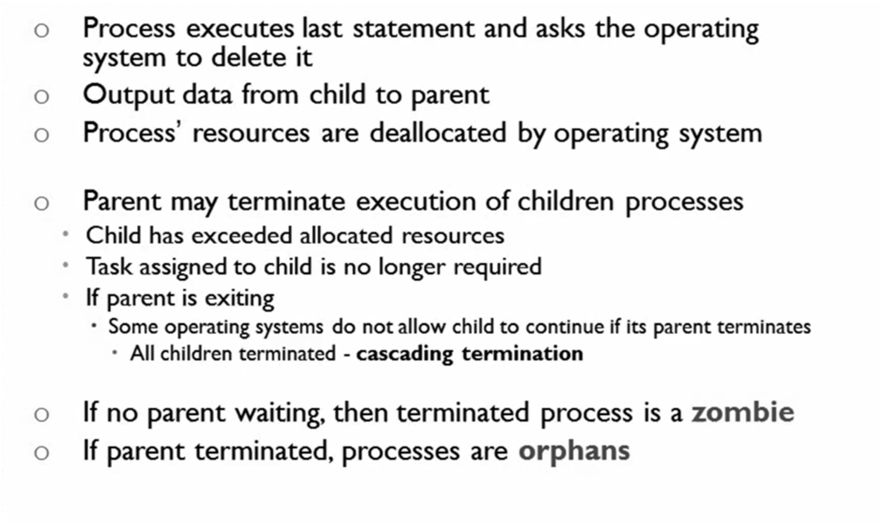
- **Voluntary**
  - ○ Normal exit
  - ○ Internal error or exception
    - Example: exit if no input file is found
- **Involuntary**

Process Termination



- **Voluntary**
  - ○ Normal exit
  - ○ Internal error or exception
    - Example: exit if no input file is found
- **Involuntary**
  - ○ Fatal error
    - Example:divide by zero/ illegal memory access
  - ○ Explicitly killed by another process
    - Example: task manager

Process Termination

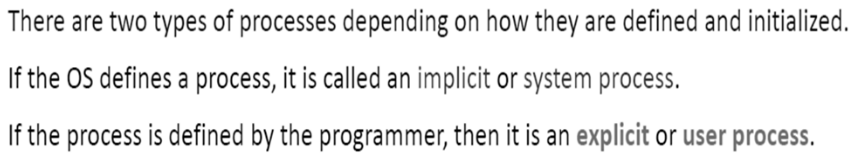


- Process executes last statement and asks the operating system to delete it
- Output data from child to parent
- Process' resources are deallocated by operating system

- Parent may terminate execution of children processes
  - Child has exceeded allocated resources
  - Task assigned to child is no longer required
  - If parent is exiting
    - Some operating systems do not allow child to continue if its parent terminates
      - All children terminated - **cascading termination**

- If no parent waiting, then terminated process is a **zombie**
- If parent terminated, processes are **orphans**

Implicit/System and Non-implicit/User processes



There are two types of processes depending on how they are defined and initialized.

If the OS defines a process, it is called an implicit or system process.

If the process is defined by the programmer, then it is an **explicit** or **user process**.

Process Relationship

Concurrent processes

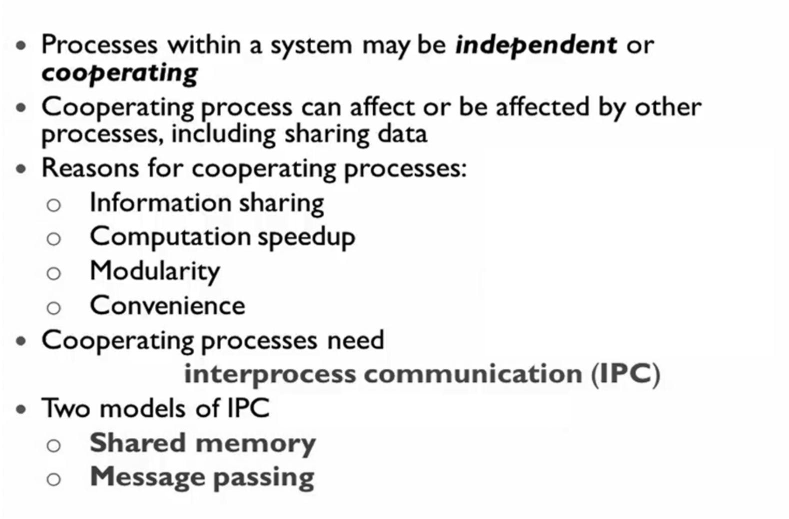
Independent processes

Interacting/cooperating processes

Parent processes

Child processes

Inter-Process Communication (IPC)



- Processes within a system may be ***independent*** or ***cooperating***
- Cooperating process can affect or be affected by other processes, including sharing data
- Reasons for cooperating processes:
  - Information sharing
  - Computation speedup
  - Modularity
  - Convenience
- Cooperating processes need **interprocess communication (IPC)**
- Two models of IPC
  - **Shared memory**
  - **Message passing**