CS3701 - Operating Systems

Chapter 2 - Processes

Department of Computer Science

College of Computer Engineering and Science

Prince Sattam bin Abdulaziz University

AY - 2025/2026

Prepared by Dr. Mahdi Khemakhem, Reviewed by Dr. Essra Aldessouky

Updated on January 22, 2026

Outline

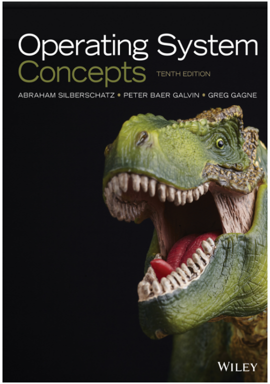
1. Process Concepts
2. Process States
3. Process Control Block (PCB)
4. Process Scheduling
5. Context Switch
6. Key Takeaways
7. Exercise

Textbook Reference

Required Reading

Refer to Chapter 3 of the textbook:

- **Title:** *Operating System Concepts*
- **Authors:** Abraham Silberschatz, Peter Baer Galvin, and Greg Gagne
- **Publisher:** John Wiley & Sons
- **Edition:** 10^{th} Edition (2018)



Outline

Process Concepts

Definition

A **process** is a *program in execution* that **progresses sequentially** through its **instructions**.
→It encompasses the *program code*, *current activity*, and *allocated resources*.

Program vs. Process

- A **program** is a **passive entity** stored as an executable file on disk.
- A **process** is an **active entity** created when *a program is loaded into memory*.
- Processes are initiated via GUI clicks, command-line execution, or system calls.
- **One program can create multiple processes (e.g., multiple users executing the same program)**

Process Concepts (Contd.)

Process Memory Components

A process in memory consists of several key components:

- **Text Section**: The executable program code.
- **Stack**: Temporary data including function parameters, return addresses, and local variables.
- **Data Section**: Global variables.
- **Heap**: Memory dynamically allocated during runtime.
- **Process State**: Current activity tracked by *program counter* and *CPU registers*.

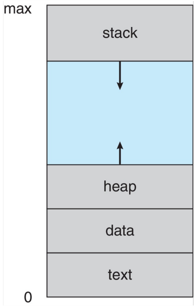


Figure 1: Process in Memory

Process Concepts (Contd.)

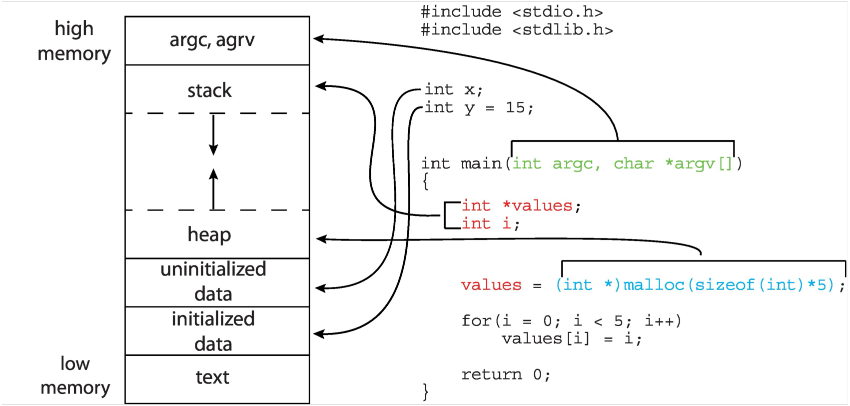


Figure 2: Memory Layout of a C-language Program

Outline

Process States

Process States

A process can be in one of several states during its lifecycle:

- **New**: Process is being **created** and **waiting to be admitted** to the *ready queue* (waiting to be loaded into memory).
- **Ready**: Process is prepared to run in the *ready queue* and **waiting for CPU allocation**.
- **Running**: Process is **currently executing** on the CPU.
- **Waiting (Blocked)**: Process is **waiting for an event** in one or more of the *waiting queues* (e.g., I/O completion).
- **Terminated**: Process has **finished execution**.

Process State Diagram

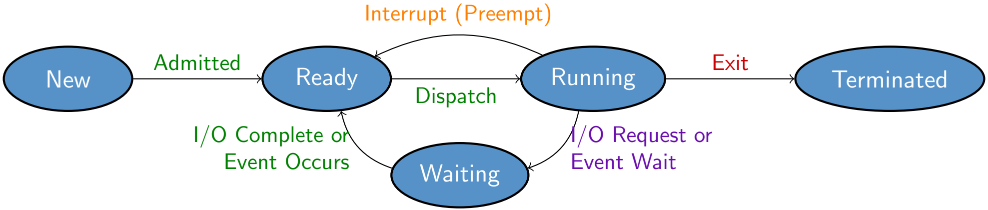


Figure 3: Process State Diagram

Process State Transitions

Process State Transitions

Key state transitions include:

- **New → Ready**: Process is **admitted** by the OS in the *ready queue* and ready to execute.
- **Ready → Running**: *Scheduler* **dispatches** process to CPU.
- **Running → Ready**: Process is **preempted** by *interrupt* or *time quantum* **expires**.
- **Running → Waiting**: Process **requests** I/O or **waits** for an event.
- **Waiting → Ready**: I/O **completes** or event **occurs**.
- **Running → Terminated**: Process **finishes** execution.

Outline

Process Control Block (PCB)

Definition

A **Process Control Block (PCB)** is a *data structure* used by the operating system to **store information** about a **specific process**.
→It contains all the necessary information for *process management* and *context switching*.

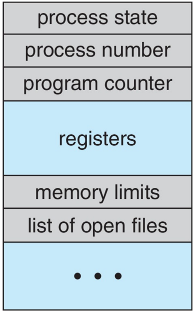


Figure 4: PCB Structure

Process Control Block (PCB) Components

PCB Components

Key components of a PCB include:

- **Process State**: Current state of the process (e.g., New, Ready, Running).
- **Process ID (PID)**: Unique identifier for the process.
- **Program Counter**: Address of the next instruction to execute.
- **CPU Registers**: Values of CPU registers when the process is not running.
- **Memory Management Information**: Details about memory allocation (e.g., base and limit registers).
- **Accounting Information**: CPU usage, time limits, etc.
- **I/O Status Information**: List of I/O devices allocated to the process.

Outline

1. Process Concepts
2. Process States
3. Process Control Block (PCB)
4. Process Scheduling
5. Context Switch
6. Key Takeaways
7. Exercise

Process Scheduling

Definition

Process Scheduling is the *mechanism* by which the operating system **decides**:

- which process in the *ready queue* will be allocated the CPU next.
- how long a process will run on the CPU before being preempted.
- which process will be moved to the *waiting queue* when it requests I/O or waits for an event.
- when a process in the *waiting queue* will be moved back to the *ready queue* after its I/O completes or event occurs.

Process Scheduling (Contd.)

Process Scheduling Key Aspects

Key aspects of process scheduling include:

- **CPU Scheduling**: Selecting processes from the ready queue to execute on the CPU (**see Chapter 4**).
- **Scheduling Algorithms**: Methods used to determine the order of process execution (e.g., FCFS, SJF, Round Robin) (**see Chapter 4**).
- **Context Switching**: Saving and restoring process states during scheduling (**see next Section**).

Process Scheduling Diagram

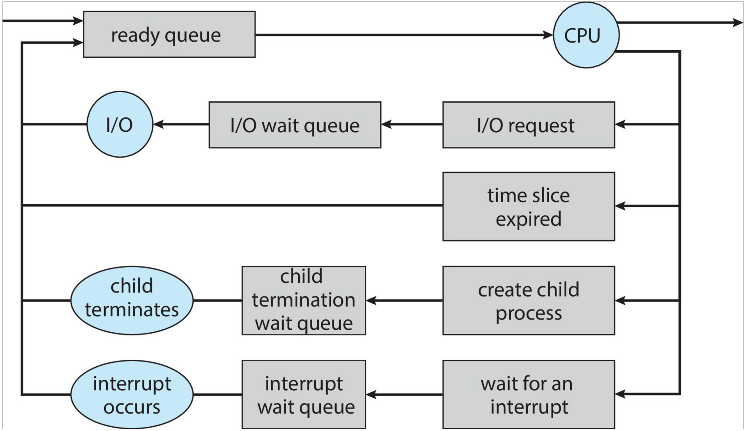


Figure 5: Process Scheduling Diagram

Outline

Context Switch

Definition

A **Context Switch** occurs when the operating system *saves the state* of a **currently running process** and *restores the state* of **another process to be executed**.
→It allows multiple processes to share a single CPU effectively.

Context Switch

Context Switch Steps

Key steps in a context switch:

1. **Save Current State**: The OS saves the current process's state (CPU registers, program counter, stack pointer) into its PCB
2. **Update Current PCB**: The OS updates the PCB to reflect the process's new state (e.g., Running → Ready or Waiting)
3. **Select Next Process**: The scheduler selects a new process from the ready queue based on the scheduling algorithm
4. **Load New State**: The OS loads the selected process's state from its PCB into the CPU registers
5. **Update New PCB**: The OS updates the new process's PCB state (Ready → Running) and resumes execution

Context Switch Diagram

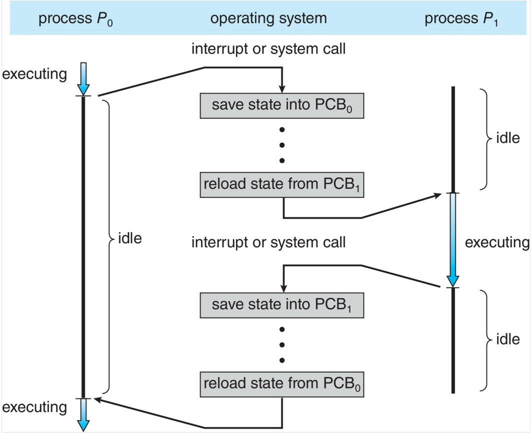


Figure 6: Context Switch Diagram

Context Switch Overhead

Context Switch Overhead

Context switching is pure overhead—no useful work is performed during the switch.

Components of Context Switch Overhead

- **Time Overhead**: Time required to save and restore process states from the PCB (typically implemented as a struct or linked list node in the kernel).
- **Resource Consumption**: CPU cycles and memory bandwidth used to access and update PCB data structures.
- **Cache Impact**: Cache misses when switching between processes, as the new process's data may not be in cache.

Context Switch Overhead Minimization

Minimizing Context Switch Overhead

Strategies to minimize context switch overhead:

- **Efficient PCB Design**: Optimize PCB structure for quick access and minimal size.
- **Batching Processes**: Reduce the frequency of context switches by allowing processes to run longer before preemption.
- **Lightweight Processes (Threads)**: Use threads within a process to reduce the need for full context switches (see Chapter 3).
- **Optimized Scheduling Algorithms**: Choose algorithms that minimize unnecessary context switches.

Outline

Key Takeaways

Key Takeaways

- A **process** is a program in execution, encompassing code, current activity, and resources.
- Processes transition through states: **New**, **Ready**, **Running**, **Waiting**, and **Terminated**.
- The **Process Control Block (PCB)** stores essential information about each process.
- **Process Scheduling** determines which process runs next and manages CPU allocation.
- A **Context Switch** saves the state of one process and restores another's, enabling multitasking.
- Context switching incurs **overhead**, which can be **minimized** through **efficient PCB design** and **scheduling strategies**.

Outline

Exercise

Exercise 2.1

We consider some context switches occurring when a CPU dispatcher switches between three processes P_1, P_2 and P_3. The following table shows the needed times (in millisecond) to save/load each process state to/from its PCB, respectively.

Process	Save Time (ms)	Load Time (ms)
P_1	a	b
P_2	c	d
P_3	e	f

Exercise (Contd.)

Exercise 2.1 (Contd.)

1. Calculate the time consumed by the context-switch if the process P_3 leaves the CPU and replaced by P_2.

   Solution 2.1.1

   Context Switch Time $=$ Save Time of $P_3 +$ Load Time of $P_2 = e + d$ ms.

2. Calculate the total time of 5 context switch between P_1 and P_3 when P_3 is initially on the CPU and P_1 is initially in the ready queue.

   Solution 2.1.2

   Total Context Switch Time $= 3 \times$ (Save Time of $P_3 +$ Load Time of P_1) $+$ $2 \times$ (Save Time of $P_1 +$ Load Time of P_3) $= 3 \times (e + b) + 2 \times (a + f)$ ms.

Exercise (Contd.)

Exercise 2.1 (Contd.)

3. We consider that the burst times of P_1, P_2 and P_3 are 9, 5, and 18 milliseconds, respectively. The CPU scheduler use the Round Robin to schedule the three processes with a time quantum $q = 4$. By considering the context switch times, calculate the total time to execute the three processes P_1, P_2 and P_3. We assume that: (i) all processes are initially in the ready queue, (ii) P_3 is at the first position in the ready queue, (iii) P_1 is at the second position in the ready queue, and (iv) when a process terminates, it is necessary to save its PCB.

Solution 2.1.3

Total Time = Total Burst Time + Total Context Switch Time =

$$\underbrace{(9+5+18)}_{\text{Burst Time}} + \underbrace{(3\times(a+b)+2\times(c+d)+4\times(e+f))}_{\text{Context Switch Time}} \text{ ms.}$$

Note: If $a = b = c = d = e = f = 1$ ms, then Total Time $= 32 + (3\times 2 + 2\times 2 + 4\times 2) = \underbrace{32}_{\text{Burst Time}} + \underbrace{18}_{\text{Context Switch Time}} = 50$ ms. The context switch overhead is $\frac{18}{50} \times 100 = 36\%$ that is considered very high!

End of Chapter 2