Information Technology Institute

Introduction to Operating Systems

Prepared by:

Zeyad Ashraf

Processes and Scheduling

Content

- Introduction to Scheduling
- Process Concept
- Process Contents
- Process States
- Process Control Block (PCB)
- Context Switching
- Scheduling

Course Outlines

2.1 Introduction to Scheduling

- One of the primary functionalities of the OS would be to provide the ability to run **multiple**, concurrent applications on the system and efficiently **manage** their access to system resources.
- As many programs try to run in parallel, there may be **competing** and **conflicting** requests to access hardware resources such as CPU, memory, and other devices.
- The operating system streamlines these requests and orchestrates the execution at runtime by **scheduling** the execution and subsequent requests to avoid conflicts.
- Before we go into the details of scheduling responsibilities and algorithms, it is important to know some background about the basic concepts of ***program execution***, ***specifically processes*** and ***threads***.

2.2 Program and Process Concept

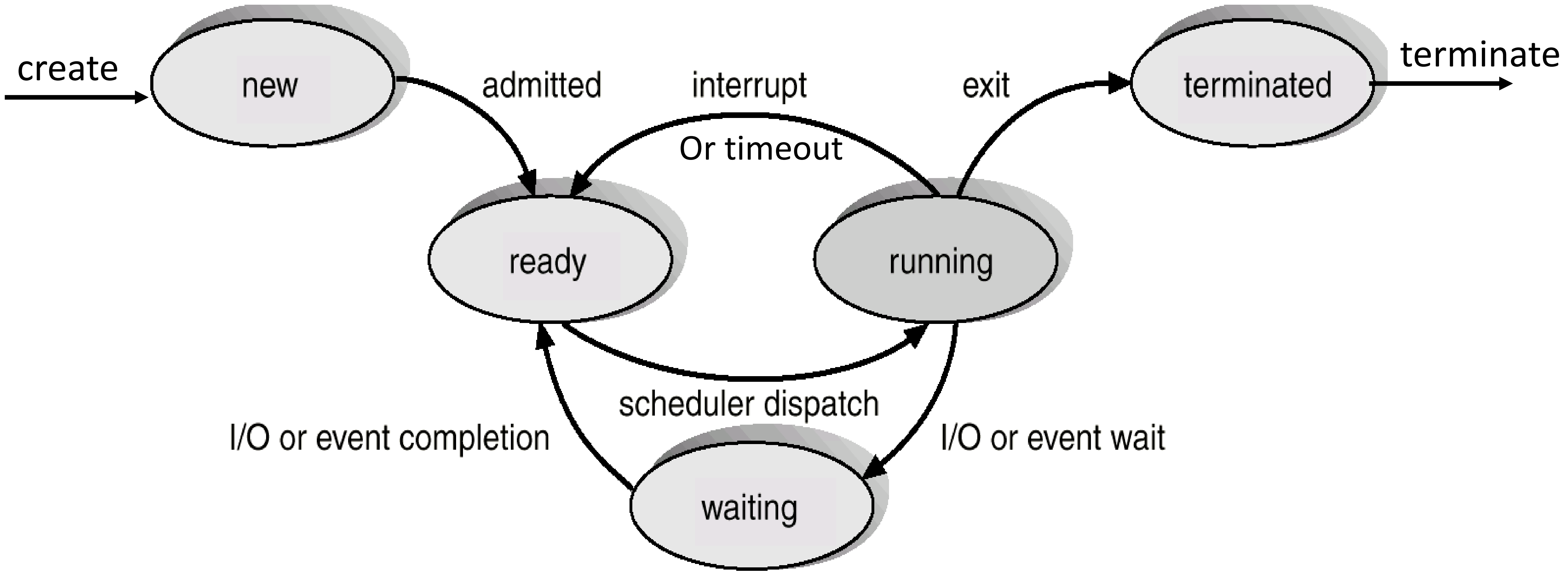
- When a software developer builds a solution, the set of capabilities it provides is usually static and embedded in the form of processed code that is built for the OS. This is typically referred to as the **program**.
- When the program gets triggered to run, the OS assigns **a process ID** and other **metrics for tracking**.
- Note that in the context of different operating systems, jobs and processes may be used interchangeably. However, ***process*** ***<u>refer to a program in execution</u>***.
- Job vs process

2.3 Process Contents

- Text section
  - Program instructions
- Program counter
  - Next instruction address
- Stack Section
  - Local variables
  - Return addresses
  - Method parameters
- Heap section
  - Global and Static Variables
  - Dynamic Allocation (at run-time)

- Text section
  - Program instructions
- Data Section
  - Global and Static Variables
- Stack Section
  - Local variables
  - Return addresses
  - Method parameters
  - **PC**: Next instruction address
- Heap section
  - Dynamic Allocation (at run-time)

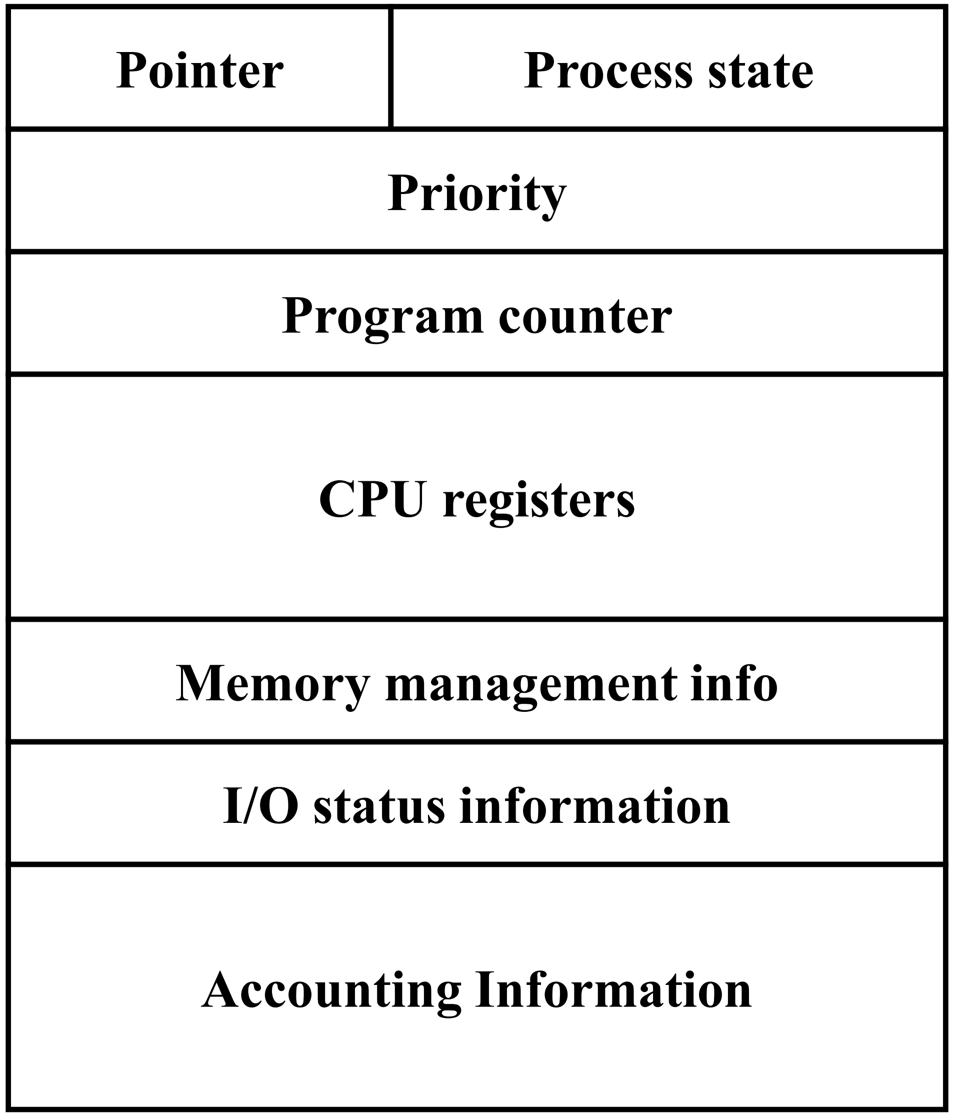
2.4 Process States



2.4 Process States

- When a program gets triggered for execution, typically say using a double click of the EXE (or using a CreateProcess() API in Windows), a new process is created.
- process typically supports multiple states of readiness in its lifecycle:
  - **New**: The process is being created.
  - **Running**: Instructions are being executed.
  - **Waiting**: The process is waiting for some event to occur.
  - **Ready**: The process is waiting to be assigned to a processor.
  - **Terminated or Exit**: The process has finished execution.
- There could be more than one CPU core on the system and hence the OS could schedule on any of the available cores.
- In order to avoid switching of context between CPU cores every time, the OS tries to limit such frequent transitions.
- The OS monitors and manages the transition of these states seamlessly and maintains the states of all such processes running on the system.

2.5 Process Control Block (PCB)



2.5 Process Control Block (PCB)

- The **process ID** is a unique identifier for the instance of the process that is to be created or currently running.
- The **process state** determines the current state of the process, described in the preceding section.
- The **pointer** could refer to the hierarchy of processes (e.g., if there was a parent process that triggered this process).
- The **priority** refers to the priority level (e.g., high, medium, low, critical, real time, etc.) that the OS may need to use to determine the scheduling.
- **Affinity and CPU register** details include if there is a need to run a process on a specific core. It may also hold other register and memory details that are needed to execute the process.

2.5 Process Control Block (PCB)

- The **program counter** usually refers to the next instruction that needs to be run.
- The **I/O status information,** like which devices assigned, limits, and so on that is used to monitor each process is also included in the structure.
- The **accounting information** such as paging requirements from memory, timers, how many time unit remaining to finish, … etc

- There could be some modifications to how the PCB looks on different OSs. However, most of the preceding are commonly represented in the PCB.

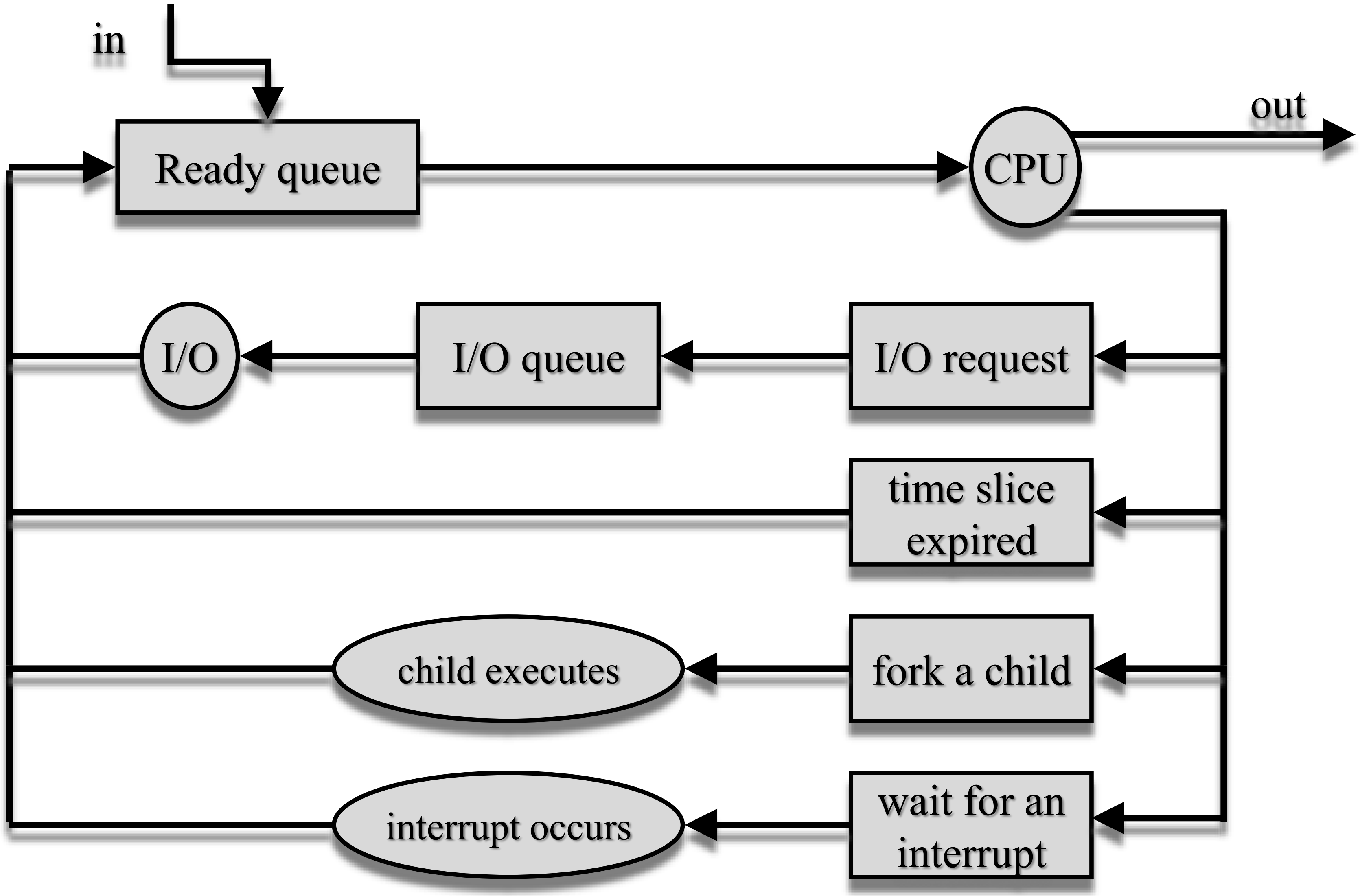
2.6 Context Switching

- The operating system may need to **swap** the currently executing process with another process to allow other applications to run, it does so with the help of **context switching**.
- When a process is executing on the CPU, the process context is determined by the program counter (instruction currently run), the processor status, register states, and various other metrics.
- When the OS needs to swap a currently executing process with another process, it must do the following steps:
  1. **Pause the currently executing process**
  2. **save** the context.
  3. **Switch** to the new process.
  4. When starting a new process, the **OS** must set the context appropriately for that process.
- This ensures that the process executes **exactly from where it was swapped.**

2.7 Scheduling

- The most frequent process states are the **Ready**, **Waiting**, and **Running** states. The operating system will receive requests to run multiple processes at the same time and may need to streamline the execution. It uses **process scheduling queues** to perform this:
  1. **Ready Queue**: When a new process is created, it transitions from New to the **Ready state**. It enters this queue indicating that it is ready to be scheduled.
  2. **Waiting Queue**: When a process gets blocked by a dependent I/O or device or needs to be suspended temporarily, it moves to the Blocked state since it is **waiting for a resource**. At this point, the OS pushes such process to the Waiting queue.
  3. **Job queue** that maintains **all the processes in the system** at any point in time. This is usually needed for bookkeeping purposes.

2.7.1 Scheduling Queues



2.7.2 Scheduling Criteria

- **CPU Utilization and Execution Runtime**: The total amount of time the process is making use of the CPU excluding NOP (no-operation) **idle** cycles.
- **Volume/Execution Throughput**: Some OSs may need to support certain execution **rates** for a given duration.
- **Responsiveness**: The time taken for **completion** of a process and the average **time spent in different queues**.
- **Resource Waiting Time**: The average **time taken on external I/Os** on the system.

  Role ➔ Maximize one and two , Minimize three and four

- ***Note*** *Most OSs try to ensure there is fairness and liveness in scheduling. There are various scheduling algorithms like First Come, First Serve (FCFS), Shortest Job First (SJF), Shortest Remaining Time First (SRT F), Round-Robin, Static/Dynamic Priority, and so on that the OS uses for scheduling of processes.*

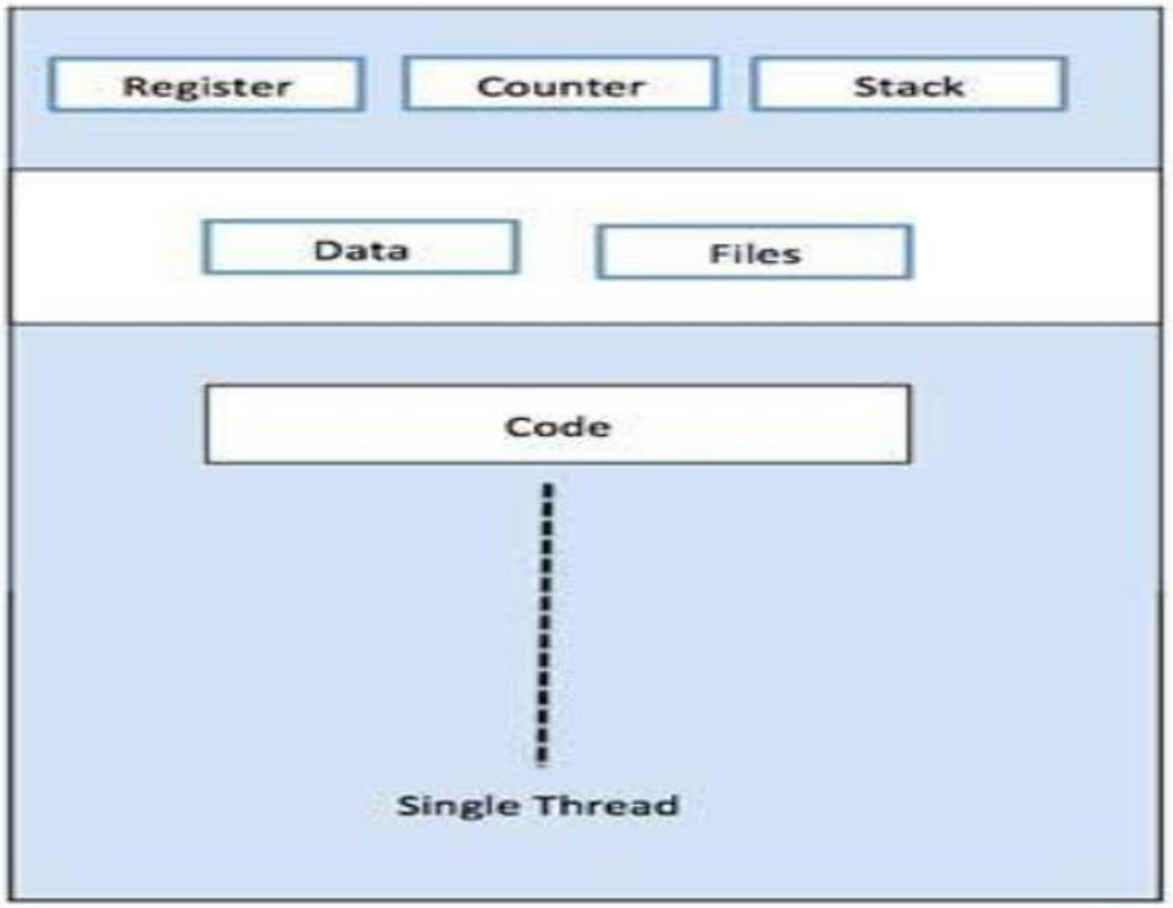
2.8 Thread Concepts

- A thread is a **lightweight process**.
- When a process gets executed, it could create one or more threads internally that can be executed on the processor. These threads have their **own PCB**; program counter, context, and register information, similar to how the process is managed.
- Threads help in performing **parallelism** within the same process.
- Examples:
  - Chatting program: the sending and receiving operations are independent, using threads
  - Strategic games: there are many actions happened at the same time independently, using threads
- *The OS may employ different types of threads, depending on whether they are run from an application. For instance, an application may leverage **user-mode** threads, and a kernel driver may leverage **kernel-mode** threads. The OS also handles switching from user-mode threads to kernel-mode threads as required by a process.*

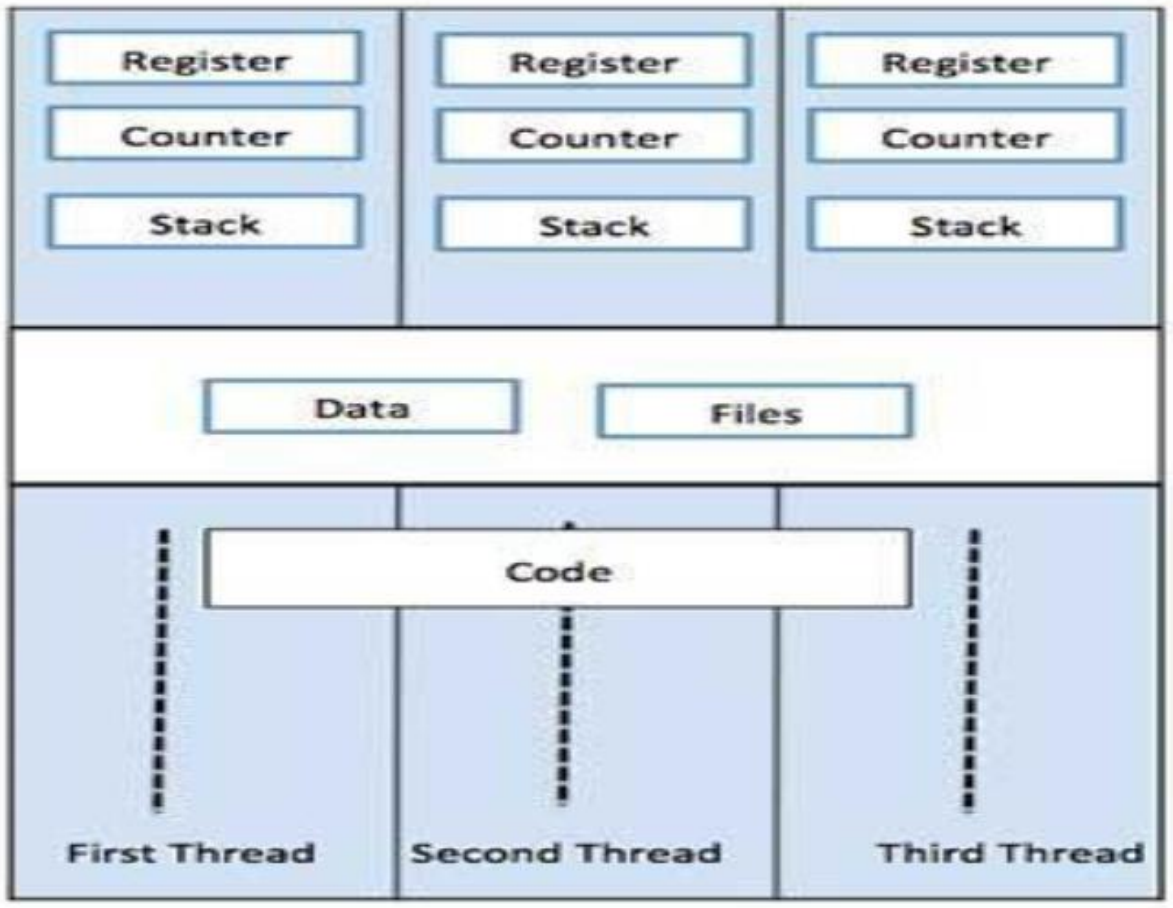
• User mode and kernel mode

Feature	User Mode	Kernel Mode
Definition	A restricted mode where applications run with limited privileges.	A privileged mode where the OS kernel executes with full access to hardware.
Access to Hardware	Cannot directly access hardware or I/O devices.	Has direct access to all hardware and system resources.
Access to Memory	Can only access its own address space (user space).	Can access both kernel and user address spaces.
Mode Bit (CPU Flag)	Mode bit = 1 (indicates user mode).	Mode bit = 0 (indicates kernel mode).
Crash Impact	If an application crashes, only that application is affected.	If the kernel crashes, the entire system can fail (system crash / blue screen).
System Calls	Needs to make a **system call** (trap) to switch to kernel mode for privileged operations (like file I/O).	Executes system calls directly without switching.
Performance	Slightly slower due to restrictions and context switches.	Faster for system-level operations.
Examples	Running apps like Chrome, Word	Running OS components

• Single-threaded & a multithreaded process.



Single Process P with single thread



Single Process P with three threads

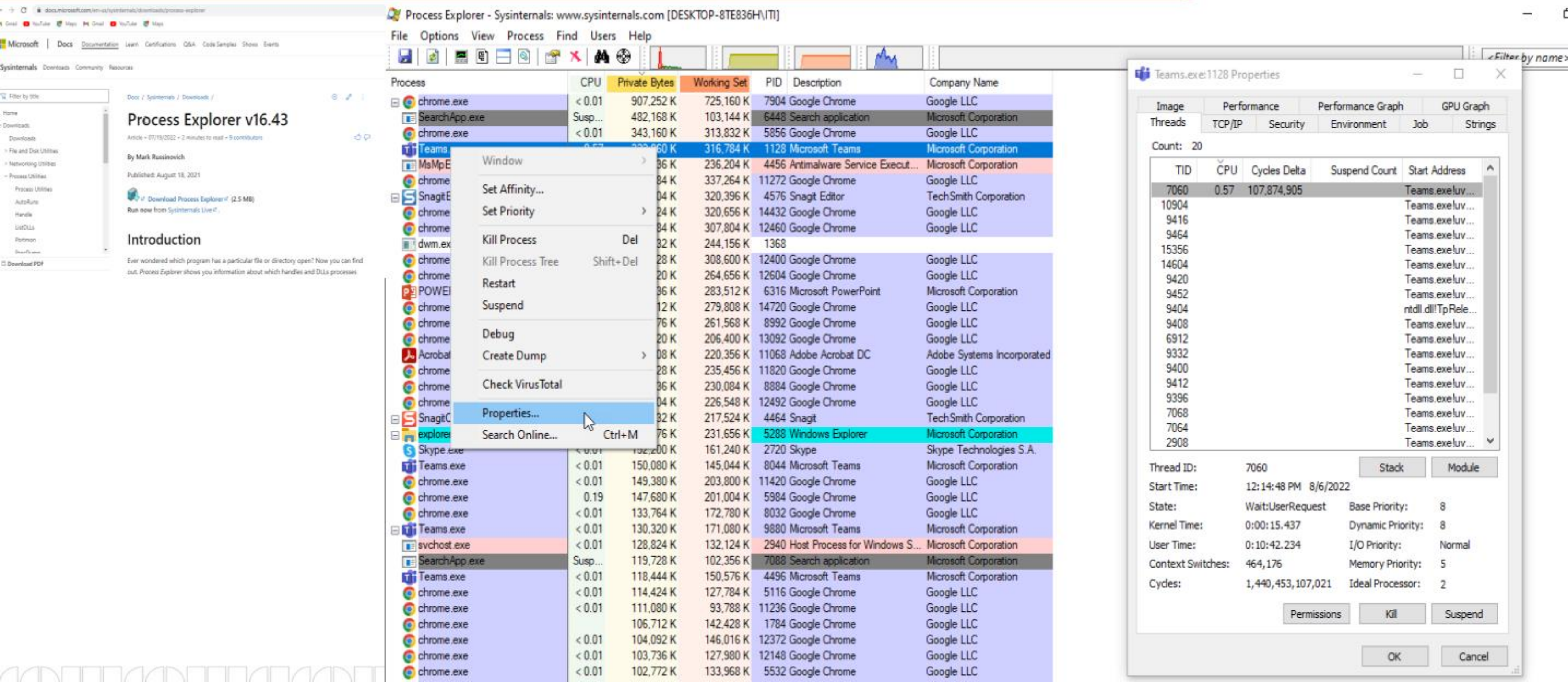
• Comparison between thread and process

S/N	Process	Thread
1	Process is heavy weight or resource intensive.	Thread is light weight, taking lesser resources than a process.
2	Process switching needs interaction with operating system.	Thread switching does not need to interact with operating system.
3	In multiple processing environments, each process executes the same code but has its own memory and file resources.	All threads can share same set of open files, child processes.
4	If one process is blocked, then no other process can execute until the first process is unblocked.	While one thread is blocked and waiting, a second thread in the same task can run.
5	Multiple processes without using threads use more resources.	Multiple threaded processes use fewer resources.
6	In multiple processes each process operates independently of the others.	One thread can read, write or change another thread's

Benefits of Threads

- **Effective Utilization of Multiprocessor system:** When you have more than one thread in one process, you can schedule more than one thread in more than one processor.
- **Faster context switch:** The context switching period between threads is less than the process context switching .
- **Enhanced throughput of the system:** the number of jobs done in the unit time increases. That is why the throughput of the system also increases.
- **Communication:** Multiple-thread communication is simpler than process communication because the threads share the same address space, while in process.
- **Resource sharing:** Resources can be shared between all threads within a process, such as code, data, and files.

• Thread Concepts



Microsoft | Docs | Documentation | Learn | Certifications | Q&A | Code Samples | Shows | Events
Sysinternals | Downloads | Community | Resources
Docs / Sysinternals / Downloads /

Process Explorer v16.43

Article • 07/19/2022 • 2 minutes to read • 9 contributors

By Mark Russinovich

Published: August 18, 2021

Download Process Explorer (2.5 MB)

Run now from Sysinternals Live

Introduction

Ever wondered which program has a particular file or directory open? Now you can find out. Process Explorer shows you information about which handles and DLLs processes

Process Explorer - Sysinternals: www.sysinternals.com [DESKTOP-8TE836H\ITI]

Process	CPU	Private Bytes	Working Set	PID	Description	Company Name
chrome.exe	< 0.01	907,252 K	725,160 K	7904	Google Chrome	Google LLC
SearchApp.exe	Susp...	482,168 K	103,144 K	6448	Search application	Microsoft Corporation
chrome.exe	< 0.01	343,160 K	313,832 K	5856	Google Chrome	Google LLC
Teams	0.57	3[illegible]60 K	316,784 K	1128	Microsoft Teams	Microsoft Corporation
MsMpE		36 K	236,204 K	4456	Antimalware Service Execut...	Microsoft Corporation
chrome		84 K	337,264 K	11272	Google Chrome	Google LLC
SnagitE		04 K	320,396 K	4576	Snagit Editor	TechSmith Corporation
chrome		24 K	320,656 K	14432	Google Chrome	Google LLC
chrome		84 K	307,804 K	12460	Google Chrome	Google LLC
dwm.ex		32 K	244,156 K	1368		
chrome		28 K	308,600 K	12400	Google Chrome	Google LLC
chrome		20 K	264,656 K	12604	Google Chrome	Google LLC
POWEI		36 K	283,512 K	6316	Microsoft PowerPoint	Microsoft Corporation
chrome		12 K	279,808 K	14720	Google Chrome	Google LLC
chrome		76 K	261,568 K	8992	Google Chrome	Google LLC
chrome		20 K	206,400 K	13092	Google Chrome	Google LLC
Acrobat		08 K	220,356 K	11068	Adobe Acrobat DC	Adobe Systems Incorporated
chrome		28 K	235,456 K	11820	Google Chrome	Google LLC
chrome		36 K	230,084 K	8884	Google Chrome	Google LLC
chrome		04 K	226,548 K	12492	Google Chrome	Google LLC
SnagitC		32 K	217,524 K	4464	Snagit	TechSmith Corporation
explorer		76 K	231,656 K	5288	Windows Explorer	Microsoft Corporation
Skype.exe	[illegible]	1[illegible],200 K	161,240 K	2720	Skype	Skype Technologies S.A.
Teams.exe	< 0.01	150,080 K	145,044 K	8044	Microsoft Teams	Microsoft Corporation
chrome.exe	< 0.01	149,380 K	203,800 K	11420	Google Chrome	Google LLC
chrome.exe	0.19	147,680 K	201,004 K	5984	Google Chrome	Google LLC
chrome.exe	< 0.01	133,764 K	172,780 K	8032	Google Chrome	Google LLC
Teams.exe	< 0.01	130,320 K	171,080 K	9880	Microsoft Teams	Microsoft Corporation
svchost.exe	< 0.01	128,824 K	132,124 K	2940	Host Process for Windows S...	Microsoft Corporation
SearchApp.exe	Susp...	119,728 K	102,356 K	7088	Search application	Microsoft Corporation
Teams.exe	< 0.01	118,444 K	150,576 K	4496	Microsoft Teams	Microsoft Corporation
chrome.exe	< 0.01	114,424 K	127,784 K	5116	Google Chrome	Google LLC
chrome.exe	< 0.01	111,080 K	93,788 K	11236	Google Chrome	Google LLC
chrome.exe		106,712 K	142,428 K	1784	Google Chrome	Google LLC
chrome.exe	< 0.01	104,092 K	146,016 K	12372	Google Chrome	Google LLC
chrome.exe	< 0.01	103,736 K	127,980 K	12148	Google Chrome	Google LLC
chrome.exe	< 0.01	102,772 K	133,968 K	5532	Google Chrome	Google LLC

Teams.exe:1128 Properties

Image | Performance | Performance Graph | GPU Graph
Threads | TCP/IP | Security | Environment | Job | Strings

Count: 20

TID	CPU	Cycles Delta	Suspend Count	Start Address
7060	0.57	107,874,905		Teams.exe!uv...
10904				Teams.exe!uv...
9416				Teams.exe!uv...
9464				Teams.exe!uv...
15356				Teams.exe!uv...
14604				Teams.exe!uv...
9420				Teams.exe!uv...
9452				Teams.exe!uv...
9404				ntdll.dll!TpRele...
9408				Teams.exe!uv...
6912				Teams.exe!uv...
9332				Teams.exe!uv...
9400				Teams.exe!uv...
9412				Teams.exe!uv...
9396				Teams.exe!uv...
7068				Teams.exe!uv...
7064				Teams.exe!uv...
2908				Teams.exe!uv...

Thread ID:	7060		
Start Time:	12:14:48 PM 8/6/2022		
State:	Wait:UserRequest	Base Priority:	8
Kernel Time:	0:00:15.437	Dynamic Priority:	8
User Time:	0:10:42.234	I/O Priority:	Normal
Context Switches:	464,176	Memory Priority:	5
Cycles:	1,440,453,107,021	Ideal Processor:	2

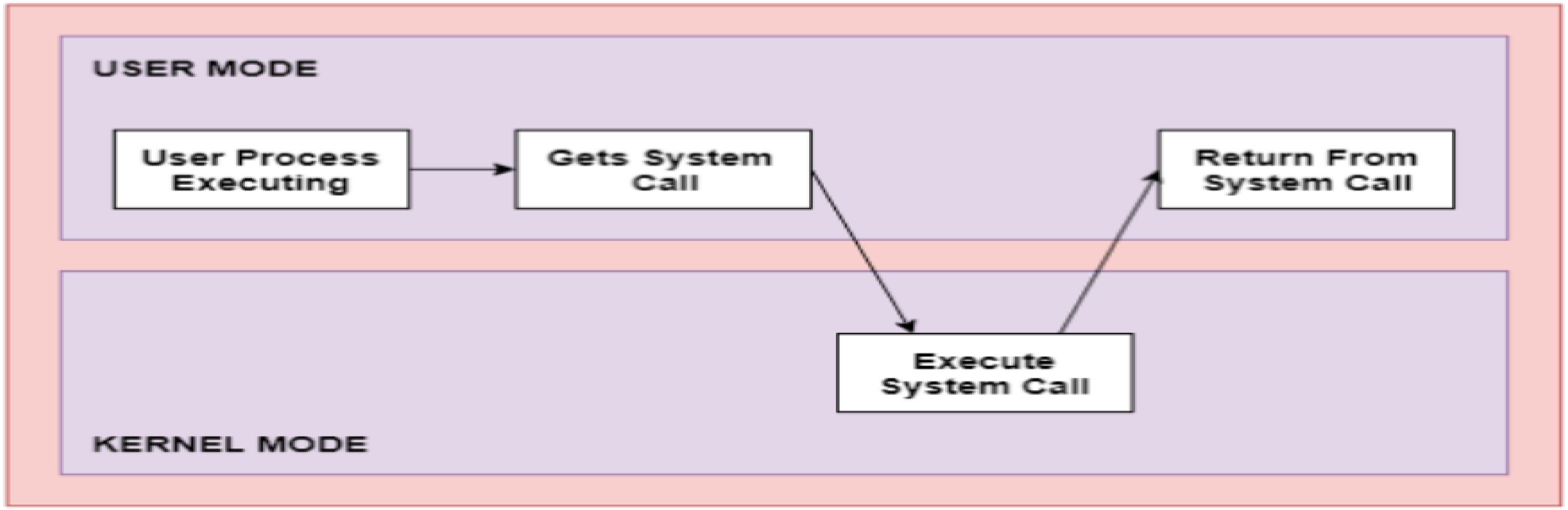
• SYSTEM CALL

The **interface between a process and an operating system** is provided by **system calls**.

In general, system calls are available as assembly language instructions. They are also included in the manuals used by the assembly level programmers.

System calls are usually made when a process (in user mode) **requires access to a resource.** Then it requests the kernel to provide the resource via a system call.

• Execution of the system call



the processes execute normally in the user mode until a system call interrupts this.
Then the system call is executed in the kernel mode. After the execution of the system call, the control returns to the user mode and execution of user processes can be resumed.

• CPU Scheduling Algorithms

1. **FCFS (First Come, First Served)**
2. **SJF (Shortest Job First) – Non-Preemptive**
3. **SJF (Shortest Job First) – Preemptive**
4. **Priority Scheduling – Non-Preemptive**
5. **Priority Scheduling – Preemptive**
6. **RR (Round Robin)**

• 1st Case : FCFS (First Come First Served)

Suppose that the processes arrive at time 0, in the order: P1 , P3 , P2 , P4

Draw Gantt Chart and calculate the average waiting time using the given table ??

Process	Burst Time
P1	3
P2	9
P3	5
P4	7

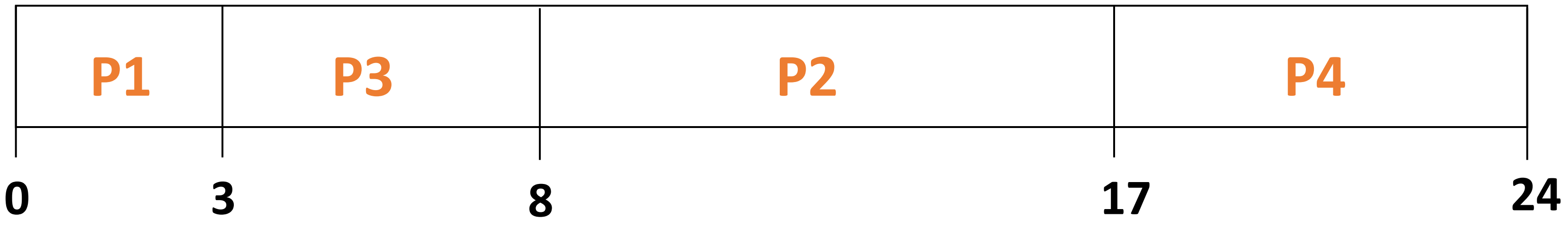
Waiting time : start time – arrival time

P1 = 0

P2 = 8

P3 = 3

P4 = 17



Average waiting time = (0 + 8 + 3 + 17) / 4 = 7

2nd Case : FCFS (First Come First Served)

Draw Gantt Chart and calculate the average waiting time using the given table ??

Process	Burst Time	Arrival Time
P1	20	0
P2	12	3
P3	4	2
P4	9	5

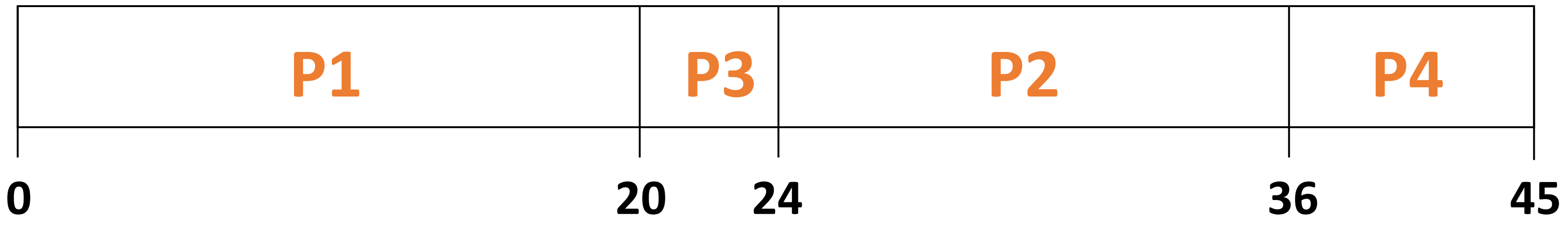
Waiting time : start time – arrival time

P1 = 0 – 0 = 0

P2 = 24 – 3 = 21

P3 = 20 – 2 = 18

P4 = 36 – 5 = 31



Average waiting time = (0 + 21 + 18 + 31) / 4 = 70 / 4

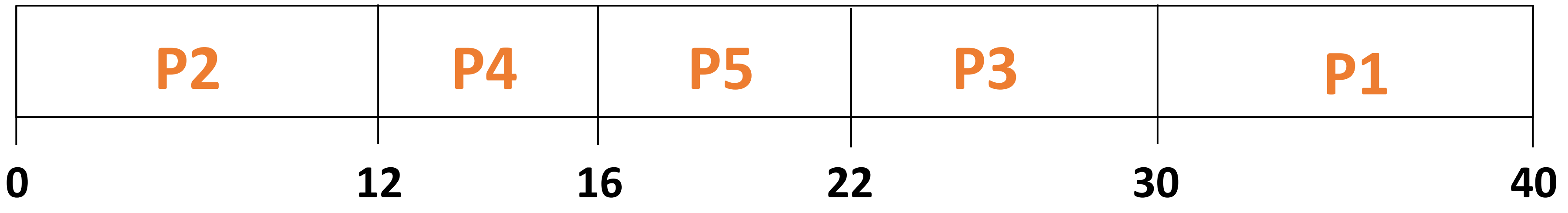
• 3rd Case : SJF (short job first) non-Preemptive

Draw Gantt Chart and calculate the average waiting time using the given table ??

Process	Burst Time	Arrival Time
P2	12	0
P3	8	3
P4	4	5
P1	10	10
P5	6	12

Waiting time : start time – arrival time

P1 = 30 – 10 = 20
P2 = 0 – 0 = 0
P3 = 22 – 3 = 19
P4 = 12 – 5 = 7
P5 = 16 – 12 = 4



Average waiting time = (20 + 0 + 19 + 7 + 4) / 5 = 50 / 5 = 10

• 4th Case : SJF (short job first) Preemptive

Draw Gantt Chart and calculate the average waiting time using the given table ??

Process	Burst Time	Arrival Time
~~P2~~	~~12~~ 9	~~0~~
~~P3~~	~~8~~ 6	~~3~~
~~P4~~	~~4~~	~~5~~
~~P1~~	~~10~~	~~10~~
~~P5~~	~~6~~	~~12~~

Waiting time : start time – arrival time

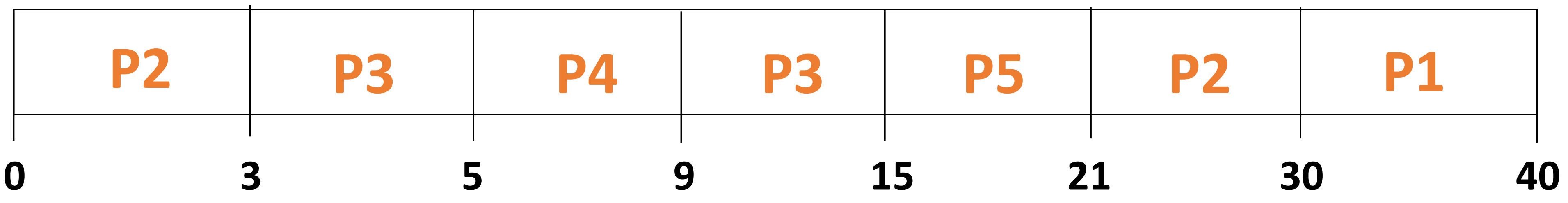
P1 = 30 – 10 = 20

P2 = (0 – 0) + (21 - 3) = 18

P3 = (3 – 3) + (9 - 5) = 4

P4 = (5 – 5) = 0

P5 = 15 – 12 = 3



Average waiting time = (20 + 18 + 4 + 0 + 3) / 5 = 45 / 5 = 9

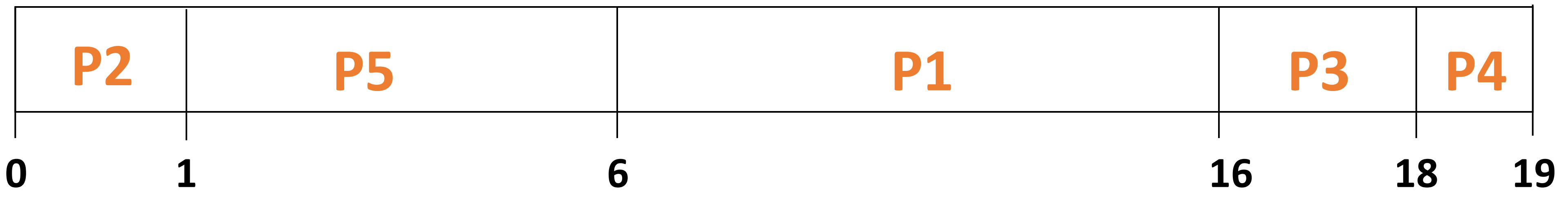
• 5th Case : Priority Scheduling non-Preemptive

Draw Gantt Chart and calculate the average waiting time using the given table ??

Process	Burst Time	Priority	Arrival Time
P1	10	3	All Processes Arrived at The Same Time
P2	1	1	
P3	2	4	
P4	1	5	
P5	5	2	

Waiting time :
start time – arrival time

P1 = 6
P2 = 0
P3 = 16
P4 = 18
P5 = 1



Average waiting time = (6 + 0 + 16 + 18 + 1) / 5 = 41 / 5 = 8.2

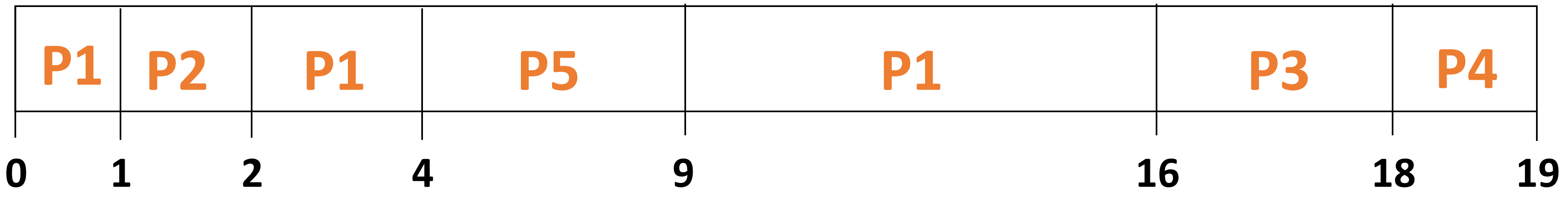
6th Case : Priority Scheduling Preemptive

Draw Gantt Chart and calculate the average waiting time using the given table ??

Process	Burst Time	Priority	Arrival Time
~~P1~~	~~10~~ ~~9~~ ~~7~~	~~3~~	~~0.0~~
P2	1	1	1.0
P3	2	4	2.0
P4	1	5	3.0
P5	5	2	4.0

Waiting time :
start time – arrival time

P1 = (0 - 0)+(2 - 1)+(9 - 4) = 6
P2 = 1 – 1 = 0
P3 = 16 – 2 = 14
P4 = 18 – 3 = 15
P5 = 4 – 4 = 0



Average waiting time = (6 + 0 + 14 + 15 + 0) / 5 = 35 / 5 = 7

• 7th Case : Round Robin (RR)

Draw Gantt Chart and Calculate The Average Waiting Time , where **Quantum = 5 ms**

Process	Burst Time
~~P1~~	~~12 7 2~~
~~P2~~	~~8 3~~
~~P3~~	~~4~~
~~P4~~	~~10 5~~
~~P5~~	~~5~~

Waiting time :

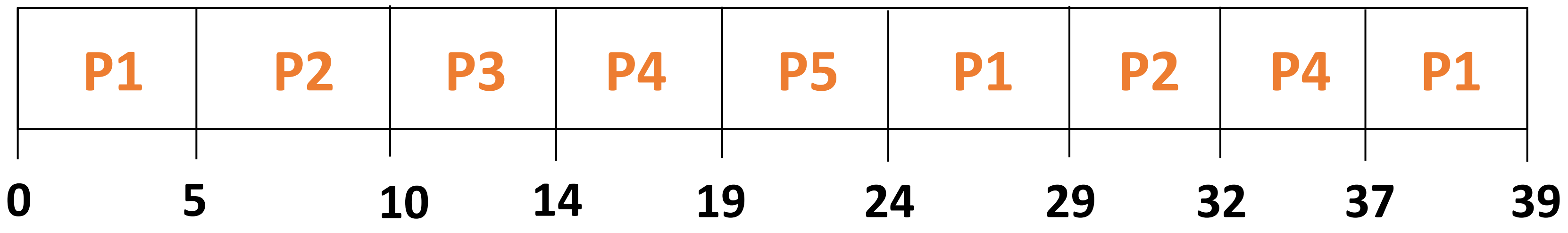
P1 = 0 + (24 - 5) + (37 - 29) = 27

P2 = 5 + (29 - 10) = 24

P3 = 10

P4 = 14 + (32 - 19) = 27

P5 = 19



Average waiting time = (27 + 24 + 10 + 27 + 19) / 5 = 107 / 5 = 21.4

Thank You

With My Best Wishes

Zeyad ashraf