PURBANCHAL UNIVERSITY

SCHOOL OF ENGINEERING (PUSOE)

# Operating System

Assignment: 02

Submitted by: **Barsha Karn**

Roll no : 13 'A'

Faculty: Computer Engineering

Batch Year : 2020

Year/ Semester: 3rd/5th

College: PUSOE, Brt

Submitted to:

**Er. Deeoranjan Dongol sir**

Assignment - 02

1. How does process differ from program? Explain the process states with the help of diagram.

⇒

| Program | Process |
|---|---|
| ① It is a set of instructions that are used to perform a certain task. | ① A process is a program being executed. |
| ② A program is stored in secondary memory and has a higher ~~priority~~ lifespan when compared with the process. | ② A process is active until the program is executed and has a short lifespan as compared to a program. |
| ③ A program does not have computational time and cost. | ③ A process consumes significant computational time to execute. |
| ④ A program does not have resource requirements. A program only needs memory space to store all the instructions. | ④ A process needs high resources as compared to a program, it requires CPU, disk, input/output, memory address, etc. |
| ⑤ A program doesn't have a significant overhead. | ⑤ A process has considerable overhead. |
| ⑥ It is considered as a passive entity. | ⑥ It is considered as active entity. |
| ⑦ Examples of programs include Microsoft word, Google Chrome, ~~r~~ Minecraft, etc. | ⑦ Examples of process include multiple tabs open in chrome, multiple documents open in notepad, Ms Excel with multiple open workbooks, etc. |

In Operating systems, a process goes through several states during its lifetime. Processes can transition between these states based on various factors, events and scheduling decision made by the operating system. The states are illustrated with the help of below diagram.

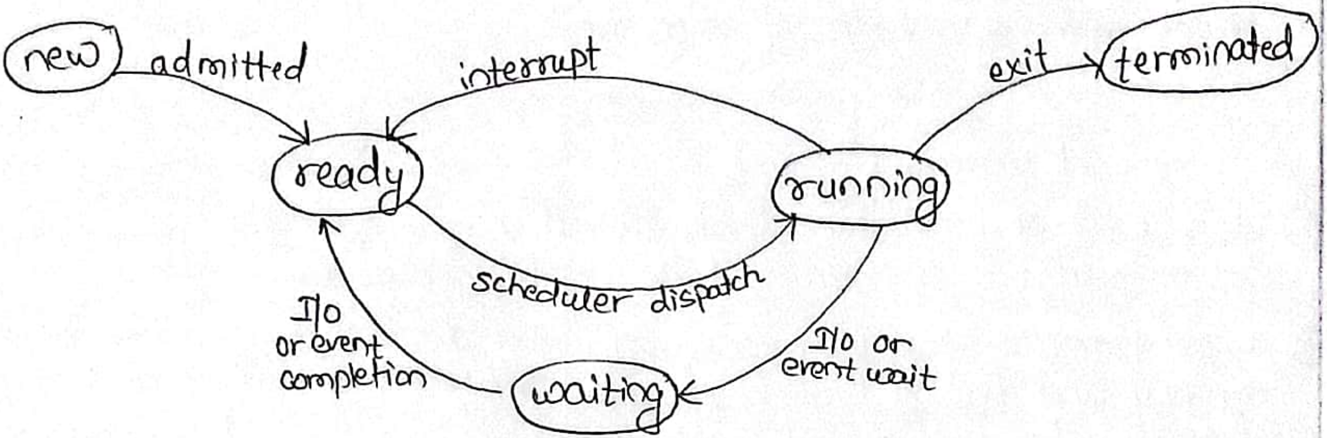


Fig: Process state transition diagram

New:- This is the first state of process life cycle. The process is created.

Ready:- When process creation gets completed, the process comes into a ready state.

Running:- When CPU is allocated to the process from the ready queues, the process state changes to running.

Waiting:- In case other event require the resources used by the process, it is brought back into main memory and the state is changed to waiting.

Terminated:- When entire set of instruction is executed and the process is completed, it is then terminated.

New → Ready :- When a process is created, it is in a new state. It moves to the ready state when the OS has allocated resources to it and it is ready to be executed.

Ready → Running :- When the CPU becomes available, the OS selects a process from the ready queue depending on various scheduling algorithms and moves to it to the running state.

Running → Ready :- When a running process is preempted by the OS, it moves to the ready state.

Running → Waiting :- When a process need to wait for event to occur, it moves to waiting state.

Waiting → Ready :- When the event for which process was waiting gets completed, the process moves to the ready state.

Running → terminated :- When the process completes, its execution or is terminated by the OS, it moves to the terminated state.

2. Explain Process Control Block (PCB) in brief.

⇒ A process control Block (PCB) is a data structure used ~~by~~ ~~as~~ an OS to manage and control processes or tasks in a computer system. It contains essential information about a specific process, allowing the OS to effectively manage and schedule multiple processes concurrently. Some key components typically found in PCB include :

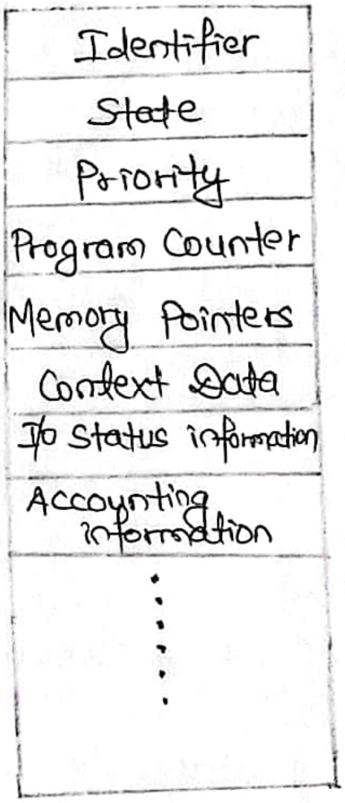
Identifier (PID) : A unique identifier assigned to each process, helping the OS keep track of and manage individual process.

State : state indicates whether the process is ready, running or waiting. This helps OS determine which processes are eligible for execution.

Priority : Some systems assign priority levels to processes to determine their importance in scheduling.

Program Counter : It stores the address of next instruction to be executed within the process.

Memory Pointers : They are data fields that store information about process's memory allocation and management. It keeps track of and manage a process's memory.



| |
|---|
| Identifier |
| State |
| Priority |
| Program Counter |
| Memory Pointers |
| Context Data |
| I/o Status information |
| Accounting information |
| ⋮ |

Fig : PCB

Context Data : It stores the state of a process at a specific point in time. This is required for context switching, allowing OS to pause the execution of one process and start or resume the execution of another.

I/O status information : It is a data structure used to keep track of a process's input/output (I/O) operations and their status which contains information related to I/O requests initiated by the process.

Accounting Information :- It refers to the section of PCB where information about the process's resource usage and accounting details is stored. It keeps data like how much CPU time, memory and other resources a process has consumed or allocated.

3. Explain how multithreading improve performance over a single threaded solution.

⇒ A thread is a single sequence stream within a process. They comprise of thread ID, Program counter, a register set and a stack. They share code section, data section and resource with other threads.

Multithreading is a technique used in OS to improve the performance and responsiveness of OS. It allows multiple threads to share the same resources of a single process such as the CPU, memory and I/O devices.

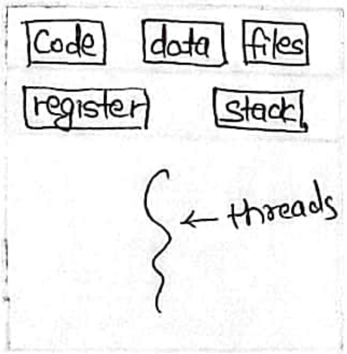


Fig: Single thread Process

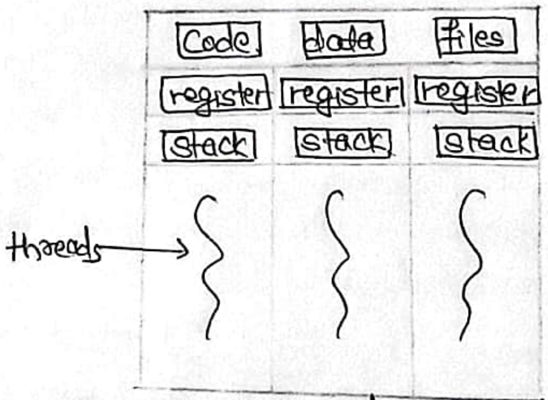


Fig: Multithread Process

Multithreading can significantly improve performance over a single threaded solution in several ways:

i) Parallelism- Multithreading allows multiple threads to run concurrently on multiple CPU cores (if available).

ii) Responsiveness- Multithreading can improve responsiveness. One thread can handle user input and interface updates, while other threads executes resource intensive tasks.

iii) Improved throughput - Multithreading is beneficial for tasks that can be divided into smaller, independent subtasks.

iv) Efficient Resource utilization- Instead of leaving CPU cores idle, multithreading enable them to be fully utilized, maximizing the CPU's processing power.

v) Reduced Latency- Multithreading reduces latency by allowing critical tasks to execute without waiting for non-critical tasks to complete.

4. Differentiate between Process and thread.

⇒

| Process | Thread |
|---|---|
| ① It is a program in execution. | ① It is a basic unit of CPU utilization. |
| ② Heavy weight | ② Light weight. |
| ③ Unit of allocation - resources, privi-ledges, etc. | ③ Unit of execution - program counter, stack pointer, register, etc. |
| ④ Inter process communication; is expen-sive; needs to context switch | ④ Inter ~~process~~ thread communication; is cheap; can use process memory & may not need to context switch. |
| ⑤ secure: one process cannot corrupt another process. | ⑤ Not secure: a thread can write the memory used by another thread. |
| ⑥ They are typically independent. | ⑥ Thread exists as subset of a process. |
| ⑦ Process carries considerable state information. | ⑦ Multiple-thread within a process share state as well as memory and other resources. |
| ⑧ Process have separate address space. | ⑧ Thread share their address space. |

5. Highlight on User space thread and kernel space thread.

⇒

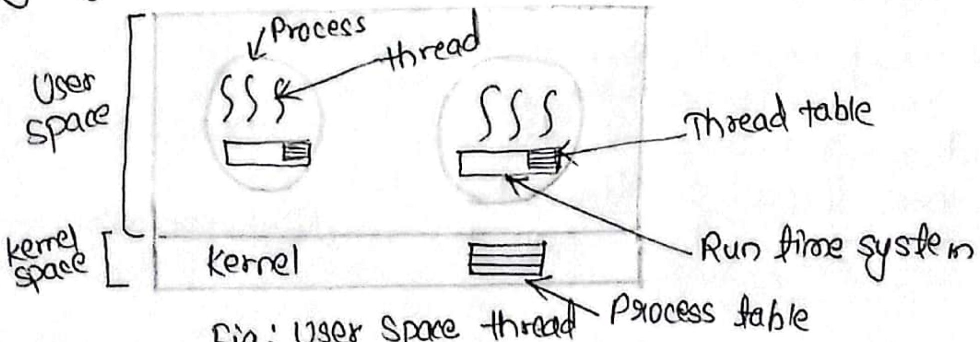


Fig: User Space thread

User Space thread is implemented by users and the kernal is not aware of the existence of these threads. The thread library contains code for creating and destroying threads, for passing message and data between threads, for scheduling thread execution and for saving and restoring thread contexts.

Advantages of User space thread-

i) Thread switching doesn't require kernel mode priviledges.

ii) Can run on any OS.

iii) Scheduling can be application specific.

iv) Fast to create and manage.

Disadvantages of user space thread-

i) In a typical OS, most system calls are blocking.

ii) Multithread application cannot take advantage of multiprocessing.

Kernel Space Thread :- They are handled by the operating system directly and the thread management is done by the kernel. The kernel performs thread creation, scheduling and managgement in kernel space.

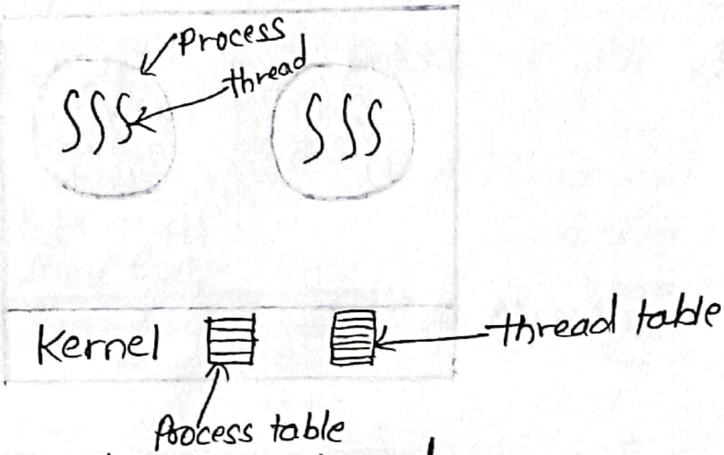


Fig: kernel space Thread

Advantages:

i) Kernel can simultaneously schedule multiple threads from the same process on multiple processes.

ii) If one thread in a process is blocked, the kernel can schedule another thread of the same process.

iii) Kernel routines themselves can be multithreaded.

Disadvantages:

i) Generally slower to create and manage than the user threads.

ii) Transfer of control from one thread to another within the same process requires a mode switch to the kernel.

6. Discuss about Multithreading Models.

⇒ The multithreading models, also known as threading models, or concurrency models, define how threads are created, scheduled and synchronized in a multithreaded application. Different types of multithreading models are:-

① One to One: Here, each user level thread corresponds to one kernel level thread managed by the OS. The model is also known as Kernel-level threads.

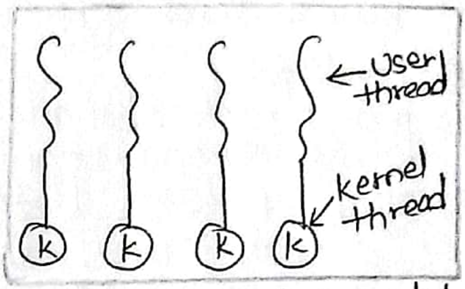


Fig: One to One model

Advantages:

i) Another thread can run when a thread makes blocking system call.

ii) Multiple thread can execute in parallel on CPU.

Disadvantages:

i) Creating a user thread requires creating corresponding kernel thread.

ii) Application may have less control over thread management.

② Many to One: This is also known as User-level threads. In this model, multiple user level threads which are managed entirely by the application without OS support are mapped to a single kernel level thread.

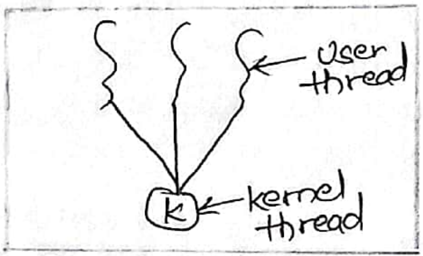


Fig: Many to one model

Advantages-

i) Allows application to create number of threads that can execute concurrently.

ii) Typically faster to create and switch.

Disadvantages:-

i) Lacks true parallelism because all threads share a single kernel thread.

ii) May not take full advantage of multicore processors.

③ Many to Many: This model combines actions of both user level and kernel level threads also known as hybrid threads. Multiple user level threads are mapped to a smaller number of kernel-level threads.

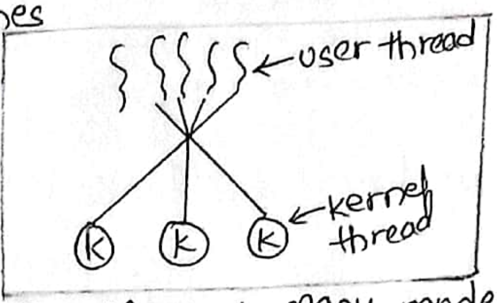


Fig: Many to many model

Advantages:

i) Many user threads and corresponding kernel thread can run in parallel on CPU.

ii) Offers balance between, lightweight thread creation and control over thread management.

Disadvantages:

i) Complex to implement

ii) If kernel thread amount is not enough then performance will be low.

⑦ Explain Preemptive and non-preemptive scheduling.

⇒ Preemptive and non-preemptive scheduling are two fundamental approaches to managing the execution of multiple processes or threads in an OS. These scheduling strategies determine how the CPU is allocated to different tasks.

- Preemptive Scheduling - In this scheduling, the OS can interrupt a running task and switch to another task at any time, without the cooperation of currently executing task. It is used when a process switches from running state to ready state. The resources (mainly CPU cycles) are allocated to the process for a limited amount of time and then taken away, and the process is again placed back in the ready queue if that process still has CPU burst time remaining. That process stays in queue till it get its next chance to execute. Algorithms based on Preemptive scheduling include Round Robin (RR), Shortest Remaining Time First (SRTF), Priority (Preemptive Version), etc.

Advantages:

i) Because a process may not monopolize the processor, it is a more reliable method.

ii) Each occurance prevents the completion of ongoing tasks.

Disadvantages:

i) Limited computational resources must be used.

ii) The low-priority process would have to wait if multiple high priority processes arrived at the same time.

- Non-Preemptive Scheduling :- This is also known as Cooperative scheduling. Here, a running task must explicitly release the CPU voluntarily before another task can run. It is used when a process terminates, or a process switches from running to waiting state. Once the resources are allocated to the process, the process holds the CPU till its gets terminated or reaches a waiting state. It doesn't stop or interrupt a process running CPU in the middle of the execution. Instead, it waits till the process complete its CPU burst time, and then it can allocate the CPU to another process. Algorithms based on non-preemptive scheduling include : Shortest Job First (SJF), Priority (non-preemptive version), etc.

Advantages:

i) It has a minimal scheduling burden.

ii) It is a very easy procedure.

iii) Less computational resources are used.

Disadvantages :

i) Its response time to the process is super.
ii) Bugs can cause a computer to freeze up.

Q) Consider the following set of processes, with the length of CPU-burst time and arrival time given in milliseconds.

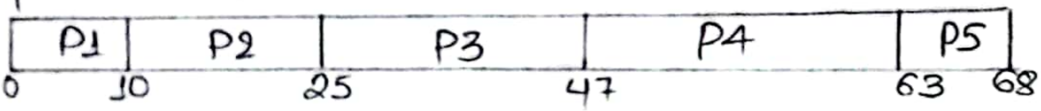
| Process | Burst time | Arrival time |
|---|---|---|
| P1 | 10 | 0 |
| P2 | 15 | 2 |
| P3 | 22 | 3 |
| P4 | 16 | 5 |
| P5 | 5 | 6 |

For the given data, draw the Gantt chart that illustrates the execution of these processes using FCFS, SJF, SRTN, RR (quantum=4) and calculate the average waiting and average turnaround time.

Ans:=>

Using FCFS:

Gantt Chart -



Waiting Time :

P1 = 0-0=0
P2 = 10-2=8
P3 = 25-3= 22
P4 = 47-5= 42
P5 = 63-6 = 57

Average Waiting Time : $= \frac{(0+8+22+42+57)}{5} = 28.8$ ms

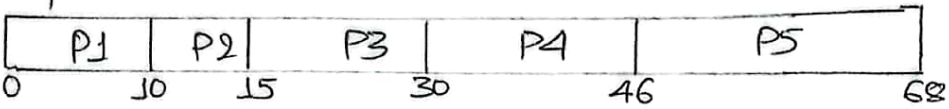
Turnaround Time:

P1 = 0+10=10<br>
P2 = 8+15=23<br>
P3 = 22+22=44<br>
P4 = 42+16=58<br>
P5 = 57+5=62

Average Turnaround time : $\frac{(10+23+44+58+62)}{5}$ = 39.4 ms

Using SJF:

Gantt Chart -



Waiting time:

P1 = 0-0=0<br>
P2 = ~~10~~15-2 = ~~8~~13<br>
P3 = 46-3=43<br>
P4 = 30-5=25<br>
P5 = 10-6 = 4

Average waiting time = $\frac{(0+13+43+25+4)}{5}$ = 17 ms

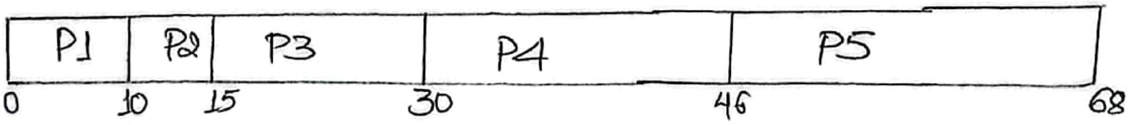
Turnaround Time:

P1 = 0+10= 10<br>
P2 = 13+15 = 28<br>
P3 = 43 +22 = 65<br>
P4 = 25+16 = 41<br>
P5 = 4+5 = 9

Average Turnaround Time: $\frac{(10+28+65+41+9)}{5}$ = 30.6 ms

Using SRTN:-

Gantt Chart -



Waiting Time :

P1 = 0-0 = 0
P2 = 15-2 = 13
P3 = 46-3 = 43
P4 = 30-5 = 25
P5 = 10-6 = 4

Average waiting Time = $\frac{(0+13+43+25+4)}{5}$ = 17 ms

Turnaround Time :

P1 = 0+10 = 10
P2 = 13+15 = 28
P3 = 43+22 = 65
P4 = 25+16-41
P5 = 4+5 = 9

Average Turnaround Time : $\frac{(10+28+65+41+9)}{5}$ = 30.6 ms

Using RR :- quantum = 4

Gantt chart -

| P1 | P2 | P3 | P1 | P4 | P5 | P2 | P3 | P1 | P4 | P5 | P2 | P3 | P4 | P2 | P3 | P4 | P3 | P3 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 4 | 8 | 12 | 16 | 20 | 24 | 28 | 32 | 34 | 38 | 39 | 43 | 47 | 51 | 54 | 58 | 62 | 64 | 68 |

Waiting Time -

P1 = (0-0)+(12-4)+(32-16) = 24
P2 = (4-2)+(24-8)+(39-28)+(51-43) = 37
P3 = (8-3)+(28-12)+(43-32)+(54-47)+(62-58) = 43
P4 = (16-5)+(34-20)+(47-38)+(58-51) = 41
P5 = (20-6)+(38-24) = 28

Average waiting Time = $\frac{(24+37+43+41+28)}{5}$ = 34.6 ms

Turnaround Time :-

P1 = 24+10 = 34
P2 = ~~4~~ 37+15 = 52
P3 = 43+22 = 65
P4 = 41+16 = 57
P5 = 28+5 = 33

Average Turnaround Time :- $\frac{(34+52+65+57+33)}{5}$

= 48.2 ms