CSC 301 – OPERATING SYSTEMS II (2 UNITS) LECTURE 2

BALOGUN JEREMIAH ADEMOLA

ASSISTANT LECTURER,

DEPARTMENT OF COMPUTER SCIENCE AND MATHEMATICS

MOUNTAIN TOP UNIVERSITY, OGUN STATE, NIGERIA

COURSE OUTLINE

- **Concurrency**
  - States and State Diagrams Structures
  - Dispatching and Context Switching
  - Interrupts
  - Concurrent Execution
  - Mutual Exclusion Problem and Some Solutions
- **Deadlock**
  - Models and Mechanisms (Semaphores, monitors etc.)
  - Producer – Consumer Problems and Synchronization
  - Multiprocessor Issues
- **Scheduling and Dispatching, Memory Management**
  - Overlays, Swapping and Partitions
  - Paging and Segmentations Placement and Replacement Policies
  - Working Sets and Trashing
  - Caching

STATES AND STATE DIAGRAMS

- A State diagram is a type of diagram used in computer science and related fields to describe the behaviour of systems.
  - State diagrams require that the system described is composed of a finite number of states.
  - State diagrams are used to give an abstract description of the behavior of a system.

- This behavior is analyzed and represented by a series of events that can occur in one or more possible states.
  - Each diagram usually represents objects of a single class and track the different states of its objects through the system.

- In a multitasking computer system, processes may occupy a variety of states.
  - These distinct states may not be recognized as such by the operating system kernel.
  - However, they are a useful abstraction for the understanding of processes.
  - In most of these states, processes are usually stored in RAM.

STATES AND STATE DIAGRAMS…

- An OS maintains one special data structure called the Process Control Block (PCB).
- All information about each process is stored in the PCB which is maintained by the OS.
- It contains a number of information with a specific process, namely:
  - **Process State** - It represents current status of the process. It may be new, ready, running or waiting.
  - **Program Counter** - It indicates the address of the next instruction to be executed for this process.
  - **CPU Registers** - They include index registers, stack pointer and general purpose registers. It is used to save process state when an interrupt occurs, so that it can resume from that state.
  - **CPU-Scheduling Information** - It contains process priority, pointer to scheduling queue.
  - **Memory Management Information** – It contains the value of the base and limit registers, page tables depending on the memory system.
  - **Accounting Information** - It contains an amount of CPU and real time used, time limits process number and so on.
  - **I/O Status Information** - It contains a list of I/O devices allocated to the process, a list of open files and so on.

STATES AND STATE DIAGRAMS…

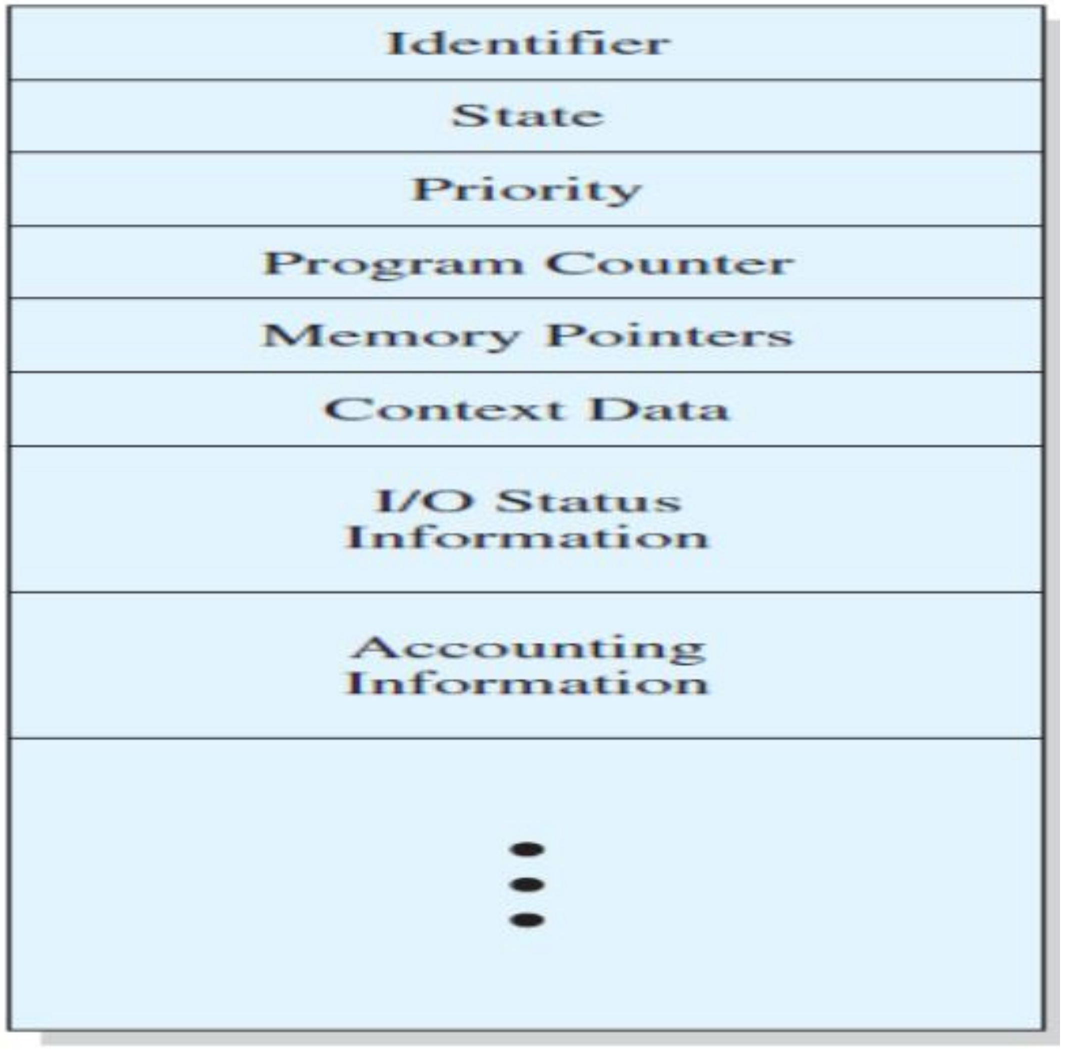


Figure 1: A Process Control Block

PROCESS STATES

- **New**
  - A program which is going to be picked up by the OS into the main memory is called a new process.
- **Ready**
  - Whenever a process is created, it directly enters in the ready state, in which, it waits for the CPU to be assigned.
  - The OS picks the new processes from the secondary memory and put all of them in the main memory.
  - The processes which are ready for the execution and reside in the main memory are called ready state processes.
  - There can be many processes present in the ready state.
- **Running**
  - One of the processes from the ready state will be chosen by the OS depending upon the scheduling algorithm.
  - Hence, if we have only one CPU in our system, the number of running processes for a particular time will always be one.
  - If we have n processors in the system then we can have n processes running simultaneously.
- **Block or Waiting**
  - From the Running state, a process can make the transition to the block or wait state depending upon the scheduling algorithm or the intrinsic behavior of the process.
  - When a process waits for a certain resource to be assigned or for the input from the user then the OS move this process to the block or wait state and assigns the CPU to the other processes.

PROCESS STATES…

- **Completion or Termination**
  - When a process finishes its execution, it comes in the termination state.
  - All the context of the process (Process Control Block) will also be deleted the process will be terminated by the Operating system.
- **Suspend Ready**
  - A process in the ready state, which is moved to secondary memory from the main memory due to lack of the resources (mainly primary memory) is called in the suspend ready state.
  - If the main memory is full and a higher priority process comes for the execution then the OS have to make the room for the process in the main memory by throwing the lower priority process out into the secondary memory.
  - The suspend ready processes remain in the secondary memory until the main memory gets available.
- **Suspend Wait**
  - Instead of removing the process from the ready queue, it's better to remove the blocked process which is waiting for some resources in the main memory.
  - Since it is already waiting for some resource to get available hence it is better if it waits in the secondary memory and make room for the higher priority process.
  - These processes complete their execution once the main memory gets available and their wait is finished.

PROCESS STATES…

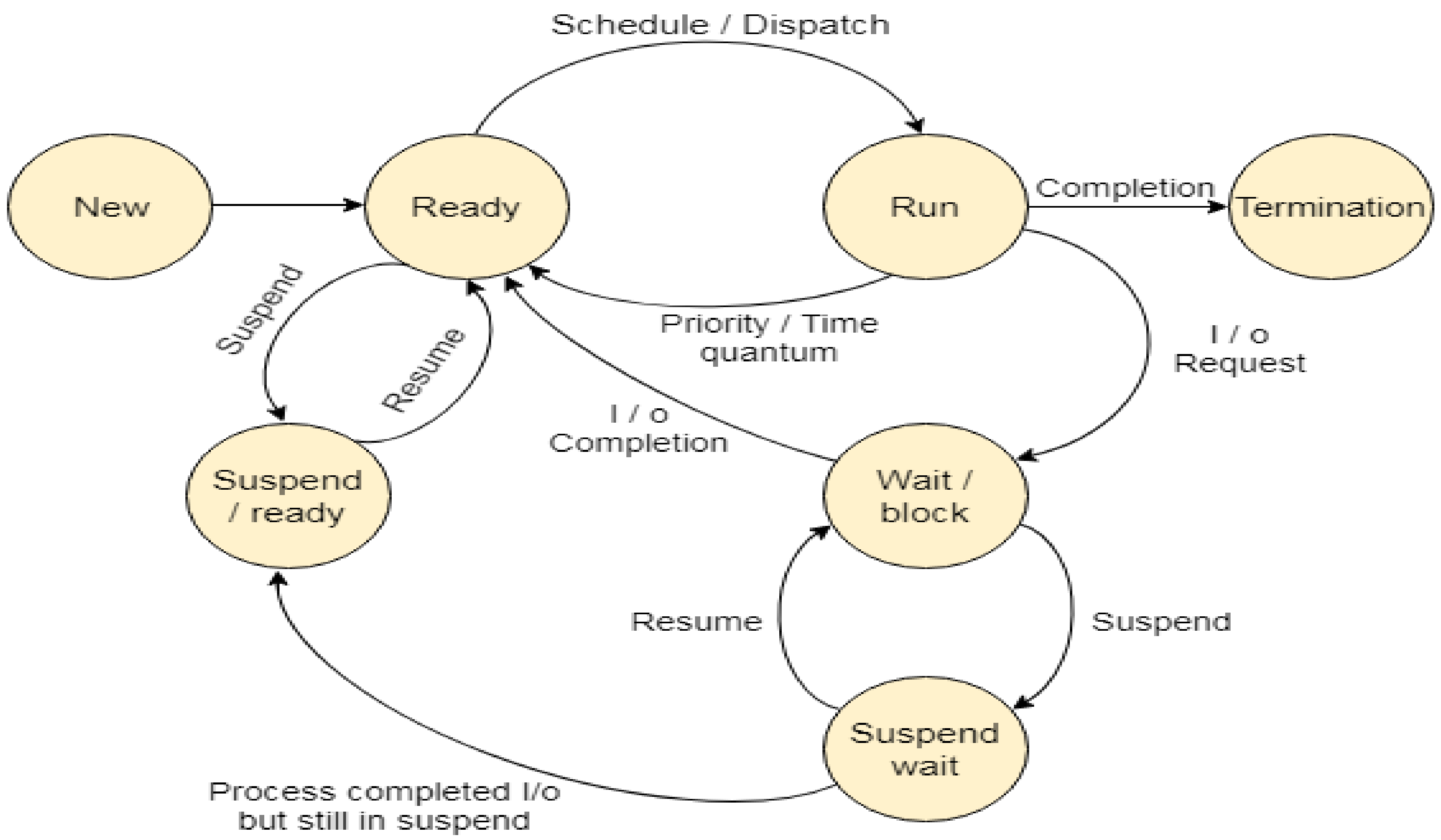


Figure 2: Process State Diagram

PROCESS OPERATIONS

- **Creation**
  - Once the process is created, it will be ready and come into the ready queue (main memory) and will be ready for the execution.
- **Scheduling**
  - Out of the many processes present in the ready queue, the Operating system chooses one process and start executing it.
  - Selecting the process which is to be executed next, is known as scheduling.
- **Execution**
  - Once the process is scheduled for the execution, the processor starts executing it.
  - Process may come to the blocked or wait state during the execution then in that case the processor starts executing the other processes.
- **Deletion/Killing**
  - Once the purpose of the process gets over then the OS will kill the process.
  - The Context of the process (PCB) will be deleted and the process gets terminated by the Operating system.

PROCESS OPERATIONS….

- **CPU and I/O Bound Processes**
  - **If the process is intensive in terms of CPU operations then it is called CPU bound process.**
  - **Similarly, If the process is intensive in terms of I/O operations then it is called IO bound process.**

- **Types of Schedulers**
  - **Long term – performance**
    - **Makes a decision about how many processes should be made to stay in the ready state, this decides the degree of multiprogramming.**
    - **Once a decision is taken it lasts for a long time hence called long term scheduler.**
  - **Short term – Context switching time**
    - **Short term scheduler will decide which process to be executed next and then it will call dispatcher.**
    - **A dispatcher is a software that moves process from ready to run and vice versa. In other words, it is context switching.**
  - **Medium term – Swapping time**
    - **Suspension decision is taken by medium term scheduler.**
    - **Medium term scheduler is used for swapping that is moving the process from main memory to secondary and vice versa.**

PROCESS SCHEDULING

- In computing, scheduling is the method by which work is assigned to resources that complete the work.
  - The work may be virtual computation elements such as threads, processes or data flows, which are in turn scheduled onto hardware resources such as processors, network links or expansion cards.
- A scheduler is what carries out the scheduling activity.
  - Schedulers are often implemented so they keep all computer resources busy:
    - via load balancing,
    - allow multiple users to share system resources effectively, or
    - to achieve a target quality of service.
- Types of Schedulers
  - Long term – performance
    - Makes a decision about how many processes should be made to stay in the ready state, this decides the degree of multiprogramming.
    - Once a decision is taken it lasts for a long time hence called long term scheduler.
  - Short term – Context switching time
    - Short term scheduler will decide which process to be executed next and then it will call dispatcher.
    - A dispatcher is a software that moves process from ready to run and vice versa. In other words, it is context switching.
  - Medium term – Swapping time
    - Suspension decision is taken by medium term scheduler.
    - Medium term scheduler is used for swapping that is moving the process from main memory to secondary and vice versa.

GOALS OF PROCESS SCHEDULING

- A scheduler may aim at one or more goals, for example:
  - maximizing throughput (the total amount of work completed per time unit);
  - minimizing wait time (time from work becoming ready until the first point it begins execution);
  - minimizing latency or response time (time from work becoming ready until it is finished in case of batch activity, or until the system responds and hands the first output to the user in case of interactive activity); or
  - maximizing fairness (equal CPU time to each process, or more generally appropriate times according to the priority and workload of each process).

- In practice, these goals often conflict (e.g. throughput versus latency), thus a scheduler will implement a suitable compromise.
  - Preference is measured by any one of the concerns mentioned above, depending upon the user's needs and objectives.

- In real-time environments, such as embedded systems for automatic control in industry (for example robotics):
  - The scheduler must ensure that processes can meet deadlines; this is crucial for keeping the system stable.
  - Scheduled tasks can also be distributed to remote devices across a network and managed through an administrative back end.

DISPATCHER

- **Another component that is involved in the CPU-scheduling function is the dispatcher, which is the module that gives control of the CPU to the process selected by the short-term scheduler.**
  - **It receives control in kernel mode as the result of an interrupt or system call.**

- **The functions of a dispatcher includes the following:**
  - **Context switches, in which the dispatcher saves the state (also known as context) of the process or thread that was previously running; the dispatcher then loads the initial or previously saved state of the new process.**
  - **Switching to user mode.**
  - **Jumping to the proper location in the user program to restart that program indicated by its new state.**

- **The dispatcher should be as fast as possible, since it is invoked during every process switch.**
  - **During the context switches, the processor is virtually idle for a fraction of time, thus unnecessary context switches should be avoided.**
  - **The time it takes for the dispatcher to stop one process and start another is known as the dispatch latency.**

SCHEDULING DISCIPLINES

- Scheduling disciplines are algorithms used for distributing resources among parties which simultaneously and asynchronously request them.
  - Scheduling disciplines are used in routers (to handle packet traffic) as well as in operating systems (to share CPU time among both threads and processes), disk drives (I/O scheduling), printers (print spooler), most embedded systems, etc.
- The main purposes of scheduling algorithms are to minimize resource starvation and to ensure fairness amongst the parties utilizing the resources.
  - Scheduling deals with the problem of deciding which of the outstanding requests is to be allocated resources. There are many different scheduling algorithms.
- In packet-switched computer networks and other statistical multiplexing, the notion of a scheduling algorithm is used as an alternative to first-come first-served queuing of data packets.
- The simplest best-effort scheduling algorithms are round-robin, fair queuing (a max-min fair scheduling algorithm), proportionally fair scheduling and maximum throughput.
  - If differentiated or guaranteed quality of service is offered, as opposed to best-effort communication, weighted fair queuing may be utilized.
- In advanced packet radio wireless networks such as HSDPA (High-Speed Downlink Packet Access) 3.5G cellular system, channel-dependent scheduling may be used to take advantage of channel state information.
  - If the channel conditions are favourable, the throughput and system spectral efficiency may be increased.
- In even more advanced systems such as LTE, the scheduling is combined by channel-dependent packet-by-packet dynamic channel allocation, or by assigning OFDMA multi-carriers or other frequency-domain equalization components to the users that best can utilize them.

SCHEDULING DISCIPLINES…

- First in, first out (FIFO), also known as first come, first served (FCFS), is the simplest scheduling algorithm.
  - FIFO simply queues processes in the order that they arrive in the ready queue.
  - This is commonly used for a task queue, for example as illustrated in this section.

- Since context switches only occur upon process termination, and no reorganization of the process queue is required, scheduling overhead is minimal.
- Throughput can be low, because long processes can be holding the CPU, causing the short processes to wait for a long time (known as the convoy effect).
- No starvation, because each process gets chance to be executed after a definite time.
- Turnaround time, waiting time and response time depend on the order of their arrival and can be high for the same reasons above.
- No prioritization occurs, thus this system has trouble meeting process deadlines.
- The lack of prioritization means that as long as every process eventually completes, there is no starvation.
  - In an environment where some processes might not complete, there can be starvation.
- It is based on queuing.

Scheduling Disciplines…

- Shortest remaining time first (SRTF) is similar to shortest job first (SJF).
  - With this strategy the scheduler arranges processes with the least estimated processing time remaining to be next in the queue.
  - This requires advanced knowledge or estimations about the time required for a process to complete.

- If a shorter process arrives during another process' execution, the currently running process is interrupted (known as preemption), dividing that process into two separate computing blocks.
  - This creates excess overhead through additional context switching.
  - The scheduler must also place each incoming process into a specific place in the queue, creating additional overhead.
- This algorithm is designed for maximum throughput in most scenarios.
- Waiting time and response time increase as the process's computational requirements increase.
  - Since turnaround time is based on waiting time plus processing time, longer processes are significantly affected by this.
  - Overall waiting time is smaller than FIFO, however since no process has to wait for the termination of the longest process.
- No particular attention is given to deadlines, the programmer can only attempt to make processes with deadlines as short as possible.
- Starvation is possible, especially in a busy system with many small processes being run.
- To use this policy we should have at least two processes of different priority

SCHEDULING DISCIPLINES…

- **Fixed Priority Pre-emptive Scheduling**
  - **The operating system assigns a fixed priority rank to every process, and the scheduler arranges the processes in the ready queue in order of their priority.**
  - **Lower-priority processes get interrupted by incoming higher-priority processes.**

- **Overhead is not minimal, nor is it significant.**
- **FPPS has no particular advantage in terms of throughput over FIFO scheduling.**
- **If the number of rankings is limited, it can be characterized as a collection of FIFO queues, one for each priority ranking.**
  - **Processes in lower-priority queues are selected only when all of the higher-priority queues are empty.**
- **Waiting time and response time depend on the priority of the process. Higher-priority processes have smaller waiting and response times.**
- **Deadlines can be met by giving processes with deadlines a higher priority.**
- **Starvation of lower-priority processes is possible with large numbers of high-priority processes queuing for CPU time.**

SCHEDULING DISCIPLINES…

- Round-Robin (RR) Scheduling
  - The scheduler assigns a fixed time unit per process, and cycles through them.
  - If process completes within that time-slice it gets terminated otherwise it is rescheduled after giving a chance to all other processes.

- RR scheduling involves extensive overhead, especially with a small time unit.
- Balanced throughput between FCFS/FIFO and SJF/SRTF, shorter jobs are completed faster than in FIFO and longer processes are completed faster than in SJF.
- Good average response time, waiting time is dependent on number of processes, and not average process length.
- Because of high waiting times, deadlines are rarely met in a pure RR system.
- Starvation can never occur, since no priority is given.
  - Order of time unit allocation is based upon process arrival time, similar to FIFO.
- If Time-Slice is large it becomes FCFS /FIFO or if it is short then it becomes SJF/SRTF.

SCHEDULING OPTIMIZATION PROBLEMS

- There are several scheduling problems in which the goal is to decide which job goes to which station at what time, such that the total makespan is minimized:

- Job shop scheduling
  - There are n jobs and m identical stations.
  - Each job should be executed on a single machine.
  - This is usually regarded as an online problem.
- Open-shop scheduling
  - There are n jobs and m different stations.
  - Each job should spend some time at each station, in a free order.
- Flow shop scheduling
  - There are n jobs and m different stations.
  - Each job should spend some time at each station, in a pre-determined order.

DIFFERENCES BETWEEN DISPATCHER AND SCHEDULER

- **Consider a situation, where various processes are residing in the ready queue waiting to be executed.**

- **The CPU cannot execute all of these processes simultaneously, so the operating system has to choose a particular process on the basis of the scheduling algorithm used.**

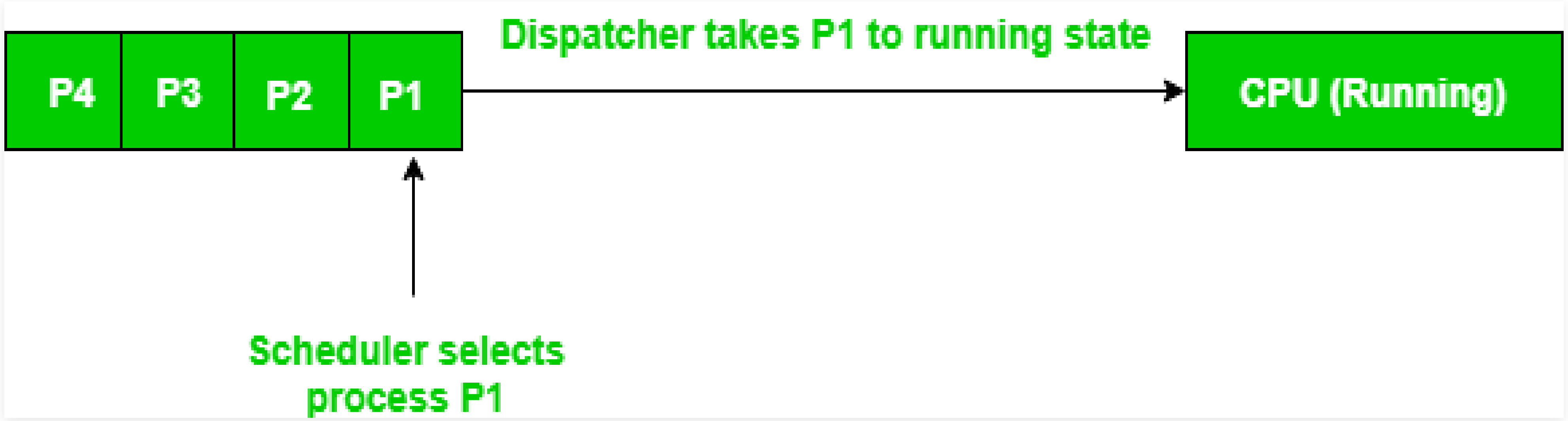
- **So, this procedure of selecting a process among various processes is done by the scheduler.**

- **Once the scheduler has selected a process from the queue, the dispatcher comes into the picture, and it is the dispatcher who takes that process from the ready queue and moves it into the running state.**

- **Therefore, the scheduler gives the dispatcher an ordered list of processes which the dispatcher moves to the CPU over time.**

DIFFERENCES BETWEEN DISPATCHER AND SCHEDULER

- **There are 4 processes in the ready queue, P1, P2, P3, P4; Their arrival times are t0, t1, t2, t3 respectively.**
- **A First in First out (FIFO) scheduling algorithm is used.**
- **Because P1 arrived first, the scheduler will decide it is the first process that should be executed, and the dispatcher will remove P1 from the ready queue and give it to the CPU.**
- **The scheduler will then determine P2 to be the next process that should be executed, so when the dispatcher returns to the queue for a new process, it will take P2 and give it to the CPU.**
- **This continues in the same way for P3, and then P4.**


P4
P3
P2
P1
Dispatcher takes P1 to running state
CPU (Running)
Scheduler selects process P1

DIFFERENCES BETWEEN DISPATCHER AND SCHEDULER

Properties	DISPATCHER	SCHEDULER
Definition	Dispatcher is a module that gives control of CPU to the process selected by short term scheduler	Scheduler is something which selects a process among various processes
Types	There are no different types in dispatcher. It is just a code segment.	There are 3 types of scheduler
Dependency	Working of dispatcher is dependent on scheduler. Means dispatcher have to wait until scheduler selects a process.	Scheduler works independently. It works immediately when needed
Algorithm	Dispatcher has no specific algorithm for its implementation	Scheduler works on various algorithm such as FCFS, SJF, RR etc.
Time Taken	The time taken by dispatcher is called dispatch latency.	Time taken by scheduler is usually negligible. Hence we neglect it.
Functions	Dispatcher is also responsible for Context Switching, Switch to user mode, Jumping to proper location when process again restarted	The only work of scheduler is selection of processes.

CONTEXT SWITCHING

- **In computing, a context switch is the process of storing the state of a process or thread, so that it can be restored and resume execution at a later point.**
  - **This allows multiple processes to share a single central processing unit (CPU), and is an essential feature of a multitasking operating system.**

- **The precise meaning of the phrase "context switch" varies.**
  - **In a multitasking context, it refers to the process of storing the system state for one task, so that task can be paused and another task resumed.**

  - **A context switch can also occur as the result of an interrupt, such as when a task needs to access disk storage, freeing up CPU time for other tasks. Some operating systems also require a context switch to move between user mode and kernel mode tasks.**

  - **The process of context switching can have a negative impact on system performance**

CONTEXT SWITCHING…

- **A context is the contents of a CPU's registers and program counter at any point in time.**

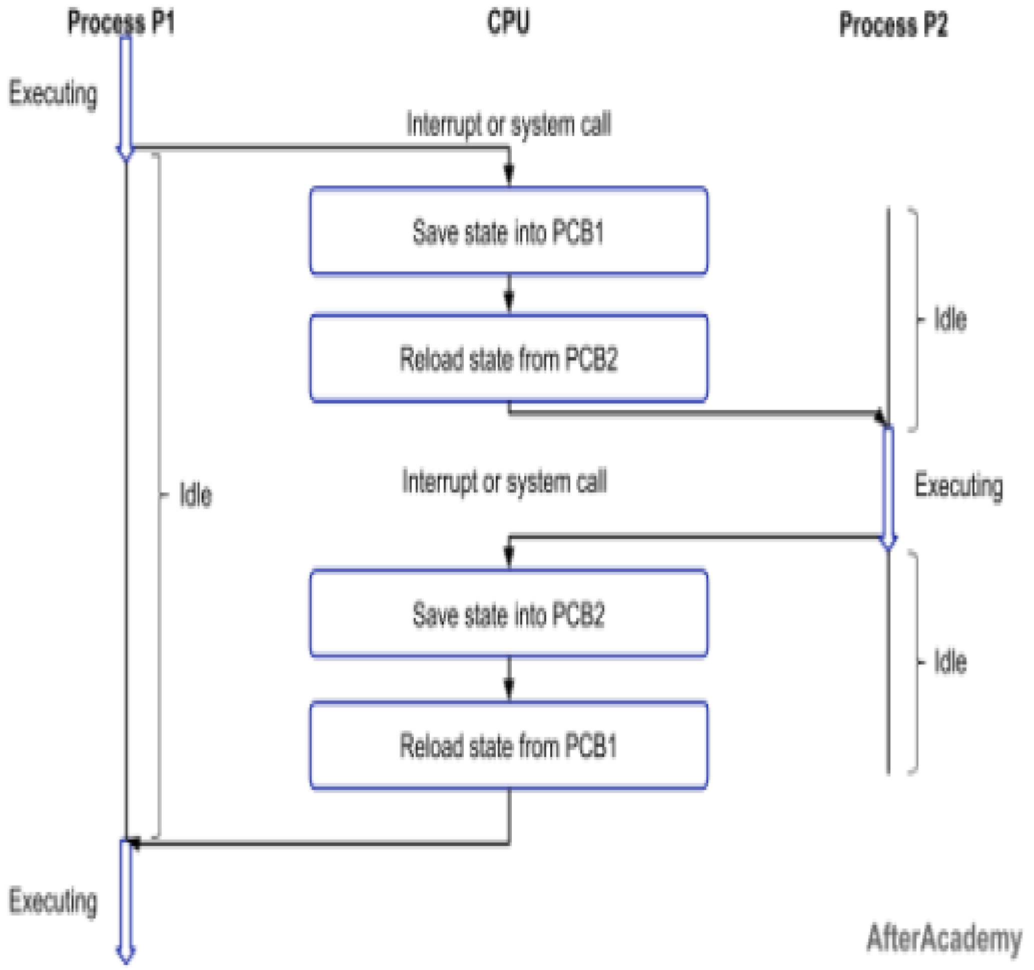
- **Context switching can happen due to the following reasons:**
  - **When a process of high priority comes in the ready state.**
  - **In this case, the execution of the running process should be stopped and the higher priority process should be given the CPU for execution.**

  - **When an interruption occurs then the process in the running state should be stopped and the CPU should handle the interrupt before doing something else.**

  - **When a transition between the user mode and kernel mode is required then you have to perform the context switching.**

CONTEXT SWITCHING…

1. Firstly, the context of the process P1 i.e. the process present in the running state will be saved in the Process Control Block of process P1 i.e. PCB1.
2. Now, you have to move the PCB1 to the relevant queue i.e. ready queue, I/O queue, waiting queue, etc.
3. From the ready state, select the new process that is to be executed i.e. the process P2.
4. Now, update the Process Control Block of process P2 i.e. PCB2 by setting the process state to running.
   - If the process P2 was earlier executed by the CPU, then you can get the position of last executed instruction so that you can resume the execution of P2.
5. Similarly, if you want to execute the process P1 again, then you have to follow the same steps as mentioned above (from step 1 to 4).


Process P1
CPU
Process P2
Executing
Interrupt or system call
Save state into PCB1
Reload state from PCB2
Idle
Idle
Executing
Interrupt or system call
Save state into PCB2
Reload state from PCB1
Idle
Executing
AfterAcademy