

<p>SCHEDULING ALGORITHM</p> <p>A scheduling algorithm is the algorithm which dictates how much CPU time is allocated to Processes and Threads.</p> <p>The objectives of scheduling algorithm are:</p> <ol style="list-style-type: none"> 1. Maximize CPU Utilization. 2. Fair allocation of CPU. 3. Maximize throughput. 4. Minimize turnaround time. 5. Minimize waiting time. 6. Minimize response time. <p>Category of Scheduling Algorithm</p> <p>1. Non-preemptive: In this once a process enters the running state; it cannot be preempted until it completes its allotted time. Ex: - FCFS, SJF, Priority, etc.</p> <p>2. Preemptive: It is based on priority where a scheduler may preempt a low priority running process anytime when a high priority process enters into a ready state. Ex: - SRTF, Priority, Round Robin, etc.</p>	<p>शेड्यूलिंग कलन विधि</p> <p>एक शेड्यूलिंग एल्गोरिदम, एक एल्गोरिदम है जो यह निर्धारित करता है कि प्रोसेस और थ्रेड के लिए कितना CPU समय आवंटित किया जाना है।</p> <p>शेड्यूलिंग एल्गोरिदम के उद्देश्य हैं:</p> <ol style="list-style-type: none"> 1. सीपीयू आवंटन को अधिकतम करें। 2. सीपीयू का उचित आवंटन। 3. थ्रूपुट अधिकतम करें। 4. टर्नअराउंड समय को कम करें। 5. प्रतीक्षा समय कम करें। 6. प्रतिक्रिया समय को कम करें। <p>शेड्यूलिंग एल्गोरिदम की श्रेणी</p> <p>1. Non-preemptive: इसमें जब एक प्रक्रिया चलती स्थिति में प्रवेश करती है; इसे तब तक छूट नहीं दी जा सकती जब तक कि यह अपना आवंटित समय पूरा न करे। Ex: - FCFS, SJF, Priority, etc.</p> <p>2. Preemptive: यह प्राथमिकता पर आधारित है जहां एक शेड्यूलर किसी भी समय एक कम प्राथमिकता वाले प्रोसेस को एक उच्च प्राथमिकता वाले प्रोसेस को अपना control दे सकता है। Ex: - SRTF, Priority, Round Robin, etc.</p>
<p>Preemptive Scheduling</p>	<p>Non-Preemptive Scheduling</p>
<p>Processor can be preempted to execute a different process in the middle of execution of any current process. प्रोसेसर को किसी भी मौजूदा प्रक्रिया के निष्पादन के बीच में एक अलग प्रक्रिया निष्पादित करने की अनुमति दी जा सकती है।</p>	<p>Once Processor starts to execute a process it must finish it before executing the other. It cannot be paused in middle. एक बार प्रोसेसर किसी एक प्रक्रिया निष्पादित करना शुरू कर देता है, तो किसे दूसरे को निष्पादित करने से पहले इसे खत्म करना होगा। इसे बीच में रोका नहीं जा सकता है।</p>
<p>CPU utilization is more. सीपीयू उपयोग अधिक है।</p>	<p>CPU utilization is less. इसमें CPU का उपयोग कम होता है।</p>
<p>Waiting time and Response time is less. प्रतीक्षा समय और प्रतिक्रिया समय कम है।</p>	<p>Waiting time and Response time is more. प्रतीक्षा समय और प्रतिक्रिया समय अधिक है।</p>
<p>The preemptive scheduling is prioritized. The highest priority process should always be the process that is currently utilized. इसमें प्राथमिकता दी जाती है। सर्वोच्च प्राथमिकता प्रक्रिया हमेशा वह होनी चाहिए जिसका उपयोग वर्तमान में किया जाता है।</p>	<p>When a process enters the state of running, the state of that process is not deleted from the scheduler until it finishes its service time. जब कोई प्रक्रिया चलने की स्थिति में प्रवेश करती है, तो उस प्रक्रिया की स्थिति शेड्यूलर से तब तक नहीं हटाई जाती जब तक कि यह अपने सेवा समय को पूरा न करे।</p>
<p>If a high priority process frequently arrives in the ready queue, low priority process may starve. यदि तैयार कतार में अक्सर उच्च प्राथमिकता प्रक्रिया आती है, तो कम प्राथमिकता प्रक्रिया भूखा हो सकती है।</p>	<p>If a process with long burst time is running CPU, then another process with less CPU burst time may starve. यदि लंबे समय के साथ एक प्रक्रिया CPU पर चल रही है, तो कम CPU वाले समय के साथ एक प्रक्रिया भूखा हो सकता है।</p>
<p>Preemptive scheduling is flexible. यह शेड्यूलिंग लचीला है।</p>	<p>Non-preemptive scheduling is rigid. यह शेड्यूलिंग कठोर है।</p>

FCFS (First Come, First Served) Scheduling Algorithm (Non-Preemptive Type)

FCFS is a scheduling algorithm used in operating systems and computer systems to manage the execution of processes or tasks in a queue. In FCFS scheduling, processes are executed in the order they arrive in the ready queue. The process that arrives first gets executed first, and so on. It is a non-preemptive scheduling algorithm, which means once a process starts executing, it continues until it finishes or goes into a waiting state.

Here's how FCFS scheduling works:

1. When a process arrives in the ready queue, it is placed at the end of the queue.
2. The CPU scheduler selects the process at the front of the queue (the one that arrived first) for execution.
3. The selected process runs until it completes or enters a waiting state (e.g., I/O operation). During this time, no other process can run.
4. Once the currently running process finishes or goes into a waiting state, the CPU scheduler selects the next process in the queue for execution.
5. This process continues until all processes in the queue have been executed.

FCFS scheduling is simple to implement and ensures fairness in the sense that the first process to arrive will be the first to be executed. However, it has some drawbacks:

1. It may lead to poor average waiting times: If a long CPU-bound process arrives first, it can cause shorter processes to wait for a long time, leading the problem of starvation (**Convoy Effect**) if the burst time of the first process is the longest among all the jobs.
2. It is not suitable for time-sensitive tasks: FCFS doesn't consider the priority or execution time of processes, so important or urgent tasks may be delayed if they arrive later.
3. It doesn't adapt well to changing workload: FCFS doesn't dynamically prioritize processes based on their characteristics or system load.

Because of its limitations, FCFS is not commonly used in modern operating systems for general-purpose scheduling. Instead, more sophisticated algorithms like Round Robin, Priority Scheduling, and Multilevel Queue

FCFS (पहले आओ, पहले पाओ) शेड्यूलिंग एल्गोरिदम (गैर-निवारक प्रकार)

FCFS शेड्यूलिंग एल्गोरिदम एक शेड्यूलिंग एल्गोरिदम है जिसका उपयोग ऑपरेटिंग सिस्टम और कंप्यूटर सिस्टम में एक कतार में प्रक्रियाओं या कार्यों के निष्पादन को प्रबंधित करने के लिए किया जाता है। एफसीएफएस शेड्यूलिंग में, प्रक्रियाओं को तैयार कतार में आने के क्रम में निष्पादित किया जाता है। जो प्रक्रिया पहले आती है वह पहले क्रियान्वित होती है, इत्यादि। यह एक गैर-प्रीमेटिव शेड्यूलिंग एल्गोरिदम है, जिसका अर्थ है कि एक बार प्रक्रिया निष्पादित होने के बाद, यह तब तक जारी रहती है जब तक यह समाप्त नहीं हो जाती या प्रतीक्षा स्थिति में नहीं चली जाती।

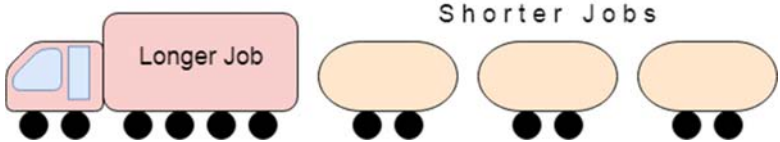
यहां बताया गया है कि एफसीएफएस शेड्यूलिंग कैसे काम करती है:

1. जब कोई प्रक्रिया तैयार कतार में आती है, तो उसे कतार के अंत में रखा जाता है।
2. सीपीयू शेड्यूलर निष्पादन के लिए कतार के सामने की प्रक्रिया (वह जो पहले पहुंची) का चयन करता है।
3. चयनित प्रक्रिया तब तक चलती है जब तक वह पूरी नहीं हो जाती या प्रतीक्षा स्थिति में प्रवेश नहीं कर जाती (उदाहरण के लिए, I/O ऑपरेशन)। इस दौरान कोई अन्य प्रक्रिया नहीं चल सकती।
4. एक बार जब वर्तमान में चल रही प्रक्रिया समाप्त हो जाती है या प्रतीक्षा स्थिति में चली जाती है, तो सीपीयू शेड्यूलर निष्पादन के लिए कतार में अगली प्रक्रिया का चयन करता है।
5. यह प्रक्रिया तब तक जारी रहती है जब तक कतार में सभी प्रक्रियाएं निष्पादित नहीं हो जातीं।

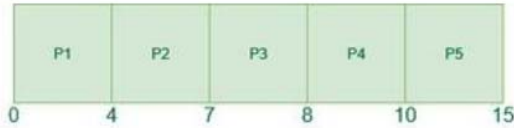
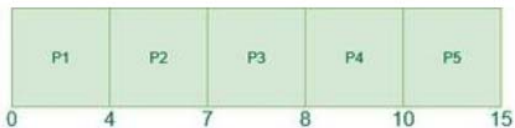
एफसीएफएस शेड्यूलिंग को लागू करना सरल है और यह इस अर्थ में निष्पक्षता सुनिश्चित करता है कि आने वाली पहली प्रक्रिया निष्पादित होने वाली पहली प्रक्रिया होगी। हालाँकि, इसमें कुछ कमियाँ हैं:

1. इससे औसत प्रतीक्षा समय खराब हो सकता है: यदि एक लंबी सीपीयू-बाउंड प्रक्रिया पहले आती है, तो इससे छोटी प्रक्रियाओं को लंबे समय तक इंतजार करना पड़ सकता है, जिससे इष्टतम प्रदर्शन कम हो सकता है।
2. यह समय-संवेदनशील कार्यों के लिए उपयुक्त नहीं है: एफसीएफएस प्रक्रियाओं की प्राथमिकता या निष्पादन समय पर विचार नहीं करता है, इसलिए महत्वपूर्ण या जरूरी कार्य देर से आने पर विलंबित हो सकते हैं।
3. यह बदलते कार्यभार के अनुकूल नहीं है: एफसीएफएस प्रक्रियाओं को उनकी विशेषताओं या सिस्टम लोड के आधार पर गतिशील रूप से प्राथमिकता नहीं देता है।

अपनी सीमाओं के कारण, एफसीएफएस का उपयोग आमतौर पर सामान्य प्रयोजन शेड्यूलिंग के लिए आधुनिक ऑपरेटिंग सिस्टम में नहीं किया जाता है। इसके बजाय, संसाधन उपयोग और

Scheduling are often employed to optimize resource utilization and responsiveness.	प्रतिक्रिया को अनुकूलित करने के लिए राउंड रॉबिन, प्रायोरिटी शेड्यूलिंग और मल्टीलेवल क्यू शेड्यूलिंग जैसे अधिक परिष्कृत एल्गोरिदम को अक्सर नियोजित किया जाता है।
Turn Around Time = Completion Time - Arrival Time Waiting Time = Turnaround time - Burst Time	
<p style="text-align: center;">The Convoy Effect, Visualized Starvation</p> 	

Example 1 of FCFS Consider the following table of arrival time and burst time for five processes P1, P2, P3, P4 and P5.	एफसीएस का उदाहरण पाँच प्रक्रियाओं P1, P2, P3, P4 और P5 के आगमन समय और बर्स्ट समय की निम्नलिखित तालिका पर विचार करें।																		
<table><tr><th>Processes</th><th>Arrival Time</th><th>Burst Time</th></tr><tr><td>P1</td><td>0</td><td>4</td></tr><tr><td>P2</td><td>1</td><td>3</td></tr><tr><td>P3</td><td>2</td><td>1</td></tr><tr><td>P4</td><td>3</td><td>2</td></tr><tr><td>P5</td><td>4</td><td>5</td></tr></table>		Processes	Arrival Time	Burst Time	P1	0	4	P2	1	3	P3	2	1	P4	3	2	P5	4	5
Processes	Arrival Time	Burst Time																	
P1	0	4																	
P2	1	3																	
P3	2	1																	
P4	3	2																	
P5	4	5																	
The FCFS CPU Scheduling Algorithm will work on the basis of steps as mentioned below: Step 0: At time = 0, <ul style="list-style-type: none">The process begins with P1As it has an arrival time 0 Step 1: At time = 1, <ul style="list-style-type: none">The process P2 arrivesBut process P1 still executing,Thus, P2 is kept on a waiting table and waits for its execution. Step 3: At time = 2, <ul style="list-style-type: none">The process P3 arrives and kept in a waiting queueWhile process P1 is still executing as its burst time is 4. Step 4: At time = 3, <ul style="list-style-type: none">The process P4 arrives and kept in the waiting queueWhile process P1 is still executing as its burst time is 4 Step 5: At time = 4, <ul style="list-style-type: none">The process P1 completes its executionProcess P5 arrives in waiting queue while process P2 starts executing Step 6: At time = 5, <ul style="list-style-type: none">The process P2 completes its execution Step 7: At time = 7.	FCFS CPU शेड्यूलिंग एल्गोरिदम नीचे बताए गए चरणों के आधार पर काम करेगा: चरण 0: समय = 0 पर, <ul style="list-style-type: none">प्रक्रिया P1 से शुरू होती हैचूँकि इसका आगमन समय 0 है चरण 1: समय = 1 पर, <ul style="list-style-type: none">प्रक्रिया P2 आती हैलेकिन प्रक्रिया P1 अभी भी क्रियान्वित हो रही है,इस प्रकार, P2 को प्रतीक्षा तालिका पर रखा जाता है और इसके निष्पादन की प्रतीक्षा की जाती है। चरण 3: समय = 2 पर, <ul style="list-style-type: none">प्रक्रिया P3 आती है और प्रतीक्षा कतार में रखी जाती हैजबकि प्रक्रिया P1 अभी भी क्रियान्वित हो रही है क्योंकि इसका विस्फोट समय 4 है। चरण 4: समय = 3 पर, <ul style="list-style-type: none">प्रक्रिया P4 आती है और प्रतीक्षा कतार में रखी जाती हैजबकि प्रक्रिया P1 अभी भी क्रियान्वित हो रही है क्योंकि इसका विस्फोट समय 4 है चरण 5: समय = 4 पर, <ul style="list-style-type: none">प्रक्रिया P1 अपना निष्पादन पूरा करती हैप्रक्रिया P5 प्रतीक्षा कतार में आती है जबकि प्रक्रिया P2 निष्पादित होना शुरू हो जाती है चरण 6: समय = 5 पर, <ul style="list-style-type: none">प्रक्रिया P2 अपना निष्पादन पूरा करती है चरण 7: समय = 7 पर,																		

<ul style="list-style-type: none"> Process P3 starts executing, it has burst time of 1 thus, it completes execution at time interval 8 <p>Step 8: At time 8,</p> <ul style="list-style-type: none"> The process of P3 completes its execution Process P4 starts executing, it has burst time of 2 thus, it completes execution at time interval 10. <p>Step 9: At time 10,</p> <ul style="list-style-type: none"> The process P4 completes its execution Process P5 starts executing, it has burst time of 5 thus, it completes execution at time interval 15. <p>Step 10: At time 15,</p> <ul style="list-style-type: none"> Process P5 will finish its execution. <p>The overall execution of the processes will be as shown below:</p> <p>Gantt chart for above execution:</p>  <p><i>Gantt chart for First come First serve Scheduling</i></p>	<ul style="list-style-type: none"> प्रक्रिया P3 निष्पादित करना शुरू करती है, इसका बर्स्ट समय 1 है, इस प्रकार यह समय अंतराल 8 पर निष्पादन पूरा करती है <p>चरण 8: समय 8 पर,</p> <ul style="list-style-type: none"> P3 की प्रक्रिया इसके निष्पादन को पूरा करती है प्रक्रिया P4 निष्पादित करना शुरू करती है, इसका बर्स्ट समय 2 है, इस प्रकार, यह समय अंतराल 10 पर निष्पादन पूरा करती है। <p>चरण 9: समय 10 पर,</p> <ul style="list-style-type: none"> प्रक्रिया P4 अपना निष्पादन पूरा करती है प्रक्रिया P5 निष्पादित करना शुरू करती है, इसका बर्स्ट समय 5 है, इस प्रकार, यह समय अंतराल 15 पर निष्पादन पूरा करती है। <p>चरण 10: समय 15 पर,</p> <ul style="list-style-type: none"> प्रक्रिया P5 अपना निष्पादन समाप्त कर देगी। <p>प्रक्रियाओं का समग्र निष्पादन नीचे दिखाए अनुसार होगा:</p> <p>उपरोक्त निष्पादन के लिए गैंट चार्ट:</p>  <p><i>Gantt chart for First come First serve Scheduling</i></p>
<p>Calculate Turnaround Time (TT)</p> $T_{P1} = 4 - 0 = 4$ $T_{P2} = 7 - 1 = 6$ $T_{P3} = 8 - 2 = 6$ $T_{P4} = 10 - 3 = 7$ $T_{P5} = 15 - 4 = 11$ <p>Calculate Waiting Time(WT)</p> $W_{P1} = 4 - 4 = 0$ $W_{P2} = 6 - 3 = 3$ $W_{P3} = 6 - 1 = 5$ $W_{P4} = 7 - 2 = 5$ $W_{P5} = 11 - 5 = 6$ <p>Average Waiting Time:</p> $W_{avg} = (0+3+5+5+6) / 5$ <p>= 3.8</p>	<p>Calculate Turnaround Time (TT)</p> $T_{P1} = 4 - 0 = 4$ $T_{P2} = 7 - 1 = 6$ $T_{P3} = 8 - 2 = 6$ $T_{P4} = 10 - 3 = 7$ $T_{P5} = 15 - 4 = 11$ <p>Calculate Waiting Time(WT)</p> $W_{P1} = 4 - 4 = 0$ $W_{P2} = 6 - 3 = 3$ $W_{P3} = 6 - 1 = 5$ $W_{P4} = 7 - 2 = 5$ $W_{P5} = 11 - 5 = 6$ <p>Average Waiting Time:</p> $W_{avg} = (0+3+5+5+6) / 5$ <p>= 3.8</p>

Example 2 of FCFS

Consider the set of 5 processes whose arrival time and burst time are given below:

Process Id	Arrival time	Burst time
P1	3	4
P2	5	3
P3	0	2
P4	5	1
P5	4	3

If the CPU scheduling policy is FCFS, calculate the average waiting time and average turn around time.

Gantt Chart:

0

2

3

7

10

13

14

P3		P1	P5	P2	P4
----	--	----	----	----	----

Here, black box represents the idle time of CPU.

Process Id	Exit time	Turn Around time	Waiting time
P1	7	$7 - 3 = 4$	$4 - 4 = 0$
P2	13	$13 - 5 = 8$	$8 - 3 = 5$
P3	2	$2 - 0 = 2$	$2 - 2 = 0$
P4	14	$14 - 5 = 9$	$9 - 1 = 8$
P5	10	$10 - 4 = 6$	$6 - 3 = 3$

Now,

Average Turn Around time = $(4 + 8 + 2 + 9 + 6) / 5 = 29 / 5 = \mathbf{5.8 \text{ unit}}$

Average Waiting Time = $(0 + 5 + 0 + 8 + 3) / 5 = 16 / 5 = \mathbf{3.2 \text{ unit}}$