# Lecture#06 Scheduling Algorithms

# First Come First Serve (FCFS)Scheduling Algorithm

# First Come First Serve (FCFS)

- Simplest CPU scheduling algorithm
- Processes are executed in order of arrival
- Non-preemptive scheduling approach
- Also known as First-In, First-Out (FIFO)

# Key Characteristics

- Processes managed using FIFO queue
- Fair and straightforward implementation
- No process starvation
- Minimal overhead in process management
- Suitable for batch processing systems

# How FCFS Works

- Processes arrives and joins queue
- CPU executes processes in arrival order
- No interruption until process completes
- Next process in queue gets CPU access
- Continues until queue is empty

# How FCFS Works - Example

Execution Time

| Process | Arrival Time (ms) | Burst Time (ms) |
|---|---|---|
| P1 | 0 | 24 |
| P2 | 1 | 3 |
| P3 | 2 | 3 |

**Gantt Chart Representation:**

Copy Edit

```
| P1 (24ms) | P2 (3ms) | P3 (3ms) |
0           24         27         30
```

# FCFS - Calculating Performance Metrics

1. **Waiting Time (WT) = Turnaround Time – Burst Time**
   - P1 = 0 - 0 = 0 ms
   - P2 = 24 - 1 = 23 ms
   - P3 = 27 - 2 = 25 ms
   - **Average WT** = (0 + 23 + 25) / 3 = **16 ms**

2. **Turnaround Time (TAT) = Completion Time – Arrival Time**
   - P1 = 24 - 0 = 24 ms
   - P2 = 27 - 1 = 26 ms
   - P3 = 30 - 2 = 28 ms
   - **Average TAT** = (24 + 26 + 28) / 3 = **26 ms**

# Advantages

- Simple to understand and implement
- Low scheduling overhead
- No complex algorithms required
- Easy to maintain
- Predictable execution order

# Disadvantages

- High average waiting time
- Poor for interactive systems
- Convoy effect possible
- Not optimal for time-sharing systems
- Performance varies with arrival order

# The Convoy Effect

- Short processes wait behind long ones
- Leads to inefficient CPU utilization
- Increases average waiting time
- Example: P1=24ms, P2=3ms, P3=3ms
- Results in poor system performance

# The Convoy Effect - Example

| Process | Arrival Time (ms) | Burst Time (ms) |
|---|---|---|
| P1 | 0 | 24 |
| P2 | 1 | 3 |
| P3 | 2 | 3 |

- P1 **(long process)** starts execution first, **blocking P2 and P3** from using the CPU.
- Even though **P2 and P3 require only 3 ms each**, they **must wait for P1** to complete **(24 ms delay)**.
- **Result:** The system is inefficient, with high waiting times.

# Performance Metrics

- Average Waiting Time
- Average Turnaround Time
- CPU Utilization
- Throughput
- Response Time

# Real-world Applications

- Batch processing systems
- Print queue management
- Simple task scheduling
- Banking queue systems
- Basic process management

# Best Practices & Implementation

- Use for processes of similar length
- Combine with other scheduling algorithms
- Monitor system performance
- Consider workload characteristics
- Implement proper queue management

# Shortest Job Next (SJN) Scheduling Algorithm

# Shortest Job Next (SJN) Scheduling Algorithm

- Simplest CPU scheduling algorithm
- Scheduling algorithm that selects the process with shortest burst time
- Non-preemptive: process runs to completion once started
- Aims to minimize average waiting time
- Optimal for batch processing systems

# Key Characteristics

- Based on burst time prediction
- Non-preemptive scheduling
- Minimizes average waiting time
- Provides optimal scheduling for given processes
- Suitable for batch processing environments

# How SJN Works

- Processes arrive with known burst times
- Scheduler selects process with shortest execution time
- Selected process runs to completion
- Next shortest job is then selected
- Process continues until all jobs complete

# Advantages

- Minimum average waiting time
- Efficient resource utilization
- Better throughput compared to FCFS
- Optimal for batch systems
- Reduces system overhead

# Disadvantages

- Potential starvation of longer processes
- Requires knowing/predicting burst time
- Not suitable for interactive systems
- Can lead to convoy effect
- Difficult to implement in practice

# Disadvantages - Example Scenario

| Process | Arrival Time | Burst Time |
|---|---|---|
| P1 | 0 ms | 6 ms |
| P2 | 1 ms | 3 ms |
| P3 | 2 ms | 8 ms |
| P4 | 3 ms | 4 ms |

**Execution Order:**

Since SJN selects **the shortest burst time first**, the order of execution is:

**P2 → P4 → P1 → P3**

**Gantt Chart Representation**

Copy Edit

```
| P2 (3ms) | P4 (4ms) | P1 (6ms) | P3 (8ms) |
0          3          7          13         21
```

# SJN - Calculating Performance Metrics

1. **Waiting Time (WT) = Turnaround Time – Burst Time**
   - P1 = 7 - 0 = **7 ms**
   - P2 = 0 - 1 = **0 ms**
   - P3 = 13 - 2 = **11 ms**
   - P4 = 3 - 3 = **3 ms**
   - **Average WT** = (7 + 0 + 11 + 3) / 4 = **5.25 ms**
2. **Turnaround Time (TAT) = Completion Time – Arrival Time**
   - P1 = 13 - 0 = **13 ms**
   - P2 = 3 - 1 = **2 ms**
   - P3 = 21 - 2 = **19 ms**
   - P4 = 7 - 3 = **4 ms**
   - **Average TAT** = (13 + 2 + 19 + 4) / 4 = **9.5 ms**

# Implementation Considerations

- Burst time estimation methods
- Queue management strategies
- Starvation prevention techniques
- Process priority handling
- System overhead management

# Real-world Applications

- Batch processing systems
- Print job scheduling
- Task scheduling in distributed systems
- Background process management
- Resource allocation in cloud computing

# END OF LECTURE!