

# OS - Chapter 5

## فصل ۵: برنامه‌ریزی پردازنده (CPU Scheduling)

### سرفصل‌های فصل (Outline)

در این فصل با مفاهیم و الگوریتم‌های مهم مربوط به زمان‌بندی پردازنده آشنا می‌شویم. مطالب اصلی فصل شامل موارد زیر است:

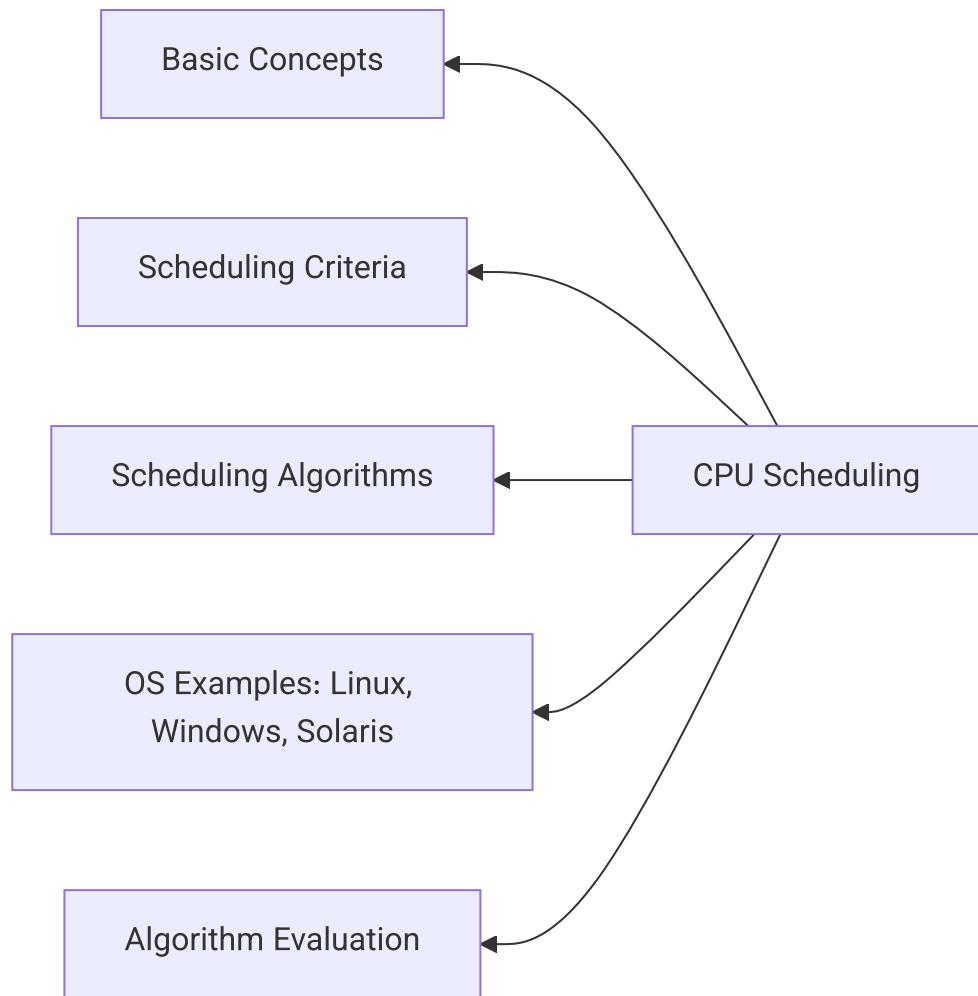
- **Basic Concepts** و تفاوت بین پردازه‌های job scheduling، تعریف صفات آماده: (مفاهیم پایه) 
- **Scheduling Criteria** معیارهایی مانند زمان پاسخ، زمان انتظار، بهره‌وری و استفاده از: (معیارهای زمان‌بندی) 
- **Scheduling Algorithms** FCFS، SJF، Round Robin وغیره.
- **Operating System Examples** بررسی نحوه پیاده‌سازی الگوریتم‌ها در سیستم‌عامل‌های: (مثال‌های سیستم‌عامل) 
- **Algorithm Evaluation** بررسی عملکرد الگوریتم‌ها: (ارزیابی الگوریتم‌ها) 

### اهداف یادگیری فصل (Objectives)

در پایان این فصل، دانشجو باید بتواند:

- الگوریتم‌های مختلف زمان‌بندی پردازنده را توضیح دهد.
- آن‌ها را بر اساس معیارهای زمان‌بندی ارزیابی کند.
- چالش‌های زمان‌بندی در سیستم‌های چندپردازنده‌ای و چنددهسته‌ای را درک کند.
- با الگوریتم‌های زمان‌بندی سیستم‌های بی‌درنگ آشنا شود.
- پیاده‌سازی الگوریتم‌های زمان‌بندی در سیستم‌عامل‌های مختلف را مقایسه کند.
- با استفاده از مدل‌سازی و شبیه‌سازی، عملکرد الگوریتم‌ها را تحلیل کند.

### دیاگرام مفاهیم اصلی فصل



## انواع زمانبندی در سیستم‌عامل (Scheduling) ⌚

در یک سیستم‌عامل چندوظیفه‌ای، برای استفاده مؤثر از منابع، باید تصمیم‌گیری‌هایی در خصوص زمان اجرای پردازه‌ها انجام شود. این تصمیم‌ها توسط انواع مختلف زمانبندی (scheduling) کنترل می‌شوند:

### ♦ زمانبندی بلندمدت (Long-term Scheduling)

- وظیفه: تصمیم‌گیری در مورد اضافه کردن پردازه‌ها به صف آماده برای اجرا.
- تأثیر: تعیین تعداد پردازه‌های فعال در سیستم.
- معمولاً توسط **job scheduler** انجام می‌شود.

### ♦ زمانبندی میانمدت (Medium-term Scheduling)

- وظیفه: تصمیم‌گیری درباره این‌که چه پردازه‌هایی باید به طور موقت از حافظه اصلی خارج یا مجدداً وارد حافظه شوند.
- معمولاً برای مدیریت بار حافظه و اجرای **swapping** استفاده می‌شود.

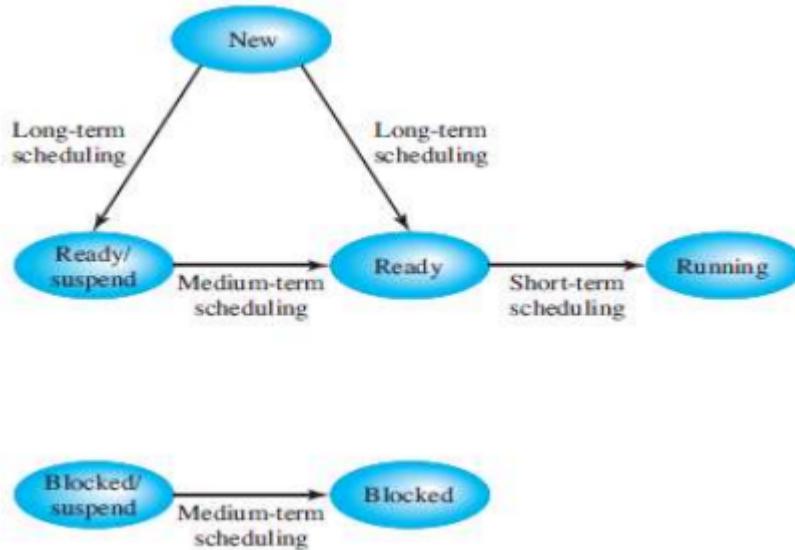
### ♦ زمانبندی کوتاه‌مدت (Short-term Scheduling)

- وظیفه: انتخاب پردازه‌ای از بین پردازه‌های آماده که قرار است روی CPU اجرا شود.
- مهم‌ترین نوع زمانبندی از نظر تأثیر مستقیم بر عملکرد سیستم.

### ♦ زمانبندی ورودی/خروجی (I/O Scheduling)

- وظیفه: تعیین این‌که کدام پردازه باید از دستگاه ورودی/خروجی استفاده کند و در چه زمانی.

## دیاگرام وضعیت‌های پردازه و نقش انواع زمانبندی



### نکات مهم:

- **زمانبندی بلندمدت** نقش مهمی در کنترل بار سیستم دارد و کمتر به طور مداوم اجرا می‌شود.
- **زمانبندی کوتاه‌مدت** بیشترین تأثیر را در تجربه کاربری دارد چون تصمیم می‌گیرد کدام پردازه به CPU برسد.
- **زمانبندی میان‌مدت** برای بهبود مدیریت حافظه فیزیکی در سیستم‌های چندوظیفه‌ای به کار می‌رود.
- **زمانبندی I/O** باعث می‌شود تداخل کمتری در استفاده از دستگاه‌های ورودی/خروجی رخ دهد.

## نمودار صف‌بندی برای زمانبندی پردازه‌ها (Queuing Diagram for Scheduling)

این نمودار نحوه حرکت پردازه‌ها بین صفحه‌های مختلف را در یک سیستم عامل چندوظیفه‌ای نشان می‌دهد. در اینجا، سه نوع پردازه اصلی داریم:

- **Batch jobs**: پردازه‌های دسته‌ای که از قبل برنامه‌ریزی شده‌اند.
- **Interactive users**: پردازه‌هایی که از طریق تعامل کاربر تولید می‌شوند.

### مسیرهای اصلی پردازه‌ها:

#### 1. Long-term scheduling:

- پردازه‌های دسته‌ای یا تعاملی ابتدا وارد صفحه "ورودی" می‌شوند.
- سپس زمان‌بند بلندمدت آنها را به صفحه آماده (Ready Queue) یا صف آماده/تعليق (Ready Suspend Queue) می‌هایت می‌کند.

#### 2. Short-term scheduling:

- از صفحه آماده، یک پردازه توسط زمان‌بند کوتاه‌مدت انتخاب و به پردازنده (Processor) فرستاده می‌شود.
- در پایان اجرای کامل یا در صورت Timeout، پردازه از CPU خارج می‌شود.

#### 3. Medium-term scheduling:

- اگر منابع محدود باشد یا نیاز به تخلیه حافظه باشد، پردازه به صف تعليق (Suspend Queue) فرستاده می‌شود.

- زمانی که منابع آزاد شدند، دوباره به صف آماده بازگردانده می‌شود.

#### 4. I/O or Event Wait:

- اگر پردازه نیاز به ورودی/خروجی داشته باشد، وارد صف Blocked می‌شود تا رویداد مورد نظر رخ دهد.
- اگر مدت زمان زیادی در Blocked بماند، ممکن است توسط زمان‌بند میان‌مدت به صف Blocked Suspend منتقل شود.

## دیاگرام Mermaid معادل:

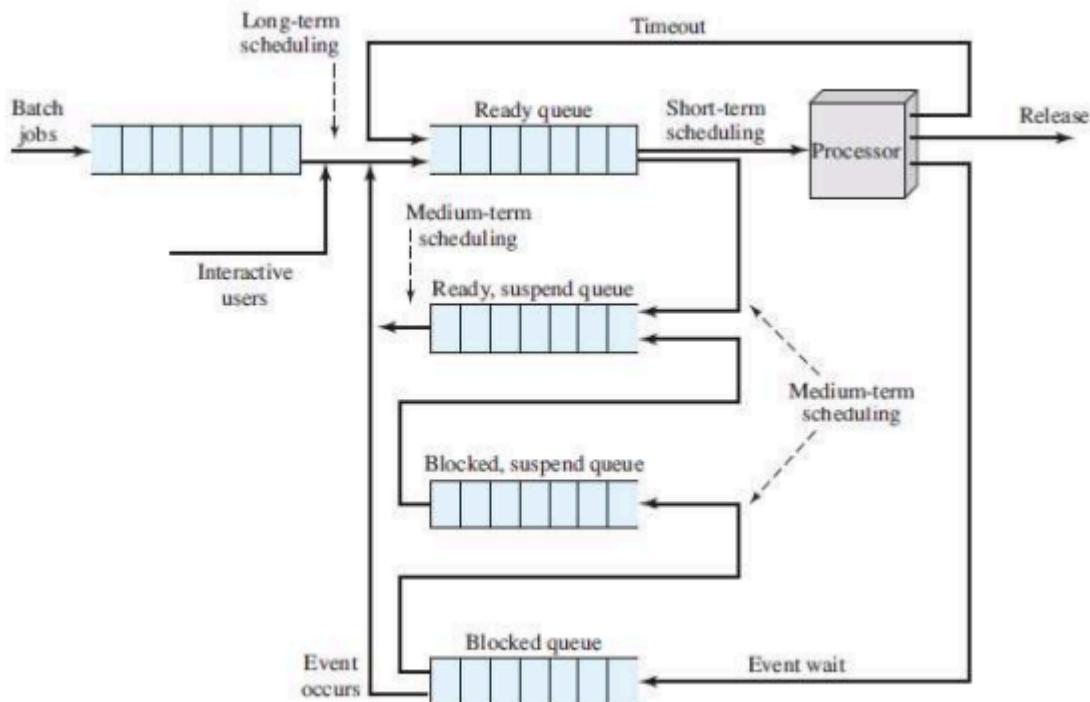


Figure 9.3 Queuing Diagram for Scheduling

## نکات کلیدی:

- صفهای Blocked و Ready جزو صفهای اصلی و فعال هستند.
- صفهای Suspend معمولاً برای مدیریت حافظه و بار زیاد سیستم استفاده می‌شوند.
- تعامل بین این صفها، از طریق زمان‌بندهای مختلف انجام می‌شود و هدف آن افزایش بهره‌وری سیستم است.

## مفاهیم پایه در زمان‌بندی پردازنده (Basic Concepts)



### هدف زمان‌بندی CPU



یکی از اهداف اصلی زمان‌بندی در سیستم‌عامل، **حداکثر استفاده از CPU (Maximum CPU Utilization)** است. این هدف معمولاً با استفاده از تکنیک **چندبرنامه‌ای (Multiprogramming)** محقق می‌شود؛ یعنی در هر لحظه، چند پردازه در سیستم وجود دارند تا در صورت انتظار یک پردازه (مثلًا برای I/O)، CPU بی‌کار نماند.

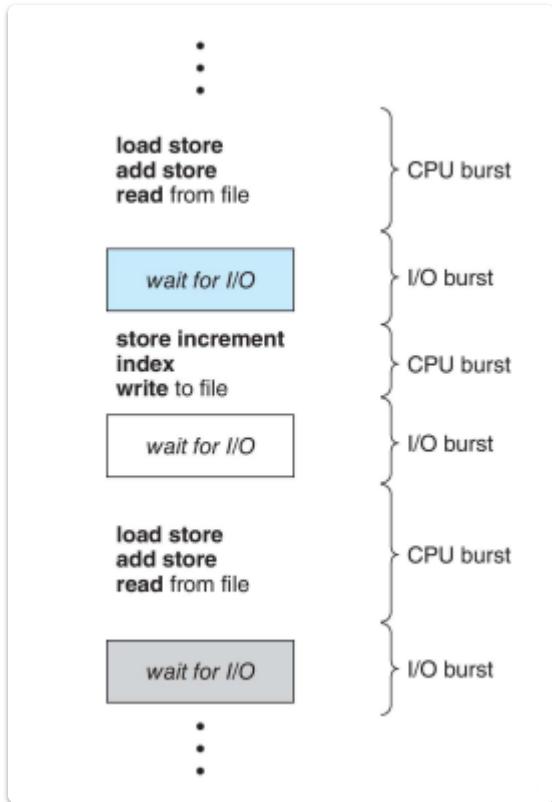
## چرخه CPU-I/O Burst (CPU-I/O Burst Cycle)



هر پردازه در زمان اجرای خود، به صورت تناوبی بین دو فاز حرکت می‌کند:

1. **CPU Burst** است و عملیات پردازشی انجام می‌دهد CPU زمانی که پردازه در حال اجرا روی **CPU انفجار** (CPU Burst) زمانی که پردازه منتظر عملیات ورودی/خروجی است (مثل خواندن از دیسک یا دریافت ورودی: **I/O انفجار** I/O Burst) کاربر.

این چرخه به شکل زیر تکرار می‌شود:



## توزيع CPU Burst ها

- یکی از مسائل مهم در طراحی الگوریتم‌های زمان‌بندی، **الگوی توزیع انفجارهای CPU** است.
- اغلب پردازه‌ها دارای **چند CPU Burst کوتاه** هستند و در میان آن‌ها، گاهی یک یا چند CPU Burst بلندتر نیز دیده می‌شود.
- این ویژگی برای الگوریتم‌های زمان‌بندی اهمیت زیادی دارد چون باعث می‌شود آن‌ها بتوانند تصمیم بگیرند کدام پردازه را برای اجرای بعدی انتخاب کنند.

## نتیجه‌گیری:

الگوریتم‌های مؤثر زمان‌بندی باید به گونه‌ای طراحی شوند که:

- از چرخه‌های CPU/I/O بهره‌برداری بهینه کنند.
- انفجارهای کوتاه CPU را سریع‌تر اجرا کنند (بهبود پاسخ‌دهی)،
- و در عین حال، انفجارهای طولانی‌تر را نیز منصفانه پردازش کنند.

## CPU (Histogram of CPU-burst Times) هیستوگرام زمان‌های انفجار

### مشاهده کلیدی:

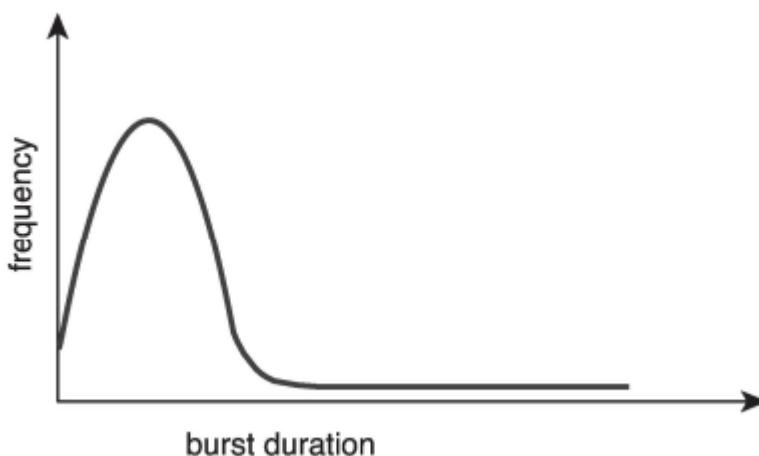
مطالعات تجربی روی عملکرد پردازه‌ها در سیستم‌های واقعی نشان داده‌اند که:

- تعداد زیادی از انفجارهای CPU بسیار کوتاه هستند
- در حالی که تعداد کمی از آنها طولانی‌اند

### توزيع هیستوگرام چگونه است؟

اگر بخواهیم طول زمان‌های انفجار CPU را به صورت هیستوگرام رسم کنیم، نمودار به شکل زیر خواهد بود:

- محور X: مدت زمان انفجار (CPU Burst Time)
- محور Y: تعداد دفعاتی که این زمان مشاهده شده است (Frequency)



نمودار دارای یک قله در بخش زمان‌های کوتاه و دنباله‌ای کشیده در بخش زمان‌های بلند است (distribution با دم بلند). (long tail –

### اهمیت برای زمان‌بندی:

- این الگو به الگوریتم‌های زمان‌بندی کمک می‌کند تا پیش‌بینی کنند کدام پردازه احتمالاً به CPU کمتری نیاز دارد.
- الگوریتم‌هایی مانند SJF (Shortest Job First) یا SRTF (Shortest Remaining Time First) از این ویژگی بهره می‌برند تا زمان پاسخ‌دهی را کاهش دهند.

### نتیجه‌گیری:

- اکثریت پردازه‌ها دارای زمان پردازش کوتاه هستند.
- تعداد کمی از پردازه‌ها انفجارهای طولانی CPU دارند.
- این توزیع پایه‌ای برای طراحی الگوریتم‌های زمان‌بندی کارآمد است.

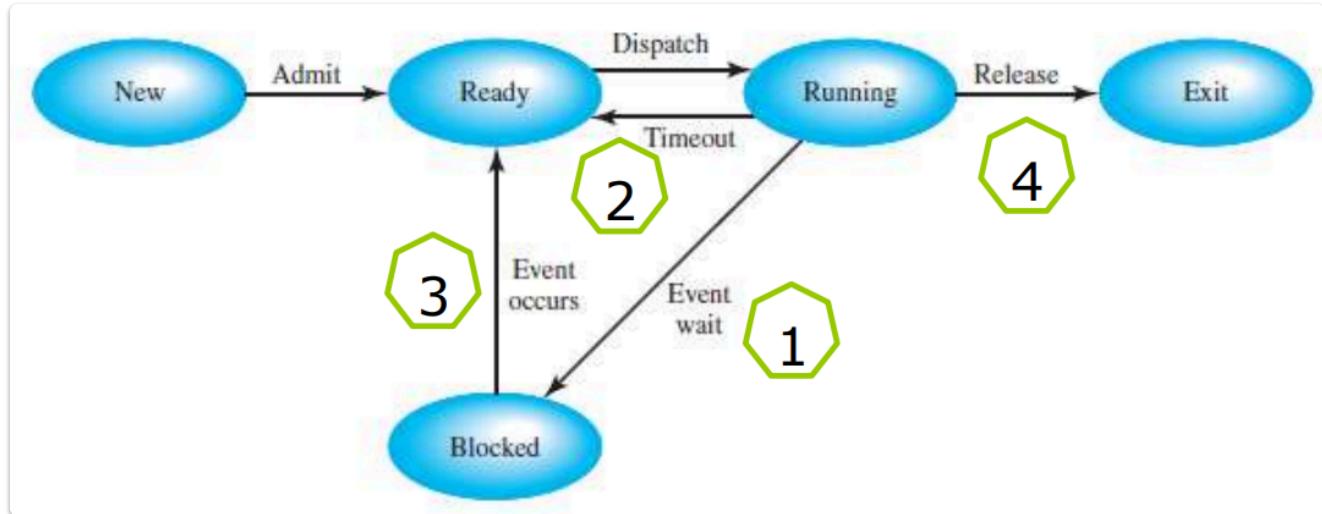
## 🧠 CPU Scheduler (Short-Term Scheduling)

### وظیفه:

زمان‌بند CPU (یا زمان‌بند کوتاه‌مدت) وظیفه دارد که از بین پردازه‌های موجود در **صف آماده** (Ready Queue) یکی را برای اجرای بعدی انتخاب کرده و آن را به یکی از هسته‌های CPU تخصیص دهد.

## موقعیت‌هایی که در آن تصمیم‌گیری زمان‌بندی انجام می‌شود:

1. پردازه از وضعیت به Waiting برود (مثلاً منتظر I/O باشد)
2. پردازه از وضعیت به Ready Running برود (مثلاً تایم‌اش تمام شده باشد - timeout)
3. پردازه از وضعیت به Ready Waiting برگرد (مثلاً عملیات I/O تمام شده باشد)
4. پردازه پایان یابد (Terminate)



در حالت‌های 1 و 4، نیازی به تصمیم‌گیری نیست، چون پردازه قبلی دیگر نمی‌تواند ادامه دهد، پس باید پردازه جدیدی انتخاب شود.  
در حالت‌های 2 و 3، زمان‌بند حق انتخاب دارد که بماند یا پردازه جدیدی را جایگزین کند.

## انواع زمان‌بندی:

### Nonpreemptive Scheduling (غیر‌پیش‌دستانه)

- فقط در حالت‌های 1 و 4 زمان‌بندی انجام می‌شود.
- وقتی CPU به پردازه‌ای داده شد، آن پردازه تا پایان یا تا زمان ورود به حالت Waiting روی CPU می‌ماند.
- سادگی و نبود شرایط رقابتی از مزایای آن است.

### Preemptive Scheduling (پیش‌دستانه)

- در تمامی حالتهای از جمله 2 و 3 نیز می‌تواند پردازه جدید را جایگزین کند.
- زمانی که پردازه‌ای با اولویت بالاتر وارد صفحه آماده شود، ممکن است پردازه فعلی از CPU کنار گذاشته شود.
- تقریباً تمامی سیستم‌عامل‌های مدرن مانند Windows، MacOS، Linux و UNIX از این نوع استفاده می‌کنند.

## ⚠ Preemptive Scheduling (Race Conditions) و شرایط رقابتی

در زمان‌بندی پیش‌دستانه، اگر دو پردازه به‌طور هم‌زمان روی داده‌ای مشترک کار کنند، امکان بروز شرایط رقابتی وجود دارد:

مثال:

پردازه اول در حال بروزرسانی داده‌ای مشترک است، اما پیش از اتمام کار، سیستم آن را متوقف کرده و پردازه دوم را اجرا می‌کند.

پردازه دوم داده‌ای را می‌خواند که در حالت ناقص و ناسازگار قرار دارد.

راه حل: استفاده از **سینکرونیزیشن (Synchronization)** و **مفاهیمی مانند قفل‌ها (Locks)** که در فصل ششم بررسی می‌شود.

## نتیجه‌گیری

- زمان‌بند کوتاه‌مدت یکی از حیاتی‌ترین بخش‌های سیستم‌عامل است.
- انتخاب نوع زمان‌بندی (Nonpreemptive یا Preemptive) بر عملکرد سیستم، تعامل با کاربران و پیچیدگی پیاده‌سازی تأثیر می‌گذارد.
- در سیستم‌های تعاملی، **پیش‌دستانه بودن** برای پاسخ‌گویی سریع‌تر ضروری است، ولی باید در برابر خطر **Race Condition** ایمن‌سازی صورت گیرد.

## .Dispatcher مفسر تصمیم زمان‌بند –

### تعريف:

ماژول **Dispatcher** یا اجراکننده‌ی تصمیمات زمان‌بند، بخشی از سیستم‌عامل است که پس از انتخاب یک پردازه توسط **CPU Scheduler**، کنترل CPU را به آن پردازه واگذار می‌کند.

### :Dispatcher وظایف

برای واگذاری CPU به پردازه جدید، Dispatcher باید این اقدامات را انجام دهد:

#### 1. **(تعویض متن اجرا):**

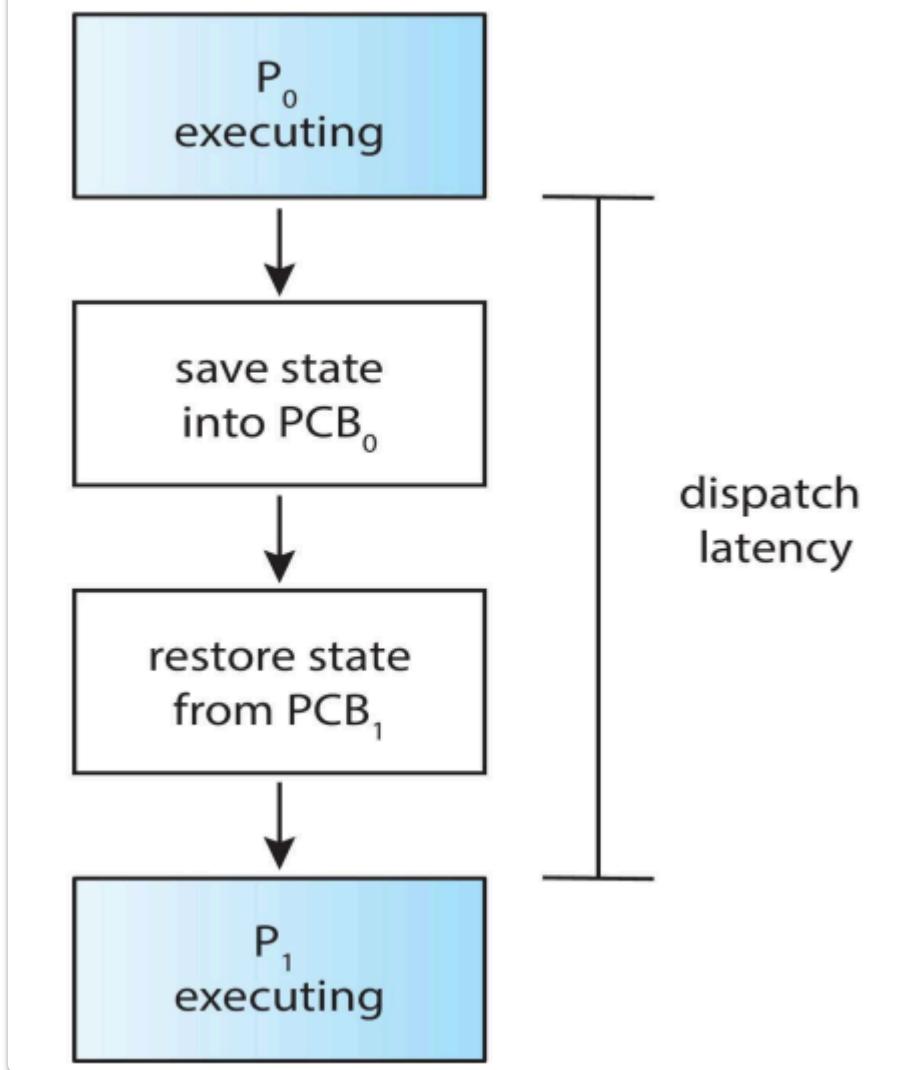
ذخیره وضعیت پردازه فعلی (مانند رجیسترها، شمارنده برنامه، پشته) و بازیابی وضعیت پردازه جدید.

#### 2. **(تغییر از حالت سیستمی به کاربری):**

برود، جایی که کدهای (User Mode) به حالت کاربری (Kernel Mode) سیستم‌عامل باید از حالت اجرایی سطح هسته برنامه کاربر اجرا می‌شوند.

#### 3. **(رفتن به موقعیت مناسب در برنامه):**

اجرای برنامه کاربر از همان جایی که قبلًا متوقف شده بود یا از نقطه شروع



## ⌚ Dispatch Latency (تأخير در انتقال):

▼ تعریف:

زمانی است که صرف می‌شود تا Dispatcher

- اجرای پردازه فعلی را متوقف کند، و
- اجرای پردازه جدید را آغاز کند.

این زمان هرچه کمتر باشد، سیستم پاسخ‌گویر و کارآثر خواهد بود.  
مخصوصاً در سیستم‌های Real-Time، *dispatch latency* باید تا حد امکان کوچک باشد.

📌 نکته مهم:

عملکرد Dispatcher در بهینه بودن **preemptive scheduling** (زمان‌بندی پیش‌دستانه) بسیار حیاتی است، چون با هر سوییچ پردازه (Context Switch)، وارد عمل می‌شود.

📊 **معیارهای زمان‌بندی (Scheduling Criteria)**

در طراحی و ارزیابی الگوریتم‌های زمان‌بندی، از مجموعه‌ای از معیارهای کلیدی استفاده می‌شود که به دو دسته کلی تقسیم می‌گردد:

## (System-Oriented Criteria) معیارهای سیستمی

### 1. CPU Utilization (استفاده از CPU)

- هدف: نگه داشتن CPU در حالت **فعال** در بیشترین زمان ممکن
- مقدار مطلوب: بین **40%** تا **90%** (بسته به نوع سیستم)

### 2. Throughput (بازده)

- تعریف: **تعداد پردازه‌های** که در یک واحد زمان به پایان می‌رسند
  - هدف: افزایش بهره‌وری کلی سیستم
-  **بیشتر throughput** به معنای کارایی بهتر سیستم است.

## (User-Oriented Criteria) معیارهای کاربرمحور

### 3. Turnaround Time (زمان برگشت)

- تعریف: مدت زمان از لحظه ورود پردازه تا پایان اجرای کامل آن  
$$\text{Turnaround} = \text{Completion time} - \text{Arrival time}$$

**Batch** معیار کلیدی در سیستم‌های

### 4. Waiting Time (زمان انتظار)

- تعریف: کل زمانی که پردازه در صفحه آماده (Ready Queue) منتظر بوده است
  - این زمان شامل زمان اجرای واقعی پردازه نمی‌شود
- الگوریتم‌های مثل SJF معمولاً این مقدار را به حداقل می‌رسانند.

### 5. Response Time (زمان پاسخ‌دهی)

- تعریف: زمان از لحظه ارسال درخواست توسط کاربر تا دریافت اولین پاسخ
  - معیار مهم در سیستم‌های تعاملی (Interactive Systems)
- حتی اگر کل کار انجام نشده باشد، کاربر انتظار دارد و اکنون اولیه سریع دریافت کند.

جمع‌بندی: 

معیار	نوع	هدف
CPU Utilization	سیستمی	فعال نگهداشتن CPU
Throughput	سیستمی	افزایش تعداد پردازه‌های تکمیل شده
Turnaround Time	کاربرمحور	کاهش زمان اجرای کامل پردازه
Waiting Time	کاربرمحور	کاهش زمان انتظار در صف
Response Time	کاربرمحور	پاسخگویی سریع اولیه به کاربر

## First-Come, First-Served (FCFS) Scheduling

### Overview

الگوریتم First-Come, First-Served (FCFS) یکی از الگوریتم‌های ساده و اولیه در سیستم‌های عامل است که بر اساس ترتیب ورود فرایندها به سیستم، آنها را اجرا می‌کند. در این الگوریتم، فرایندهایی که زودتر وارد می‌شوند ابتدا اجرا می‌شوند.

: مثال

فرایندهای زیر با زمان‌های burst به ترتیب در سیستم وارد می‌شوند:

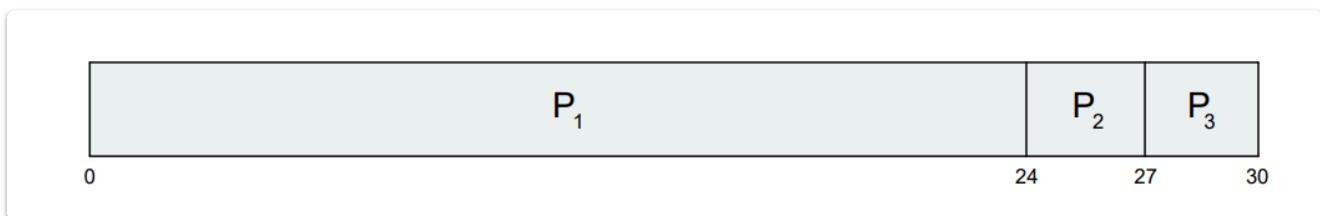
Process	Burst Time
P1	24
P2	3
P3	3

فرض می‌کنیم که فرایندها به ترتیب زیر وارد می‌شوند:

- P1
- P2
- P3

### Gantt Chart

مربوط به این زمان‌بندی به صورت زیر خواهد بود:



در این چارت، فرایندها به ترتیب به اجرای خود ادامه می‌دهند و زمان شروع و پایان هر فرایند مشخص شده است.

**محاسبه زمان‌های انتظار:**

- زمان انتظار برای P1: چون P1 اولین فرایند است، زمان انتظار آن برابر با صفر است.

Waiting Time for P1 = 0

- زمان انتظار برای P2: فرایند P2 بعد از P1 شروع می‌شود، بنابراین زمان انتظار آن برابر با زمان تکمیل P1 خواهد بود.

Waiting Time for P2 = 24

- زمان انتظار برای P3: فرایند P3 بعد از P1 و P2 شروع می‌شود، بنابراین زمان انتظار آن برابر با زمان تکمیل P2 خواهد بود.

Waiting Time for P3 = 27

### محاسبه زمان انتظار متوسط:

برای محاسبه زمان انتظار متوسط، از فرمول زیر استفاده می‌کنیم:

$$\text{Average Waiting Time} = (0 + 24 + 27) / 3 = 17$$

بنابراین، زمان انتظار متوسط برابر با 17 واحد زمانی است.

### نکات:

- الگوریتم FCFS بسیار ساده است، اما ممکن است منجر به تأخیرهای زیادی برای فرایندهای بعدی (به خصوص اگر یک فرایند طولانی مدت ابتدا وارد شود) شود.
- این الگوریتم به دلیل ویژگی "اولین آمده، اولین اجرا می‌شود"، ممکن است در شرایط خاص کارایی پایین‌تری داشته باشد.

## First-Come, First-Served (FCFS) Scheduling (Cont.)

### Overview

در این بخش، همانطور که قبلاً ذکر شد، الگوریتم First-Come, First-Served (FCFS) به ترتیب ورود فرایندها عمل می‌کند. در این مثال، ترتیب ورود فرایندها تغییر کرده و این امر تأثیر زیادی بر زمان‌های انتظار فرایندها دارد.

### فرضیات:

فرایندها با زمان‌های burst زیر وارد می‌شوند:

Process	Burst Time
P1	24
P2	3
P3	3

در این سناریو، فرض می‌کنیم که فرایندها به ترتیب زیر وارد می‌شوند:

- P2
- P3
- P1

### Gantt Chart

مربوط به این ترتیب ورود به صورت زیر خواهد بود:



در این چارت، فرایندها به ترتیب به اجرای خود ادامه می‌دهند و زمان شروع و پایان هر فرایند مشخص شده است.

### محاسبه زمان‌های انتظار:

- **زمان انتظار برای P1:** در اینجا، فرایند P1 پس از P2 و P3 شروع می‌شود. بنابراین، زمان انتظار آن برابر با زمان تکمیل P2 و P3 خواهد بود.

Waiting Time for P1 = 6

- **زمان انتظار برای P2:** چون P2 اولین فرایند است، زمان انتظار آن برابر با صفر خواهد بود.

Waiting Time for P2 = 0

- **زمان انتظار برای P3:** فرایند P3 بعد از P2 شروع می‌شود، بنابراین زمان انتظار آن برابر با زمان تکمیل P2 خواهد بود.

Waiting Time for P3 = 3

### محاسبه زمان انتظار متوسط:

برای محاسبه زمان انتظار متوسط، از فرمول زیر استفاده می‌کنیم:

$$\text{Average Waiting Time} = (6 + 0 + 3) / 3 = 3$$

بنابراین، زمان انتظار متوسط در این حالت برابر با 3 واحد زمانی است.

## تحلیل

- در مقایسه با حالت قبلی، این سناریو زمان انتظار متوسط کمتری دارد، بنابراین عملکرد بهتری در این مورد دیده می‌شود.
- اثر کاروان (Convoy Effect):** این پدیده زمانی رخ می‌دهد که یک فرایند کوتاه مدت پس از یک فرایند بلند مدت قرار گیرد و زمان انتظار زیادی ایجاد کند. در این مثال، اگر P1 یک فرایند طولانی مدت باشد، باعث ایجاد تأخیر در فرایندهای بعدی می‌شود.
- فرایندهای CPU-bound و I/O-bound:** زمانی که در سیستم فرایندهای CPU-bound (که به زمان پردازش طولانی نیاز دارند) و I/O-bound (که بیشتر به عملیات ورودی/خروجی نیاز دارند) وجود دارند، استفاده از الگوریتم FCFS می‌تواند به شدت تأثیرگذار باشد. در اینجا، فرایندهای I/O-bound ممکن است در پشت فرایندهای CPU-bound گرفتار شوند، که موجب ایجاد تأخیر بیشتر می‌شود.

## نکات:

- مزیت:** یک الگوریتم ساده است که برای سیستم‌هایی با بار کم و پردازش‌های متنوع خوب عمل می‌کند.
- معایب:** برای سیستم‌هایی که فرایندهای کوتاه و طولانی به طور همزمان دارند، FCFS می‌تواند منجر به تأخیرهای زیاد شود و اثر "کاروان" را ایجاد کند.



## FCFS Scheduling (Cont.)

Table 9.4 Process Scheduling Example

Process	Arrival Time	Service Time
A	0	3
B	2	6
C	4	4
D	6	5
E	8	2

Process	Arrival Time	Service Time ( $T_s$ )	Start Time	Finish Time	Turnaround Time ( $T_r$ )	$T_r/T_s$
W	0	1	0	1	1	1
X	1	100	1	101	100	1
Y	2	1	101	102	100	100
Z	3	100	102	202	199	1.99
Mean					100	26

## Shortest-Job-First (SJF) Scheduling

### Overview

الگوریتم Shortest-Job-First (SJF) یکی از الگوریتم‌های زمان‌بندی است که در آن، فرایندی که کوتاه‌ترین زمان CPU را نیاز دارد، ابتدا اجرا می‌شود. این الگوریتم بر اساس پیش‌بینی طول مدت اجرای فرایندها عمل می‌کند.

## اصول کلی:

- برای هر فرایند، طول زمان **بعدی** CPU burst پیش‌بینی می‌شود.
- فرایند با کمترین زمان **بعدی** برای اجرا انتخاب می‌شود.
- SJF** به طور نظری بهینه است و کمترین **زمان انتظار متوسط** را برای مجموعه‌ای از فرایندها به همراه دارد.

## نسخه پیشرفته: Shortest-Remaining-Time-First (SRTF)

نسخه **پیش‌امدی** (preemptive) SJF که به آن Shortest-Remaining-Time-First (SRTF) می‌گویند، به این صورت است که اگر فرایند جدیدی وارد سیستم شود که مدت زمان آن از باقی‌مانده زمان یک فرایند در حال اجرا کمتر باشد، فرایند در حال اجرا متوقف شده و فرایند جدید اجرا می‌شود.

## چگونه طول زمان CPU بعدی را پیش‌بینی کنیم؟

- می‌توان از کاربر خواست که این زمان را وارد کند.
- یا می‌توان با استفاده از **برآورد** (estimation) این مقدار را تخمین زد.

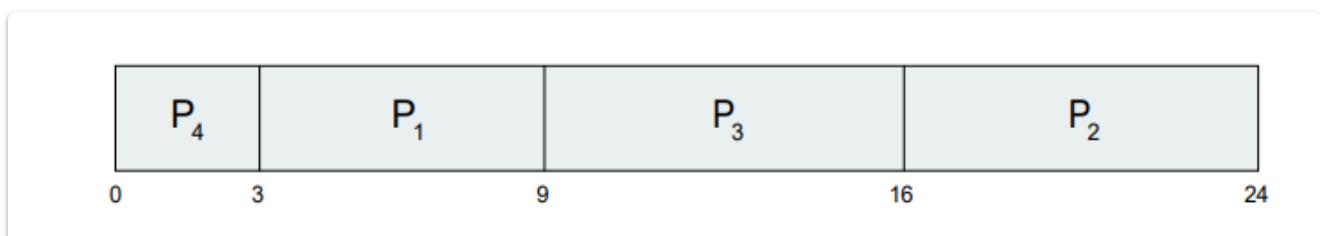
## مثال:

در اینجا فرایندهایی با زمان‌های burst مشخص داریم:

Process	Burst Time
P1	6
P2	8
P3	7
P4	3

## Gantt Chart

با استفاده از الگوریتم SJF، چارت زمان‌بندی به صورت زیر خواهد بود:



در این چارت، فرایندها به ترتیب با کمترین زمان burst انتخاب و اجرا می‌شوند.

## محاسبه زمان‌های انتظار:

- زمان انتظار برای  $P_1$ :** زمان انتظار  $P_1$  برابر با زمان تکمیل  $P_4$  است که 3 واحد زمانی است.

Waiting Time for  $P_1 = 3$

- زمان انتظار برای P2: زمان انتظار P2 برابر با زمان تکمیل P3 است که 16 واحد زمانی است.

Waiting Time for P2 = 16

- زمان انتظار برای P3: زمان انتظار P3 برابر با زمان تکمیل P1 است که 9 واحد زمانی است.

Waiting Time for P3 = 9

- زمان انتظار برای P4: چون P4 اولین فرایند است، زمان انتظار آن برابر با صفر است.

Waiting Time for P4 = 0

### محاسبه زمان انتظار متوسط:

برای محاسبه زمان انتظار متوسط، از فرمول زیر استفاده می‌کنیم:

$$\text{Average Waiting Time} = (3 + 16 + 9 + 0) / 4 = 7$$

بنابراین، زمان انتظار متوسط برابر با 7 واحد زمانی است.

### نکات:

- به طور تئوری بهترین زمان انتظار را فراهم می‌کند SJF.
- با این حال، SJF غیرقابل پیش‌بینی است زیرا به پیش‌بینی زمان‌های بعدی burst نیاز دارد که می‌تواند با خطا همراه باشد.
- نسخه پیش‌امدنا (SRTF) می‌تواند عملکرد بهتری در شرایط خاص داشته باشد اما پیچیده‌تر است.

## Determining Length of Next CPU Burst

### Overview

برای پیش‌بینی طول زمان CPU بعدی، معمولاً نمی‌توانیم مقدار دقیق آن را تعیین کنیم، اما می‌توانیم تخمین بزنیم که این مقدار مشابه با زمان‌های قبلی CPU خواهد بود. برای این کار، معمولاً از روش‌های میانگین نمایی (Exponential Averaging) استفاده می‌شود.

### اصول کلی:

- طول زمان CPU بعدی باید مشابه زمان‌های قبلی باشد.
- برای پیش‌بینی طول زمان CPU بعدی، از زمان‌های قبلی استفاده کرده و با استفاده از **میانگین نمایی** (Exponential) تخمین زده می‌شود.
- یکی از مقادیر معمولی برای  $\alpha$ ، مقدار  $\frac{1}{2}$  است.

## پیش‌بینی طول زمان CPU بعدی:

پیش‌بینی طول زمان CPU بعدی را می‌توان با استفاده از فرمول زیر انجام داد:

$$\tau_{n+1} = \alpha * \tau_n + (1 - \alpha) * \tau_n$$

در اینجا:

- $\tau_{n+1}$ : بعدی طول زمان CPU
- $\tau_n$ : فعلی CPU زمان
- $\alpha$ : ضریب نمایی (معمولًاً مقدار آن  $\frac{1}{2}$  است)

## مثال‌ها از میانگین نمایی:

**حالت 1:  $\alpha = 0$**

در این حالت، فقط از آخرین زمان CPU استفاده می‌شود و تاریخچه اخیر به حساب نمی‌آید:

$$\tau_{n+1} = \tau_n$$

این بدان معناست که تنها از مقدار زمان CPU قبلی استفاده می‌شود و هیچ وزنی به زمان‌های قبلی داده نمی‌شود.

**حالت 2:  $\alpha = 1$**

در این حالت، تنها از زمان فعلی CPU استفاده می‌شود و هیچ وزن‌دهی به تاریخچه داده نمی‌شود:

$$\tau_{n+1} = \alpha * \tau_n = \tau_n$$

**حالت عمومی:**

اگر فرمول را گسترش دهیم، به صورت زیر خواهیم داشت:

$$\tau_{n+1} = \alpha * \tau_n + (1 - \alpha) * \alpha * \tau_{n-1} + (1 - \alpha)^2 * \alpha * \tau_{n-2} + \dots + (1 - \alpha)^{n+1} * \tau_0$$

در اینجا، هر کدام از مقادیر تاریخچه قبلی وزنی کمتر از مقدار قبلی خواهند داشت، زیرا  $\alpha$  و  $(1 - \alpha)$  هر دو کمتر از یا مساوی 1 هستند.

## نکات:

- پارامتری است که میزان تأثیر زمان‌های قبلی در پیش‌بینی زمان بعدی را کنترل می‌کند  $\alpha$ .
  - اگر  $\alpha$  نزدیک به 1 باشد، فقط آخرین زمان CPU در پیش‌بینی استفاده خواهد شد.
  - اگر  $\alpha$  نزدیک به 0 باشد، تاریخچه کامل زمان‌های قبلی تأثیر زیادی در پیش‌بینی نخواهد داشت.
  - این روش برای تخمین طول زمان‌های CPU بعدی بسیار موثر است، اما همیشه دارای خطاهایی خواهد بود، چرا که پیش‌بینی‌ها هیچ‌گاه 100% دقیق نخواهند بود.

## Shortest Remaining Time First Scheduling

### Overview

الگوریتم **Shortest Job Next** (SJN) نسخه پیش‌امدنی (preemptive) از الگوریتم **Shortest Remaining Time First (SRTF)** است. در این الگوریتم، زمانی که فرایند جدیدی به صفت آماده می‌آید، تصمیم‌گیری در مورد اینکه کدام فرایند باید اجرا شود مجدداً با استفاده از الگوریتم SJN انجام می‌شود.

### ویژگی‌ها:

- در SRTF، هر زمان که فرایند جدیدی وارد صفت آماده شود، الگوریتم تصمیم می‌گیرد که کدام فرایند با توجه به زمان باقی‌مانده باید اجرا شود.
- الگوریتم SRTF معمولاً زمان انتظار متوسط کمتری نسبت به SJN دارد، زیرا می‌تواند هر زمان که فرایند جدیدی وارد شود، تصمیم خود را تغییر دهد.

### آیا SRTF نسبت به SJN بهینه‌تر است؟

- الگوریتم SRTF از نظر زمان انتظار متوسط ممکن است بهینه‌تر از SJN باشد، زیرا می‌تواند در هر لحظه تصمیم خود را تغییر دهد و فرایندهایی که زمان باقی‌مانده کمتری دارند را زودتر اجرا کند.
- اما این الگوریتم نیاز به پیاده‌سازی پیچیده‌تری دارد و ممکن است باعث پیش‌آمدگی (preemption) مکرر شود که می‌تواند مشکلاتی مانند overhead را ایجاد کند.

### مثال: الگوریتم Shortest Remaining Time First (SRTF)

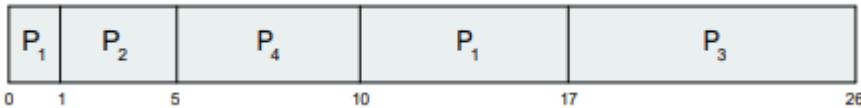
در این مثال، به بررسی الگوریتم زمان‌بندی **کوتاه‌ترین زمان باقی‌مانده اول** (SRTF) می‌پردازیم که نسخه Preemptive از الگوریتم **Shortest Job First (SJF)** است.

در این حالت، فرآیندی که **کوتاه‌ترین زمان اجرای باقی‌مانده** را دارد، اولویت اجرا می‌گیرد، حتی اگر این باعث **وقفه** در اجرای یک فرآیند در حال اجرا شود.

### مشخصات فرآیندها:

فرآیند	زمان ورود (Arrival Time)	زمان اجرای مورد نیاز (Burst Time)
P <sub>1</sub>	0	8
P <sub>2</sub>	1	4
P <sub>3</sub>	2	9
P <sub>4</sub>	3	5

## نمودار گانت (Gantt Chart)



نکته: نمودار بالا نشان می‌دهد که فرآیندها به ترتیب زیر اجرا شده‌اند:

- ابتدا P<sub>1</sub> برای 1 واحد اجرا شد.
- سپس P<sub>2</sub> با زمان باقی‌مانده کمتر وارد شد و تا پایان اجرا شد.
- بعد P<sub>4</sub> وارد شد و اجرا شد.
- سپس P<sub>1</sub> به اجرای باقی‌مانده‌اش ادامه داد.
- در نهایت P<sub>3</sub> با زمان زیاد ولی بدون وقفه اجرا شد.

## محاسبه زمان انتظار میانگین (Average Waiting Time)

فرمول زمان انتظار برای هر فرآیند:

$$\text{Waiting Time} = (\text{Finish Time} - \text{Arrival Time} - \text{Burst Time})$$

محاسبه برای هر فرآیند:

- P<sub>1</sub>: 9 = 8 - 0 - 17 → پایان در 17
- P<sub>2</sub>: 0 = 4 - 1 - 5 → پایان در 5
- P<sub>3</sub>: 15 = 9 - 2 - 26 → پایان در 26
- P<sub>4</sub>: 2 = 5 - 3 - 10 → پایان در 10

میانگین زمان انتظار:

$$6.5 = \frac{4}{26} = \frac{4}{(2 + 15 + 0 + 9)}$$

نکات مهم:

- الگوریتم SRTF در لحظه ورود یک فرآیند جدید بررسی می‌کند که آیا باید فرآیند فعلی را قطع کند یا نه.
- این الگوریتم برای کاهش زمان انتظار میانگین بسیار مؤثر است اما ممکن است باعث گرسنگی فرآیندهای بلندتر شود.

## الگوریتم Round Robin (RR)

الگوریتم Round Robin (RR) یکی از رایج‌ترین الگوریتم‌های زمان‌بندی Preemptive در سیستم‌عامل‌هاست که برای اشتراک‌گذاری منصافانه CPU بین فرآیندها استفاده می‌شود.

تعریف:

- هر فرآیند یک واحد زمانی کوچک از CPU (به نام time slice یا time quantum) دریافت می‌کند.
- مقدار معمول برای time quantum بین 10 تا 100 میلیثانیه است.
- پس از پایان این زمان، اگر فرآیند کامل نشده باشد:

  - وقفه (preemption) ایجاد می‌شود، فرآیند به انتهای صف آماده (Ready Queue) اضافه می‌شود،
  - فرآیند بعدی برای اجرا انتخاب می‌شود.

### ویژگی‌های کلیدی:

- اگر  $n$  فرآیند در صف آماده وجود داشته باشد و اندازه time quantum برابر  $q$  باشد:

  - هر فرآیند در هر چرخه، حداقل  $q$  واحد زمانی از CPU دریافت می‌کند.
  - هیچ فرآیندی بیشتر از  $(n-1) \times q$  منتظر نمی‌ماند.

- تایم سیستم پس از هر time quantum پس از هر وقفه ایجاد می‌کند تا فرآیند بعدی زمان‌بندی شود.

### عملکرد الگوریتم:

توضیح	رفتار الگوریتم	مقدار $q$
مثل الگوریتم صف اول وارد، اول خارج	FIFO (FCFS)	بزرگ $q$
زمان‌بندی عادلانه‌تر ولی با سربار بیشتر	Round Robin	واقعی $q$ کوچک $q$

### نکته مهم:

| اندازه‌ی time quantum باید بزرگ‌تر از زمان context switch باشد، در غیر این صورت:

- زمان زیادی صرف تعویض بین فرآیندها می‌شود.
- بهره‌وری CPU کاهش می‌یابد.

### مزایا و معایب:

- ✓ منصفانه برای همهی فرآیندها
- ✓ پاسخ‌دهی سریع برای پردازش‌های تعاملی
- ✗ در صورت انتخاب نامناسب  $q$ ، کارایی پایین می‌آید
- ✗ امکان تاخیر بالا برای فرآیندهای طولانی‌تر

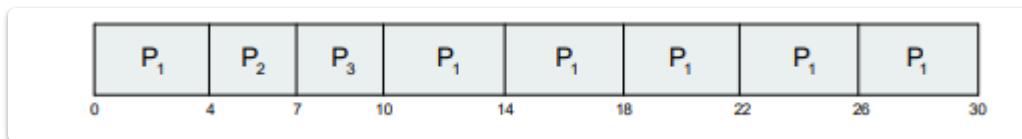
## مثال از الگوریتم Time Quantum = 4 با Round Robin

در این مثال، الگوریتم زمان‌بندی Round Robin با time quantum 4 واحد زمانی بررسی می‌شود.

## مشخصات فرآیندها:

فرآیند	زمان اجرای مورد نیاز (Burst Time)
P <sub>1</sub>	24
P <sub>2</sub>	3
P <sub>3</sub>	3

## نمودار گانت (Gantt Chart) :



## تحلیل:

- ابتدا P<sub>1</sub> اجرا شده و پس از 4 واحد، چون تمام نشده، به انتهای صفحه بازمی‌گردد.
- سپس P<sub>2</sub> و P<sub>3</sub> که زمان اجرایشان 3 واحد است، در یک time slice اجرا شده و تمام می‌شوند.
- سپس P<sub>1</sub> به صورت پیاپی در هر بار 4 واحد اجرا می‌شود تا در نهایت مجموع 24 واحد زمانش به پایان برسد.

## نکات کلیدی:

- معمولًا میانگین زمان چرخش (Turnaround Time) در Round Robin بیشتر از الگوریتم SJF است.
- اما زمان پاسخگویی (Response Time) بهتر است، مخصوصاً برای فرآیندهای تعاملی (Interactive).

## :Time Quantum درباره انتخاب

باید بزرگ‌تر از زمان  $q$  باشد.

مؤلفه	مقدار توصیه شده
time quantum (q)	بین 10 تا 100 میلیثانیه
context switch time	کمتر از 10 میکروثانیه

در غیر این صورت، سربار تعویض بین فرآیندها زیاد می‌شود و بهره‌وری کاهش می‌یابد.

## نتیجه‌گیری:

الگوریتم Round Robin time quantum می‌تواند زمان پاسخ‌گویی فرآیندها را بهبود دهد اما باید تعادل خوبی بین اندازه‌ی ۹ و سربار تعویض فرآیندها برقرار باشد.

## Time Quantum و Context Switch Time

در این اسلاید، تأثیر اندازه‌ی time quantum بر تعداد سوئیچ‌های زمینه‌ای (context switches) در اجرای یک فرآیند واحد با زمان پردازش ۱۰ واحد بررسی شده است.

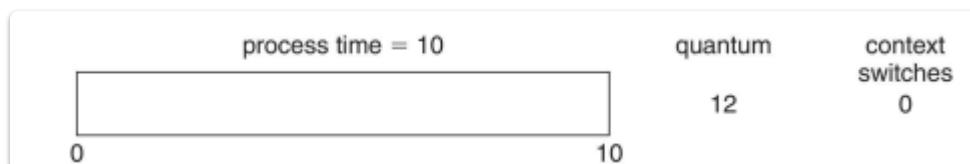
**فرض:**

- زمان اجرای یک فرآیند برابر ۱۰ واحد زمانی است.
- بررسی با ۳ مقدار مختلف برای time quantum

Time Quantum (q)	تعداد context switch
12	0
6	1
1	9

**تحلیل بصری:**

**حالت اول:**  $q = 12$

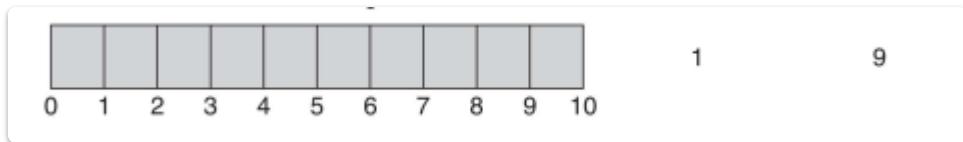


- چون  $q = 12$  از زمان اجرای کامل فرآیند (10) بیشتر است، فرآیند بدون وقفه اجرا می‌شود.
- هیچ context switch رخ نمی‌دهد.

**حالت دوم:**  $q = 6$



- فرآیند ابتدا برای 6 واحد اجرا شده، سپس context switch اتفاق افتاده و ادامه‌ی کار در نوبت دوم انجام شده.
- 1 بار context switch



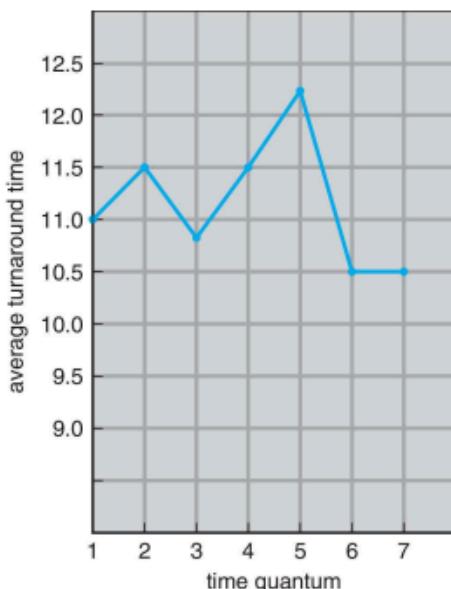
- فرآیند 10 بار اجرا شده، هر بار فقط 1 واحد.
- بین هر اجرای متوالی، 1 context switch اتفاق افتاده است.
- مجموعاً 9 بار context switch انجام شده.

### نتیجه‌گیری:

- هرچه time quantum کوچک‌تر باشد، تعداد context switch‌ها بیشتر می‌شود.
  - اگر  $q$  خیلی کوچک باشد، زمان زیادی صرف جابه‌جایی بین فرآیندها می‌شود که منجر به افزایش سربار سیستم و کاهش بهره‌وری می‌شود.
  - انتخاب  $q$  مناسب باید با در نظر گرفتن زمان context switch واقعی سیستم انجام شود.
- تعادل بین زمان پاسخ‌گویی و کارایی با انتخاب time quantum مناسب حاصل می‌شود.



## Turnaround Time Varies With The Time Quantum



process	time
$P_1$	6
$P_2$	3
$P_3$	1
$P_4$	7

80% of CPU bursts  
should be shorter than q



## Process Scheduling Example

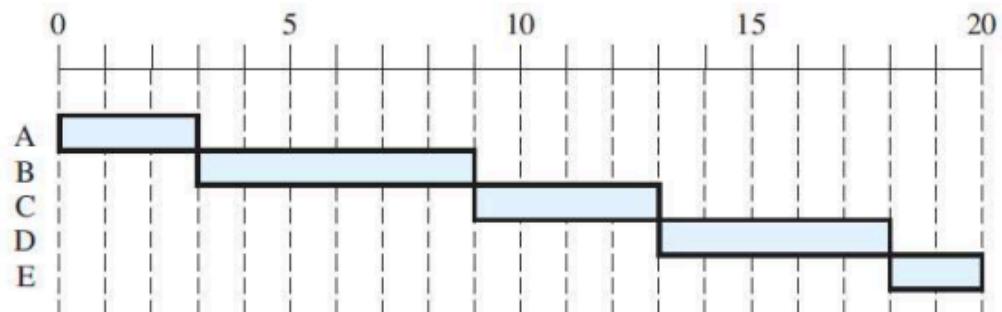
**Table 9.4 Process Scheduling Example**

Process	Arrival Time	Service Time
A	0	3
B	2	6
C	4	4
D	6	5
E	8	2



## FCFS

First-come-first served (FCFS)

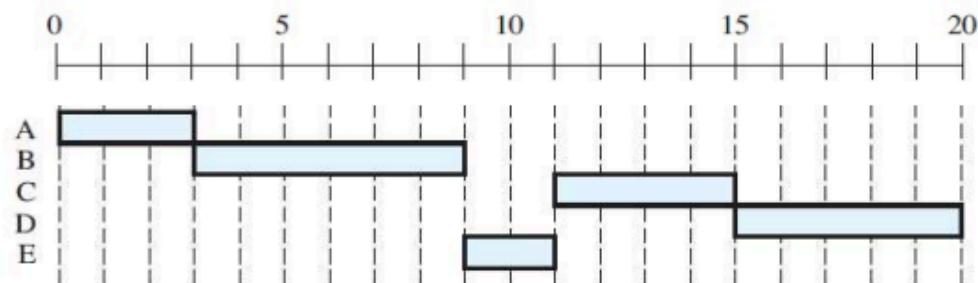


Process	A	B	C	D	E	
Arrival Time	0	2	4	6	8	
Service Time ( $T_s$ )	3	6	4	5	2	Mean
FCFS						
Finish Time	3	9	13	18	20	
Turnaround Time ( $T_r$ )	3	7	9	12	12	8.60
$T_r/T_s$	1.00	1.17	2.25	2.40	6.00	2.56



## SPN

Shortest process next (SPN)



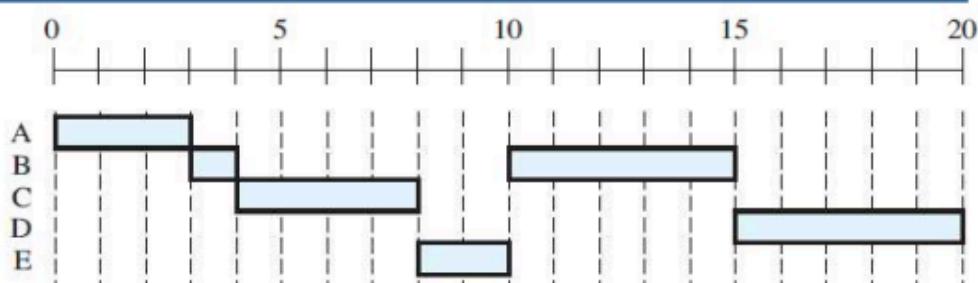
## SPN

Finish Time	3	9	15	20	11	
Turnaround Time ( $T_r$ )	3	7	11	14	3	7.60
$T_r/T_s$	1.00	1.17	2.75	2.80	1.50	1.84



## SRT

Shortest remaining time (SRT)



## SRT

Finish Time	3	15	8	20	10	
Turnaround Time ( $T_r$ )	3	13	4	14	2	7.20
$T_r/T_s$	1.00	2.17	1.00	2.80	1.00	1.59

## HRRN (Highest Response Ratio Next) الگوریتم

الگوریتم HRRN یکی از الگوریتم‌های غیر پیش‌گیرنده (Non-preemptive) زمان‌بندی فرآیندها است که در آن تصمیم‌گیری بر اساس نسبت پاسخ‌دهی (Response Ratio) انجام می‌شود.

## هدف:

ترکیب مزایای الگوریتمهای:

- **FCFS** (اولین وارد، اولین خارج)
- **SJF** (کوتاهترین زمان پردازش)

برای کاهش زمان انتظار و افزایش عدالت بین فرآیندها.

## فرمول نسبت پاسخدهی (Response Ratio)

$$\text{Response Ratio} = (\text{Waiting Time} + \text{Burst Time}) / \text{Burst Time}$$

یا به شکل دیگر:

$$R = 1 + (\text{Waiting Time} / \text{Burst Time})$$

## شیوه کار:

1. در هر گام از زمانبندی، فرآیند با بیشترین مقدار R انتخاب می‌شود.
2. این باعث می‌شود فرآیندهایی که زمان اجرای کوتاه دارند زودتر اجرا شوند (مثل SJF).
3. و فرآیندهایی که مدت زیادی منتظر مانده‌اند نیز در اولویت قرار گیرند.

## مثال:

فرض کن سه فرآیند با مشخصات زیر داریم:

Process	Arrival Time	Burst Time	Waiting Time	Response Ratio
P1	0	3	4	(4+3)/3 = 2.33
P2	1	6	2	(2+6)/6 = 1.33
P3	2	4	4	(4+4)/4 = 2.00

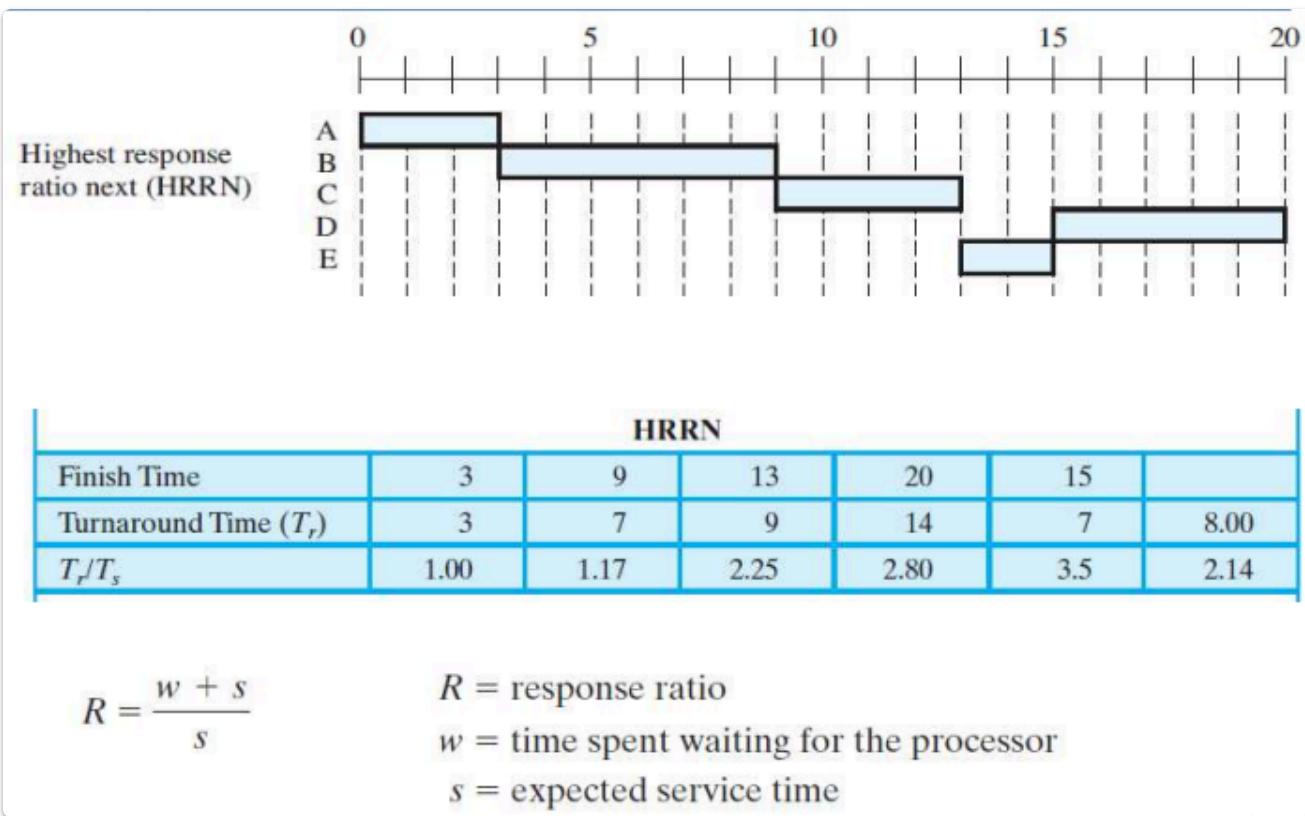
در این حالت، P1 با نسبت پاسخدهی 2.33 بیشترین مقدار را دارد و انتخاب می‌شود.

## مزایا:

- ✓ نسبت به SJF عادلانه‌تر است، چون فرآیندهای بلند نیز بالاخره نوبت می‌گیرند.
- ✓ باعث کاهش **Starvation** (گرسنگی فرآیندها) می‌شود.
- ✓ در بارهای کاری متعادل، عملکرد خوبی دارد.

## معایب:

نیاز به محاسبه Response Ratio برای هر فرآیند در هر گام.  
مناسب سیستم‌های بلادرنگ (Real-time) نیست.



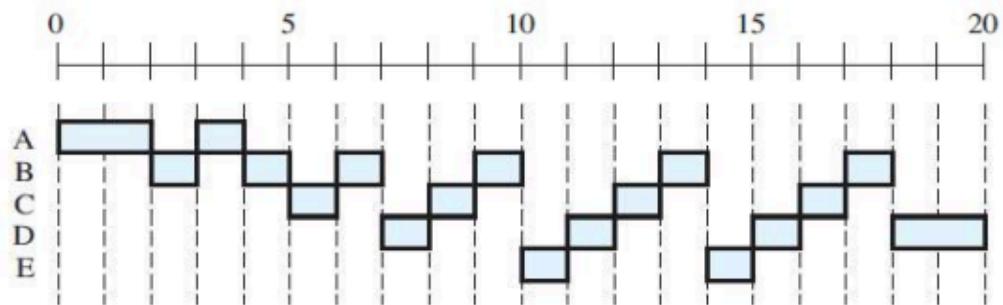
### جمع‌بندی:

الگوریتم HRRN یک روش مؤثر برای تعادل بین پاسخ‌دهی سریع و جلوگیری از گرسنگی فرآیندهاست، مخصوصاً در سیستم‌های [Batch Processing](#).



## RR

Round-robin  
(RR),  $q = 1$



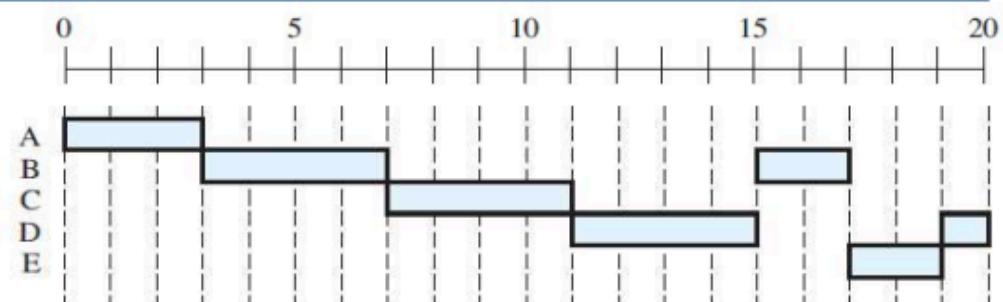
**RR  $q = 1$**

Finish Time	4	18	17	20	15	
Turnaround Time ( $T_r$ )	4	16	13	14	7	10.80
$T_r/T_s$	1.33	2.67	3.25	2.80	3.50	2.71



## RR

Round-robin  
(RR),  $q = 4$



**RR  $q = 4$**

Finish Time	3	17	11	20	19	
Turnaround Time ( $T_r$ )	3	15	7	14	11	10.00
$T_r/T_s$	1.00	2.5	1.75	2.80	5.50	2.71

## نمودار صفحه زمانبندی Round-Robin مجازی (Virtual Round-Robin)

: مقدمه

در زمانبندی **Virtual Round-Robin (VRR)**، مفهومی از **صفهای چندگانه** برای رسیدگی به فرآیندها با اولویت‌های مختلف به کار می‌رود، به گونه‌ای که تعامل بین زمانبندی **preemptive** و پاسخ‌دهی سریع فرآیندهای آماده بهتر مدیریت شود.

## تفاوت با Round Robin معمولی:

Virtual Round Robin	المعمولی Round Robin	ویژگی
ندارد (صف چندبخشی دارد)	دارد	صف واحد
دارد	ندارد	mekanizm اجرایی مجدد سریع
بهتر	ضعیف	مدیریت I/O-bound بهتر

## اجزای صف‌بندی در VRR:

### 1. Active Queue

- فرآیندهایی که آماده اجرا هستند.
- از این صف زمان‌بند فرآیند را انتخاب می‌کند.

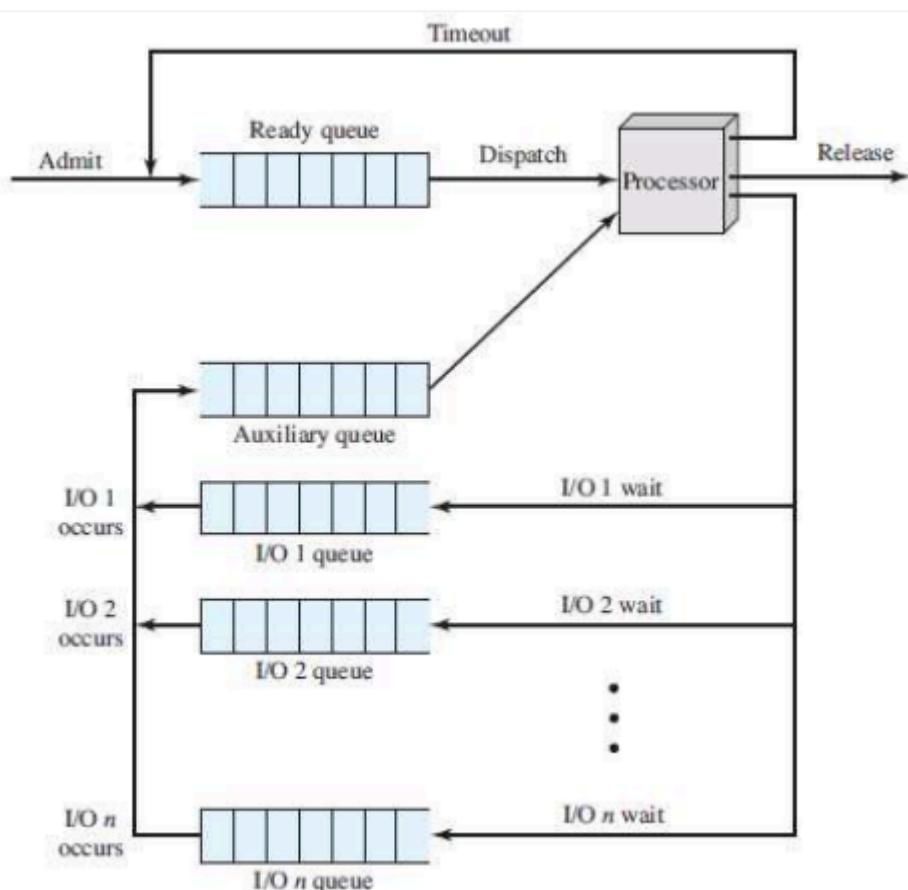
### 2. Expired Queue

- فرآیندهایی که time quantum خود را مصرف کرده‌اند.
- پس از اتمام یک دور، active و expired با هم جایه‌جا می‌شوند.

### 3. I/O Queue / Bonus Queue (در نسخه‌هایی از VRR)

- فرآیندهایی که زودتر از پایان time quantum خود برای عملیات I/O متوقف شده‌اند.
- به آن‌ها پاداش داده می‌شود و پیش از ورود به expired، به active بازمی‌گردند.

## نمودار صف در زمان‌بند VRR:



## ویژگی کلیدی:

- فرآیندهایی که زودتر به I/O می‌روند، سریع‌تر به Active Queue برمی‌گردند. این باعث بهبود پاسخ‌دهی برای فرآیندهای I/O-bound می‌شود.

## مزایا:

- پاسخ‌دهی بهتر نسبت به RR معمولی
- عملکرد بهتر برای ترکیب فرآیندهای CPU-bound و I/O-bound
- جلوگیری از starvation با تقسیم عادلانه زمان

## کاربرد:

این نوع زمان‌بندی در سیستم‌های مانند زمان‌بند (1) در نسخه‌های قدیمی‌تر هسته لینوکس استفاده شده است.

## نتیجه‌گیری:

باعث بهبود، I/O و صفات اضافی برای فرآیندهای Active و Expired، با استفاده از صفاتی کارایی Round Robin و پاسخ‌دهی در زمان‌بندی فرآیندها نسبت به معمولی می‌شود.

# Multilevel Feedback Queue Scheduling

## تعريف

در صفحه‌چندسطحی با بازخورد (Multilevel Feedback Queue)، فرآیندها می‌توانند بین صفاتی حرکت کنند. این ویژگی باعث می‌شود سیستم انعطاف‌پذیرتر و پاسخ‌گویتر به نیازهای فرآیندها باشد.

## پارامترهای تعیین‌کننده‌ی این الگوریتم:

- تعداد صفات (Queues)
- الگوریتم زمان‌بندی هر صفت (مثلاً RR یا FCFS)
- روش ارتقاء فرآیند از یک صفت به صفت بالاتر
- روش تنزل فرآیند به صفت پایین‌تر
- قانون تعیین صفت ورود فرآیند جدید

## مزایا:

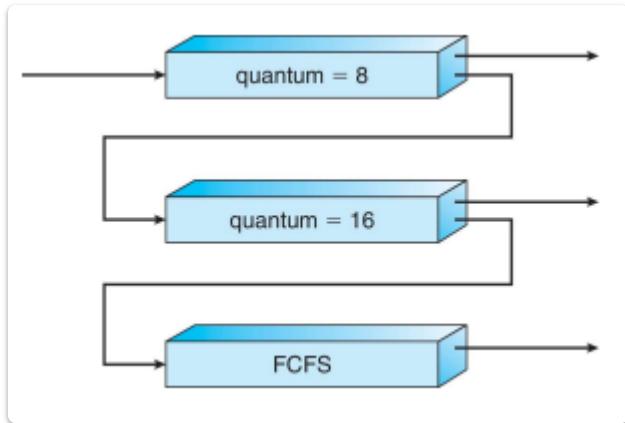
- را می‌توان با این الگوریتم پیاده‌سازی کرد؛ بنابراین از گرسنگی فرآیندهای کم‌اولویت جلوگیری می‌شود (پیر شدن)
- ترکیب انعطاف‌پذیری الگوریتم RR با بهره‌وری FCFS

# Multilevel Feedback Queue Scheduling

مثال از ساختار صفها:



صف	الگوریتم	زمان کوانتوم
Q0	Round Robin (RR)	8 ms
Q1	Round Robin (RR)	16 ms
Q2	FCFS	-



## نحوه اجرای فرایندها:

- فرایند جدید در ابتدا وارد صف Q0 می‌شود.
- در Q0، با الگوریتم RR، تا 8 میلیثانیه اجرا می‌شود.
- اگر کارش تمام نشد، به صف Q1 منتقل می‌شود.
- در Q1، با RR و تا 16 میلیثانیه اجرا می‌شود.
- اگر باز هم تمام نشد، به صف Q2 می‌رود و با الگوریتم FCFS تا پایان اجرا خواهد شد.

## تحلیل رفتاری

- فرایندهای کوتاه، معمولاً در صفحهای بالا اجرا می‌شوند و سریع پاسخ می‌گیرند.
- فرایندهای طولانی به تدریج به صفحهای پایین‌تر منتقل می‌شوند تا CPU برای فرایندهای کوتاه آزاد شود.
- اگر فرایند برای مدتی در حال اجرا نباشد، ممکن است با سازوکار "aging" به صف بالاتر بازگردد.



## Feedback Scheduling

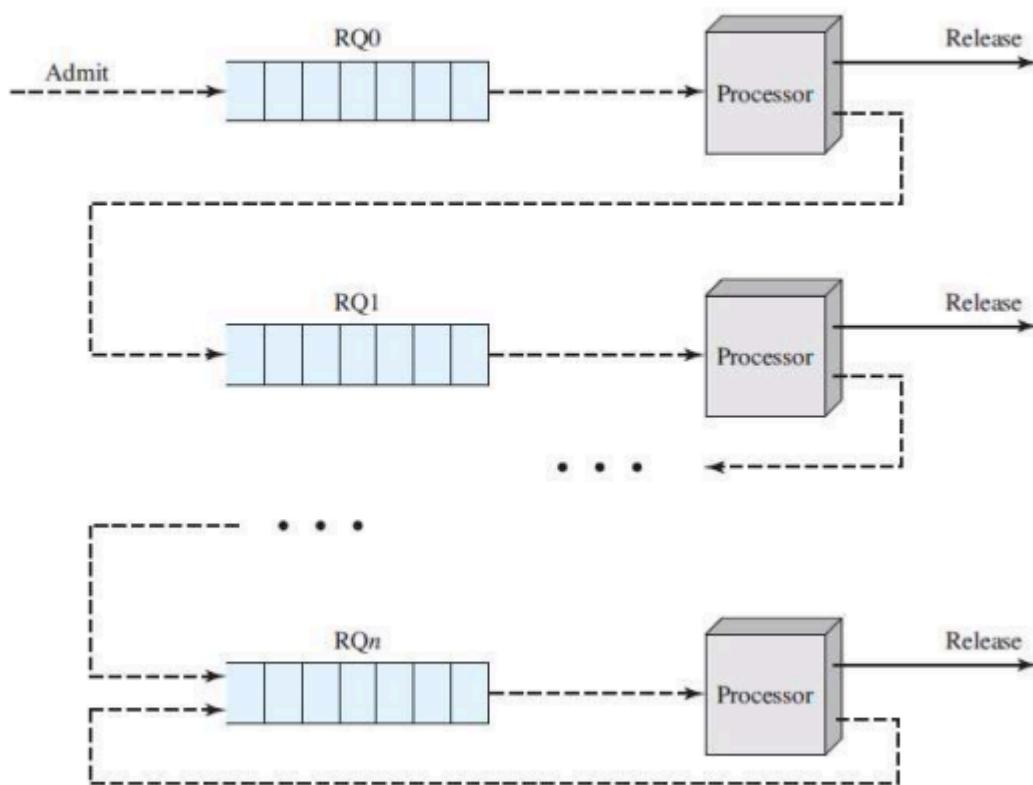
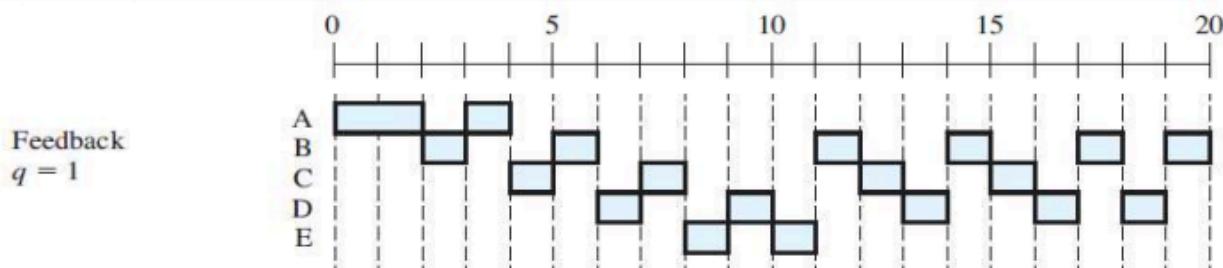


Figure 9.10 Feedback Scheduling



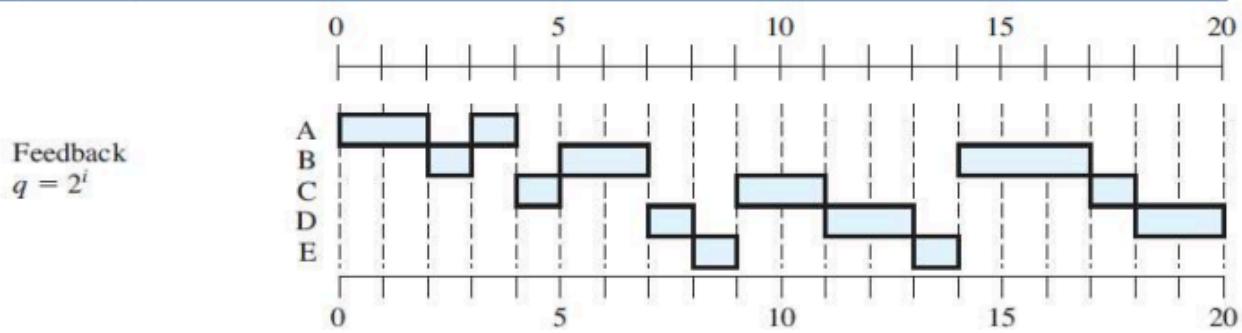
## Feedback Scheduling



FB $q = 1$						
Finish Time	4	20	16	19	11	
Turnaround Time ( $T_r$ )	4	18	12	13	3	10.00
$T_r/T_s$	1.33	3.00	3.00	2.60	1.5	2.29



## Feedback Scheduling



FB $q = 2^i$						
Finish Time	4	17	18	20	14	
Turnaround Time ( $T_r$ )	4	15	14	14	6	10.60
$T_r/T_s$	1.33	2.50	3.50	2.80	3.00	2.63

## Priority Scheduling

### تعريف

در الگوریتم Priority Scheduling، به هر فرایند یک عدد اولویت (Priority Number) نسبت داده می‌شود.

- عدد کوچکتر به معنای اولویت بالاتر است.
- به فرایندی اختصاص می‌یابد که بیشترین اولویت (کمترین عدد) را دارد CPU.

### أنواع الگوریتم:

- از فرایند جاری گرفته شده و به آن CPU، اگر فرایندی با اولویت بالاتر وارد صف آماده شود: (پیش‌دستانه) اختصاص داده می‌شود.
- فرایند جاری حتی در صورت رسیدن فرایند با اولویت بالاتر، تا پایان اجرا خواهد: (غیر پیش‌دستانه) شد.

### SJF ارتباط با

الگوریتم Shortest-Job-First (SJF) یک نوع خاص از Priority Scheduling است که در آن:

- اولویت بر اساس مدت زمان CPU بعدی مورد نیاز فرایند تعیین می‌شود.
- هر چه زمان پیش‌بینی شده کمتر، اولویت بیشتر.

### Starvation مشکل:

فرایندهایی با اولویت پایین ممکن است به دلیل ورود پی‌درپی فرایندهای با اولویت بالا، هرگز به CPU نرسند.

### Aging: راه حل

با گذشت زمان، اولویت فرایندهای در حال انتظار افزایش می‌یابد (عدد اولویت کاهش می‌یابد) تا امکان اجرای آن‌ها فراهم شود.

## مثال عددی

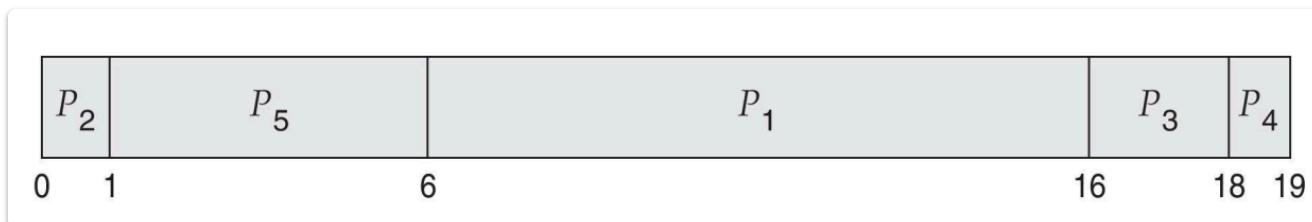
داده‌ها:

اولویت	زمان اجرای CPU (Burst Time)	فرایند
3	10	P1
1	1	P2
4	2	P3
5	1	P4
2	5	P5

ترتیب اجرا بر اساس اولویت:

P2 → P5 → P1 → P3 → P4

### Gantt Chart



### محاسبه زمان انتظار

فرایند	زمان شروع اجرا	زمان انتظار
P2	0	0
P5	1	1
P1	6	6
P3	16	14
P4	18	17

میانگین زمان انتظار:

$$\text{Average Waiting Time} = \frac{0 + 1 + 6 + 14 + 17}{5} = \frac{38}{5} = 7.6$$

نکته: در اسلاید اصلی میانگین زمان انتظار 8.2 ذکر شده، اما با ترتیب فوق و محاسبات صحیح، مقدار 7.6 به دست می‌آید.

# Priority Scheduling + Round-Robin

## تعريف

در این نوع الگوریتم ترکیبی:

- فرایندها بر اساس اولویت اجرا می‌شوند.
- اگر چند فرایند دارای اولویت یکسان باشند، بین آنها از الگوریتم Round-Robin استفاده می‌شود.
- اجرای هر فرایند در Round-Robin به اندازه‌ی یک time quantum (مثلاً ۲ میلیثانیه) انجام می‌شود.

## داده‌های مسئله

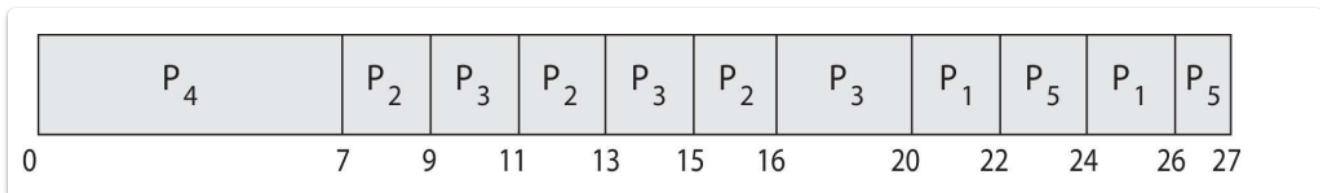
Priority	Burst Time	فرایند
3	4	P1
2	5	P2
2	8	P3
1	7	P4
3	3	P5

- Time Quantum = 2

• ترتیب اجرا بر اساس اولویت (کمترین عدد = بالاترین اولویت):

1. P4 (اولویت 1)
2. P2, P3 اولویت 2 → با (Round-Robin)
3. P1, P5 اولویت 3 → با (Round-Robin)

## Gantt Chart



## نکات

- بهنهایی دارای بالاترین اولویت است؛ پس از ابتدا اجرا می‌شود.
- سپس P2 و P3 به صورت Round-Robin اجرا می‌شوند (هر بار 2 واحد).
- بعد از اتمام اولویت 2، نوبت به P1 و P5 با اولویت 3 می‌رسد.

# Multilevel Queue Scheduling

## تعريف

در الگوریتم صفحه‌چندسطحی (Multilevel Queue Scheduling) :

- صف آماده (Ready Queue) به چند زیرصف (Queue) مجزا تقسیم می‌شود.
- هر صف می‌تواند الگوریتم زمان‌بندی متفاوتی داشته باشد.
- بین صفحه‌ها معمولاً از Priority Scheduling استفاده می‌شود (مثلاً صف اول از صف دوم با اولویت بالاتری برخوردار است).

## پارامترهای اصلی الگوریتم:

- تعداد صفحه‌ها (مثلاً 3 صف: سیستم تعاملی، دسته‌ای، پیش‌زمینه)
- الگوریتم زمان‌بندی داخل هر صف
- مثلاً RR (Round-Robin) برای صف تعاملی
- برای صف دسته‌ای
- روش تخصیص فرایندها به صف مناسب
  - بر اساس نوع برنامه یا ویژگی‌های مشخص (مثلاً فرایند I/O-bound به صف تعاملی برود)
- روش زمان‌بندی بین صفحه‌ها
  - معمولأً صف با اولویت بالاتر اول سرویس می‌گیرد

## مثال مفهومی

فرض کنید 3 صف داریم:

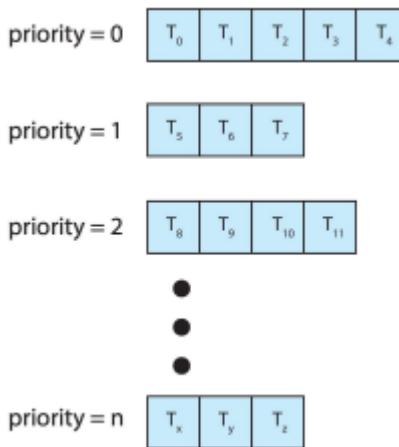
صف	نوع برنامه‌ها	الگوریتم داخلی	اولویت
Q0	سیستم تعاملی (Interactive)	با کوانتم 8 RR	بالا
Q1	دسته‌ای (Batch)	FCFS	متوسط
Q2	پیش‌زمینه (Background)	FCFS	پایین

در این حالت:

- اول تمامی فرایندهای Q0 اجرا می‌شوند.
- اگر Q0 خالی بود، به سراغ Q1 می‌رویم.
- اگر Q1 هم خالی بود، Q2 اجرا می‌شود.

## تفاوت با Multilevel Feedback Queue

در Multilevel Queue، فرایند پس از ورود به یک صف تا پایان اجرای خود در همان صف باقی می‌ماند. اما در Multilevel Feedback Queue، فرایند می‌تواند بین صفحه‌ها جابجا شود (ارتقا یا تنزل).



## (اولویت بر اساس نوع فرایند) (Multilevel Queue Scheduling)

### تعريف کلی

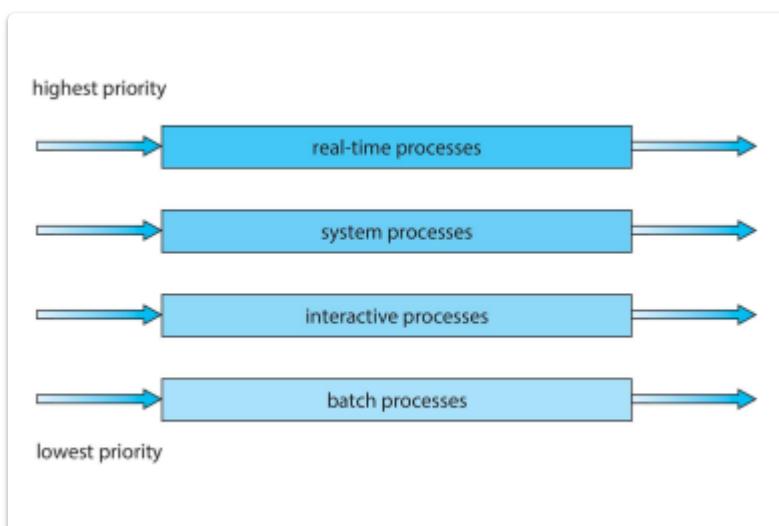
در مدل **Multilevel Queue Scheduling**، صف آماده (Ready Queue) به چند صف مجزا تقسیم می‌شود. فرایندها بر اساس نوعشان (process type) در یکی از این صفها قرار می‌گیرند و دیگر از آن صف خارج نمی‌شوند.

### معیار اولویت: نوع فرایند (Process Type)

فرایندها به دسته‌های مختلفی تقسیم می‌شوند، مانند:

- فرایندهای تعاملی (Interactive)
- فرایندهای دسته‌ای (Batch)
- فرایندهای سیستمی (System)
- فرایندهای زمان واقعی (Real-Time)

سپس به هر دسته یک صف جداگانه اختصاص می‌یابد. این صفها دارای **اولویت‌های مختلف** هستند.



### سیاست‌های رایج:

- فرایندهای تعاملی معمولاً در صف با اولویت بالا قرار می‌گیرند.
- فرایندهای Batch یا پس زمینه در صفحه‌ای با اولویت پایین‌تر.
- صفحه‌ها ممکن است از الگوریتم‌های مختلفی مثل Priority، FCFS یا RR استفاده کنند.

## ✳️ ویژگی‌ها

- اولویت‌دهی سختگیرانه: صفحه با اولویت بالا باید خالی شود تا صفحه بعدی بررسی شود.
- فرایندها به صفت ثابت تعلق دارند و جابجا نمی‌شوند.
- مناسب برای سیستم‌هایی با تفکیک مشخص بین نوع فرایندها.

# Characteristics of Various Scheduling Policies

در این بخش ویژگی‌های مهم چند الگوریتم زمان‌بندی مختلف با هم مقایسه شده‌اند.

Feedback	HRRN	SRT	SPN	Round Robin	FCFS	ویژگی
(رجوع به متن اصلی)	$\max((w + s)/s)$	$\min(s - e)$	$\min(s)$	ثابت (constant)	$\max(w)$	تابع انتخاب Selection ) (function
پیش‌دستی در زمان کوانتم	غیرقابل پیش‌دستی	پیش‌دستی هنگام ورود Preemptive ) (at arrival	غیرقابل پیش‌دستی	پیش‌دستی در زمان کوانتم Preemptive ) at time (quantum	غیرقابل پیش‌دستی Non-) (preemptive	حال تصمیم‌گیری Decision ) (mode
برجسته نیست	بالا	بالا	بالا (High)	ممکن است پایین باشد اگر کوانتم کوچک باشد	برجسته نیست Not ) (emphasized	توان عملیاتی (Throughput)
برجسته نیست	زمان پاسخ خوب	زمان پاسخ خوب	زمان پاسخ خوب برای فرایندهای کوتاه	زمان پاسخ خوب برای فرایندهای کوتاه	ممکن است زیاد باشد، خصوصاً در صورت تنوع زیاد در زمان اجرای فرایندها	زمان پاسخ‌دهی Response ) (time
ممکن است زیاد باشد	ممکن است زیاد باشد	ممکن است زیاد باشد	ممکن است زیاد باشد	حداقل	حداقل (Minimum)	بار اضافی Overhead)
ممکن است به نفع فرایندهای I/O محور باشد	تعادل خوب Good ) (balance	فرایندهای طولانی را جریمه می‌کند	فرایندهای طولانی را جریمه می‌کند	رفتار منصفانه Fair ) (treatment	فرایندهای کوتاه و فرایندهای I/O محور را جریمه می‌کند	تأثیر بر فرایندها Effect on ) (processes
ممکن است	ممکن است	ممکن است	ممکن است (Possible)	خیر	خیر (No)	گرسنگی Starvation)

## توضیحات تکمیلی درباره ویژگی‌های مهم:

- **تابع انتخاب (Selection function)**:
  - مشخص می‌کند کدام فرایند در اولویت اجرا قرار گیرد.
  - مثلاً FCFS براساس بیشترین زمان انتظار ( $\max(w)$ ) انتخاب می‌کند.
- **حالت تصمیم‌گیری (Decision mode)**:
  - مشخص می‌کند آیا سیستم می‌تواند اجرای فرایندی را در وسط قطع کند (preemptive) یا نه (non-preemptive).
- **توان عملیاتی (Throughput)**:
  - تعداد فرایندهای تکمیل شده در واحد زمان.
- **زمان پاسخ‌دهی (Response time)**:
  - فاصله زمانی از زمان ورود فرایند تا اولین پاسخ سیستم.
- **بار اضافی (Overhead)**:
  - منابع اضافی مورد نیاز برای مدیریت زمان‌بندی.
- **تأثیر بر فرایندها (Effect on processes)**:
  - نحوه تأثیر الگوریتم بر فرایندهای کوتاه، بلند، و I/O محور.
- **گرسنگی (Starvation)**:
  - حالتی که یک فرایند هیچ‌گاه به CPU دست پیدا نمی‌کند.

## نکته مهم:

انتخاب الگوریتم مناسب به ماهیت بار کاری (Workload) و اهداف سیستم بستگی دارد. هیچ الگوریتمی برای همه شرایط بهترین نیست!

## Algorithm Evaluation and Deterministic Evaluation

### انتخاب الگوریتم زمان‌بندی CPU در سیستم عامل

- یکی از وظایف اصلی سیستم عامل، انتخاب الگوریتم مناسب برای زمان‌بندی CPU است.
- برای این انتخاب:
  - باید ابتدا **معیارهای ارزیابی** مشخص شوند (مانند زمان انتظار، زمان پاسخ‌دهی، بهره‌وری و ...).
  - سپس الگوریتم‌ها بر اساس این معیارها مورد بررسی قرار گیرند.

### (Deterministic Modeling) مدل‌سازی قطعی

- **مدل قطعی** یکی از روش‌های **ارزیابی تحلیلی** (Analytic Evaluation) است.
  - در این روش:
    - یک حجم کاری ثابت (مجموعه‌ای از فرایندها با زمان‌های اجرای مشخص) در نظر گرفته می‌شود.
    - عملکرد الگوریتم‌ها روی این ورودی تحلیل می‌شود.
  - این روش:
    - سریع و ساده است،
    - اما نتایج آن فقط برای همان مجموعه داده معتبر خواهد بود.

### مثال عملی: ۵ فرایند ورودی

فرایندها در زمان ۰ وارد سیستم شده‌اند و زمان اجرای (Burst Time) آنها به صورت زیر است:

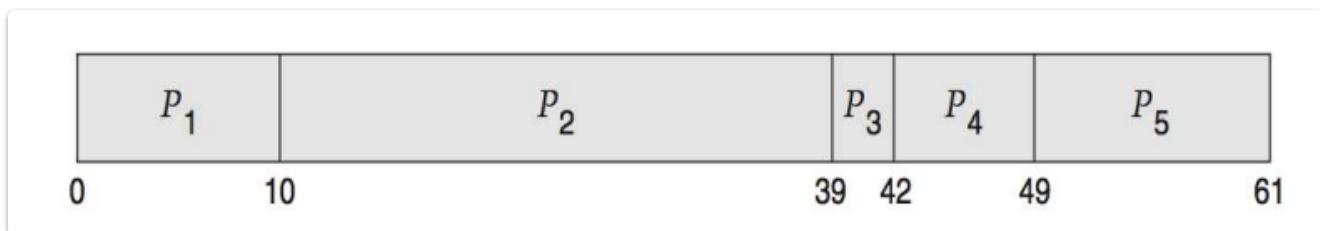
فرآیند	زمان اجرا (Burst Time)
$P_1$	10
$P_2$	29
$P_3$	3
$P_4$	7
$P_5$	12

## ارزیابی الگوریتم‌های مختلف روی ورودی مشخص شده

### 1. الگوریتم First-Come, First-Served (FCFS)

- میانگین زمان انتظار: 28 میلیثانیه

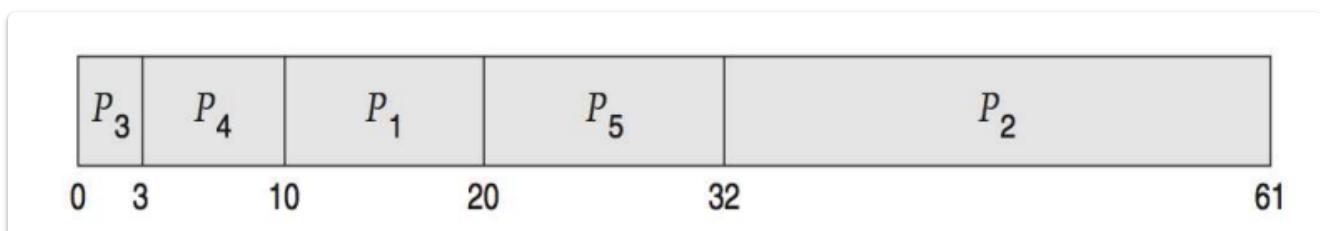
نمودار گانت :FCFS



### 2. الگوریتم Shortest Job First (SJF) – غیرقابل پیش‌دستی (Non-preemptive)

- میانگین زمان انتظار: 13 میلیثانیه

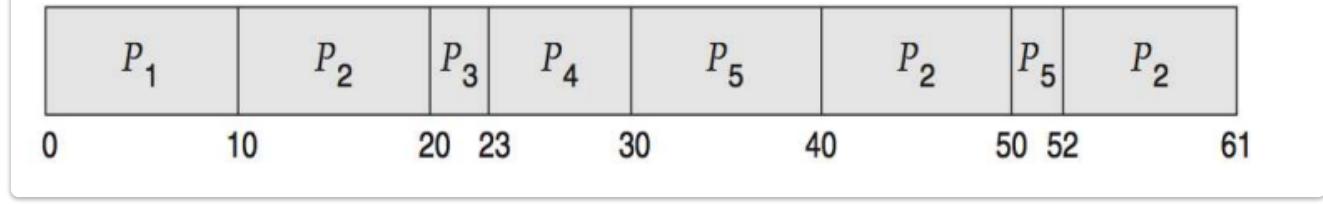
نمودار گانت :SJF



### 3. الگوریتم Round Robin (RR)

- میانگین زمان انتظار: 23 میلیثانیه.
- فرض: کوانتوم زمانی (Time Quantum) برابر 10 میلیثانیه.

نمودار گانت :RR



## نکات مهم:

- در روش ارزیابی قطعی، مقایسه بین الگوریتم‌ها تنها برای ورودی خاص معتبر است.
- در عمل، برای داشتن تحلیل عمومی‌تر، باید از روش‌های ارزیابی آماری یا شبیه‌سازی استفاده شود.

## مدل‌های صف (Queueing Models)



### تعريف کلی

مدل‌های صف برای تحلیل زمان‌بندی پردازه‌ها و ارزیابی کارایی سیستم استفاده می‌شوند.

## ویژگی‌های مدل صف

### توصیف آماری:

- ورودی پردازه‌ها و مدت زمان CPU و I/O به صورت **احتمالات** (Probabilistic) مدل‌سازی می‌شود.
- معمولًاً توزیع‌ها **نمایی** (Exponential) هستند.
- توصیف بر اساس **میانگین** (Mean) صورت می‌گیرد.

### محاسبات قابل انجام:

- Throughput:** نرخ انجام پردازه‌ها در واحد زمان
- Utilization:** یا منابع دیگر CPU درصد استفاده از میانگین زمان انتظار در صف
- Average Waiting Time:** میانگین طول صف
- Average Queue Length:** میانگین طول صف

## سیستم به عنوان شبکه‌ای از سرورها

سیستم کامپیوترا به صورت مجموعه‌ای از سرورها مدل می‌شود که:

- هر سرور دارای صفحه از پردازه‌های منتظر است.
- با دانستن نرخ ورود (λ) و نرخ سرویس‌دهی (μ) می‌توان پارامترهای عملکردی را محاسبه کرد.

## فرمول لیتل (Little's Formula)



### تعريف پارامترها:

- $n$  = میانگین تعداد پردازه‌ها در صف
- $W$  = میانگین زمان انتظار در صف
- $\lambda$  = نرخ ورود پردازه‌ها به صف

رابطه اصلی: 

$$n = \lambda \times W$$

در حالت پایدار (Steady-State)، نرخ ورود پردازه‌ها برابر با نرخ خروج است.

کاربرد: 

این رابطه برای **هر نوع الگوریتم زمان‌بندی** و **هر نوع توزیع ورودی** معتبر است.

مثال: 

اگر در هر ثانیه به طور میانگین ۷ پردازه وارد صف شوند، و در صف به طور میانگین ۱۴ پردازه وجود داشته باشد:

$$W = n / \lambda = 14 / 7 = 2$$

یعنی هر پردازه به طور میانگین ۲ ثانیه در صف منتظر می‌ماند.

## مزیت اصلی:

مدل‌های صف و فرمول لیتل ابزارهای ریاضی قدرتمندی برای:

- تحلیل دقیق کارایی سیستم
  - پیش‌بینی بار سیستم
  - بهینه‌سازی منابع
- هستند.

## Simulations in CPU Scheduling Evaluation

چرا به شبیه‌سازی نیاز داریم؟

- محدودیت مدل‌های صف (Queueing Models):
- مدل‌های تئوریک صفت‌بندی برای شرایط واقعی کافی نیستند.
- فرضیات ساده‌کننده‌ای دارند که همیشه معتبر نیستند.
- مزایای شبیه‌سازی:

- دقیقت از مدل‌های تحلیلی ساده است.
- با شبیه‌سازی، می‌توان عملکرد الگوریتم‌ها را تحت شرایط متنوع بررسی کرد.

### ویژگی‌های شبیه‌سازی:

- مدل برنامه‌نویسی شده از سیستم کامپیوتری ساخته می‌شود.
- ساعت (Clock) در شبیه‌سازی به عنوان یک متغیر عمل می‌کند.
- داده‌های عملکردی جمع‌آوری شده و برای تحلیل الگوریتم‌ها استفاده می‌شوند.

### روش‌های تهیه داده برای شبیه‌سازی:

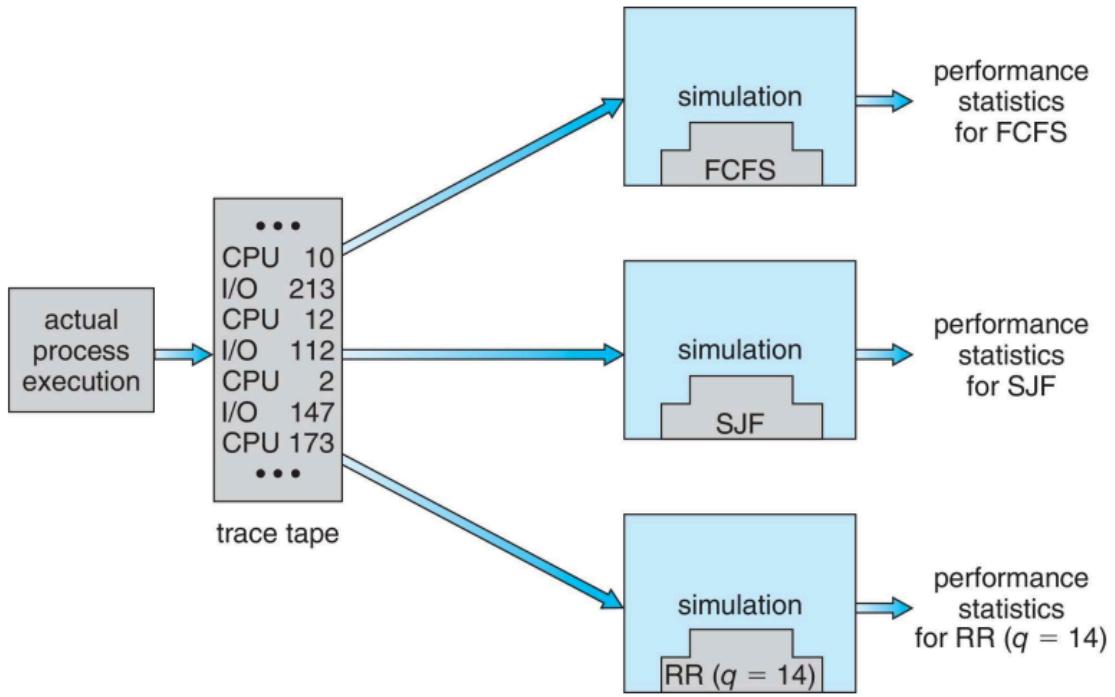
- استفاده از **تولیدکننده اعداد تصادفی** بر اساس احتمالات.
- **تعریف توزیع‌ها** به صورت ریاضی یا تجربی.
- استفاده از نوارهای ردیابی (Trace tapes):
- ثبت دنباله‌ای از رویدادهای واقعی در سیستم‌های عملیاتی.

## Evaluation of CPU Schedulers by Simulation

### نحوه ارزیابی:

1. **تهیه داده‌های واقعی:**
  - اطلاعات واقعی از اجرای فرایندها (مانند زمان اجرای CPU، درخواست I/O و ...) ثبت می‌شود.
  - داده‌ها در قالب یک **Trace Tape** ذخیره می‌شوند.
2. **اجرای شبیه‌سازی:**
  - داده‌های Trace Tape به عنوان ورودی به شبیه‌سازهای مختلف داده می‌شوند.
  - برای هر الگوریتم (مثل FCFS، SJF، RR و ...) یک شبیه‌ساز مجزا ساخته می‌شود.
3. **جمع‌آوری آمار عملکردی:**
  - آمار عملکرد مانند زمان پاسخ، بهره‌وری، میزان گرسنگی و ... برای هر الگوریتم محاسبه می‌شود.

### دیاگرام فرایند شبیه‌سازی:



## نکته مهم:

شبیه‌سازی یک روش بسیار قدرتمند برای ارزیابی الگوریتم‌ها در شرایط واقعی است اما به زمان و منابع محاسباتی بیشتری نسبت به تحلیل‌های ساده نیاز دارد.

## Implementation of CPU Schedulers

### محدودیت‌های شبیه‌سازی:

- حتی شبیه‌سازی‌ها هم دقت محدودی دارند.
- همیشه نمی‌توان شرایط واقعی را به طور کامل در شبیه‌سازی بازآفرینی کرد.

### راهکار نهایی: پیاده‌سازی واقعی

- پیاده‌سازی مستقیم** الگوریتم زمان‌بندی در سیستم واقعی و تست آن.
- این روش دقیقترين اطلاعات درباره عملکرد الگوریتم می‌دهد اما:
  - هزینه‌ی بالایی دارد (High cost).
  - ریسک بالایی دارد (High risk).
- محیط‌های اجرایی (environments) از سیستمی به سیستم دیگر متفاوت هستند.

### چالش‌های پیاده‌سازی:

- تفاوت محیط‌های کاری باعث می‌شود نتایج در یک سیستم قابل تعمیم به دیگر سیستم‌ها نباشد.
- نیاز به تطبیق الگوریتم با شرایط و نیازمندی‌های هر سایت یا سیستم.

### راهکارهای افزایش انعطاف‌پذیری:

- طراحی Scheduler هایی که بتوانند برای هر سایت یا سیستم سفارشی‌سازی شوند.
- ارائه‌ی API هایی برای تغییر اولویت‌ها یا تنظیم پارامترهای زمان‌بندی در زمان اجرا.

با این حال، به دلیل تغییرپذیری محیط‌ها، نتایج همیشه قابل پیش‌بینی نیستند.

## خلاصه:

پیاده‌سازی واقعی دقیق‌ترین روش ارزیابی زمان‌بندی CPU است، ولی به دلیل هزینه، ریسک و تغییرپذیری محیطی، باید با دقت و در صورت نیاز به انعطاف بالا انجام شود.

**End of Chapter 5 :)**