

Operation Systems

Operating System Concepts

TENTH EDITION

ABRAHAM SILBERSCHATZ • PETER BAER GALVIN • GREG GAGNE



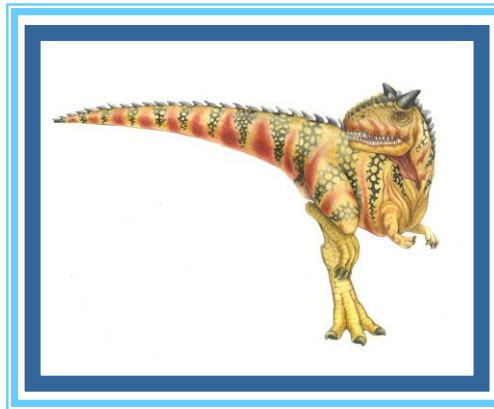
WILEY



Dr. A. Taghinezhad

Website: ataghinezhad.github.io, Email: a0taghinezhad@gmail.com

فصل ۵: زمان بندی پردازنده





Outline

- معیارهای زمان بندی (**Scheduling Criteria**): معیارهایی برای ارزیابی الگوریتم‌های زمان بندی CPU
- الگوریتم‌های زمان بندی (**Scheduling Algorithms**): روش‌هایی که سیستم عامل برای تخصیص CPU به فرایندها استفاده می‌کند
- زمان بندی رشته (**Thread Scheduling**): تخصیص زمان پردازنده به رشته‌های درون یک فرایند
- زمان بندی چند پردازنده (**Multi-Processor Scheduling**): تخصیص منابع پردازشی در سیستم‌های چند پردازنده
- زمان بندی زمان حقیقی (**Real-Time CPU Scheduling**): الگوریتم‌های خاص برای سیستم‌های با زمان پاسخگویی حیاتی
- مثال‌های سیستم عامل (**Operating Systems Examples**)
- ارزیابی الگوریتم (**Algorithm Evaluation**): روش‌های سنجش کارایی الگوریتم‌های زمان بندی

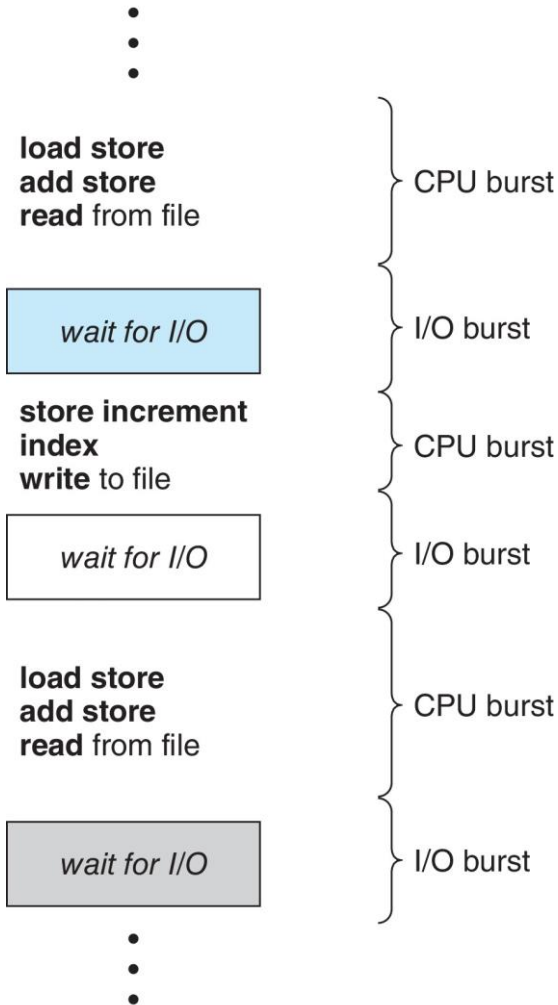


اهداف

- **الگوریتم‌های مختلف زمان بندی CPU:** انواع الگوریتم‌های زمان بندی CPU که برای مدیریت صف (queue) فرایندهای آماده‌ی اجرا به کار می‌روند، توضیح داده خواهد شد.
- **ارزیابی الگوریتم‌های زمان بندی CPU بر اساس معیارهای زمان بندی:** روش‌های ارزیابی کارایی الگوریتم‌های زمان بندی CPU بر اساس معیارهایی مانند زمان پاسخ‌دهی، زمان turnaround، زمان انتظار و توان عملیاتی (throughput) شرح داده می‌شود.
- **مسائل مربوط به زمان بندی در سیستم‌های چندپردازنده و چند هسته‌ای:** چالش‌ها و راه‌حل‌های مربوط به زمان بندی در سیستم‌های با چند پردازنده (multiprocessor) و چند هسته (multicore) مورد بحث قرار خواهد گرفت.
- **الگوریتم‌های مختلف زمان بندی زمان حقیقی:** انواع الگوریتم‌های زمان بندی که برای سیستم‌های زمان حقیقی (real-time) که نیازمند پاسخ‌دهی تضمین شده هستند، معرفی خواهند شد.



مفاهیم اولیه



حداکثر استفاده ممکن از CPU با چندبرنامگی (Multiprogramming) به معنی حداکثر استفاده از CPU

با استفاده از تکنیک چندبرنامگی (اجرای همزمان چند برنامه در حافظه) و تأثیر آن بر کارایی سیستم توضیح داده خواهد شد.

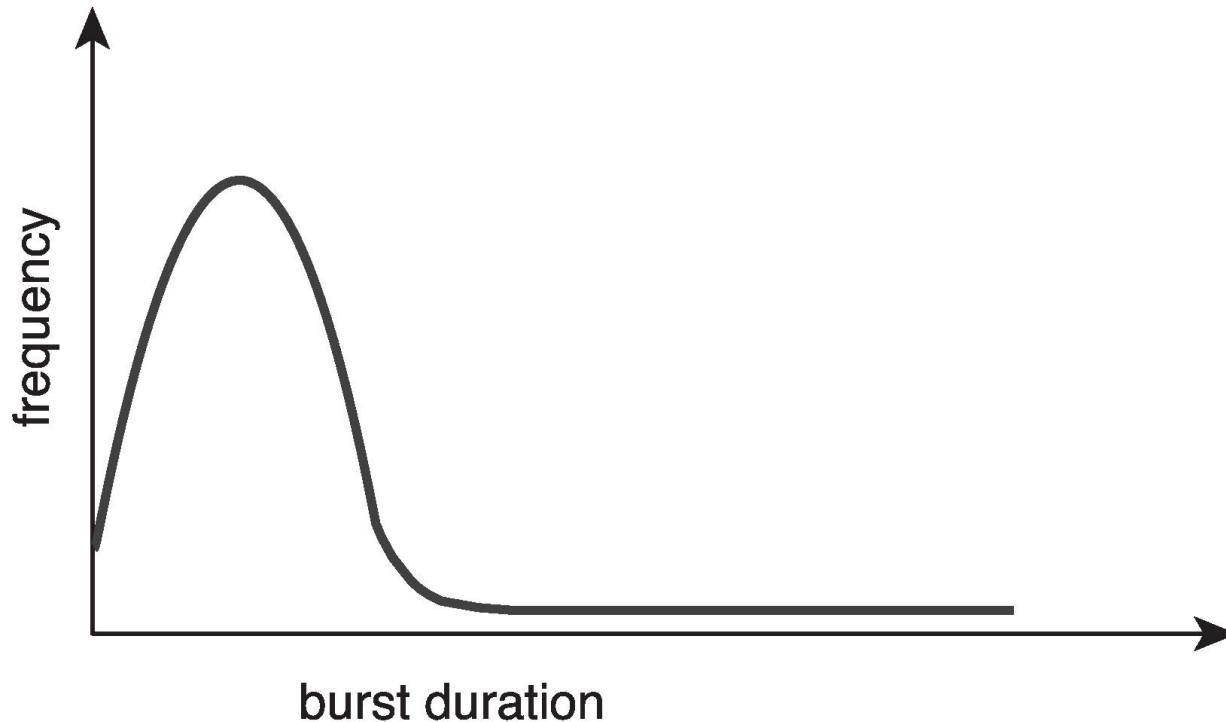
چرخه‌ی وقفه: CPU-I/O مفهوم چرخه‌ی وقفه CPU-I/O که ماهیت اجرای فرایندها را به صورت تناوب بین محاسبات CPU و انتظار برای I/O نشان می‌دهد، تشریح خواهد شد.

دنباله‌ی وقفه‌های CPU و I/O اهمیت توزیع دنباله‌ی وقفه‌های پردازنده و ورودی/خروجی مدت زمان صرف شده برای محاسبات و انتظار برای ورودی/خروجی در انتخاب الگوریتم زمان‌بندی مناسب مورد بحث قرار خواهد گرفت.



Histogram of CPU-burst Times

- توزیع بازه‌های اجرای CPU اهمیت ویژه‌ای دارد:
 - تعداد زیاد بازه‌های کوتاه
 - تعداد کم بازه‌های بلند





زمان بندی پردازنده

- زمان بند پردازنده از میان فرایندهای موجود در صف آماده، یک فرایند را برای اجرا بر روی یک هسته پردازشی انتخاب می کند.
- صف آماده می تواند به روش های مختلف اولویت گذاری شود.



زمان بندی پردازنده

تصمیم گیری زمان بندی (CPU scheduling decisions)

تصمیم گیری برای زمان بندی در زمان های زیر رخ می دهد:

1. تغییر وضعیت یک فرایند از در حالت اجرا به انتظار (وقتی فرایند درخواست I/O

می دهد)

2. تغییر وضعیت یک فرایند از حالت اجرا به آماده (وقتی فرایند به خاتمه رسیده ولی

منتظر خروجی باشد)

3. تغییر وضعیت یک فرایند از انتظار به آماده (وقتی عملیات I/O به اتمام رسیده

باشد)

4. اتمام یک فرایند (خاتمه ی طبیعی یا خطا)

در موارد ۱ و ۴، انتخابی برای زمان بندی وجود ندارد. در صورت وجود فرایند در صف آماده، باید یک فرایند جدید برای اجرا انتخاب شود.

اما در موارد ۲ و ۳، امکان انتخاب وجود دارد.



■ زمان بندی انحصاری (Nonpreemptive Scheduling)

- زمانی که فرایند کنترل پردازنده را به دست می آورد، تا زمانی که آن را رها نکند (خاتمه یا انتظار) به اجرای خود ادامه می دهد.
- تقریباً تمامی سیستم های عامل مدرن از زمان بندی غیر انحصاری (Preemptive Scheduling) استفاده می کنند.



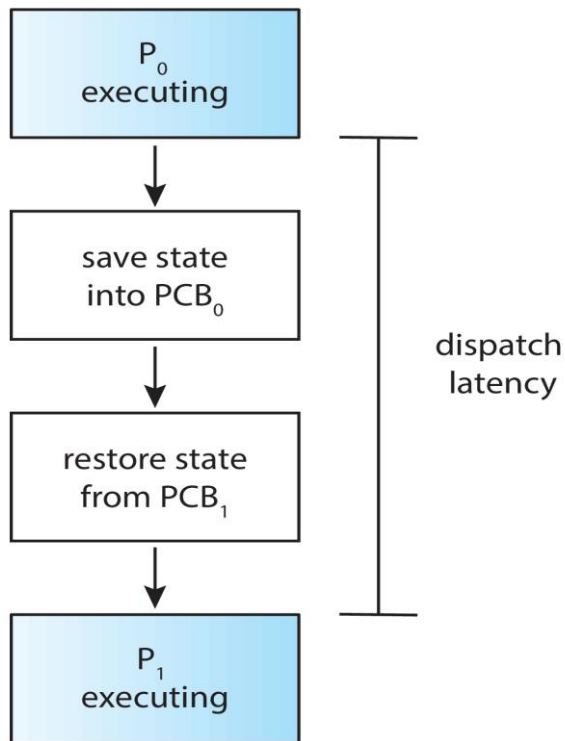
Preemptive Scheduling and Race Conditions

■ زمان بندی غیرانحصاری

■ می تواند منجر به شرایط مسابقه شود، زمانی که چندین فرایند به داده های مشترک دسترسی داشته باشند.

■ فرض کنید دو فرایند داریم که داده ای را به اشتراک گذاشته اند. در حالی که یک فرایند در حال به روزرسانی داده ها است، سیستم عامل کنترل پردازنده را از آن گرفته و به فرایند دیگری می دهد. در نتیجه، فرایند دوم تلاش می کند داده ها را بخواند، در حالی که داده ها در یک وضعیت نامعتبر قرار دارند.

توزیع کننده Dispatcher



■ ماژول توزیع کننده (Dispatcher module)

■ ماژول توزیع کننده، کنترل پردازنده را به فرایندی که توسط زمان بند پردازنده انتخاب شده است، واگذار می کند. این فرآیند شامل موارد زیر است:

- تغییر متن (Switching context)
- تغییر به حالت کاربر (Switching to user mode)

• پرش به مکان مناسب در برنامه کاربر برای راه اندازی مجدد آن برنامه

■ تاخیر توزیع **dispatch latency**: زمانی که برای توزیع کننده طول میکشد تا یک فرایند را متوقف کرده و دیگری را شروع کند.



Scheduling Criteria

- بهره‌وری پردازنده (**CPU utilization**) مشغول نگه داشتن پردازنده تا حد ممکن
- توان عملیاتی: (**Throughput**) تعداد فرایندهایی که اجرای خود را در واحد زمان به پایان می‌رسانند.
- زمان گردش: (**Turnaround time**) کل زمانی که برای اجرای یک فرایند خاص صرف می‌شود.
- زمان انتظار: (**Waiting time**) مدتی که یک فرایند در صف آماده منتظر مانده است.
- زمان پاسخ: (**Response time**) مدتی که از زمان ارسال یک درخواست تا تولید اولین پاسخ طول می‌کشد.

معیارهای بهینه‌سازی الگوریتم زمان بندی



- ❖ حداکثر بهره‌وری پردازنده
- ❖ حداکثر توان عملیاتی
- ❖ حداقل زمان گردش
- ❖ حداقل زمان انتظار
- ❖ حداقل زمان پاسخ



الگوریتم زمان بندی FCFS

- ❖ اولین ورود و اولین خدمت رسانی
- ❖ اولویت بر فرآیندهایی هست که زودتر به صف وارد شده‌اند.
- ❖ از نوع زمان بندی انحصاری



First- Come, First-Served (FCFS) Scheduling

| فرآیندها | زمان سوختن یا اجرا |
|----------|--------------------|
| P_1 | 24 |
| P_2 | 3 |
| P_3 | 3 |

فرض کنید فرایندها به ترتیب زیر وارد سیستم می شوند:

■ P_1 , P_2 , P_3

■ نمودار گانت برای زمان بندی به صورت زیر است:



■ زمان انتظار $P_1 = 0$; $P_2 = 24$; $P_3 = 27$

■ میانگین زمان انتظار: $(0 + 24 + 27)/3 = 17$

Turn Around Time= Completion Time – Arrival Time

Waiting Time= Turn Around Time – Burst Time

FCFS Scheduling (Cont.)



■ فرض کنید فرایندها به ترتیب زیر وارد سیستم می‌شوند از چپ به راست:

■ P2 , P3 , P1

■ نمودار گانت برای زمان‌بندی به صورت زیر است:



■ زمان انتظار برای P1 = 6; P2 = 0; P3 = 3

• میانگین زمان انتظار: $3 = 3 / (3 + 0 + 6)$

• این مقدار بسیار بهتر از مثال قبلی است.

■ اثر کاروان (Convoy Effect)

▶ اثر کاروان زمانی رخ می‌دهد که یک فرایند کوتاه پشت سر یک فرایند بلند قرار گیرد.



Shortest-Job-First (SJF) Scheduling

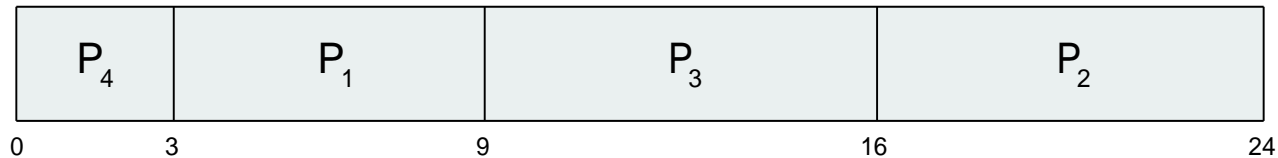
- در نظر گرفتن طول بازه فرآیند بعدی هر فرایند برای زمان بندی
- زمان بندی فرایندی با کوتاه ترین بازه فرآیند بعدی
- زمان بندی انحصاری هست.
- SJF بهینه است و برای مجموعه ای از فرایندهای خاص، حداقل میانگین زمان انتظار را به دست می دهد.
- نسخه ی غیرانحصاری از SJF با نام زمان باقیمانده کوتاه ترین در ابتدا (Shortest Remaining Time First)
- چگونه طول بازه فرایند بعدی را تعیین کنیم؟
 - پرسیدن از کاربر
 - تخمین



Example of SJF

| <u>Process</u> | <u>Burst Time</u> |
|----------------|-------------------|
| P_1 | 6 |
| P_2 | 8 |
| P_3 | 7 |
| P_4 | 3 |

- SJF scheduling chart



- Average waiting time = $(3 + 16 + 9 + 0) / 4 = 7$



تعیین طول اجرای بعدی پردازنده

- طول واقعی قابل پیش‌بینی نیست، اما باید به طول بازه قبلی نزدیک باشد.
- انتخاب فرایند با کوتاه‌ترین بازه پردازنده پیش‌بینی شده بعدی
- این کار با استفاده از طول بازه‌های پردازنده قبلی و میانگین‌گیری نمایی قابل انجام است.

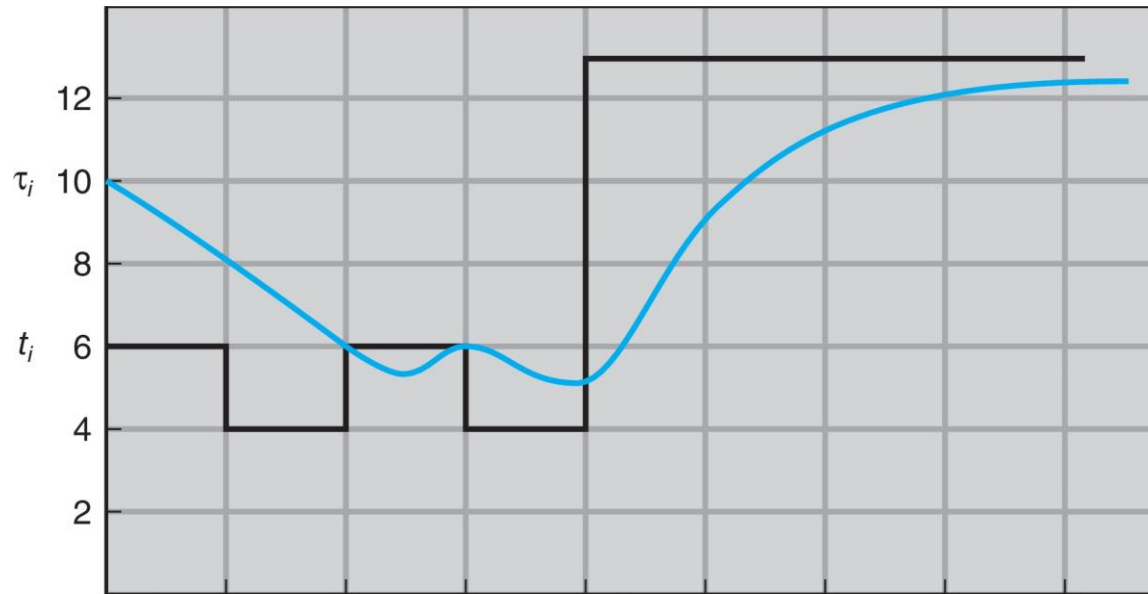
1. t_n = actual length of n^{th} CPU burst
2. τ_{n+1} = predicted value for the next CPU burst
3. $\alpha, 0 \leq \alpha \leq 1$
4. Define: $\tau_{n+1} = \alpha t_n + (1 - \alpha)\tau_n$.

$$\tau_{n+1} = \alpha t_n + (1 - \alpha)\alpha t_{n-1} + \dots + (1 - \alpha)^j \alpha t_{n-j} + \dots + (1 - \alpha)^{n+1} \tau_0$$

- مقدار α به طور معمول برابر با $1/2$ در نظر گرفته می‌شود.
- مقدار t_n شامل اطلاعات جدید و مقدار τ_n تاریخچه را ذخیره می‌کند.



Prediction of the Length of the Next CPU Burst



| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | ... |
|----------------------|----|---|---|---|---|----|----|----|-----|
| CPU burst (t_i) | | 6 | 4 | 6 | 4 | 13 | 13 | 13 | ... |
| "guess" (τ_i) | 10 | 8 | 6 | 6 | 5 | 9 | 11 | 12 | ... |

$$\tau_{n+1} = \alpha t_n + (1 - \alpha)\alpha \cdot t_{n-1} + \dots + (1 - \alpha)^j \alpha \cdot t_{n-j}$$

$$\tau_2 = \frac{1}{2}8 + (1 - 0.5)0.5 \cdot t_0 = 4 + 0.25 * 10 = 6.5$$



Shortest Remaining Time First Scheduling

- زمان بندی غیرانحصاری با **SJN (Shortest Job Next)**
- هر زمان که یک فرایند جدید وارد صف آماده می شود، تصمیم گیری در مورد اینکه کدام فرایند را زمان بندی کند، با استفاده از الگوریتم **SJN** انجام می شود.
- مقایسه **SJN** و **SRT**
- آیا **SRT** از نظر حداقل میانگین زمان انتظار برای مجموعه ای از فرایندهای خاص، «بهینه تر» از **SJN** است؟

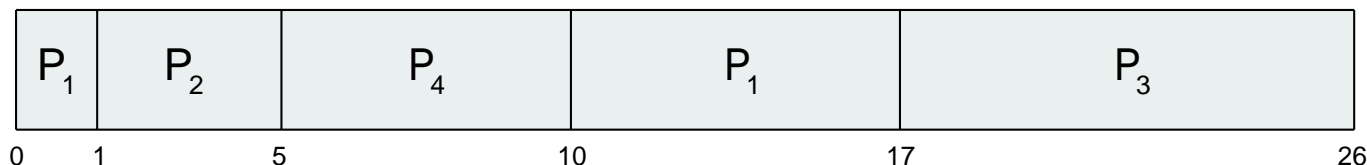


Example of Shortest-remaining-time-first

■ اضافه کردن مفاهیم تغییر زمان ورود و غیرانحصاری بودن:

| Process | <u>Arrival Time</u> | <u>Burst Time</u> |
|---------|---------------------|-------------------|
| P_1 | 0 | 8 |
| P_2 | 1 | 4 |
| P_3 | 2 | 9 |
| P_4 | 3 | 5 |

■ *Preemptive SJF Gantt Chart*



$$\text{Average waiting time} = [(10-1)+(1-1)+(17-2)+(5-3)]/4 = 26/4 = 6.5$$

$$\text{Average Response time} = [(17-0)+(5-1)+(26-2)+(10-3)]/4 = 52/4 = 13$$

زمان انتظار: زمان سرویس - زمان رسیدن
زمان پاسخ: زمان پایان اجرا - زمان ورود به سیستم



Round Robin (RR)

- به هر فرایند یک واحد کوچک زمان پردازنده (مقدار زمانی کوانتومی q) اختصاص داده می شود که معمولاً ۱۰ تا ۱۰۰ میلی ثانیه است.
 - بعد از گذشتن این زمان، فرایند گرفته شده و به انتهای صف آماده اضافه می شود.
 - اگر n فرایند در صف آماده وجود داشته باشد و مقدار زمانی کوانتومی q باشد، هر فرایند $\frac{1}{n}$ از زمان پردازنده را به صورت بخش‌هایی با حداکثر q واحد زمانی در یک مرحله دریافت می کند.
 - هیچ فرایندی بیش از $(n - 1)q$ از واحد زمان منتظر نمی ماند.
 - یک وقفه زمان سنج در هر واحد زمانی کوانتومی رخ می دهد تا فرایند بعدی زمان بندی شود.
- عملکرد

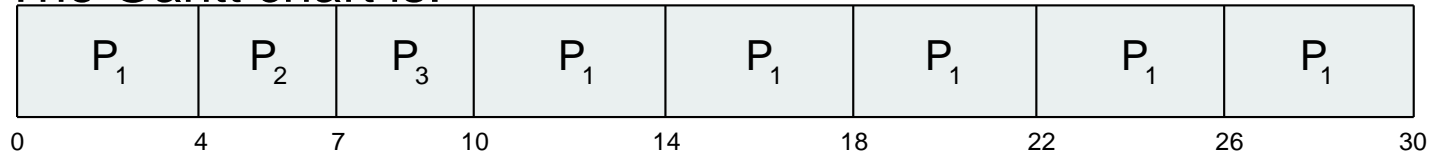
- مقدار q بزرگ: شبیه به الگوریتم FIFO اولین درخواست، اولین سرویس
- مقدار q کوچک: شبیه به الگوریتم RR
- توجه داشته باشید که مقدار q باید نسبت به زمان جابجایی متن (Context Switch) بزرگ باشد، در غیر این صورت سربار (Overhead) سیستم عامل بالا می رود.



Example of RR with Time Quantum = 4

| <u>Process</u> | <u>Burst Time</u> |
|----------------|-------------------|
| P_1 | 24 |
| P_2 | 3 |
| P_3 | 3 |

- The Gantt chart is:



- به طور کلی، میانگین زمان گردش در الگوریتم RR از SJF بیشتر است، اما زمان پاسخ بهتری دارد.
- مقدار q باید نسبت به زمان جابجایی متن بزرگ باشد.
- مقدار معمول q بین ۱۰ تا ۱۰۰ میلی ثانیه است.
- زمان جابجایی متن کمتر از ۱۰ میکروثانیه است.



Round Robin (RR)

■ مزایای الگوریتم RR

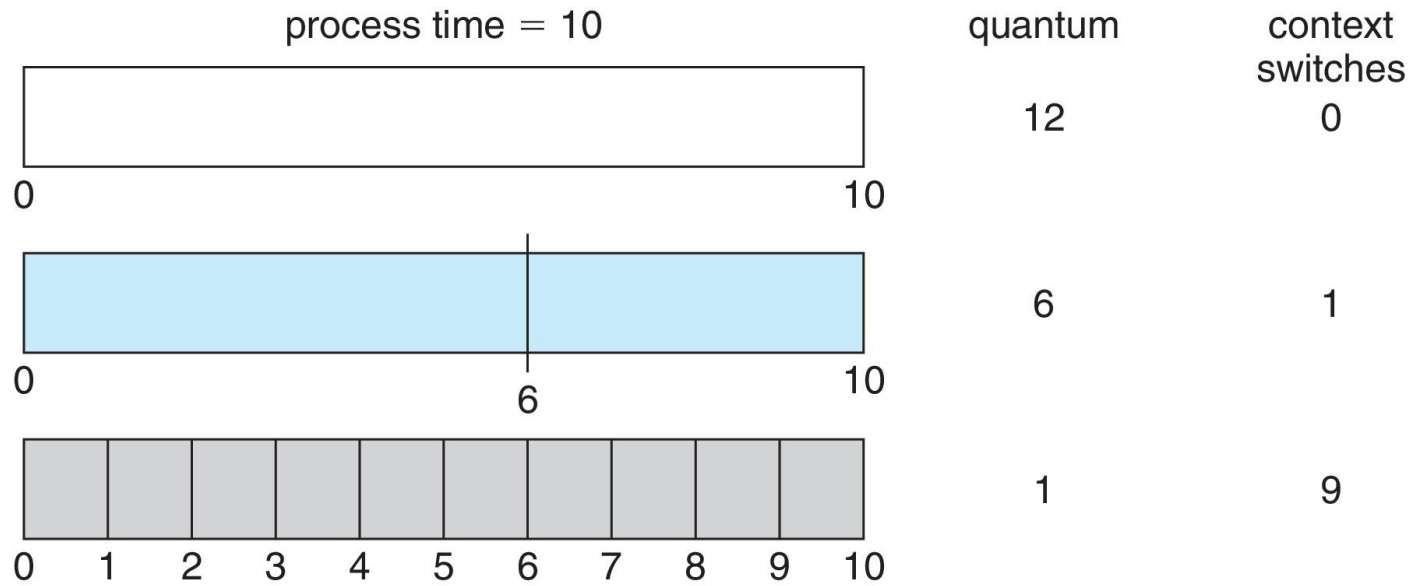
- تضمین زمان پاسخ مطلوب برای کارهای معمولی کوچک
- نداشتن قحطی و گرسنگی در سیستم
- سادگی اجرا
- عملکرد عادلانه

■ معایب الگوریتم RR در سیستم عامل

- انتخاب برهه زمانی (مدت زمان کوانتوم) یکی از معیارهای مشخص کننده عملکرد الگوریتم RR است. اگر کوانتوم زمانی خیلی کوچک باشد، توان عملیاتی (throughput سیستم عامل) کم خواهد شد.
- سربار تعداد زیاد تعویض میان اجرای فرآیندها
- میانگین زمان اجرای نسبتاً بالا در پردازش‌های طولانی

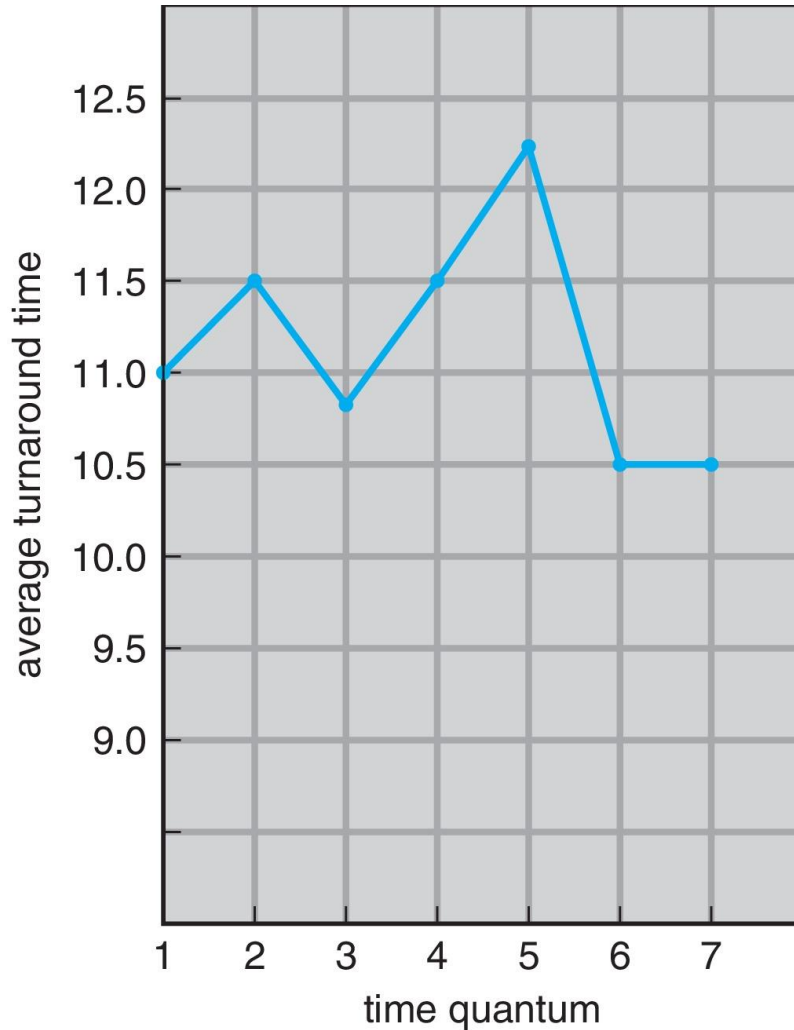


Time Quantum and Context Switch Time





Turnaround Time Varies With The Time Quantum



| process | time |
|---------|------|
| P_1 | 6 |
| P_2 | 3 |
| P_3 | 1 |
| P_4 | 7 |

- ۸۰ درصد انفجارهای پردازش باید کوتاهتر از q باشد



زمان‌بندی اولویت محور

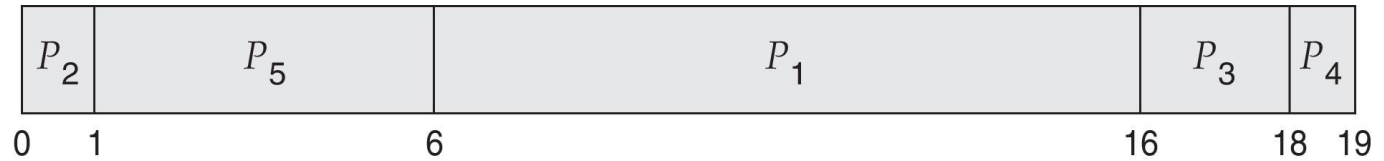
- **زمان‌بندی بر اساس اولویت (Priority Scheduling)**
 - به هر فرایند یک عدد اولویت (عدد صحیح) اختصاص داده می‌شود.
 - پردازنده به فرایندی با بالاترین اولویت (کوچک‌ترین عدد صحیح نشان‌دهنده بالاترین اولویت) اختصاص می‌یابد.
 - این الگوریتم می‌تواند به دو صورت **انحصاری و غیرانحصاری** عمل کند.
 - الگوریتم **SJF** نوعی زمان‌بندی بر اساس اولویت است که در آن اولویت معکوس طول پیش‌بینی‌شده‌ی بازه **CPU** بعدی در نظر گرفته می‌شود.
- **مشکل: گرسنگی - (Starvation)** فرایندهای با اولویت پایین ممکن است هرگز اجرا نشوند.
- **راه‌حل: افزایش اولویت فرایند با گذشت زمان (Aging)**



Example of Priority Scheduling

| فرآیند | زمان انفجار | اولویت |
|--------|-------------|--------|
| P_1 | 10 | 3 |
| P_2 | 1 | 1 |
| P_3 | 2 | 4 |
| P_4 | 1 | 5 |
| P_5 | 5 | 2 |

■ گانت چارت زمان بندی اولویت



■ $8.2 =$ میانگین زمان انتظار



Priority Scheduling w/ Round-Robin

- فرآیند را با بالاترین اولویت اجرا کنید.
- فرآیندهای با **اولویت یکسان** به صورت **دوره ای** اجرا می شوند

- Example:

| <u>Process</u> | <u>Burst Time</u> | <u>Priority</u> |
|----------------|-------------------|-----------------|
| P_1 | 4 | 3 |
| P_2 | 5 | 2 |
| P_3 | 8 | 2 |
| P_4 | 7 | 1 |
| P_5 | 3 | 3 |

- Gantt Chart with time quantum = 2





صف چند سطحی (Multilevel Queue)

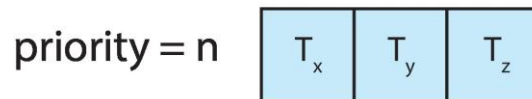
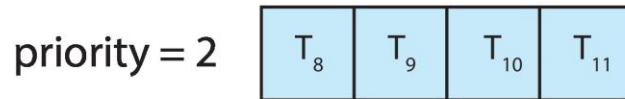
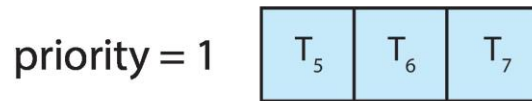
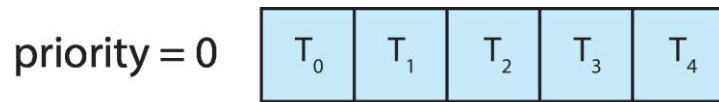
- صف آماده از چندین صف تشکیل شده است.
- یک زمان بند صف چند سطحی با پارامترهای زیر تعریف می شود:
 - تعداد صفها
 - الگوریتم زمان بندی مشخصی برای هر صف وجود دارد
 - روشی برای تعیین اینکه یک فرایند هنگام نیاز به سرویس وارد کدام صف می شود، استفاده می شود.
 - روش زمان بندی بین صفها



صف چند سطحی

■ در زمان بندی بر اساس اولویت، برای هر اولویت یک صف جداگانه وجود دارد

■ فرایند موجود در صف با بالاترین اولویت زمان بندی می شود!

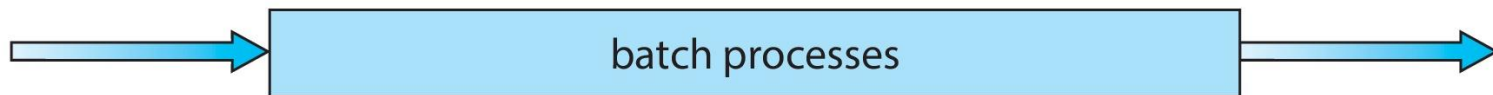
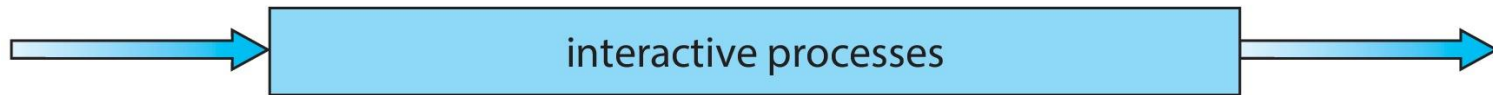
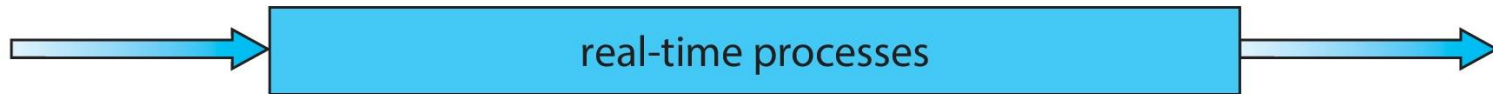




صف چند سطحی

■ اولویت بندی بر اساس نوع فرایند انجام می شود.

highest priority



lowest priority



Multilevel Feedback Queue صف بازخورد چند سطحی

- یک فرایند می‌تواند بین صف‌های مختلف جا بجا شود.
- یک زمان‌بند صف چند سطحی با بازخورد با پارامترهای زیر تعریف می‌شود:
 - تعداد صف‌ها
 - الگوریتم زمان‌بندی برای هر صف
 - روشی که برای تعیین زمان ارتقای یک فرایند به صف بالاتر استفاده می‌شود.
 - روشی که برای تعیین زمان تنزل یک فرایند به صف پایین‌تر استفاده می‌شود.
 - روشی که برای تعیین اینکه یک فرایند هنگام نیاز به سرویس وارد کدام صف می‌شود، استفاده می‌شود.
- با استفاده از صف چند سطحی با بازخورد می‌توان پیری (Aging) را پیاده‌سازی کرد.



Example of Multilevel Feedback Queue

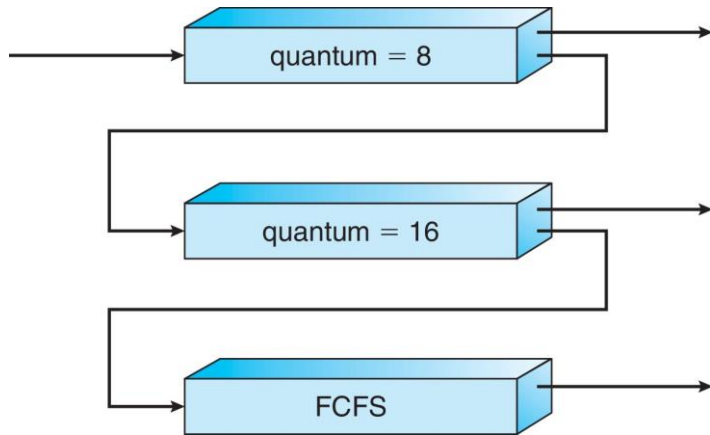
سه صف:

• **Q0 – RR** با مقدار زمانی کوانتومی ۸ میلی ثانیه

• **Q1 – RR** با مقدار زمانی کوانتومی ۱۶ میلی ثانیه

• **Q3-FCFS** اولین درخواست، اولین سرویس

■ زمان بندی



• یک فرایند جدید وارد صف Q0 می شود که با الگوریتم RR سرویس داده می شود.

○ هنگامی که این فرایند کنترل پردازنده را به دست

می آورد، ۸ میلی ثانیه زمان پردازنده دریافت می کند.

○ اگر در این ۸ میلی ثانیه به اتمام نرسد، به صف Q1

منتقل می شود.

• در Q1، فرایند مجدداً با الگوریتم RR سرویس داده می شود و

۱۶ میلی ثانیه زمان CPU اضافی دریافت می کند.

○ اگر همچنان کامل نشده باشد، پیش گرفته شده و به صف

Q2 منتقل می شود

زمان بندی رشته یا ریسمان



■ در سیستم‌عامل‌های چند-رشته‌ای (multi-threaded)، از دو روش برای زمان‌بندی رشته‌ها (thread scheduling) استفاده می‌شود: دامنه رقابت سیستم (System Contention Scope) و دامنه رقابت فرایند (Process Contention Scope). این دو روش در محدوده رقابت برای منابع پردازنده با یکدیگر تفاوت دارند.

• دامنه رقابت فرایند (PCS):

- در این روش، رقابت بر سر منابع پردازنده تنها بین رشته‌های هم‌تراز (equivalents) درون یک فرایند (process) یا برنامه کاربردی (application) رخ می‌دهد. رشته‌های هم‌تراز، برای اجرا در یک فرایندهای سبک‌وزن LWP در دسترس زمان‌بندی می‌شوند.
- تعداد فرایندهای سبک‌وزن اندازه محدودی دارند.
- PCS معمولاً برای مدل‌های «یک‌به‌چند» (one-to-many) و «چندبه‌چند» (many-to-many) استفاده می‌شود.
- این روش سبک‌تر و کم‌هزینه‌تر از SCS است
- سیستم‌عامل‌هایی مانند ویندوز، سولاریس و لینوکس از PCS برای زمان‌بندی رشته‌های کاربر (user-level threads) استفاده می‌کنند.

زمان بندی رشته یا ریسمان



• دامنه رقابت سیستم (SCS):

- در این روش، تمام رشته‌های موجود در سیستم عامل، برای استفاده از پردازنده با یکدیگر رقابت می‌کنند.
- SCS معمولاً برای مدل «یک‌به‌یک» (one-to-one) استفاده می‌شود.
- به ازای هر فرآیند کاربر یک LWP ساخته خواهد .
- این روش بسیار قابل پیش‌بینی است، زیرا هسته سیستم عامل (kernel) بر تمام رشته‌ها نظارت مستقیم دارد.
- با این حال، SCS به دلیل لزوم مدیریت مرکزی تمام رشته‌ها، می‌تواند پرهزینه‌تر از PCS باشد.
- سیستم عامل‌هایی مانند لینوکس که از مدل یک‌به‌یک پشتیبانی می‌کنند، از SCS برای زمان بندی رشته‌های هسته (kernel-level threads) استفاده می‌کنند.

■ به‌طور خلاصه، PCS برای رقابت درون یک فرآیند یا برنامه کاربردی و SCS برای رقابت بین تمام رشته‌های سیستم کاربرد دارد. انتخاب روش مناسب به مدل رشته‌بندی و اولویت‌های سیستم عامل بستگی دارد.



زمان بندی رشته یا ریسمان

- رابط برنامه‌نویسی (API) به کاربر اجازه می‌دهد تا در هنگام ایجاد رشته، بین PCS یا SCS یکی را انتخاب کند:
- `PTHREAD_SCOPE_PROCESS` با استفاده از زمان بندی PCS ، رشته‌ها را زمان بندی می‌کند.
- `PTHREAD_SCOPE_SYSTEM` با استفاده از زمان بندی SCS ، رشته‌ها را زمان بندی می‌کند.
- این انتخاب ممکن است توسط سیستم عامل محدود شود. برای مثال، لینوکس و macOS فقط به `PTHREAD_SCOPE_SYSTEM` اجازه می‌دهند.



زمان بندی چند پردازنده ای

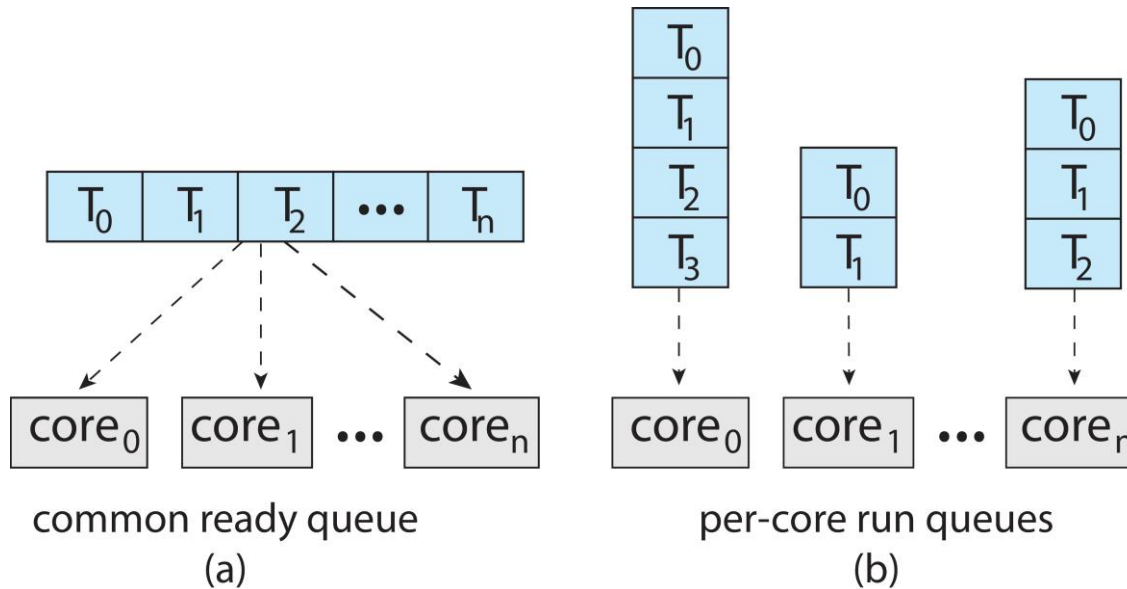
■ زمان بندی پردازنده در سیستم های چند پردازنده CPU (scheduling in Multiprocessor Systems)

- زمان بندی CPU زمانی که چندین پردازنده در دسترس باشد، پیچیده تر می شود.
- مفهوم چند پردازنده می تواند شامل انواع مختلفی از معماری ها باشد:
 - پردازنده های چند هسته ای (Multicore CPUs)
 - هسته های چند رشته ای (Multithreaded cores)
 - سیستم های دسترسی غیر یکنواخت به حافظه NUMA
 - پردازش ناهمگن (Heterogeneous multiprocessing)



زمان بندی چند پردازنده ای

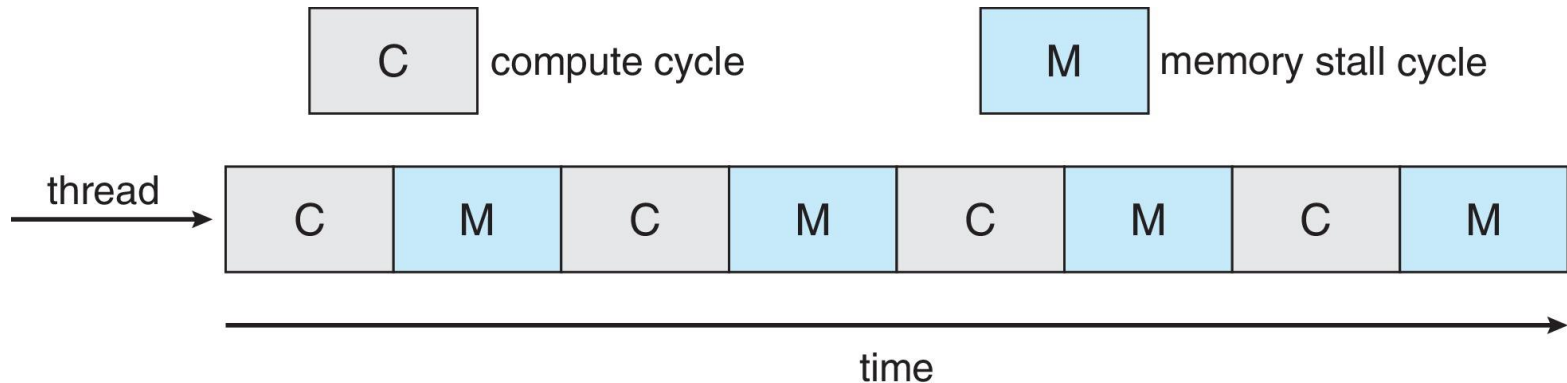
- چند پردازنده متقارن (SMP) حالتی است که در آن هر پردازنده به صورت مستقل، خود را زمان بندی می کند.
- در این حالت، همه رشته ها می توانند در یک صف آماده ای مشترک قرار بگیرند. شکل الف
- همچنین، هر پردازنده می تواند صف اختصاصی خود را برای رشته ها داشته باشد. شکل ب





پردازنده‌های چند هسته‌ای

- زمانی که پردازنده به حافظه دسترسی پیدا می‌کند، زمان قابل توجهی را صرف انتظار برای در دسترس قرار گرفتن داده‌ها می‌کند. این وضعیت که به **توقف موقت حافظه** شناخته می‌شود،
 - عمدتاً به این دلیل رخ می‌دهد که پردازنده‌های مدرن با سرعت بسیار بالاتری نسبت به حافظه کار می‌کنند.
 - وقفه حافظه همچنین می‌تواند به دلیل **خطای حافظه نهان** (دسترسی به داده‌هایی که در حافظه نهان وجود ندارند) رخ دهد.
 - شکل زیر یک وقفه حافظه را نشان می‌دهد. در این سناریو، پردازنده می‌تواند تا ۵۰ درصد از زمان خود را صرف انتظار برای دریافت داده از حافظه کند.





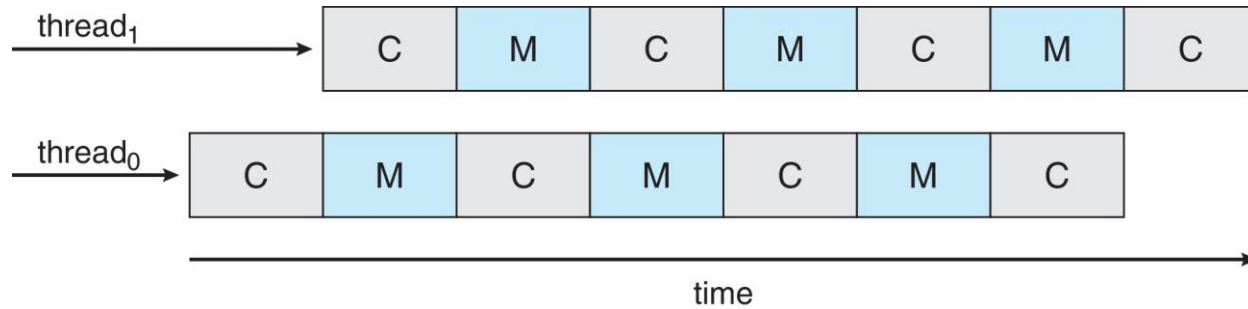
پردازنده‌های چند هسته‌ای

- تمایل اخیر این است که چندین هسته‌ی پردازنده را روی یک تراشه‌ی فیزیکی واحد قرار دهند.
- این روش باعث افزایش سرعت و کاهش مصرف انرژی می‌شود.
- استفاده از چندین رشته در هر هسته نیز رو به افزایش است.
- این کار از توقف موقت حافظه (memory stall) برای پیشرفت در یک رشته‌ی دیگر در حین بازیابی حافظه استفاده می‌کند.

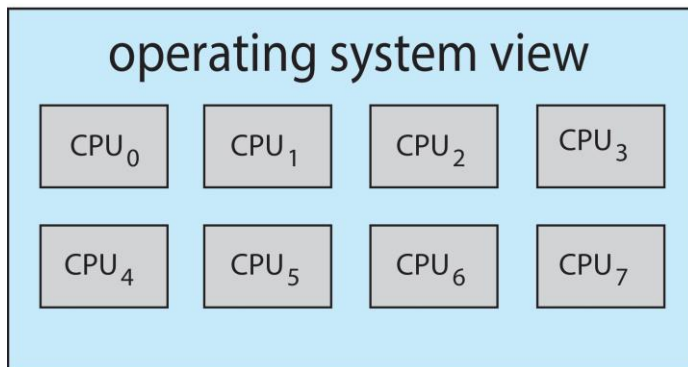
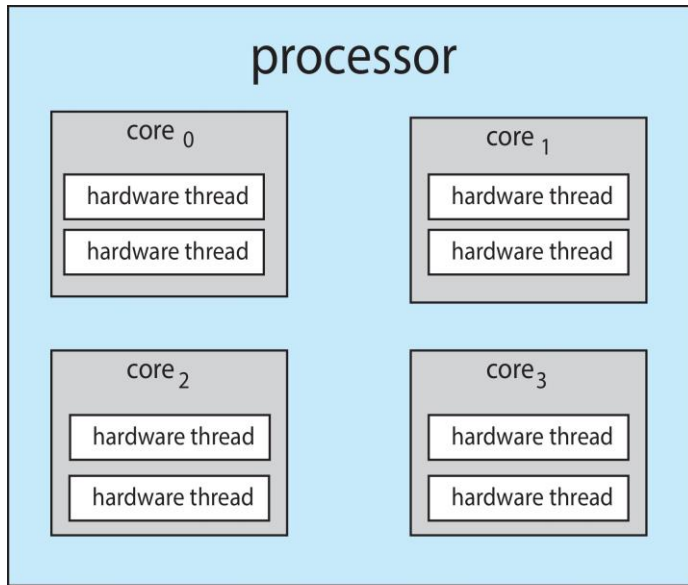


سیستم های چندنخی چندهسته ای

- هر هسته بیش از یک رشته ی سخت افزاری دارد.
- اگر یک رشته دچار توقف موقت حافظه شود، به یک رشته ی دیگر سوئیچ کنید!



سیستم های چندنخی چندهسته ای



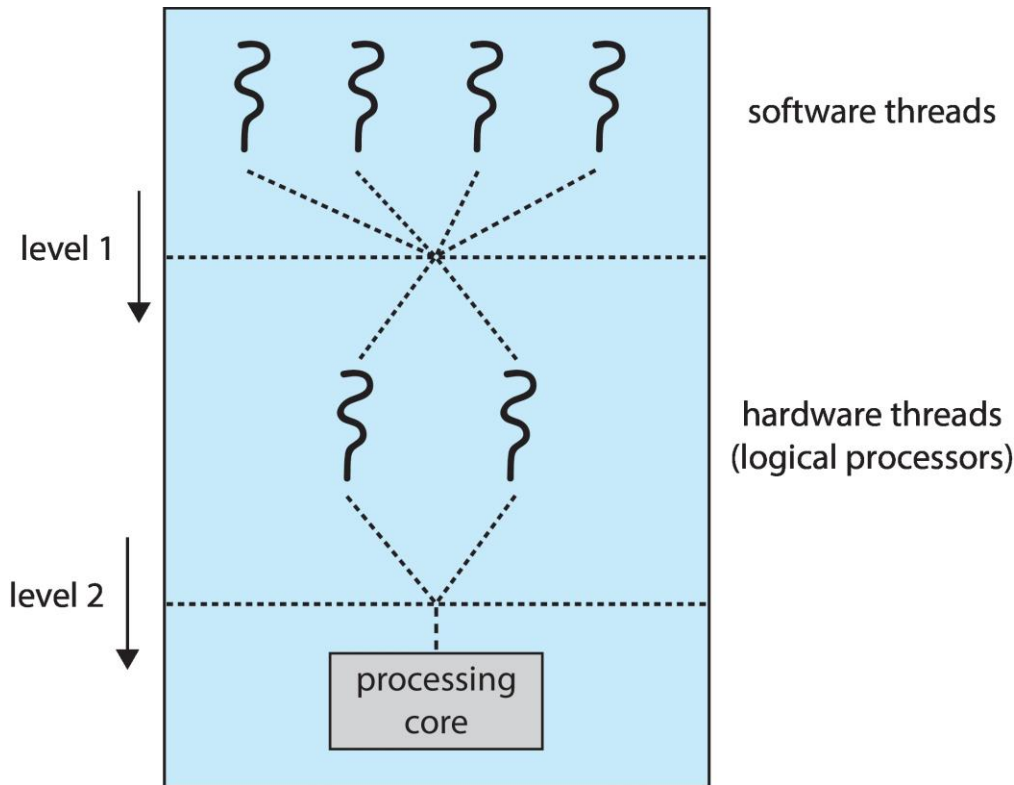
■ چند رشته ای کردن تراشه

Chip-multithreading

■ چندین رشته ای سخت افزاری را به هر هسته اختصاص می دهد. اینتل به این قابلیت، ابرنخی (Hyperthreading) می گوید .

■ بر روی یک سیستم چهار هسته ای با ۲ رشته ای سخت افزاری به ازای هر هسته، سیستم عامل ۸ پردازنده ای منطقی را می بیند .

سیستم های چندنخی چندهسته ای



■ دو سطح از زمان بندی وجود دارد:

- سیستم عامل تصمیم می گیرد که کدام رشته ی نرم افزاری روی یک پردازنده منطقی یا نخ سخت افزاری اجرا شود.
- هر هسته چطور تصمیم می گیرد کدام رشته ی سخت افزاری (پردازنده منطقی) را روی هسته ی فیزیکی اجرا کند.



زمان بندی چند پردازنده ای - توازن بار

- در حالت SMP ، برای کارایی بهتر، باید همه ی پردازنده ها را لود نگه داشت.
- توازن بار (Load balancing) تلاش می کند تا حجم کاری به طور مساوی توزیع شود:
- مهاجرت فشاری (Push migration): یک کار دوره ای بار روی هر پردازنده را بررسی می کند و در صورت مشاهده ی اضافه بار، وظیفه را از پردازنده پر بار به پردازنده های دیگر منتقل می کند.
- مهاجرت کششی (Pull migration): پردازنده های بیکار، وظیفه ی منتظر مانده را از پردازنده ی شلوغ می کشند.

تمایل پردازنده - زمان بندی چند پردازنده ای

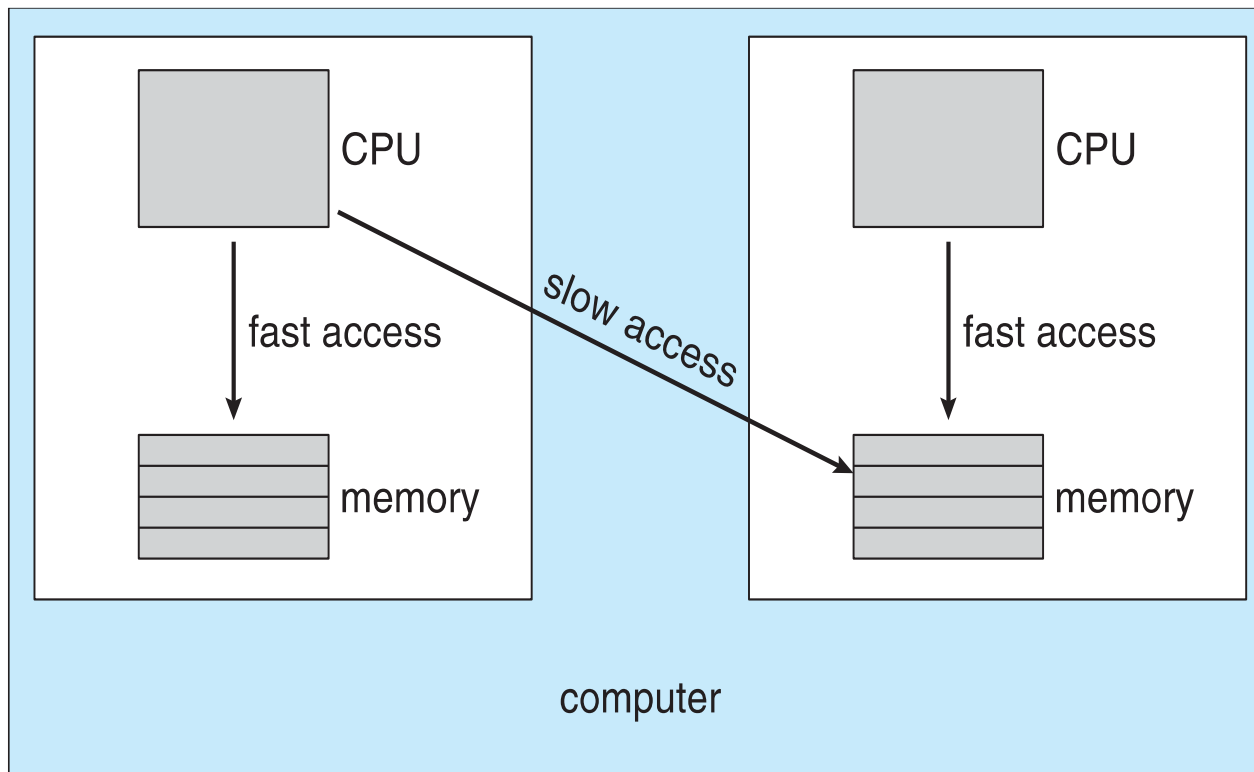


- زمانی که یک رشته روی یک پردازنده اجرا می‌شود، محتویات کش آن پردازنده، دسترسی‌های حافظه‌ی آن رشته را ذخیره می‌کند و دسترسی‌های بعدی احتمالاً از کش خواهد بود. ما این را به عنوان تمایل یک رشته به یک پردازنده (یعنی «تمایل پردازنده») در نظر می‌گیریم.
- توازن بار ممکن است بر تمایل پردازنده تأثیر بگذارد، زیرا ممکن است یک رشته برای متعادل کردن بارها از یک پردازنده به پردازنده‌ی دیگری منتقل شود، اما در این حالت، محتوای کش قبلی خود را از دست می‌دهد.
- **تمایل نرم: (Soft affinity)** سیستم عامل تلاش می‌کند تا یک رشته را روی همان پردازنده نگه دارد، اما هیچ تضمینی وجود ندارد.
- **تمایل سخت: (Hard affinity)** به یک فرایند اجازه می‌دهد مجموعه پردازنده‌هایی را که می‌تواند روی آن‌ها اجرا شود، مشخص کند.



زمان بندی پردازنده و دسترسی غیر یکنواخت

- اگر سیستم عامل از NUMA (دسترسی غیر یکنواخت به حافظه) آگاه باشد، برای رشته در حال اجرا حافظه‌ی نزدیک به پردازنده‌ای که رشته روی آن اجرا می‌شود را اختصاص می‌دهد.





زمان بندی برخط پردازنده

■ میتواند چالشهای واضحی داشته باشد.

- سیستم‌های بلادرنگ نرم (Soft real-time systems): وظایف حیاتی بلادرنگ دارای بالاترین اولویت هستند، اما هیچ تضمینی برای زمان بندی آنها وجود ندارد.
- سیستم‌های بلادرنگ سخت (Hard real-time systems): یک وظیفه باید تا مهلت تعیین شده‌ی خود سرویس شود.

Windows Scheduling



- هسته‌ی ویندوز از زمان‌بندی غیرانحصاری مبتنی بر اولویت استفاده می‌کند.
- رشته‌ی دارای بالاترین اولویت، در مرحله‌ی بعد اجرا می‌شود.
- توزیع‌کننده (Dispatcher) همان زمان‌بند است.
- یک رشته تا زمانی که:
 - ۱. بلاک شود،
 - ۲. از تکه‌ی زمانی خود استفاده کند، یا
 - ۳. توسط یک رشته‌ی با اولویت بالاتر پیش‌گرفته شود، اجرا می‌شود.
- رشته‌های بلادرگ می‌توانند رشته‌های غیر بلادرنگ را قبضه کرده و متوقف کنند.
- زمان‌بندی صف اولویت در ویندوز دارای **۳۲ سطح** است.
 - کلاس متغیر ۱ تا ۱۵ را در بر می‌گیرد.
 - کلاس بلادرنگ ۱۶ تا ۳۱ را در بر می‌گیرد.
 - **اولویت صفر متعلق به رشته‌ی مدیریت حافظه است.**
 - برای هر اولویت یک صف وجود دارد.
 - اگر هیچ رشته‌ی قابل‌اجرایی وجود نداشته باشد، یک رشته‌ی بیکار اجرا می‌شود.



Algorithm Evaluation

- چگونه الگوریتم زمان بندی CPU را برای یک سیستم عامل انتخاب کنیم؟
- معیارها را تعیین کنید، سپس الگوریتمها را ارزیابی کنید.
- مدل سازی قطعی (Deterministic modeling)
- نوعی ارزیابی تحلیلی است.
- یک حجم کاری از پیش تعیین شدهی خاص را در نظر می گیرد و عملکرد هر الگوریتم را برای آن حجم کاری تعریف می کند.
- فرض کنید ۵ فرایند در زمان ۰ وارد می شوند:

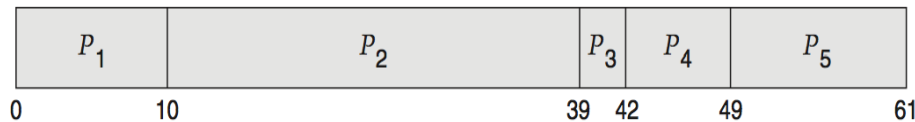
| <u>Process</u> | <u>Burst Time</u> |
|----------------|-------------------|
| P_1 | 10 |
| P_2 | 29 |
| P_3 | 3 |
| P_4 | 7 |
| P_5 | 12 |

Deterministic Evaluation

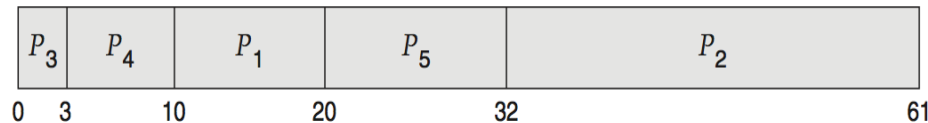


- برای هر الگوریتم، حداقل میانگین زمان انتظار را محاسبه کنید.
- این روش ساده و سریع است، اما به اعداد دقیق برای ورودی نیاز دارد و فقط برای آن ورودی‌ها قابل اعمال است.

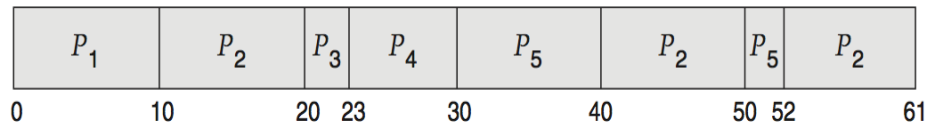
- FCS is 28ms:



- Non-preemptive SFJ is 13ms:



- RR is 23ms:





Queueing Models

- این مدل‌ها ورود فرایندها و همچنین وقفه‌های CPU و I/O را به صورت احتمالی توصیف می‌کنند.
- **توزیع‌های (احتمالی) رایج**: به طور معمول از توزیع نمایی (exponential) استفاده می‌شود که با میانگین آن توصیف می‌شود.
- **محاسبه‌ی معیارهای عملکرد**: این مدل‌ها امکان محاسبه‌ی میانگین توان عملیاتی (throughput)، میزان استفاده (utilization)، زمان انتظار و غیره را فراهم می‌کنند.
- فرض کنید سیستم کامپیوتری به عنوان شبکه‌ای از سرورها در نظر گرفته شود که هر کدام دارای صف (queue) از فرایندهای در حال انتظار هستند.
- **با دانستن نرخ ورود (arrival rate) و نرخ سرویس‌دهی (service rate)**: می‌توان میزان استفاده، میانگین طول صف، میانگین زمان انتظار و غیره را محاسبه کرد.



Little' s Formula

■ n = میانگین طول صف

■ W = میانگین زمان انتظار در صف

■ λ = میانگین نرخ ورود به صف

■ قانون لیتل: در حالت پایدار (steady state) ، تعداد فرایندهایی که صف را ترک می کنند باید برابر با تعداد فرایندهای ورودی باشد، بنابراین:

$$n = \lambda \times W$$

■ این رابطه برای هر الگوریتم زمان بندی و توزیع ورود معتبر است.

■ برای مثال، اگر به طور میانگین ۷ فرایند در ثانیه وارد شوند و به طور معمول ۱۴ فرایند در صف باشند، در این صورت میانگین زمان انتظار برای هر فرایند ۲ ثانیه خواهد بود.



■ پایان بخش اصلی فصل