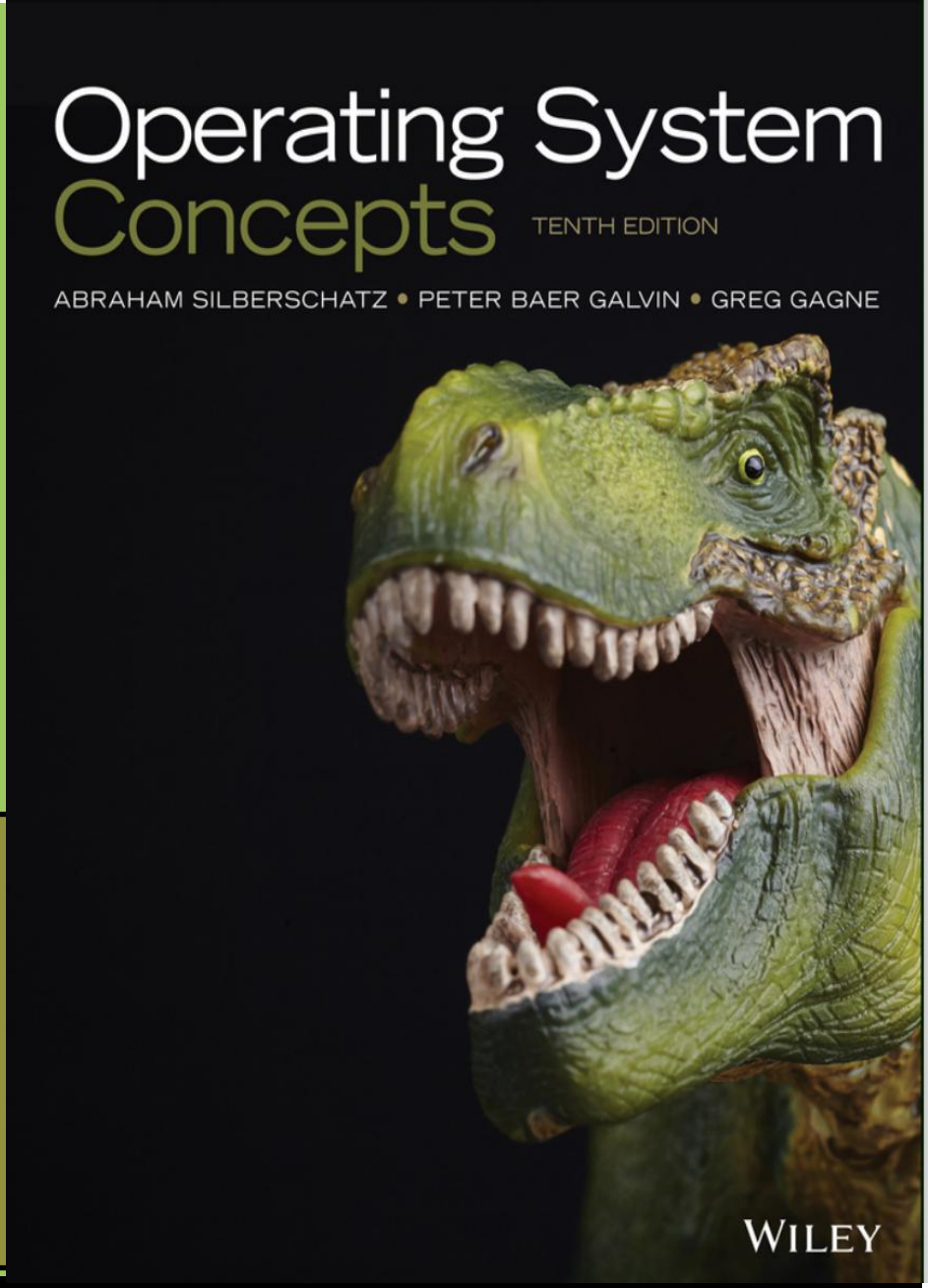
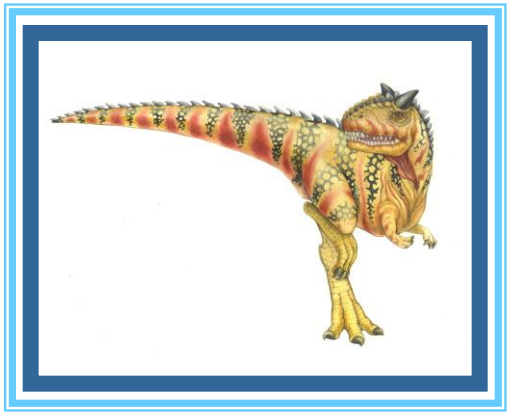
Operation Systems

Dr. A. Taghinezhad



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

فصل ۵: زمان بندی پردازنده



Outline

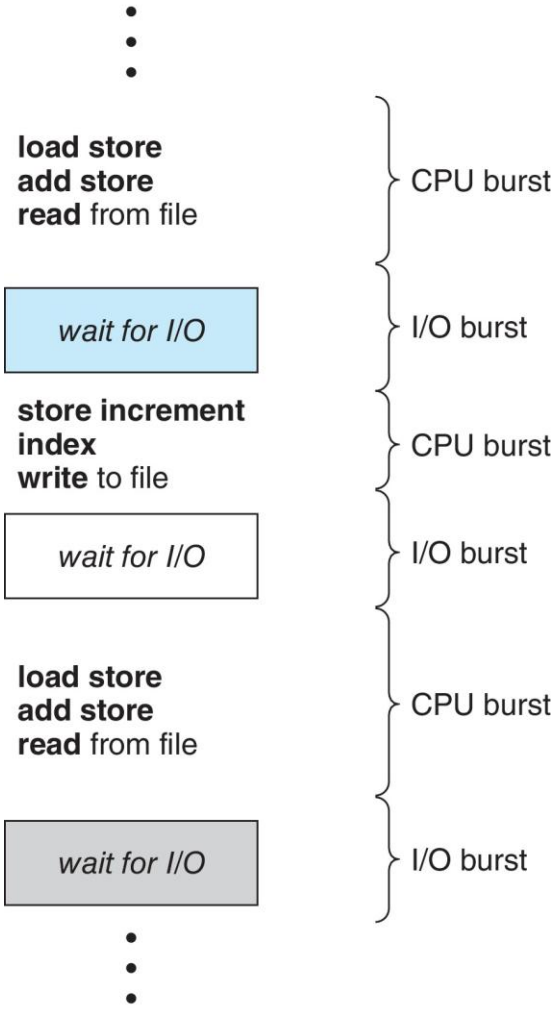
- **معیارهای زمان‌بندی (Scheduling Criteria):** معیارهایی برای ارزیابی الگوریتم‌های زمان‌بندیCPU
- **الگوریتم‌های زمان‌بندی (Scheduling Algorithms):** روش‌هایی که سیستم عامل برای تخصیص CPU به فرایندها استفاده می‌کند
- **زمان‌بندی رشته (Thread Scheduling):** تخصیص زمان پردازنده به رشته‌های درون یک فرایند
- **زمان‌بندی چند پردازنده (Multi-Processor Scheduling):** تخصیص منابع پردازشی در سیستم‌های چند پردازنده
- **زمان‌بندی زمان حقیقی CPU (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 زمان‌بندی

اهداف

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

.

مفاهیم اولیه



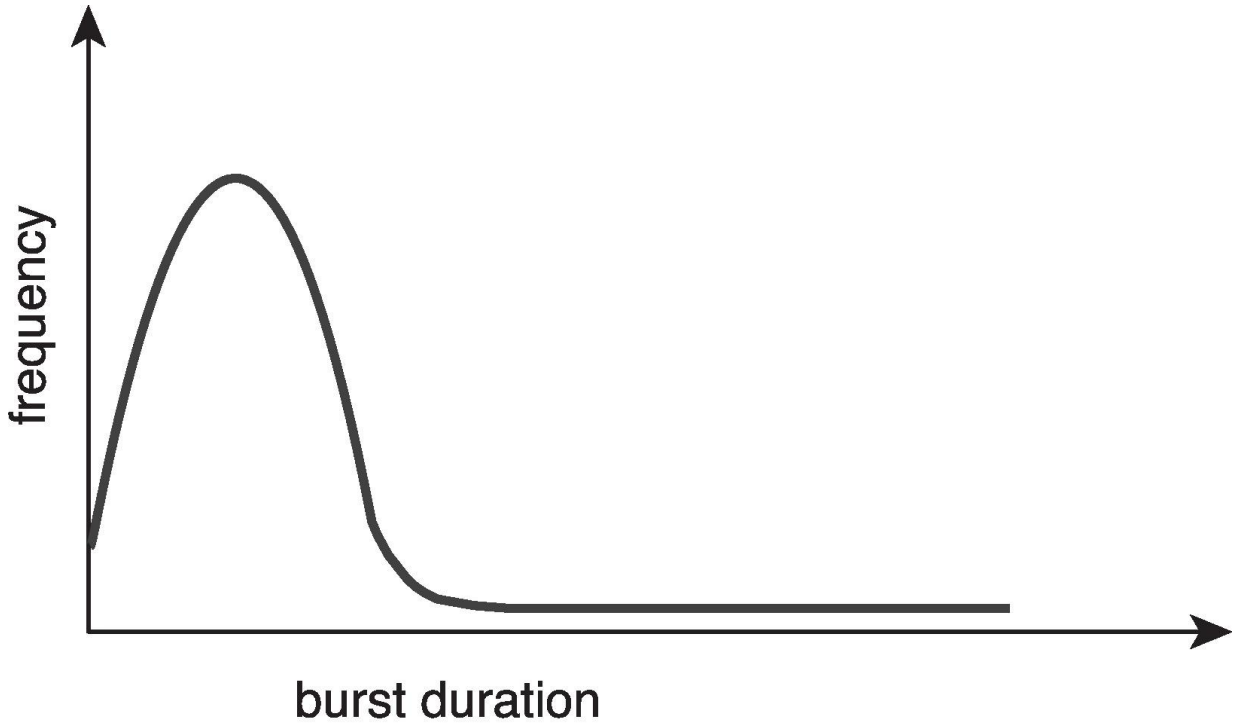
- **حداکثر استفاده ممکن از CPU با چندبرنامگی (Multiprogramming)** به معنی حداکثر استفاده از CPU با استفاده از تکنیک چندبرنامگی (اجرای همزمان چند برنامه در حافظه) و تأثیر آن بر کارایی سیستم توضیح داده خواهد شد.

- **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



زمان‌بندی پردازنده

- زمان‌بند پردازنده از میان فرایندهای موجود در صف آماده، یک فرایند را برای اجرا بر روی یک هسته پردازشی انتخاب می‌کند.
- صف آماده می‌تواند به روش‌های مختلف اولویت‌گذاری شود.

زمان‌بندی پردازنده

- **تصمیم‌گیری زمان‌بندی(CPU scheduling decisions)**

- تصمیم‌گیری برای زمان‌بندی در زمان‌های زیر رخ می‌دهد:
  1. **تغییر وضعیت** یک فرایند از در حالت **اجرا** به **انتظار** (وقتی فرایند درخواست I/O می‌دهد)
  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 (خاتمه‌ی طبیعی یا خطا)
- در موارد ۱ و ۴، انتخابی برای زمان‌بندی 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.

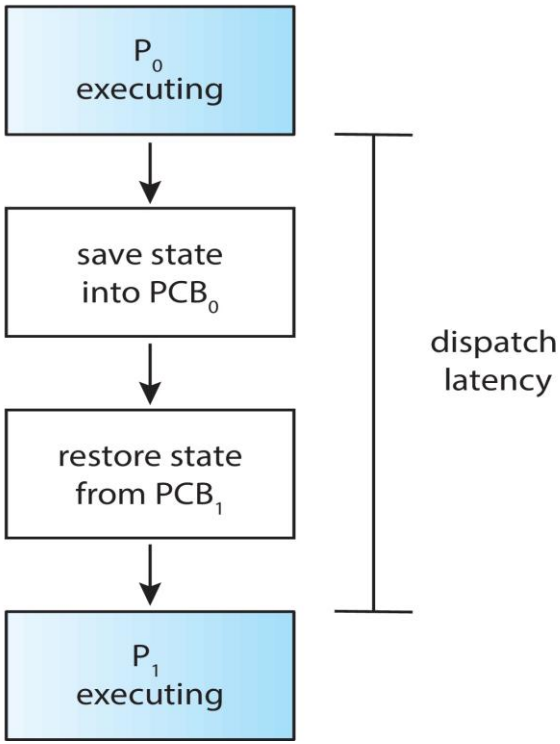
زمان‌بندی انحصاری و غیرانحصاری(پس‌گرفتنی)

- **زمان‌بندی انحصاری(Nonpreemptive Scheduling)**
  - زمانی که فرایند کنترل پردازنده را به دست می‌آورد، تا زمانی که آن را رها نکند (خاتمه یا انتظار) به اجرای خود ادامه می‌دهد.
  - تقریبا تمامی سیستم‌های عامل مدرن از زمان‌بندی غیرانحصاری (Preemptive Scheduling)استفاده می‌کنند.

Preemptive Scheduling and Race Conditions

- **زمان‌بندی غیرانحصاری**
  - می‌تواند منجر به **شرایط مسابقه** شود، زمانی که چندین فرایند به داده‌های مشترک دسترسی داشته باشند.
  - فرض کنید دو فرایند داریم که داده‌ای را به اشتراک گذاشته‌اند. در حالی که یک فرایند در حال به‌روزرسانی داده‌ها است، سیستم عامل کنترل پردازنده را از آن گرفته و به فرایند دیگری می‌دهد. در نتیجه، فرایند دوم تلاش می‌کند داده‌ها را بخواند، در حالی که داده‌ها در یک وضعیت نامعتبر قرار دارند.

Dispatcher توزیع‌کننده



- **ماژول توزیع‌کننده(Dispatcher module)**
- ماژول توزیع‌کننده، کنترل پردازنده را به فرایندی که توسط زمان‌بند پردازنده انتخاب شده است، واگذار می‌کند. 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 راه‌اندازی 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 زمان‌بندی

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

الگوریتم زمان‌بندی FCFS

- اولین ورود و اولین خدمت‌رسانی
- اولویت بر فرآیندهایی هست که زودتر به صف وارد شده‌اند.
- از نوع زمانبندی انحصاری

First- Come, First-Served (FCFS) Scheduling

فرآیندها	زمان‌سوختن یا اجرا
P_1	24
P_2	3
P_3	3

- فرض کنید فرایندها به ترتیب زیر وارد سیستم می‌شوند:
  - P1 , P2 , P3
- نمودار گانت برای زمان‌بندی به صورت زیر است:

- زمان انتظار Waiting time for P_1 = 0; P_2 = 24; P_3= 27
- میانگین زمان انتظار Average waiting time: (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: (۶ + ۰ + ۳) / ۳ = ۳
  - 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.

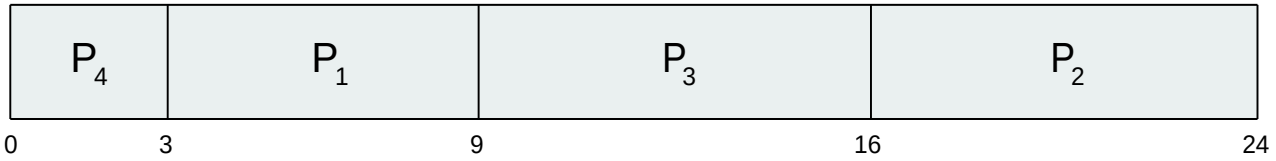
Shortest-Job-First (SJF) Scheduling

- در نظر گرفتن طول بازه فرآیند بعدی هر فرایند برای زمان‌بندی
- زمان‌بندی فرایندی با کوتاه‌ترین بازه فرآیند بعدی
- زمان‌بندی انحصاری هست.
- SJFبهینه است و برای مجموعه‌ای از فرایندهای خاص، حداقل میانگین زمان انتظار را به دست می‌دهد.
- نسخه‌ی غیرانحصاری از SJF با نام زمان باقیمانده کوتاه‌ترین در ابتدا (Shortest Remaining Time First)
- **چگونه طول بازه فرایند بعدی را تعیین کنیم؟**
  - پرسیدن از کاربر
  - تخمین

Example of SJF

Process	Burst Time
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

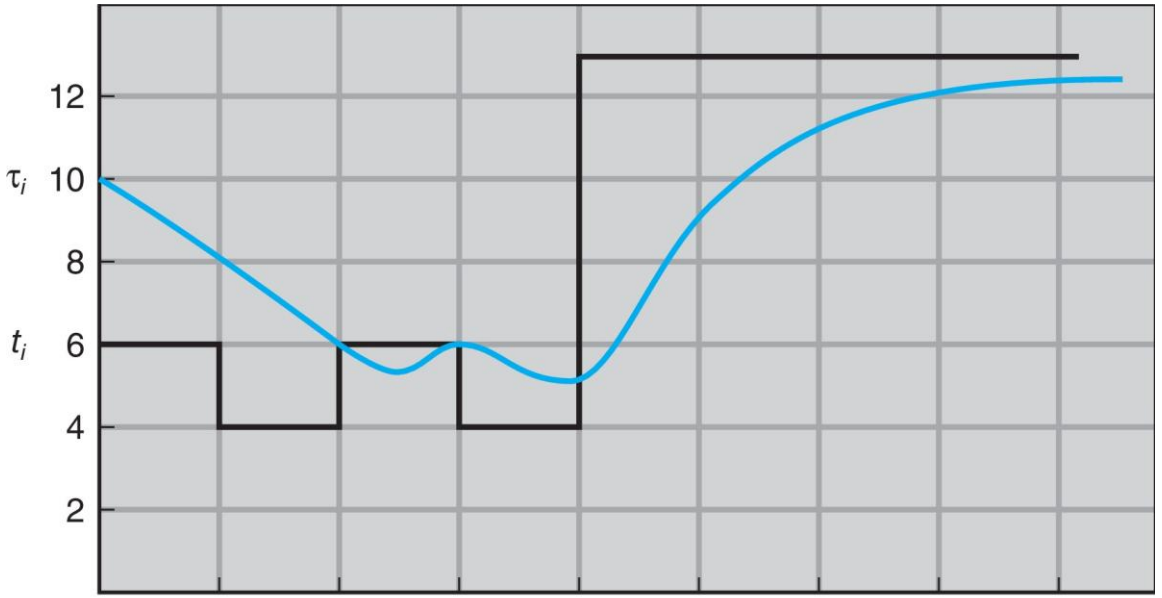
- طول واقعی قابل پیش‌بینی نیست، اما باید به طول بازه قبلی نزدیک باشد.
- انتخاب فرایند با کوتاه‌ترین بازه پردازنده پیش‌بینی‌شده بعدی
- این کار با استفاده از طول بازه‌های پردازنده قبلی و میانگین‌گیری نمایی قابل انجام است.

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

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

- مقدار α به طور معمول برابر با ½ 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



CPU burst (t_i)		6	4	6	4	13	13	13	...
"guess" (τ_i)	10	8	6	6	5	9	11	12	...

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

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

Shortest Remaining Time First Scheduling

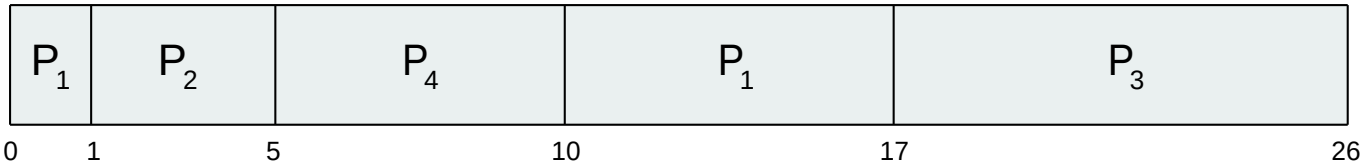
- **زمان‌بندی غیرانحصاری با SJN (Shortest Job Next)**
  - هر زمان که یک فرایند جدید وارد صف آماده می‌شود، تصمیم‌گیری در مورد اینکه کدام فرایند را زمان‌بندی کند، با استفاده از الگوریتم SJN انجام می‌شود.
- **مقایسه SRT و SJN**
- آیا SRT از نظر حداقل میانگین زمان انتظار برای مجموعه‌ای از فرایندهای خاص، «بهینه‌تر» از SJN است؟

Example of Shortest-remaining-time-first

- اضافه کردن مفاهیم تغییر زمان‌ورود و غیرانحصاری بودن:

Process	*Arrival* Time	Burst Time
P_1	0	8
P_2	1	4
P_3	2	9
P_4	3	5

- *Preemptive* SJF Gantt Chart



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

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 کوانتومیq) اختصاص داده می‌شود که معمولا ۱۰ تا ۱۰۰ میلی‌ثانیه است.
- بعد از گذشتن این زمان، فرایند گرفته شده و به انتهای صف آماده اضافه می‌شود.
- 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 می‌کند.
- هیچ فرایندی بیش $(n-1)q$ ازواحد زمان منتظر نمی‌ماند.
- یک **وقفه زمان‌سنج** در هر واحد زمانی کوانتومی رخ می‌دهد تا فرایند بعدی زمان‌بندی شود.

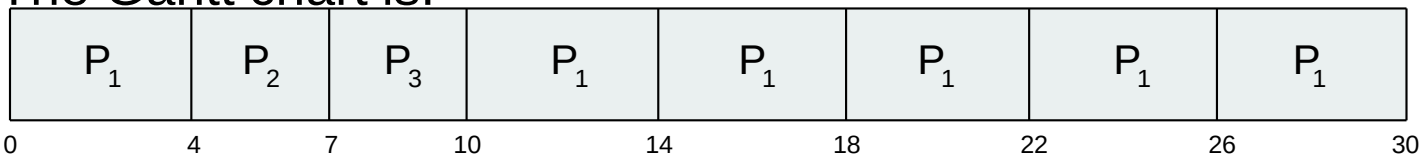
عملکرد

- مقدار q بزرگ: شبیه به الگوریتم FIFO اولین درخواست، اولین سرویس
- مقدار q کوچک: شبیه به الگوریتمRR
- 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 می‌رود.

Example of RR with Time Quantum = 4

Process	Burst Time
P_1	24
P_2	3
P_3	3

- The Gantt chart is:

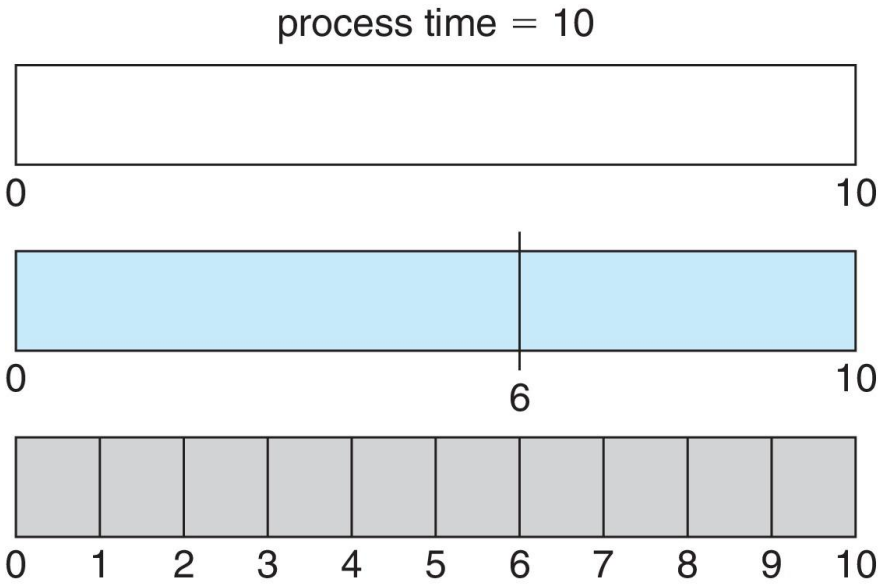


- به طور کلی، میانگین زمان گردش در الگوریتم RR از SJF بیشتر است، اما زمان پاسخ بهتری دارد.
- مقدار q باید نسبت به زمان جابجایی متن بزرگ باشد.
- مقدار معمول q بین ۱۰ تا ۱۰۰ میلی‌ثانیه است.
- زمان جابجایی متن کمتر از ۱۰ میکروثانیه است.

Round Robin (RR)

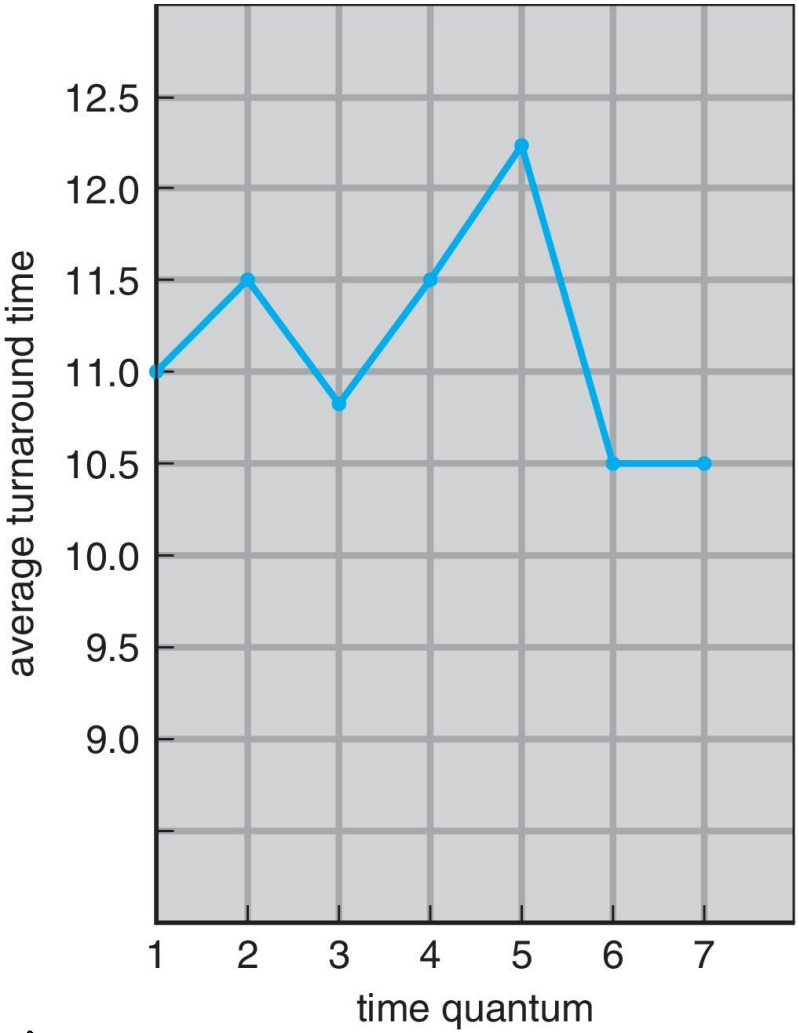
- مزایای الگوریتم RR
  - تضمین زمان پاسخ مطلوب برای کارهای معمولی کوچک
  - نداشتن قحطی و گرسنگی در سیستم
  - سادگی اجرا
  - عملکرد عادلانه
- معایب الگوریتم RR در سیستم عامل
  - انتخاب برهه زمانی (مدت زمان کوانتوم) یکی از معیارهای مشخص‌کننده عملکرد الگوریتم RR است. اگر کوانتوم زمانی خیلی کوچک باشد، توان 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



quantum	context switches
12	0
6	1
1	9

Turnaround Time Varies With The Time Quantum



process	time
P_1	6
P_2	3
P_3	1
P_4	7

- ۸۰ درصد انفجارهای پردازه باید کوتاهتر از q باشد

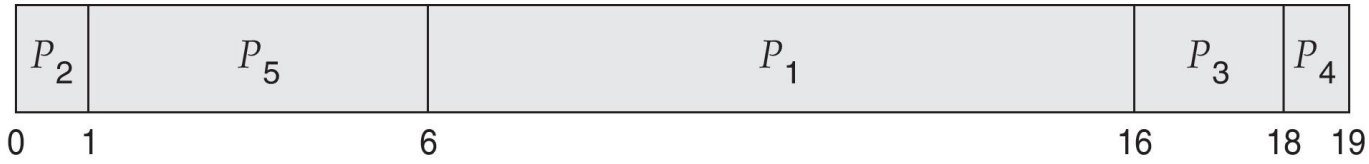
زمان‌بندی اولویت محور

- **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 زمان‌بندی اولویت



- میانگین زمان انتظار= 8.2

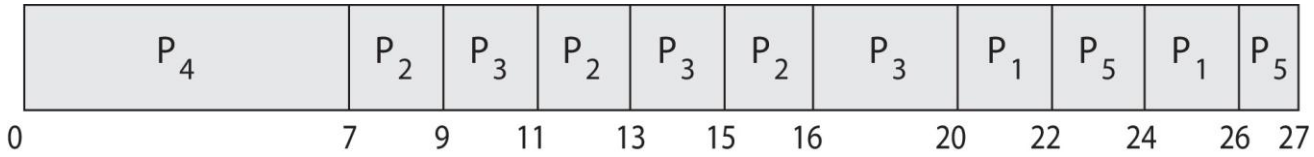
Priority Scheduling w/ Round-Robin

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

- Example:

Process	Burst Time	Priority
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

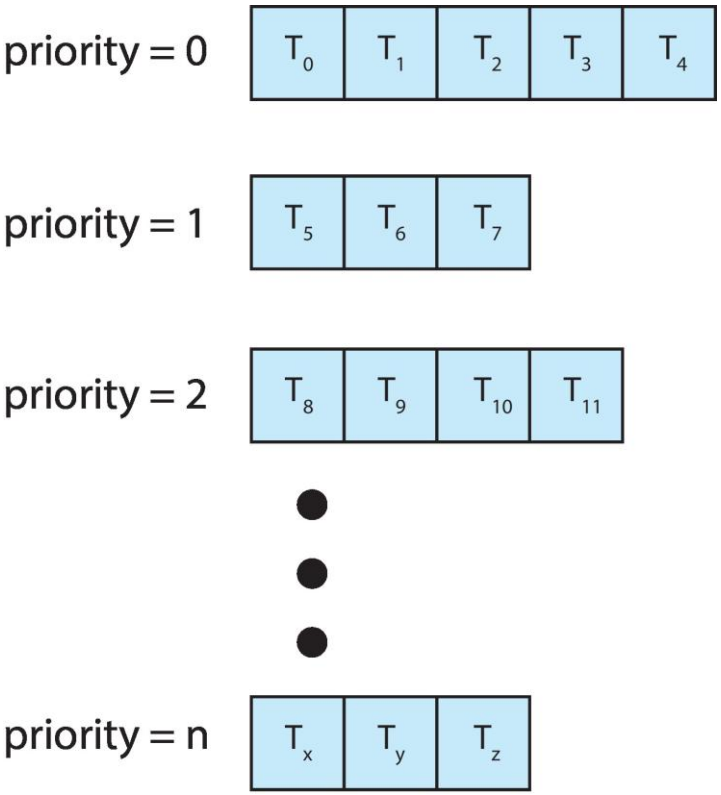


صف چند سطحی(Multilevel Queue)

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

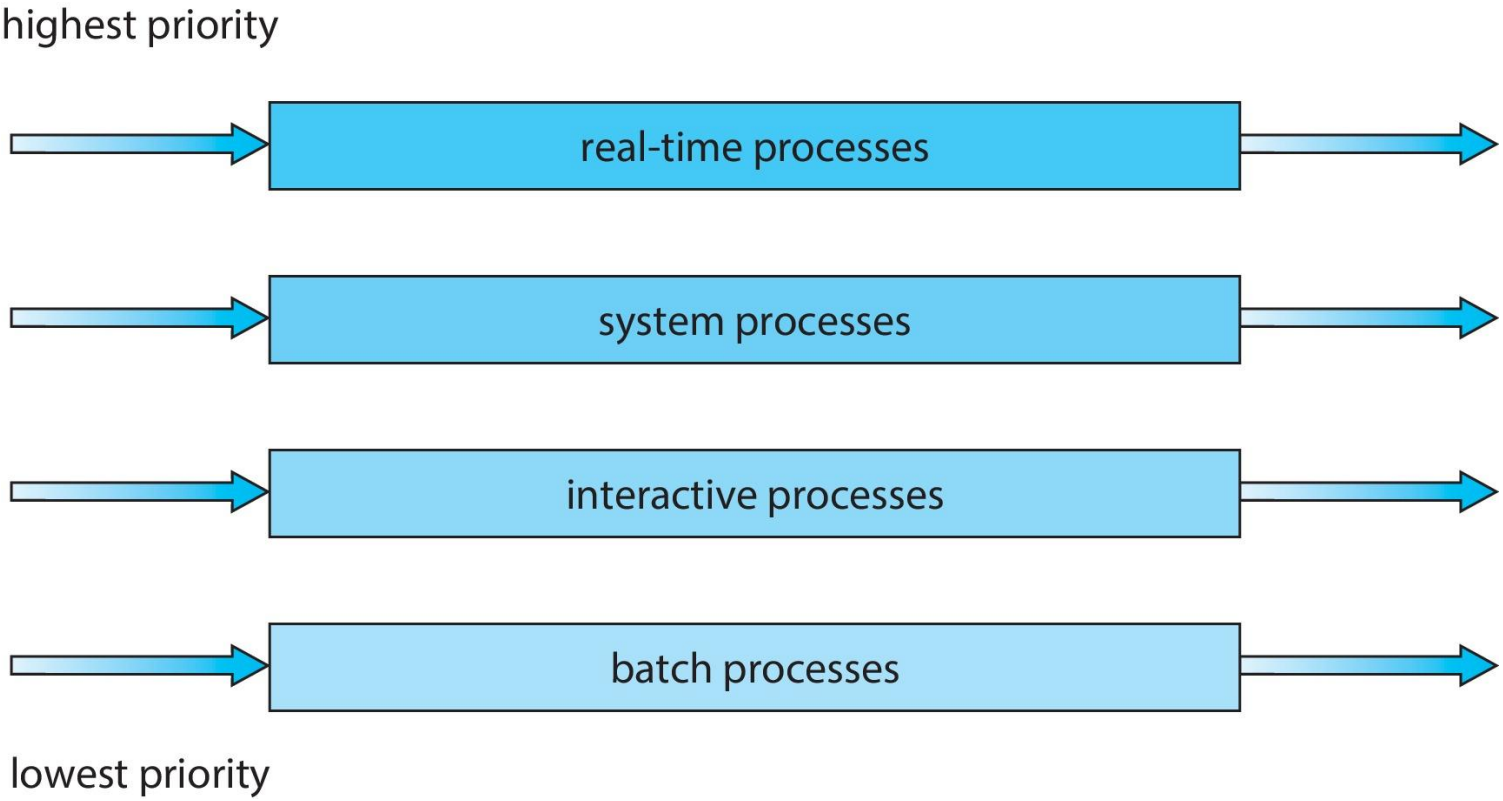
صف چند سطحی

- در زمان‌بندی بر اساس اولویت، برای هر اولویت یک صف جداگانه وجود دارد
- فرایند موجود در صف با بالاترین اولویت زمان‌بندی می‌شود!.



صف چند سطحی

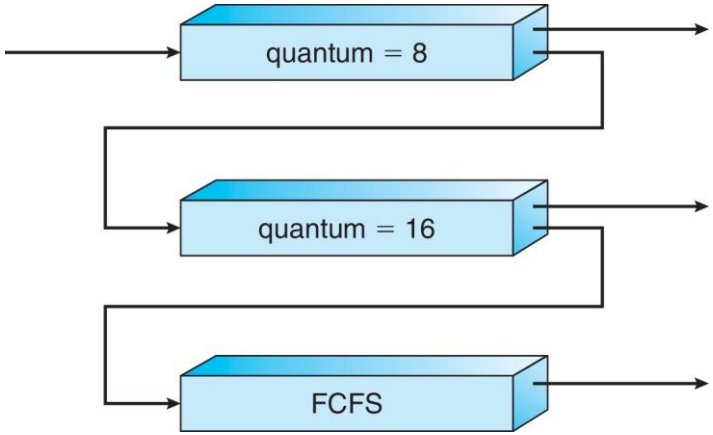
- اولویت‌بندی بر اساس نوع فرایند انجام می‌شود.



Multilevel Feedback Queue صف بازخورد چند سطحی

- یک فرایند می‌تواند بین **صف‌های مختلف جابجا** 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 میلی‌ثانیه**
  - **Q1 – RR با مقدار زمانی کوانتومی ۱۶ میلی‌ثانیه**
  - **Q2-FCFS اولین درخواست، اولین سرویس**
- **زمان‌بندی**
  - یک فرایند جدید وارد صف Q0 می‌شود که با الگوریتم RR سرویس داده می‌شود.
    - هنگامی که این فرایند کنترل پردازنده را به دست می‌آورد، ۸ میلی‌ثانیه زمان پردازنده دریافت می‌کند.
    - اگر در این ۸ میلی‌ثانیه به اتمام نرسد، به صف Q1 منتقل می‌شود.
  - در Q1 ، فرایند مجددا با الگوریتم RR سرویس داده می‌شود و ۱۶ میلی‌ثانیه زمان CPU اضافی دریافت می‌کند.
    - اگر همچنان کامل نشده باشد، پیش‌گرفته شده و به صف Q2 منتقل می‌شود

Example of Multilevel Feedback Queue

- فرایندها (فرضی):

فرایند	زمان ورود(ms)	زمان اجرای کل(ms)
P1	0	25
P2	2	12
P3	4	35

P1(Q0)	P2(Q0)	P3(Q0)	P1(Q1)	P2(Q1)	P3(Q1)	P1(Q2)	P3(Q2)
0-8	8-16	16-24	24-40	40-44	44-60	60-61	61-72

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.

زمان بندی رشته یا ریسمان

- **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 معمولا 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 فرایند (PCS):**
- در این مدل، نخ‌های pthreadفقط **درون خود برنامه (process)** با هم رقابت می‌کنند.
- سیستم‌عامل تنها **تعدادی LWP فرآیند سبک‌وزن** به برنامه اختصاص می‌دهد.
- خود برنامه) با کمک کتابخانه‌ی (pthread تصمیم می‌گیرد که **کدام نخ کاربر** روی کدام LWPاجرا شود.
- سیستم‌عامل فقط **LWPها** را می‌شناسد، و هیچ اطلاعی از نخ‌های سطح کاربر pthread)ها ندارد.

زمان بندی رشته یا ریسمان

- **دامنه رقابت سیستم (SCS):**
  - در این روش، **تمام** رشته‌های موجود در سیستم‌عامل، برای استفاده از پردازنده با یکدیگر رقابت می‌کنند.
  - SCS معمولا 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 سیستم‌عامل (kernel) بر تمام رشته‌ها نظارت مستقیم دارد.(میداند چه رشته‌هایی وجود دارد و کدام باید اجرا شود)
  - بااین‌حال، SCS به دلیل لزوم **مدیریت مرکزی تمام رشته‌ها در سیستم**، می‌تواند پرهزینه‌تر از PCS باشد.
  - سیستم‌عامل‌هایی مانند لینوکس که از مدل یک‌به‌یک پشتیبانی می‌کنند، از **SCS** برای زمان‌بندی **رشته‌های هسته** (kernel-level threads) استفاده می‌کنند.

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

زمان بندی رشته یا ریسمان

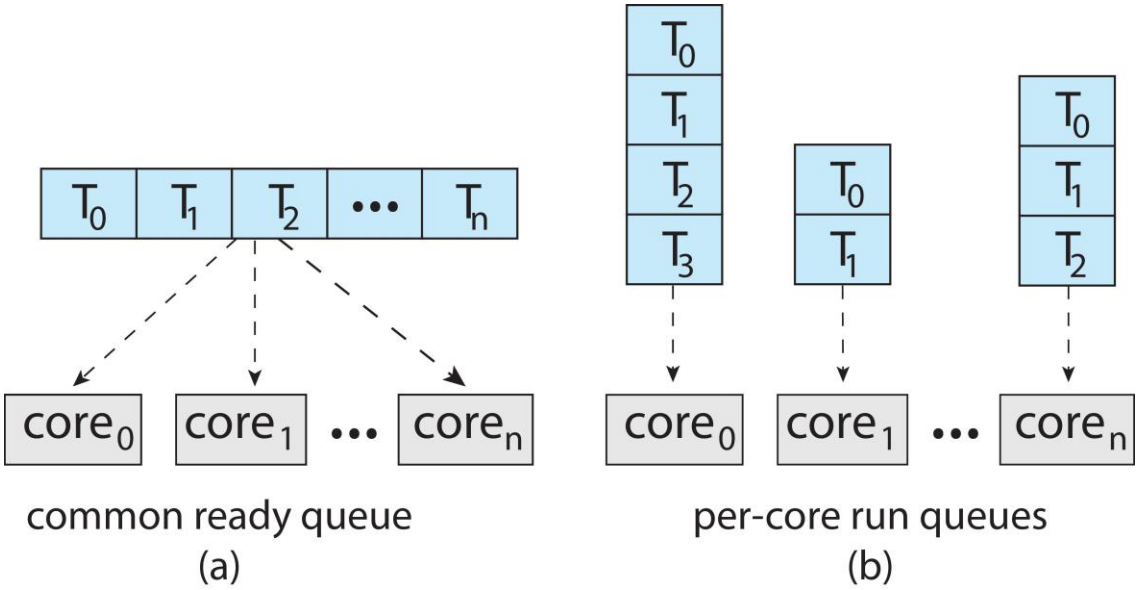
- رابط برنامه‌نویسی (API) به کاربر اجازه می‌دهد تا در هنگام ایجاد رشته، بین PCSیا SCS یکی را انتخاب کند:
  - PTHREAD_SCOPE_PROCESSبا استفاده از زمان‌بندیPCS ، رشته‌ها را زمان‌بندی می‌کند.
  - PTHREAD_SCOPE_SYSTEMبا استفاده از زمان‌بندیSCS ، رشته‌ها را زمان‌بندی می‌کند.
- این انتخاب ممکن است توسط سیستم عامل محدود شود. برای مثال، لینوکس و macOSفقط به PTHREAD_SCOPE_SYSTEM اجازه می‌دهند.

زمان بندی چند پردازنده ای

- **زمان‌بندی پردازنده در سیستم‌های چند پردازنده (CPU scheduling in Multiprocessor Systems)**
- زمان‌بندی CPU زمانی که چندین پردازنده در دسترس باشد، پیچیده‌تر می‌شود.
- مفهوم چند پردازنده می‌تواند شامل انواع مختلفی از معماری‌ها باشد:
  - پردازنده‌های چند هسته‌ای (Multicore CPUs)
  - هسته‌های چند رشته‌ای (Multithreaded cores)
  - سیستم‌های دسترسی غیر یکنواخت به حافظه NUMA
  - پردازش ناهمگن (Heterogeneous multiprocessing)

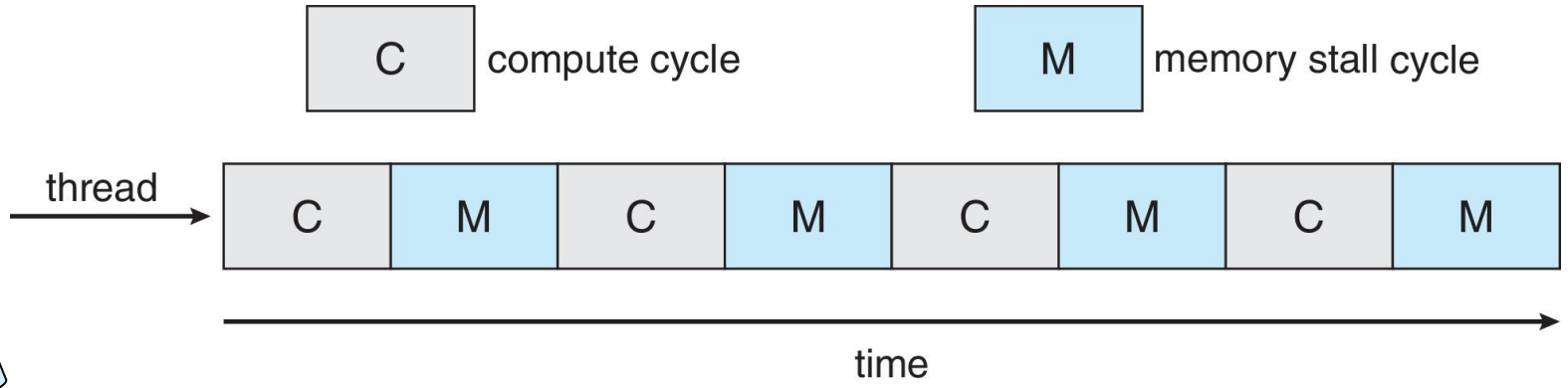
زمان بندی چند پردازنده ای

- **چندپردازنده متقارن (SMP)** حالتی است که در آن هر **پردازنده** به صورت **مستقل**، خود را **زمان‌بندی** می‌کند.
  - در این حالت، همه رشته‌ها می‌توانند در یک صف آماده‌ی مشترک قرار بگیرند. شکل الف
  - همچنین، هر پردازنده می‌تواند صف اختصاصی خود را برای رشته‌ها داشته باشد. شکل ب



پردازنده‌های چندهسته ای

- 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.

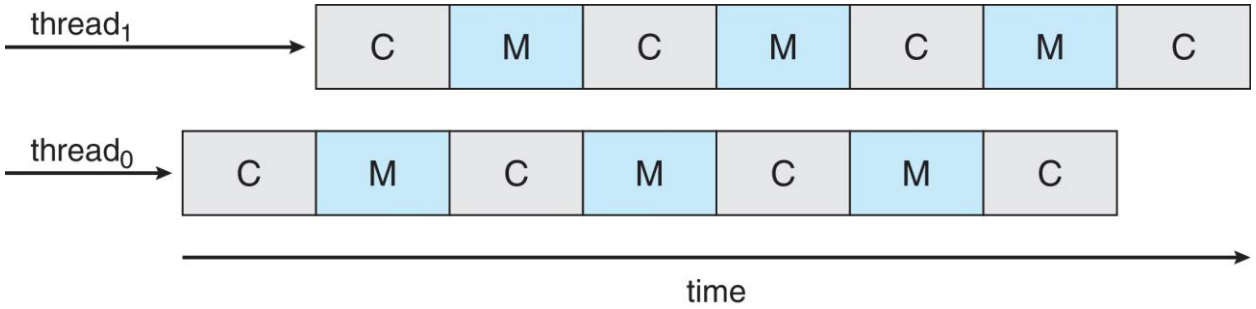


پردازنده‌های چندهسته ای

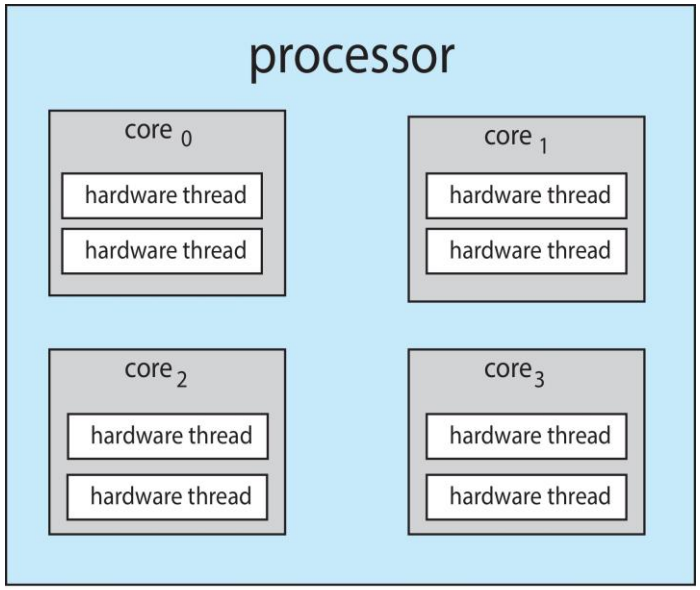
- تمایل اخیر این است که چندین هسته‌ی پردازنده را روی یک تراشه‌ی فیزیکی واحد قرار دهند.
- این روش باعث **افزایش سرعت** و **کاهش مصرف انرژی** می‌شود.
- استفاده از **چندین رشته در هر هسته** نیز رو به افزایش است.
  - این کار از **توقف موقت حافظه** (memory stall) برای پیشرفت در یک رشته‌ی دیگر در حین بازیابی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

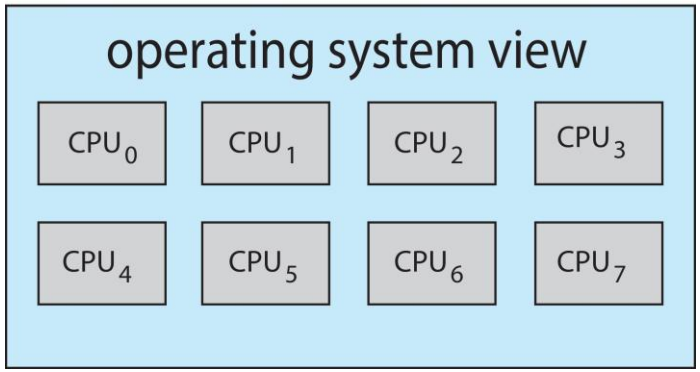
سیستم های چندنخی چندهسته ای

- هر هسته بیش از **یک رشته‌ی سخت‌افزاری** دارد.
- اگر یک رشته **دچار توقف موقت حافظه** شود، به یک رشته‌ی دیگر سوئیچ کنید!



سیستم های چندنخی چندهسته ای





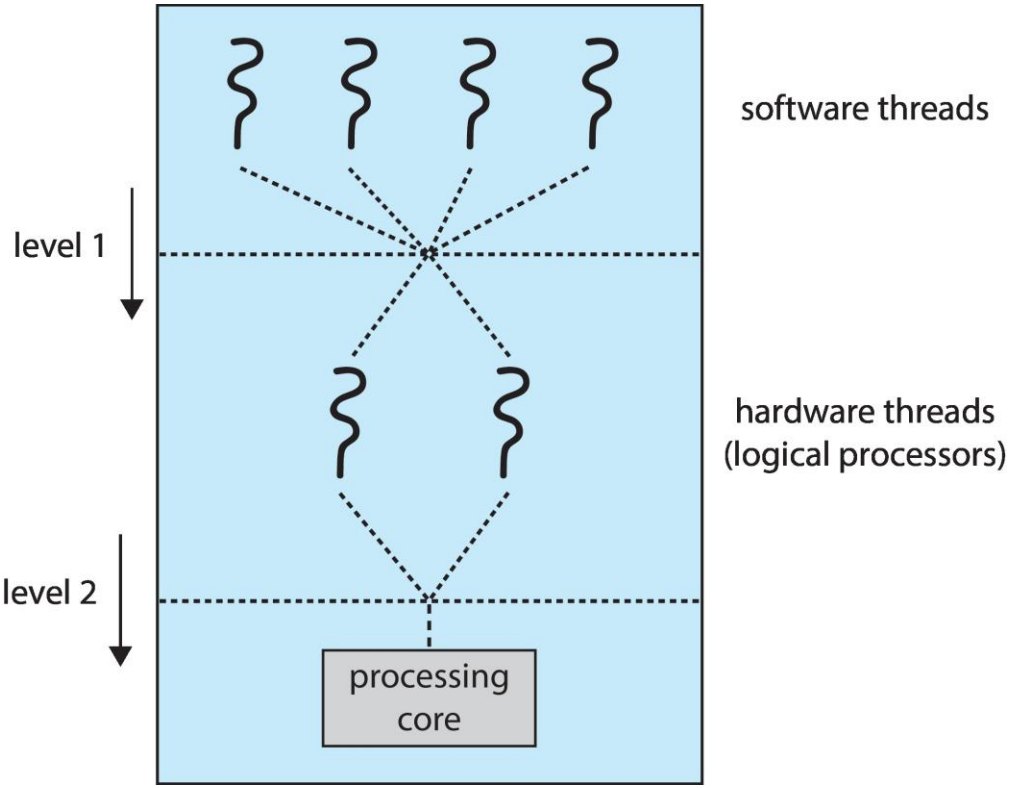
- چند رشته‌ای کردن تراشه

Chip-multithreading

- چندین رشته‌ی سخت‌افزاری را به هر هسته اختصاص می‌دهد. اینتل به این قابلیت، ابرنخی (Hyperthreading)می گوید .
- بر روی یک سیستم چهار هسته‌ای با ۲ رشته‌ی سخت‌افزاری به ازای هر هسته، سیستم عامل ۸ پردازنده‌ی منطقی را می‌بیند.

سیستم های چندنخی چندهسته ای

- دو سطح از زمان‌بندی وجود دارد:
  - سیستم عامل تصمیم می‌گیرد که کدام رشته‌ی نرم‌افزاری روی یک پردازنده منطقی یا نخ سخت‌افزاری اجرا شود.
  - هر هسته چطور تصمیم می‌گیرد کدام رشته‌ی سخت‌افزاری(پردازنده منطقی) را روی هسته‌ی فیزیکی اجرا کند.



زمان بندی چند پردازنده ای- توازن بار

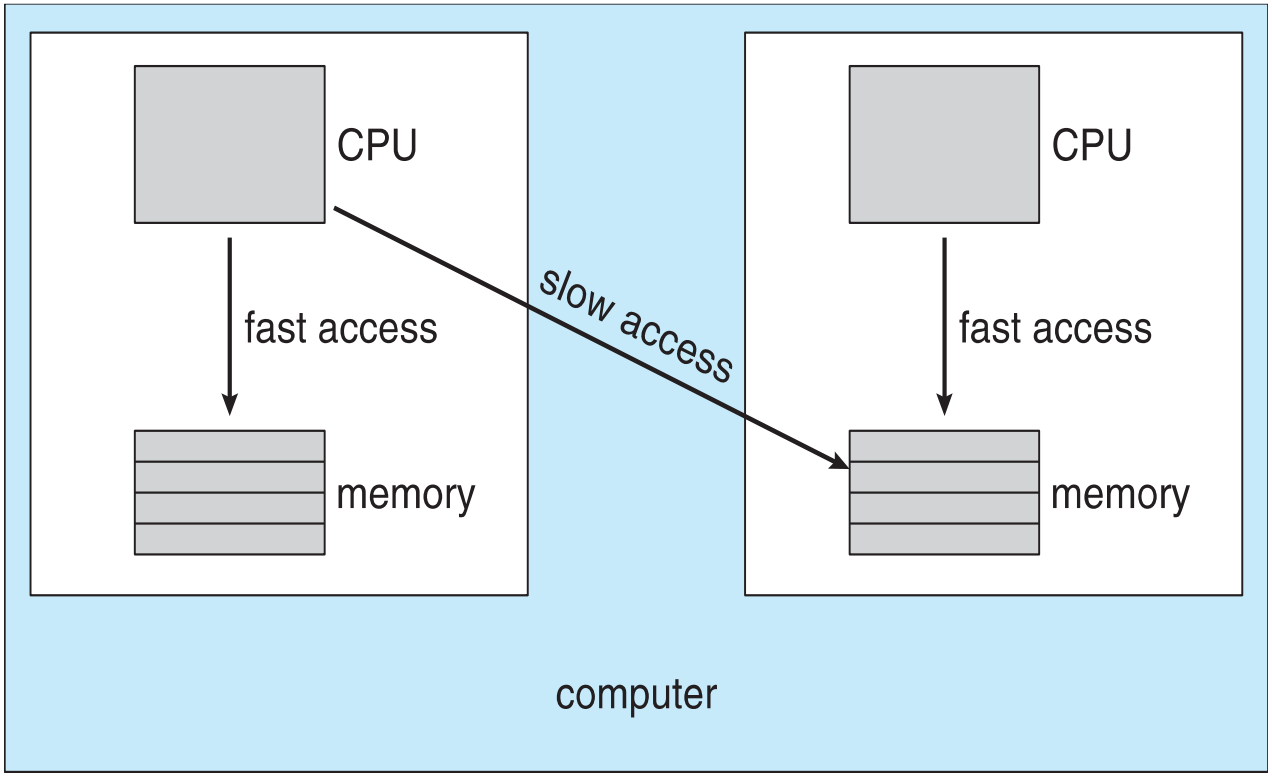
- در حالتSMP ، برای کارایی بهتر، باید همه‌ی پردازنده‌ها را لود نگه داشت.
- توازن بار (Load balancing) تلاش می‌کند تا حجم کاری به طور مساوی توزیع شود:
- مهاجرت فشاری :(Push migration) یک کار دوره‌ای بار روی هر پردازنده را بررسی می‌کند و در صورت مشاهده‌ی اضافه بار، وظیفه را از پردازنده پر بار به پردازنده های دیگر منتقل می‌کند.
- مهاجرت کششی :(Pull migration) پردازنده‌های بیکار، وظیفه‌ی منتظر مانده را از پردازنده‌ی شلوغ می‌کشند.

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 دسترسی های بعدی احتمالا 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 غیریکنواخت

- 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 زمان‌بندی آن‌ها وجود ندارد.
  - **سیستم‌های بلادرنگ سخت (Hard real-time systems):** یک وظیفه باید تا مهلت تعیین‌شده‌ی خود سرویس شود.

زمان بندی ویندوز

- 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 قابل اجرایی وجود نداشته باشد، یک رشته‌ی بیکار اجرا می‌شود.

ارزیابی الگوریتم

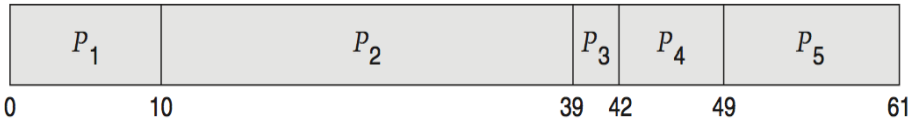
- **چگونه الگوریتم زمان‌بندی CPU را برای یک سیستم عامل انتخاب کنیم؟**
- معیارها را تعیین کنید، سپس الگوریتم‌ها را ارزیابی کنید.
- **مدل‌سازی قطعی(Deterministic modeling)**
  - نوعی ارزیابی تحلیلی است.
  - یک حجم کاری از پیش تعیین‌شده‌ی خاص را در نظر می‌گیرد و عملکرد هر الگوریتم را برای آن حجم کاری تعریف می‌کند.
- فرض کنید ۵ فرایند در زمان ۰ وارد می‌شوند:

Process	Burst Time
P_1	10
P_2	29
P_3	3
P_4	7
P_5	12

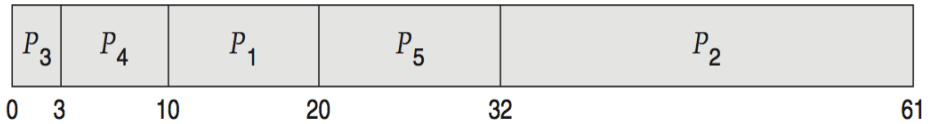
Deterministic Evaluation

- 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.

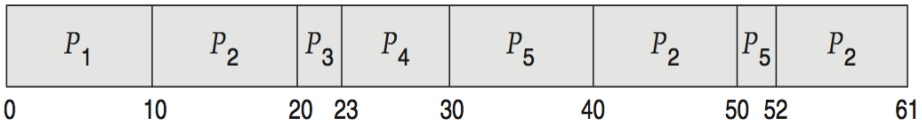
- FCS is 28ms:



- Non-preemptive SFJ is 13ms:



- RR is 23ms:



مدلهای صف کردن

- 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.

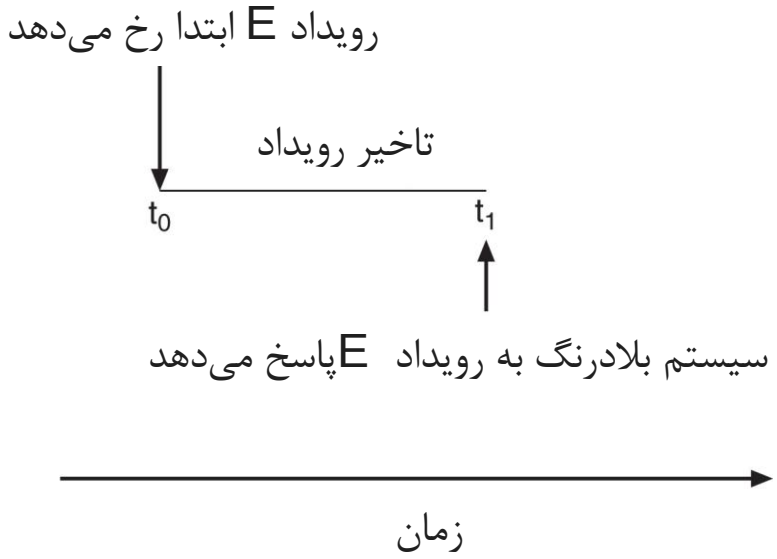
Little's Formula

- 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 می‌کنند باید برابر با تعداد فرایندهای ورودی باشد، بنابراین:
- $n = \lambda \times W$
- این رابطه برای هر الگوریتم زمان‌بندی 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.

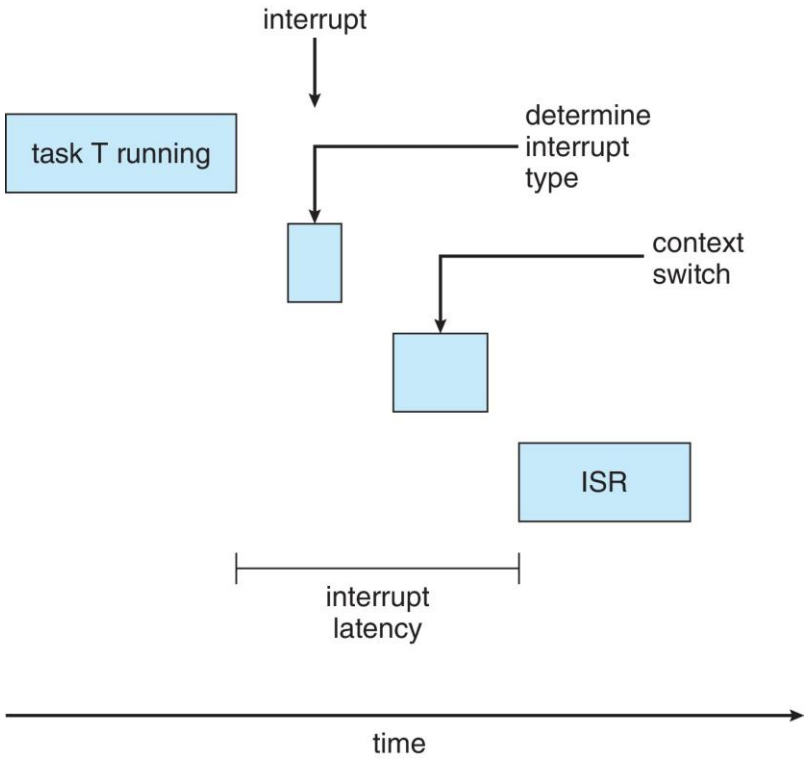
- پایان بخش اصلی فصل

زمان بندی برخط پردازنده

- **تأخیر رویداد (Event latency):**
  مدت زمانی که از وقوع یک رویداد تا زمان سرویس آن می‌گذرد.
- دو نوع تأخیر بر عملکرد تأثیر می‌گذارد:
- **تأخیر وقفه (Interrupt latency):**
  زمان از رسیدن وقفه تا شروع روتینی که سرویس‌دهنده‌ی وقفه است.
- **تأخیر زمان‌بندی (Dispatch latency):**
  زمان لازم برای اینکه زمان‌بند، فرایند جاری را از پردازنده خارج کند و به فرایند دیگری سوئیچ کند.

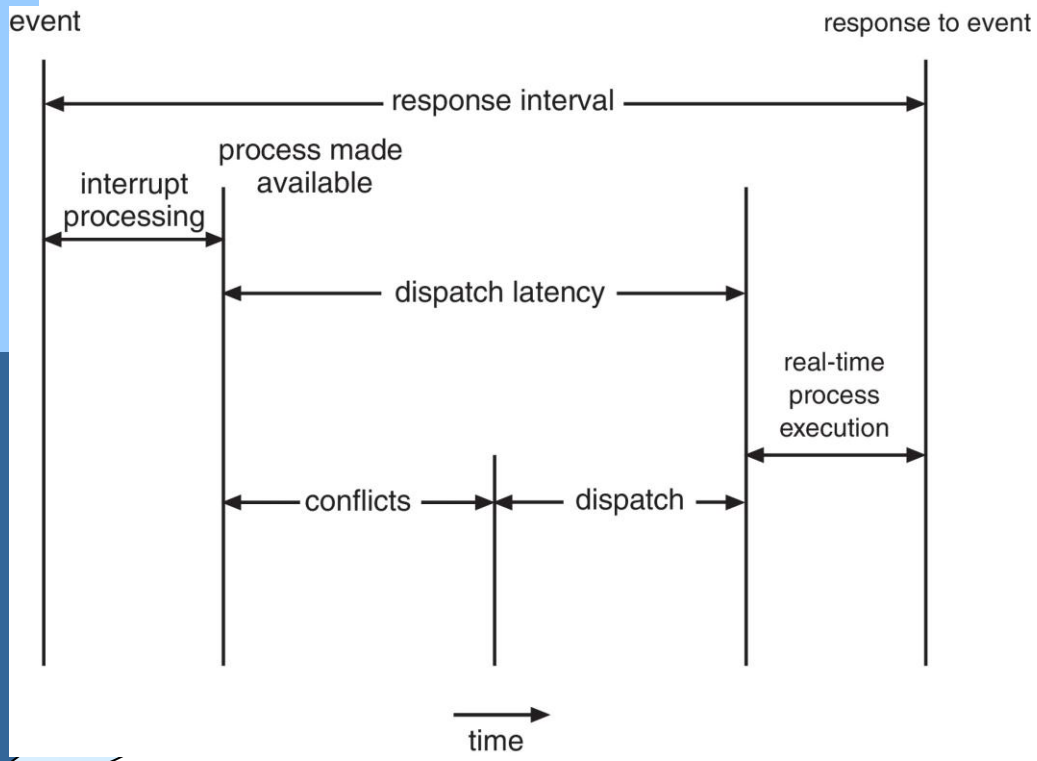


تاخیر وقعه


interrupt
task T running
determine
interrupt
type
context
switch
ISR
interrupt
latency
time

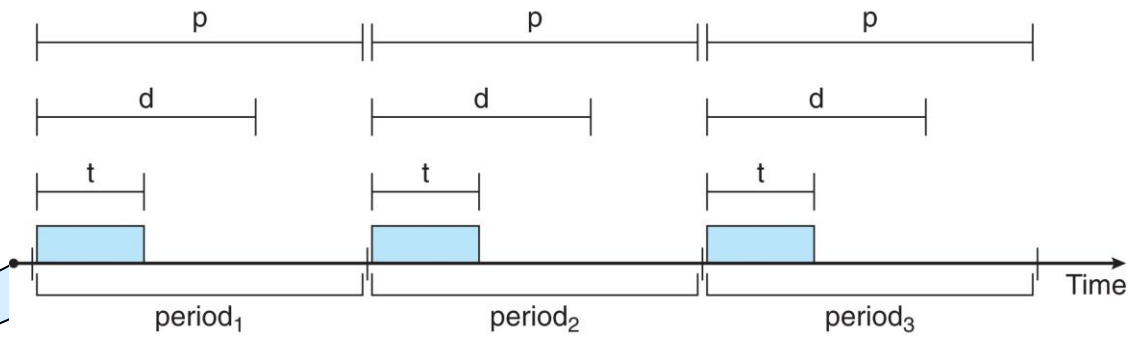
تاخیر توزیع

- فاز تضاد در تأخیر زمان‌بندی (Conflict phase of dispatch latency): دو جز دارد.
  - قبضه کردن هر فرایندی که در حالت هسته یا کرنل اجرا می‌شود.
  - آزاد شدن منابع فرایندهای با اولویت پایین توسط فرایندهای با اولویت بالا



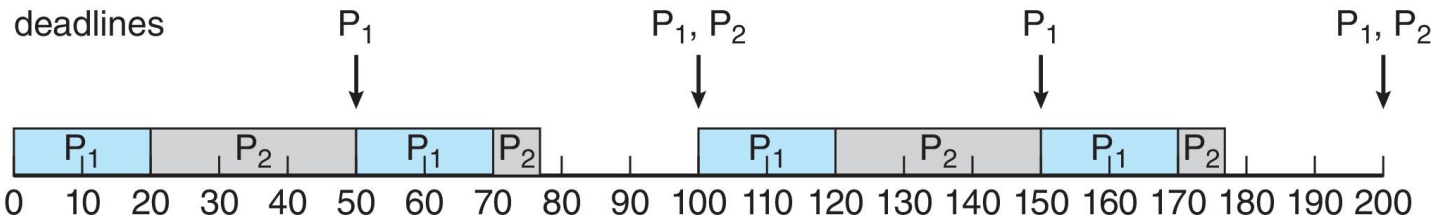
Priority-based Scheduling

- برای زمان‌بندی بلادرنگ، زمان‌بند باید از زمان‌بندی انحصاری مبتنی بر اولویت پشتیبانی کند.
  - اما این روش فقط بلادرنگ نرم را تضمین می‌کند.
- برای بلادرنگ سخت، همچنین باید توانایی رسیدن به مهلت‌های مقرر را فراهم کند.
- فرایندها دارای ویژگی‌های جدیدی هستند: فرایندهای دوره‌ای به فواصل زمانی ثابت به پردازنده نیاز دارند.
  - دارای زمان پردازش t ، مهلت مقرر d ، دوره‌ی زمانی p است.
  - $0 \leq t \leq d \leq p$
  - نرخ وظیفه‌ی دوره‌ای 1/p است.



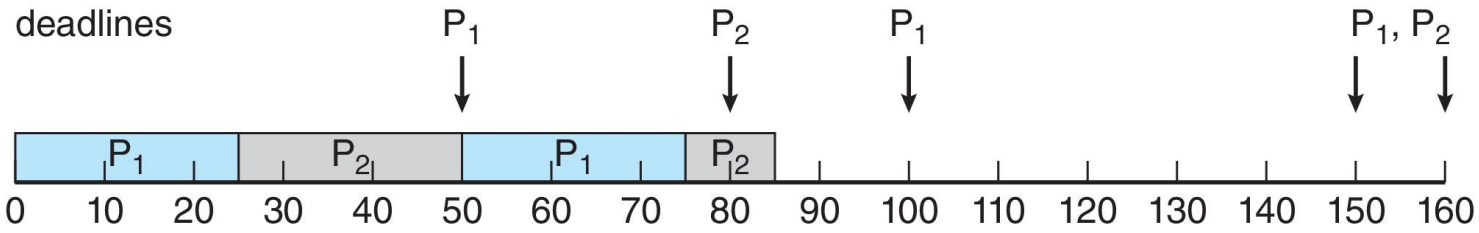
Rate Monotonic Scheduling

- بر اساس معکوس دوره‌ی آن، یک اولویت اختصاص داده می‌شود.
- دوره‌های کوتاه‌تر = اولویت بالاتر؛
- دوره‌های طولانی‌تر = اولویت پایین‌تر
- اولویت بالاتری به P1 نسبت به P2 اختصاص داده می‌شود.



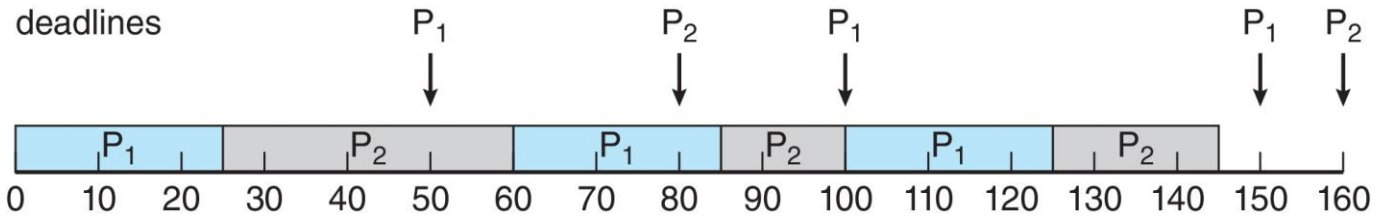
Missed Deadlines with Rate Monotonic Scheduling

- فرایند P2 در زمان ۸۰ مهلت مقرر خود را از دست می‌دهد.
- شکل



Earliest Deadline First Scheduling (EDF)

- اولویت‌ها بر اساس مهلت‌های مقرر اختصاص داده می‌شوند:
- هرچه مهلت مقرر زودتر باشد، اولویت بالاتر است.
- هرچه مهلت مقرر دیرتر باشد، اولویت پایین‌تر است.
- شکل



Proportional Share Scheduling

- **اشتراک‌گذاری T**
- **(T shares)**سهم‌هایی از زمان کل CPU بین تمام فرایندهای سیستم تخصیص داده می‌شود.
- یک برنامه N سهم دریافت می‌کند که در آن N < T است.
- این کار اطمینان می‌دهد که هر برنامه N/T از کل زمان پردازنده را دریافت کند.

POSIX Real-Time Scheduling

- **استاندارد POSIX.1b**
- این استاندارد یک رابط برنامه‌نویسی (API) برای مدیریت رشته‌های بلادرنگ ارائه می‌دهد.
- دو کلاس زمان‌بندی را برای رشته‌های زمان حقیقی تعریف می‌کند:
  - SCHED_FIFO - رشته‌ها با استفاده از یک استراتژی FCFS با یک صف FIFO زمان‌بندی می‌شوند. بر خلاف زمان‌بندی دوره‌ای، هیچ تکه‌بندی زمانی برای رشته‌های با اولویت برابر وجود ندارد.
  - SCHED_RR - مشابه SCHED_FIFO است، به جز این‌که تکه‌بندی زمانی برای رشته‌های با اولویت برابر رخ می‌دهد.
- همچنین این استاندارد دو تابع برای دریافت و تنظیم سیاست زمان‌بندی تعریف می‌کند:
  1. `pthread_attr_getsched_policy(pthread_attr_t *attr, int *policy)`
  2. `pthread_attr_setsched_policy(pthread_attr_t *attr, int policy)`

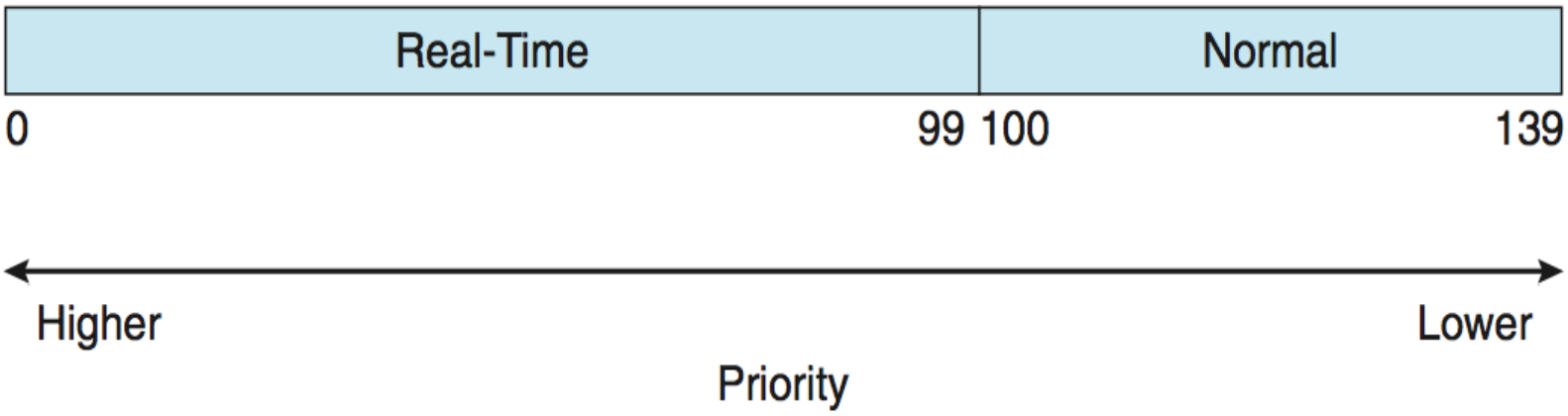
Operating System Examples

- Linux scheduling
- Windows scheduling
- Solaris scheduling

Linux Scheduling (Cont.)

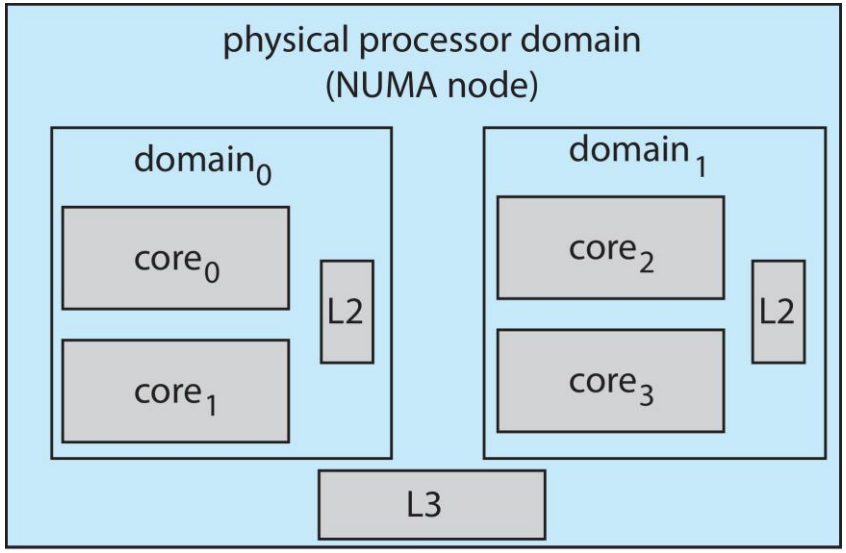
زمان‌بندی زمان حقیقی بر اساس POSIX.1b (Real-time scheduling according to POSIX.1b)

- وظایف بلادرنگ دارای اولویت‌های استاتیک هستند.
- اولویت‌های بلادرنگ به همراه اولویت‌های عادی در یک طرح اولویت کلی نگاشت می‌شوند.
- مقدار nice برابر با ۲۰- به اولویت کلی ۱۰۰ نگاشت می‌شود.
- مقدار nice برابر با ۱۹+ به اولویت ۱۳۹ نگاشت می‌شود.



Linux Scheduling (Cont.)

- هسته‌ی لینوکس از توازن بار پشتیبانی می‌کند، اما همچنین از NUMA (دسترسی غیر یکنواخت به حافظه) آگاه است.
- **دامنه‌ی زمان‌بندی (Scheduling domain):** مجموعه‌ای از هسته‌های CPU است که می‌توان آن‌ها را در برابر یکدیگر متعادل کرد.
- دامنه‌ها بر اساس اشتراک هایشان (به عنوان مثال، حافظه‌ی کش) سازماندهی می‌شوند.
- هدف این است که از مهاجرت رشته‌ها بین دامنه‌ها جلوگیری شود.



Windows Priority Classes

- Win32 APIدر چندین کلاس اولویت را تعریف می‌کند که یک فرایند می‌تواند به آن‌ها تعلق داشته باشد
- REALTIME_PRIORITY_CLASS, HIGH_PRIORITY_CLASS, ABOVE_NORMAL_PRIORITY_CLASS,NORMAL_PRIORITY_CLASS, BELOW_NORMAL_PRIORITY_CLASS, IDLE_PRIORITY_CLASS
  - به جز کلاسREALTIME ، همه‌ی این موارد متغیر هستند.
    - یک رشته‌ی درون یک کلاس اولویت خاص، یک اولویت نسبی دارد:
  - TIME_CRITICAL, HIGHEST, ABOVE_NORMAL, NORMAL, BELOW_NORMAL, LOWEST, IDLE
  - کلاس اولویت و اولویت نسبی با هم ترکیب می‌شوند تا اولویت عددی را به دست دهند.
  - اولویت پایه در یک کلاس، مقدار NORMAL است.
  - اگر کوانتوم منقضی شود، اولویت کاهش می‌یابد، اما هرگز از مقدار پایه پایین‌تر نمی‌رود.

Windows Priority Classes (Cont.)

- اگر انتظار رخ دهد، اولویت بسته به اینکه برای چه چیزی انتظار کشیده شده است، افزایش می‌یابد.
- به پنجره‌ی پیش‌زمینه، ضریب افزایش اولویت ۳ برابری داده می‌شود.
- ویندوز ۷ زمان‌بندی حالت کاربر (UMS - User-Mode Scheduling) را اضافه کرد.
- برنامه‌ها به صورت مستقل از هسته، رشته‌ها را ایجاد و مدیریت می‌کنند.
- برای تعداد زیادی از رشته‌ها، بسیار کارآمدتر است.
- زمان‌بندی‌گرهای UMS از کتابخانه‌های زبان‌های برنامه‌نویسی مانند فریمورک C++ Concurrent Runtime (ConcRT)می‌آیند.

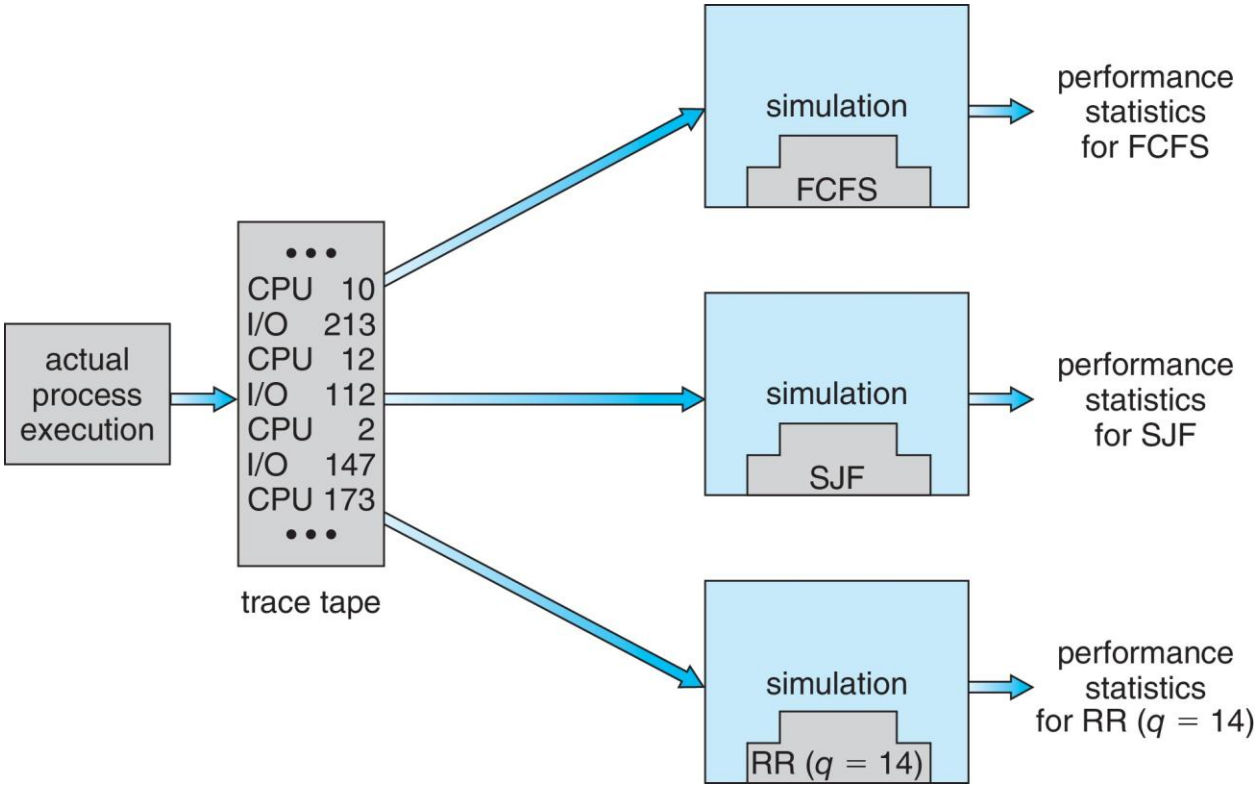
Windows Priorities

	real-time	high	above normal	normal	below normal	idle priority
time-critical	31	15	15	15	15	15
highest	26	15	12	10	8	6
above normal	25	14	11	9	7	5
normal	24	13	10	8	6	4
below normal	23	12	9	7	5	3
lowest	22	11	8	6	4	2
idle	16	1	1	1	1	1

Simulations

- محدودیت‌های مدل‌های صف‌بندی
- دقت مدل‌های صف‌بندی محدود است. روش‌های شبیه‌سازی (Simulations)از دقت بالاتری برخوردار هستند.
- **شبیه‌سازی(Simulations)**
- شبیه‌سازی یک مدل برنامه‌ریزی‌شده از یک سیستم کامپیوتری است.
- **ساعت شبیه‌سازی :**یک متغیر است که زمان را شبیه‌سازی می‌کند.
- **گردآوری آمار :**برای نشان دادن عملکرد الگوریتم، آمار جمع‌آوری می‌شود.
- **داده‌های ورودی به شبیه‌سازی:**
  - از طریق تولیدکننده‌ی اعداد تصادفی بر اساس احتمالات به دست می‌آید.
  - توزیع‌ها به صورت ریاضی یا تجربی تعریف می‌شوند.
  - Trace tapeها توالی رویدادهای واقعی را در سیستم‌های واقعی ثبت می‌کنند.

Evaluation of CPU Schedulers by Simulation



پیاده سازی

حتی شبیه‌سازی‌ها نیز از دقت محدودی برخوردار هستند.

بهترین راهکار برای ارزیابی یک الگوریتم زمان‌بندی، پیاده‌سازی آن در سیستم واقعی و تست کردن آن است.

- اما این روش پرهزینه و پرخطر است.
- محیط‌های مختلف می‌توانند بر عملکرد الگوریتم تأثیر بگذارند.

برخی از زمان‌بندی‌ها انعطاف‌پذیر را می‌توان به صورت اختصاصی برای هر سایت یا هر سیستم اصلاح کرد.

همچنین ممکن است API هایی برای تغییر اولویت‌ها وجود داشته باشد.

اما باز هم باید توجه داشت که محیط‌های مختلف می‌توانند بر عملکرد الگوریتم تأثیر بگذارند.

پایان فصل زمان بندی پردازنده