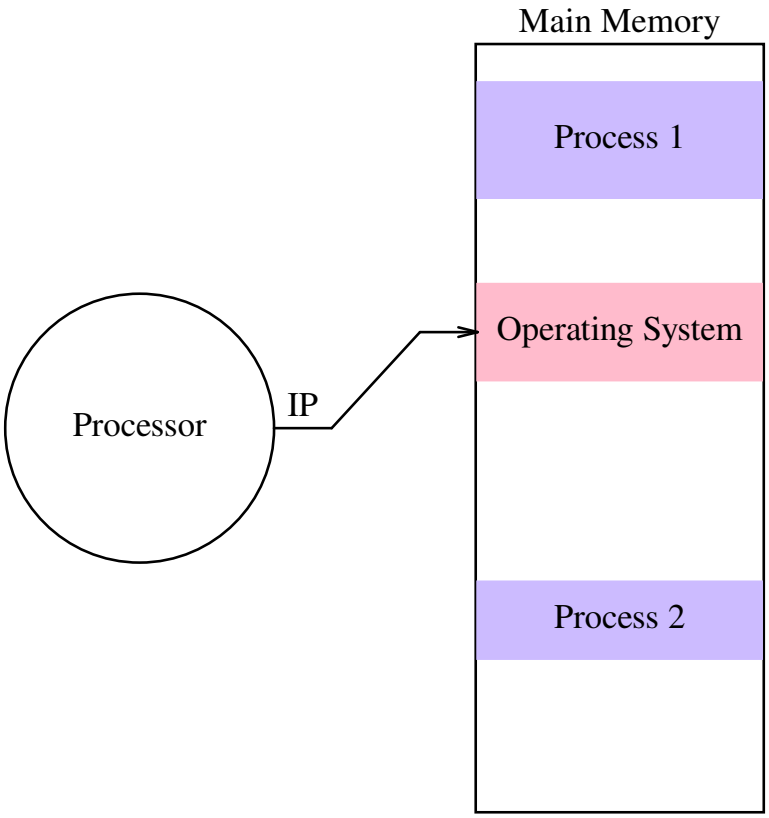
یادداشت‌های درس سیستم‌های عامل - بخش هفتم

در این بخش از درس زمانبندی پردازنده در سیستم عامل را مطالعه می‌کنیم و الگوریتم‌های پایه‌ی زمانبندی پردازنده را بررسی می‌نماییم.

- زمانبند پردازنده: تعیین می‌کند از بین پردازه‌هایی که آماده‌ی اجرا هستند پردازنده به چه پردازه‌ای و به چه مدتی داده شود.
- الگوریتم‌های متفاوتی برای زمانبندی پردازنده موجود هستند.

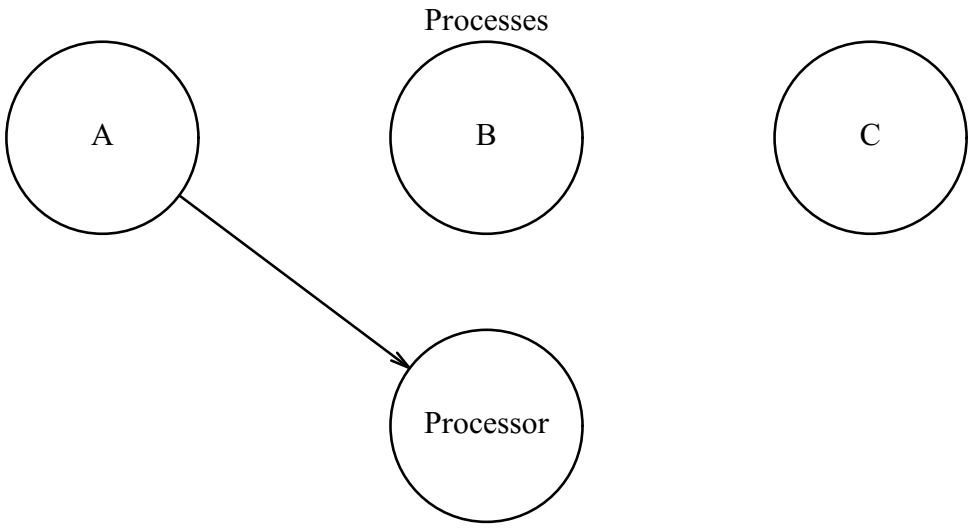


- هر پردازه در زمان اجرایش یا به پردازنده احتیاج دارد یا یک عمل ورودی یا خروجی انجام می‌دهد و منتظر آن می‌ماند.
- به بازه‌هایی که پردازه در آن پردازش انجام می‌دهد CPU Burst و به بازه‌هایی که پردازه در آن منتظر کامل شدن عمل ورودی یا خروجی هست I/O Burst گفته می‌شود.
- هر پردازه در طول عمر خود، به صورت متناوب در CPU Burst و I/O Burst قرار می‌گیرد.

```
CPU Burst (3ms)
I/O Burst
CPU Burst (20ms)
I/O Burst
CPU Burst (5ms)
...
```

- در بسیاری از پردازه‌ها، طول CPU Burst بسیار کوتاه و کمتر از ده میلی‌ثانیه هست.
- برای نمونه در یک برنامه‌ی محاوره‌ای، پردازه منتظر فشار دادن دکمه‌ها توسط کاربر می‌شود، در فایل‌ها می‌نویسد، اطلاعاتی را از شبکه انتقال می‌دهد، از کارت صدا پخش می‌کند؛ پس از هر یک از این عملیات ورودی یا خروجی (I/O Burst)، پردازه زمانی در CPU Burst قرار می‌گیرد.

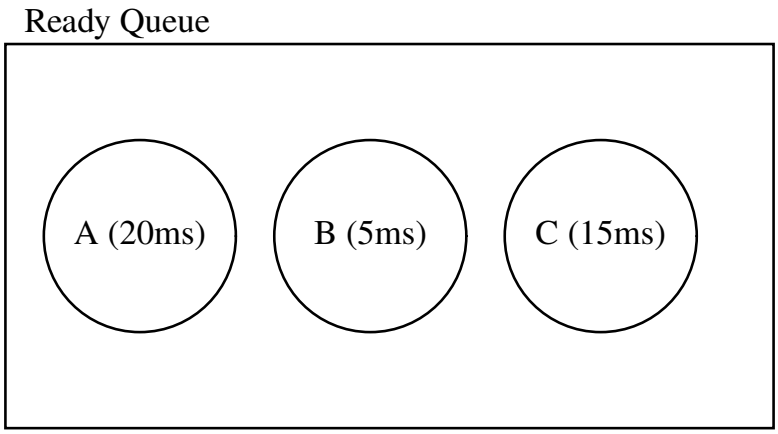
- سیستم‌های عامل قدیمی توانایی اجرای فقط یک پردازه را در هر لحظه داشتند.
- در نتیجه، اگر پردازه به دلیل انجام یک عمل ورودی یا خروجی (مثلا خواندن از دیسک یا دریافت اطلاعات از کارت شبکه) منتظر می‌شد، پردازنده بیکار می‌ماند.
- برای استفاده بهتر از پردازنده، سیستم‌های عامل با ویژگی چند برنامگی (Multiprogramming) طراحی شدند که توانایی اجرای چند پردازه را به صورت همزمان داشتند.



- اگر پردازه‌ی در حال اجرا، با انجام یک عمل ورودی یا خروجی منتظر شود، سیستم عامل تعویض متن انجام می‌دهد و پردازنده را در اختیار پردازه‌ی دیگری که آماده‌ی اجرا هست قرار می‌دهد.
- به این صورت با وجود چند برنامگی از پردازنده به شکل بهتری استفاده می‌شود.
- برای اینکه پردازنده در بازه‌های کوتاهی به هر پردازه داده شود لازم است همه‌ی پردازه‌ها پس از مدتی پردازش، داوطلبانه منتظر شوند تا پردازنده در اختیار سایر پردازه‌ها قرار گیرد. به همین دلیل به چند برنامگی وظیفه‌ای تعاونی یا مبتنی بر همکاری (Cooperative Multitasking) نیز گفته می‌شود.
- قبلا دیده‌ایم که در سیستم‌های عامل اشتراک زمانی (Time-sharing) چگونه زمان اجرای هر پردازه به کمک وقفه‌های سخت‌افزاری (به کمک Timer) محدود می‌شود. به سیستم‌های عامل اشتراک زمانی گاهی چند وظیفه‌ای قبضه‌ای (Preemptive Multitasking) هم گفته می‌شود.

ورودی زمانبند پردازنده

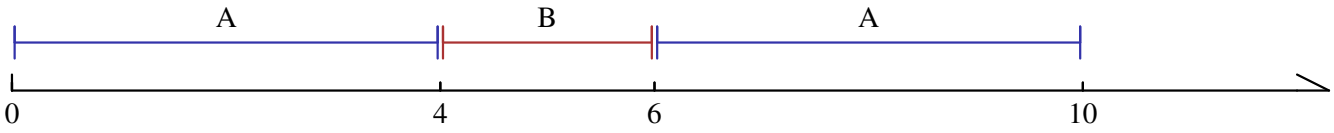
- در هر لحظه تعدادی از پردازه‌های موجود در یک سیستم عامل در حالت آماده‌ی اجرا (Ready) قرار دارند. این پردازه‌ها در صف آماده‌باش (Ready Queue) قرار می‌گیرند.
- هر پردازه‌ای که در حالت آماده‌ی اجرا قرار می‌گیرد باید به مدت مشخصی روی پردازنده‌ای اجرا شود.



- بنابراین، هر پردازه در شروع CPU Burst در صف آماده باش قرار می‌گیرد و تا پایان آن در آن باقی می‌ماند.
- زمانبندی پردازنده در سیستم عامل مشخص می‌کند پردازنده به کدام پردازه‌ی آماده‌ی اجرا داده شود.
- سیستم عامل معمولا طول CPU Burst را نمی‌داند.

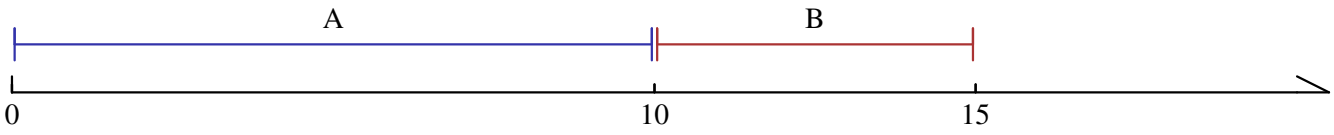
نمودار گانت

- در ادامه از نمودار گانت (Gantt) برای نمایش پردازه‌ی در حال اجرا روی پردازنده در طول زمان استفاده می‌کنیم.
- در نمودار گانت، محور زمان را نشان می‌دهد.
- نمودار زیر را در نظر بگیرید.

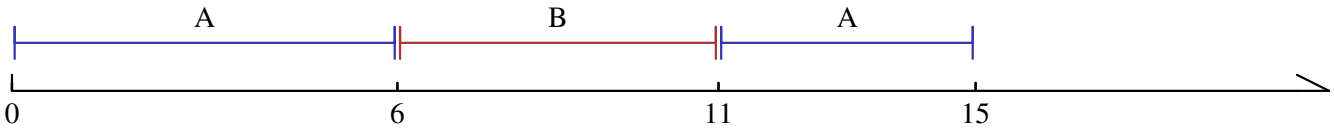


انواع الگوریتم‌های زمانبندی

- الگوریتم‌های زمانبندی به دو دسته تقسیم می‌شوند: الگوریتم‌های Preemptive (قبضه‌ای) و الگوریتم‌های Nonpreemptive (بدون قبضه).
- مصدر Preemption یعنی (به زور) گرفتن و Preemptive یعنی الگوریتمی که در آن پردازنده به زور از پردازه گرفته می‌شود.
- در الگوریتم‌های Nonpreemptive اگر پردازنده در اختیار پردازه‌ای قرار گیرد، سیستم عامل پردازنده را از آن پردازه نمی‌گیرد تا CPU Burst آن پردازه تمام شود (کاری که در چند برنامگی انجام می‌شد).
- در الگوریتم‌های Preemptive سیستم عامل می‌تواند قبل از تمام شده CPU Burst پردازه‌ای که پردازنده را در اختیار دارد، تعویض متن انجام دهد و پردازنده را در اختیار پردازه‌ی دیگری قرار دهد.
- برای مثال فرض کنید دو پردازه در صف آماده باش حاضر باشند. پردازه‌ی A با طول CPU Burst ده میلی‌ثانیه و پردازه‌ی B با طول CPU Burst پنج میلی‌ثانیه.
- اگر از یک الگوریتم زمانبندی Nonpreemptive استفاده شود و پردازنده در زمان صفر در اختیار پردازه‌ی A قرار گیرد، باید CPU Burst این پردازه تمام شود تا پردازنده را به پردازه‌ی B داد (شکل زیر).



- اما در یک الگوریتم زمانبندی Preemptive می‌توان زمانی که CPU Burst پردازه‌ی A هنوز تمام نشده است، پردازنده را از این پردازه گرفت و به پردازه‌ی دیگر داد و تا پردازه‌ی A بعدا اجرای خود را ادامه دهد (مانند شکل زیر).



- در ادامه چند معیار برای ارزیابی الگوریتم‌های زمانبندی را معرفی می‌کنم.

الف) CPU Load

ب) Throughput

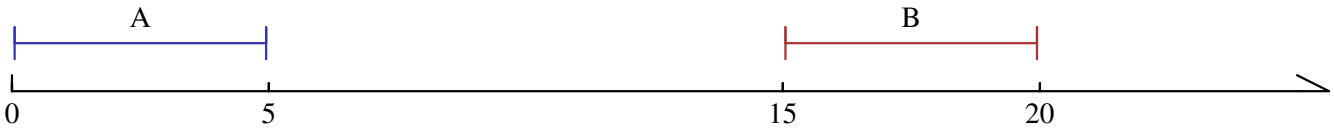
ج) Turnaround Time

د) Waiting Time

ه) Response Time

بار کاری پردازنده

- بار کاری پردازنده (CPU Utilization یا CPU Load): نشـان می‌دهد چه درصدی از زمان، پردازنـده در حال اجرای پردازه‌های کاربر بوده است.
- برای نمونه، نمودار زیر را در نظر بگیرید.

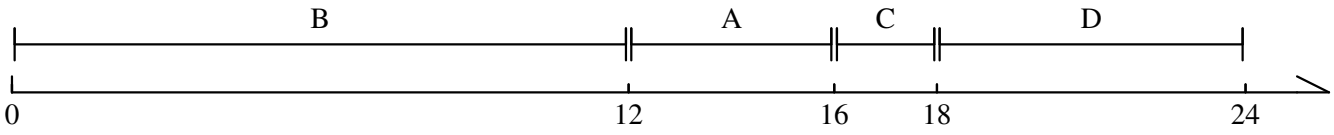


- در بازه‌ی زمانی صفر تا بیست میلی‌ثانیه پردازنده چند درصد مواقع مشغول بوده است؟
- در الگوریتم‌هایی که در درس می‌بینیـم بار کـاری صد در صد اسـت مگر اینکـه پردازه‌ی آمـاده‌ی اجرایی مـوجود نباشد.

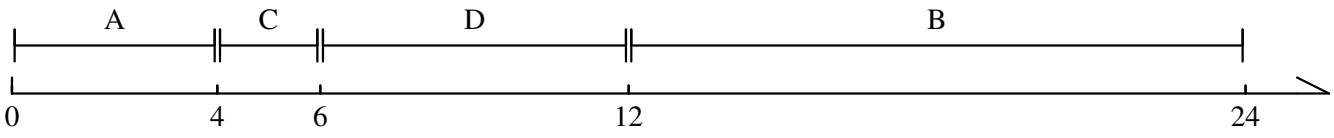
- توان عملیاتی یا خروجی (Throughput): تعداد پردازه‌هایی که در واحد زمان از صف آماده‌باش خارج می‌شوند.
- فرض کنید چهار پردازه‌ی زیر در صف زمانبندی قرار داشته باشند.

Process	CPU Burst
A	4ms
B	12ms
C	2ms
D	6ms

- بازه‌ی زمانی صفر تا پانزده میلی‌ثانیه در زمانبندی زیر را در نظر بگیرید. در این بازه، CPU Burst یک پردازه خاتمه می‌یابد. بنابراین در این بازه‌ی زمانی توان عملیاتی این الگوریتم زمانبندی ۰/۰۷ پردازه در میلی‌ثانیه است.



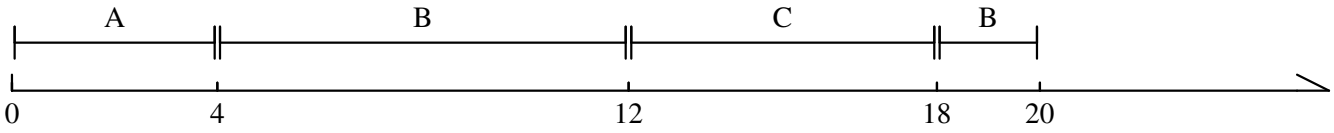
- بازه‌ی زمانی صفر تا پانزده میلی‌ثانیه در زمانبندی زیر را در نظر بگیرید. در این بازه، سه صف آماده باش را ترک می‌کنند. بنابراین در این بازه توان عملیاتی این الگوریتم زمابندی ۰/۲ پردازه در میلی‌ثانیه است.



- ج) زمان رفت و برگشت (Turnaround Time): فاصله‌ی زمانی بین زمان ورود پردازه به صف آماده‌باش تا خاتمه‌ی CPU Burst آن.
- سه پردازه‌ی زیر را در نظر بگیرید. ستون آخر زمانی است که پردازه وارد صف آماده‌باش شده است.

Process	CPU Burst	Arrival
A	4ms	0ms
B	10ms	2ms
C	6ms	4ms

- زمانبندی زیر را در نظر بگیرید.

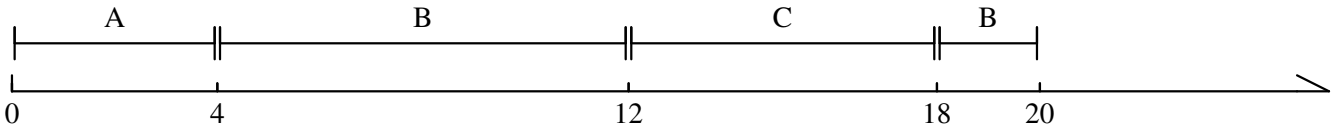


- پردازه‌ی A در زمان صفر وارد شده است و در زمان چهار میلی‌ثانیه CPU Burst خودش را اجرا کرده است. بنابراین زمان رفت و برگشت این پردازه چهار میلی‌ثانیه است.
- پردازه‌ی B در زمان دو میلی‌ثانیه وارد شده است و در زمان بیست میلی‌ثانیه CPU Burst خودش را اجرا کرده است. بنابراین زمان رفت و برگشت این پردازه هجده میلی‌ثانیه است.
- پردازه‌ی C در زمان چهار میلی‌ثانیه وارد شده است و در زمان هجده میلی‌ثانیه CPU Burst خودش را اجرا کرده است. بنابراین زمان رفت و برگشت این پردازه چهارده میلی‌ثانیه است.

- د) زمان انتظار (Waiting Time): مجموع زمانی که پردازه در صف آماده‌باش منتظر بوده است.
- سه پردازه‌ی زیر را در نظر بگیرید.

Process	CPU Burst	Arrival
A	4ms	0ms
B	10ms	2ms
C	6ms	4ms

- زمانبندی زیر را در نظر بگیرید.

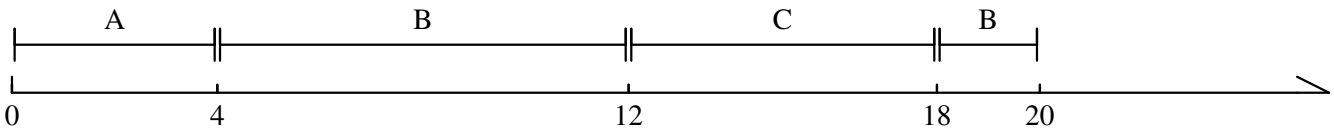


- پردازه‌ی A در زمان اجرا هیچگاه منتظر نشده است. بنابراین زمان انتظار این پردازه صفر میلی‌ثانیه است.
- پردازه‌ی B در زمان دو میلی‌ثانیه قبل از شروعش و شش میلی‌ثانیه از زمان دوازده میلی‌ثانیه منتظر شده است. بنابراین زمان انتظار این پردازه هشت میلی‌ثانیه هست.
- پردازه‌ی C در زمان چهار میلی‌ثانیه وارد شده است و قبل از اجرا هشت میلی‌ثانیه منتظر شده است. بنابراین زمان انتظار این پردازه هشت میلی‌ثانیه است.
- زمان انتظار را می‌توان از تفریق CPU Burst از زمان رفت و برگشت محاسبه کرد.

- ه) زمان پاسخ (Response Time): فاصله‌ی زمانی که یک پردازه وارد صف آماده‌باش می‌شود تا وقتی که بتواند اولین پاسخ خودش را تولید کند (پردازنده در اختیارش قرار بگیرید).
- در برنامه‌های محاوره‌ای (مثل یک ویرایشگر متن) زمان پاسخ اهمیت دارد؛ اگر زمان پاسخ زیاد باشد، کاربر ناراضی می‌شود (برای مثال فرض کنید پاسخ کلیک‌ها در یک ویرایشگر متن یا یک بازی با یک ثانیه تأخیر ایجاد شود).
- سه پردازه‌ی زیر را در نظر بگیرید.

Process	CPU Burst	Arrival
A	4ms	0ms
B	10ms	2ms
C	6ms	4ms

- زمانبندی زیر را در نظر بگیرید.

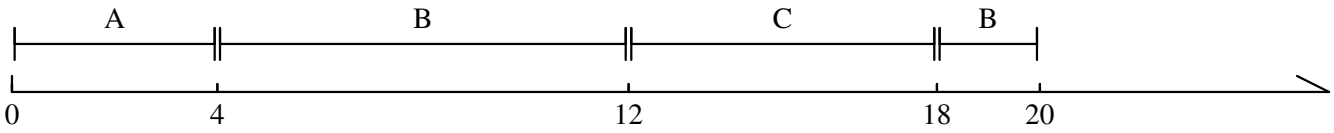


- فاصله‌ی زمانی ورود پردازه‌ی A تا زمانی که پردازنده برای اولین بار در اختیارش گذاشته شده است صفر میلی‌ثانیه است در نتیجه زمان پاسخ این پردازه صفر میلی‌ثانیه است.
- فاصله‌ی زمانی ورود پردازه‌ی B تا زمانی که پردازنده برای اولین بار در اختیارش گذاشته شده است و در نتیجه زمان پاسخ این پردازه دو میلی‌ثانیه است.
- فاصله‌ی زمانی ورود پردازه‌ی C تا زمانی که پردازنده برای اولین بار در اختیارش گذاشته شده است و در نتیجه زمان پاسخ این پردازه هشت میلی‌ثانیه است.

- سه پردازه‌ی زیر را در نظر بگیرید.

Process	CPU Burst	Arrival	Turnaround T.	Waiting T.	Response T.
A	4ms	0ms	4ms	0ms	0ms
B	10ms	2ms	18ms	8ms	2ms
C	6ms	4ms	14ms	8ms	8ms

- در جدول بالا مقدار سه معیار زمان رفت و برگشت، زمان انتظار و زمان پاسخ را برای زمانبندی زیر می‌بینید.



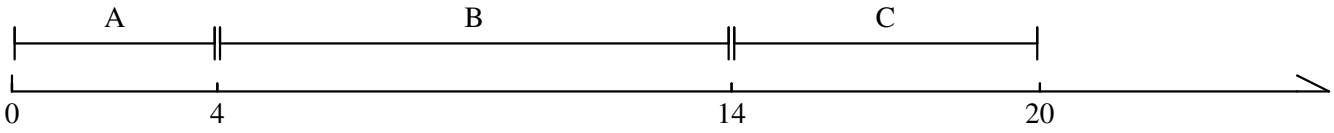
- بدیهی است که هر چه زمان این سه معیار کوچک‌تر باشد الگوریتم زمانبندی بهتر عمل می‌کند.

- در ادامه چند الگوریتم زمانبندی پایه را بررسی می‌کنیم.

- در الگوریتم First-Come First-Served (FCFS) پردازه‌ای که زودتر به صف آماده‌باش وارد شود زودتر پردازش می‌شود.
- این الگوریتم یک الگوریتم Nonpreemptive هست و پردازه‌ها در نوبت خودشان CPU Burst را کامل اجرا می‌کنند.
- سه پردازه‌ی زیر را در نظر بگیرید.

Process	CPU Burst	Arrival
A	4ms	0ms
B	10ms	2ms
C	6ms	4ms

- زمانبندی زیر خروجی الگوریتم First-Come First-Served هست.

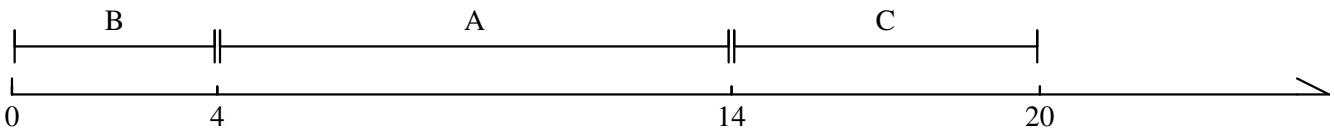


- متوسط زمان انتظار سه پردازه در این مثال ۴/۶۷ میلی‌ثانیه هست.
- با تغییر ترتیب اجرای پردازه‌ها می‌توان متوسط زمان انتظار را کاهش داد (چگونه؟).

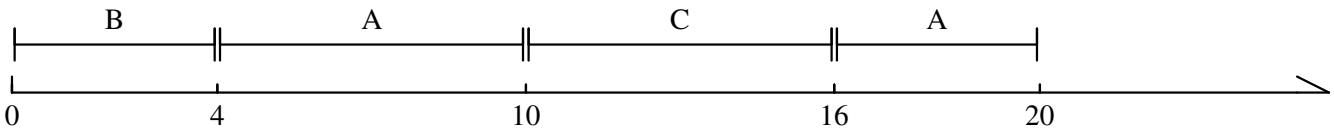
- در الگوریتم زمانبندی اولویت (Priority) هر پردازه‌ی ورودی یک مقدار اولویت نیز دارد.
- این الگوریتم پردازنده را در اختیار پردازه‌ای قرار می‌دهد که بالاترین اولویت را داشته باشد.
- این الگوریتم دو حالت Preemptive و Nonpreemptive دارد.
- گاهی اولویت را با یک عدد نمایش می‌دهند؛ در برخی از سیستم‌های عامل هر چه عدد اولویت کوچک‌تر باشد، اولویت پردازه بیشتر است.
- مثال زیر را در نظر بگیرید. ستون آخر اولویت پردازه‌ها را مشخص می‌کند. فرض کنید مقدار اولویت کمتر در این مثال به مفهوم اولویت بیشتر باشد.

Process	CPU Burst	Arrival	Priority
A	10ms	0ms	2
B	4ms	0ms	1
C	6ms	10ms	1

- نمودار زیر خروجی الگوریتم را در حالت Nonpreemptive نشان می‌دهد.



- تفاوت الگوریتم در حالت Nonpreemptive و حالت Preemptive در زمان ورود پردازه‌های جدید است.
- نمودار زیر خروجی الگوریتم را در حالت Preemptive نشان می‌دهد.

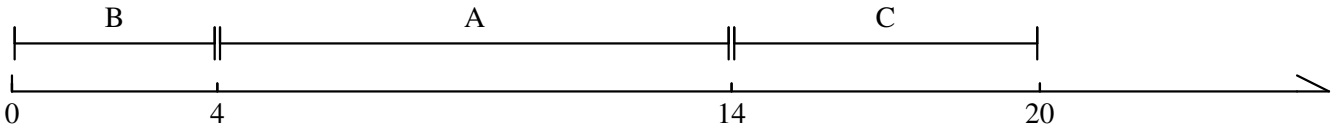


- فرض کنید اولویت پردازه‌ای در سیستم عامل نسبت به سایرین کم باشد.
- در این صورت، امکان دارد همواره پردازه‌ی با اولویت‌تری در صف آماده‌باش موجود باشد.
- در این حالت امکان دارد پردازنده هیچگاه در اختیار پردازه‌ی کم اولویت قرار نگیرد و در نتیجه این پردازه دچار گرسنگی شود.
- یک راه برای جلوگیری از این اتفاق، افزایش اولویت پردازه‌هایی است که مدت زیادی در صف اولویت قرار دارند. به این کار Aging گفته می‌شود.

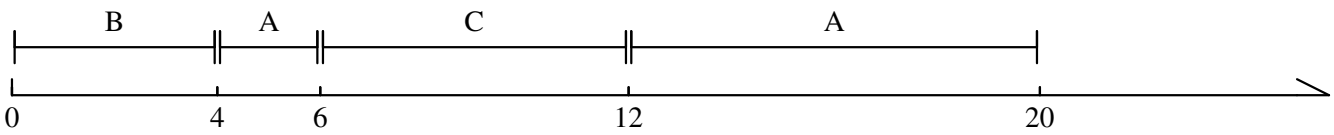
- در الگوریتم کوتاه‌ترین وظیفه اول (Shortest Job First یا SJF) پردازنده در اختیار پردازه‌ای قرار می‌گیرد که به کمترین زمان پردازش احتیاج داشته باشد.
- این الگوریتم دو حالت Preemptive و Nonpreemptive دارد؛ به حالت Preemptive این الگوریتم گاهی Shortest Remaining Time First (کوتاه‌ترین زمان باقی‌مانده اول) گفته می‌شود.
- سه پردازه‌ی زیر را در نظر بگیرید.

Process	CPU Burst	Arrival
A	10ms	0ms
B	4ms	0ms
C	6ms	6ms

- خروجی الگوریتم Shortest Job First در حالت Nonpreemptive را در نمودار زیر می‌بینید.



- خروجی الگوریتم Shortest Job First در حالت Preemptive را در نمودار زیر می‌بینید.



- تفاوت در زمان ورود پردازه‌ی C هست.

- می‌توان اثبات کرد که متوسط زمان انتظار پردازه‌ها در الگوریتم SJF حداقل هست.
- اما این الگوریتم به زمان پردازش مورد نیاز پردازه‌ها (CPU Burst) احتیاج دارد. در عمل معمولا سیستم عامل پیشاپیش نمی‌داند طول CPU Burst پردازه‌ها چقدر است.
- برای استفاده از الگوریتم SJF می‌توان مقدار CPU Burst پردازه‌ها را تخمین زد.
- یک روش ساده برای تخمین طول CPU Burst بعدی یک پردازه، در نظر گرفتن میانگین طول CPU Burst-های قبلی همان پردازه است (بدیهی است که طول CPU Burst-ها قبلی مشخص است چون انجام شده‌اند). مثال زیر را در نظر بگیرید.

Row	CPU Burst	Average
1	2ms	-
2	4ms	2ms
3	3ms	3ms
4	100ms	3ms
5	110ms	27ms
6	100ms	43ms

- ستون آخر میانگین طول CPU Burst-های قبلی است.
- برای تخمین دومین زمان پردازش، میانگین زمان‌های پردازش قبلی یعنی عدد دو در نظر گرفته می‌شود.
- برای تخمین سومین زمان پردازش، میانگین دو و چهار در نظر گرفته می‌شود.
- برای تخمین سومین زمان پردازش، میانگین دو و چهار و سه در نظر گرفته می‌شود و ...
- بدی میانگین ساده در گام‌های پنجم و ششم مشخص می‌شود: در گام پنجم، زمان پردازش قبلی صد بوده است اما مقدار تخمین برای زمان پردازش بعدی فقط ۲۷ میلی‌ثانیه هست.
- در گام ششم اگر چه دو زمان پردازش اخیر حداقل صد میلی‌ثانیه بوده‌اند، میانگین کل زمان‌های پردازش فقط ۴۳ هستند.
- اگر چه چند زمان پردازش آخر بزرگ هستند، زمان‌های پردازش اولیه (که کوچک بوده‌اند) مانع از این می‌شوند که میانگین به سرعت رشد کند و این مسئله منجر به تخمین بدی می‌شود.

- بدی در نظر گرفتن میانگین ساده این است که تأثیر تغییر اندازه‌ی نسبی CPU Burst-ها کم است.
- برای حل این مشکل می‌توان از میانگین نمایی (Exponential Averaging) استفاده کرد.
- فرض کنید a_i مقدار میانگین نمایی در گام i-ام و t_i طول CPU Burst i-ام باشد.
- در میانگین نمایی ثابت α عددی بین اعشاری از صفر تا یک است.
- تخمین در گام $i+۱$-ام با توجه به تخمین در مرحله‌ی قبلی و زمان پردازش قبلی تعیین می‌شود:

$$a_{i+۱} = \alpha t_i + (۱ - \alpha) a_i$$

- فرض کنید α برابر ۰/۵ باشد (فرض کنید $a_۱$ برابر پنج باشد).

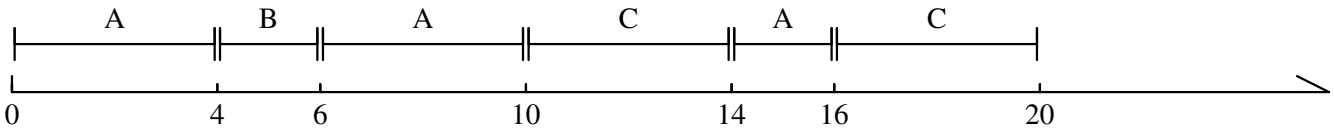
Row	CPU Burst	Average
1	2ms	5ms
2	4ms	3.5ms
3	3ms	3.8ms
4	100ms	3.4ms
5	110ms	51.7ms
6	100ms	80.9ms

- در مقایسه با میانگین ساده، در گام‌های پنجم و ششم میانگین نمایی بعد از تغییر اندازه‌ی نسبی زمان پردازش بهتر زمان پردازش را تخمین زده است.
- هر چه ضریب α به صفر نزدیک‌تر باشد، حافظه‌ی میانگین نمایی قوی‌تر می‌شود و تأثیر زمان‌های پردازش اولیه بیشتر در تخمین باقی می‌ماند.
- هر چه ضریب α به یک نزدیک‌تر باشد، حافظه‌ی میانگین نمایی ضعیف‌تر می‌شود و تأثیر چند زمان پردازش آخر در تخمین بیشتر خواهد بود.

- در الگوریتم Round Robin زمانی به عنوان برش زمانی (Time Slice یا Time Quantum) تعیین می‌شود که مشخص می‌کند هر پردازه در هر نوبت چقدر می‌تواند از پردازنده استفاده کند.
- الگوریتم Round Robin به صورت نوبتی پردازنده را در اختیار هر پردازه‌ی آماده‌ی اجرا قرار می‌دهد اما اگر پردازه‌ای برش زمانی خودش را کامل استفاده کند، اجرایش قطع می‌شود و به آخر صف انتقال می‌یابد.
- بدیهی است که الگوریتم Round Robin به صورت Preemptive کار می‌کند.
- مثال زیر را در نظر بگیرید.

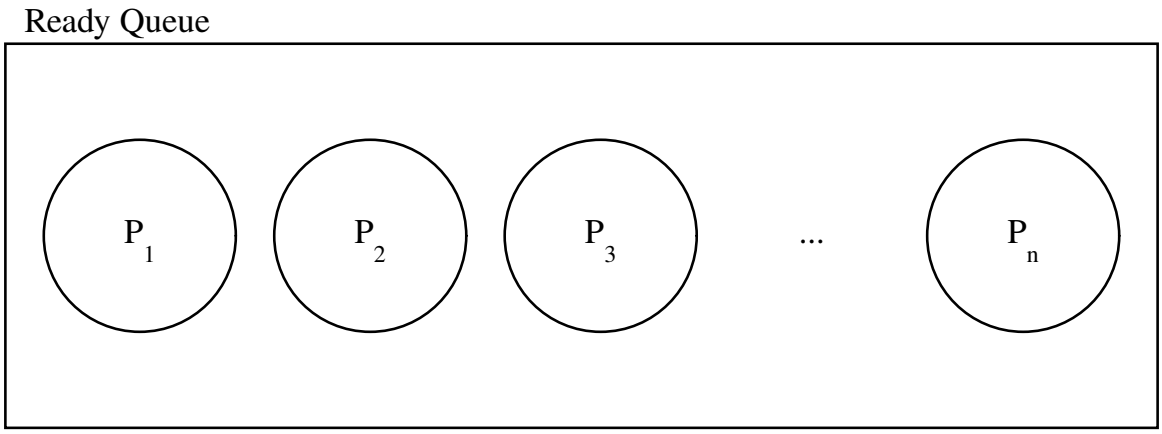
Process	CPU Burst	Arrival
A	10ms	0ms
B	2ms	2ms
C	8ms	8ms

- فرض کنید برش زمانی در این مثال چهار میلی‌ثانیه باشد.
- خروجی الگوریتم Round Robin برای این پردازه‌ها نمودار زیر است.



Round Robin - ویژگی‌ها

- یکی از ویژگی‌های خوب الگوریتم Round Robin زمان پاسخ آن است.
- فرض کنید در سیستم عاملی n پردازه وجود داشته باشند و برش زمانی q باشد.
- قبل از هر پردازه‌ی آماده‌ی اجرا حداکثر $n-1$ پردازه در صف آماده باش هستند، حتما بعد از $n-1$ پردازه، یک بار پردازنده در اختیار هر پردازه‌ی منتظر قرار می‌گیرد.
- بنابراین، زمان پاسخ حداکثر $(n-1) \cdot q$ هست.



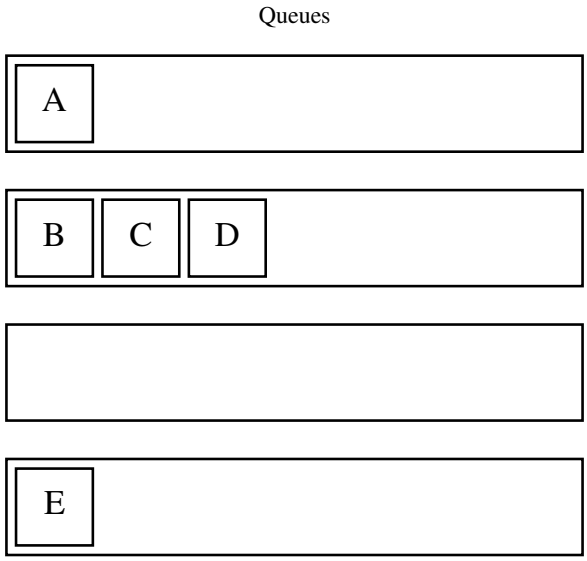
- برش زمانی معمولا عددی بین ده تا صد میلی‌ثانیه است.
- هر چه برش زمانی بزرگ‌تر باشد، مقدار زمان پاسخ افزایش می‌یابد و الگوریتم مشابه الگوریتم FCFS عمل می‌کند.
- اگر برش زمانی خیلی کوچک باشد زمان پاسخ کاهش می‌یابد ولی تعداد و سربار تعویض متن افزایش می‌یابد؛ پردازنده زمان قابل توجهی را صرف تعویض متن خواهد کرد.

- الگوریتم‌هایی که در ادامه می‌بینیم از چند صف آماده‌باش بهره می‌برند.
- در هر یک از این صف‌ها، یک الگوریتم زمانبندی مجزا اجرا می‌شود.

- در صف چند رده‌ای (Multi-Level Queue) پردازه‌های آماده‌ی اجرا در چند صف قرار می‌گیرند.
- هر یک از این صفحه‌ها از الگوریتم مشخصی استفاده می‌کند.
- صف مربوط به هر پردازه مشخص است.
- در صف چند رده‌ای، الگوریتمی برای زمانبندی بین صف‌ها نیز لازم است.
- معمولا از دو الگوریتم زمانبندی بین صفحه‌ای استفاده می‌شود:

الف) اولویت: پردازه‌های یک صف اجرا می‌شوند اگر همه‌ی صف‌های با اولویت‌تر خالی باشند.

ب) برش زمانی: به هر صف زمان مشخصی تخصیص می‌یابد و صف‌ها به صورت متناوب اجرا می‌شوند.



صف چند رده‌ای - مثال

- دو صف: در صف اول از الگوریتم Round Robin با برش زمانی چهار میلی‌ثانیه و در صف دوم از الگوریتم Shortest Job First در حالت Preemptive استفاده می‌شود.

Queues

Queue 1: Round Robin (4ms)

Queue 2: Shortest Job First (Preemptive)

- پردازه‌های زیر را در نظر بگیرید.

Process	CPU Burst	Arrival	Queue
A	10ms	0ms	2
B	8ms	5ms	1
C	3ms	6ms	1
D	3ms	7ms	2

- اگر اولویت صف اول از صف دوم بیشتر باشد و زمانبندی بین صف‌ها به کمک الگوریتم اولویت انجام شود، زمانبندی زیر حاصل می‌شود.

صف چند رده‌ای - مثال

- مثال صفحه‌ی قبل و چهار پردازه‌ی زیر را دوباره در نظر بگیرید.

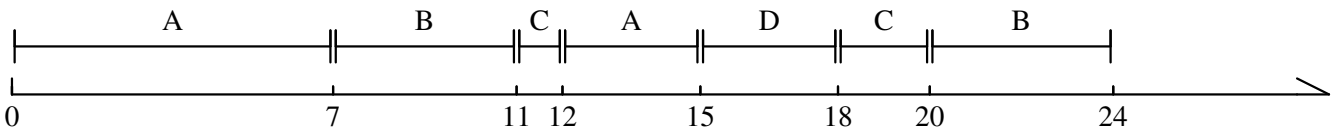
Process	CPU Burst	Arrival	Queue
A	10ms	0ms	2
B	8ms	5ms	1
C	3ms	6ms	1
D	3ms	7ms	2

Queues

Queue 1: Round Robin (4ms)

Queue 2: Shortest Job First (Preemptive)

- اگر از برش زمانی برای زمانبندی بین صف‌ها استفاده شود و به ازای هر پنج میلی‌ثانیه در صف اول، هفت میلی‌ثانیه به صف دوم تخصیص یابد، زمانبندی زیر حاصل می‌شود.



- در الگوریتم زمانبندی صف چند رده‌ای با بازخورد (Multi-Level Feedback Queue) صف پردازه‌ها تغییر می‌کند.
- در این الگوریتم باید موارد زیر مشخص شود:

  الف) الگوریتم هر یک از صف‌ها

  ب) صف شروع (پردازه‌های جدید به این صف وارد می‌شوند)

  ج) شرایط تغییر صف پردازه‌ها

  د) الگوریتم زمانبندی بین صف‌ها
- هر پردازه‌ی جدید وارد صف شروع می‌شود (بر خلاف الگوریتم قبل، صف پردازه‌ها از قبل مشخص نیست).
- با توجه به رفتار یک پردازه، صف پردازه عوض می‌شود.

صف چند رده‌ای با بازخورد - مثال

- فرض کنید در الگوریتم صف چند رده‌ای با بازخورد، دو صف موجود باشند:

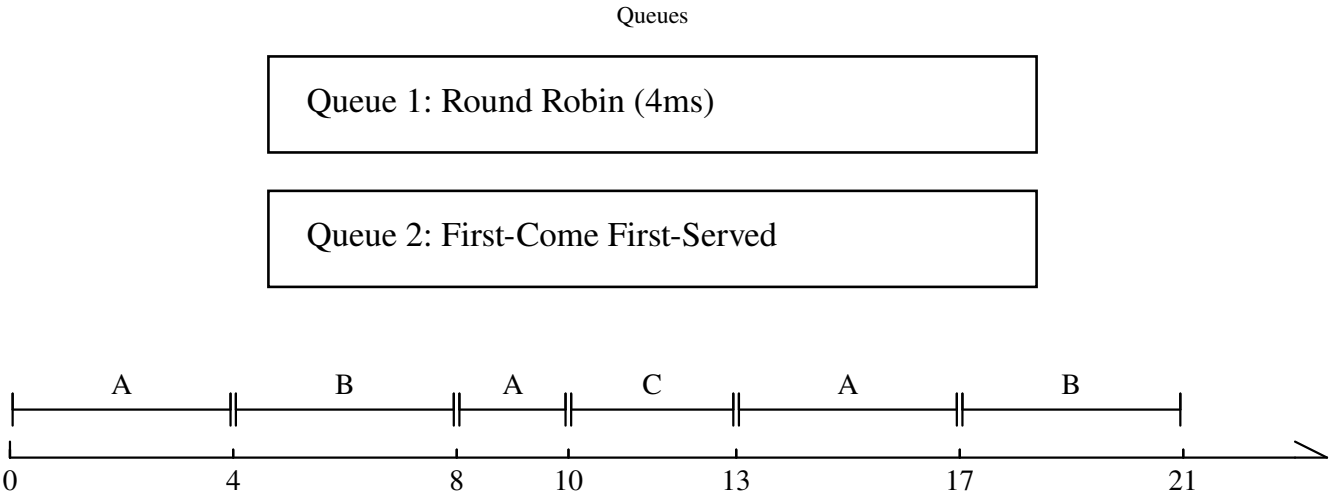
  صف اول: از الگوریتم Round Robin با برش زمانی چهار میلی‌ثانیه استفاده می‌شود.

  صف دوم: از الگوریتم FCFS استفاده می‌شود.

  بین صف‌ها نیز از الگوریتم اولویت استفاده می‌شود (اولویت صف اول بیشتر از دوم است).

- صف شروع صف اول است.
- پردازه‌های موجود در صف اول اگر برش زمانی خود را کامل استفاده کنند به صف دوم انتقال می‌یابند.

Process	CPU Burst	Arrival
A	10ms	0ms
B	8ms	3ms
C	3ms	10ms

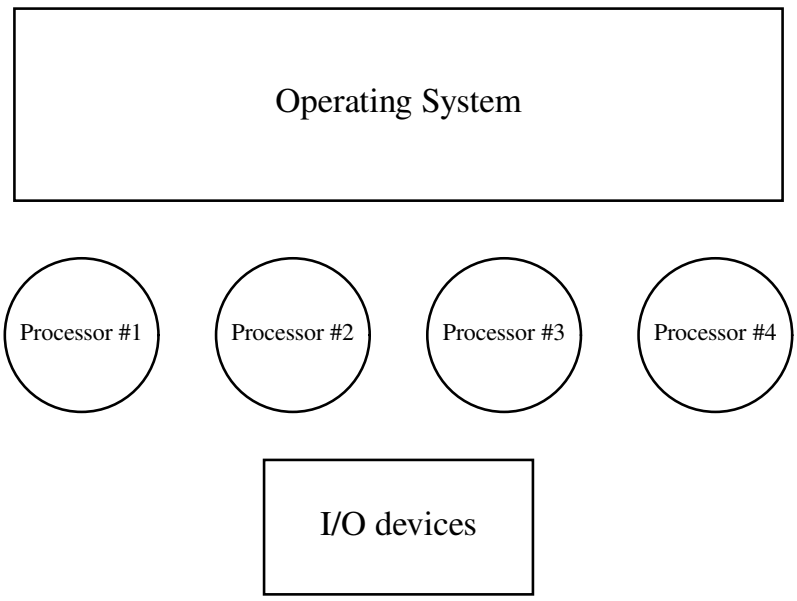


- در حالت کلی سیستم‌های سخت‌افزاری و نرم‌افزاری چند پردازنده‌ها در دو دسته قرار می‌گیرند:

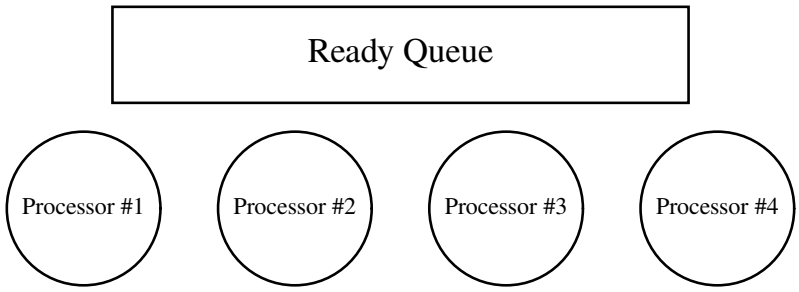
  الف) چند پردازندگی متقارن (Symmetric Multiprocessing (SMP)).

  ب) چند پردازندگی نامتقارن (Asymmetric Multiprocessing).

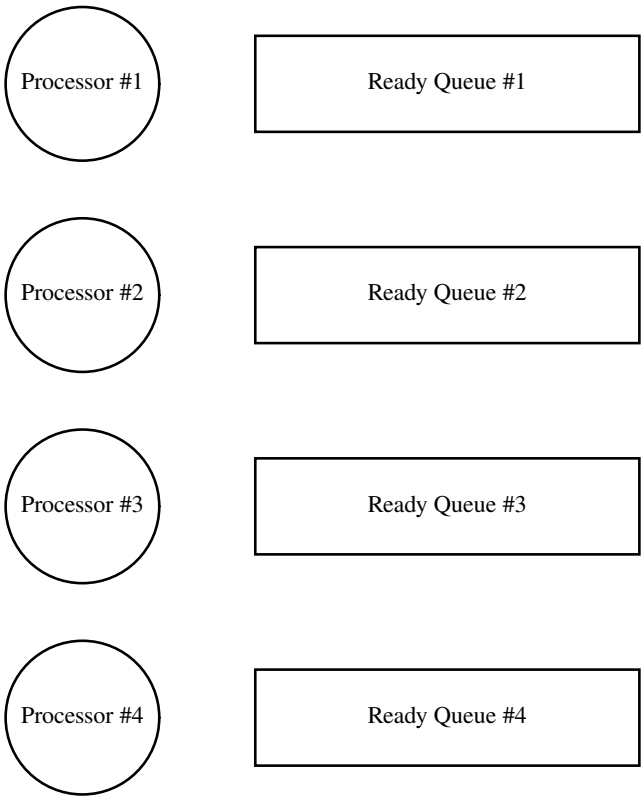
- در چند پردازندگی نامتقارن توانایی‌های همه‌ی پردازنده‌ها یکسان نیستند؛ برای مثال فقط یکی از پردازنده‌ها کد سیستم عامل را اجرا می‌کند یا فقط یکی وقفه‌های سخت‌افزاری را دریافت می‌کند یا عمل ورودی یا خروجی انجام می‌دهد.
- در چند پردازندگی متقارن همه‌ی پردازنده‌ها مشابه هم هستند، همه کد سیستم عامل را اجرا می‌کنند و می‌توانند عملیاتی مثل ورودی یا خروجی را انجام دهند.



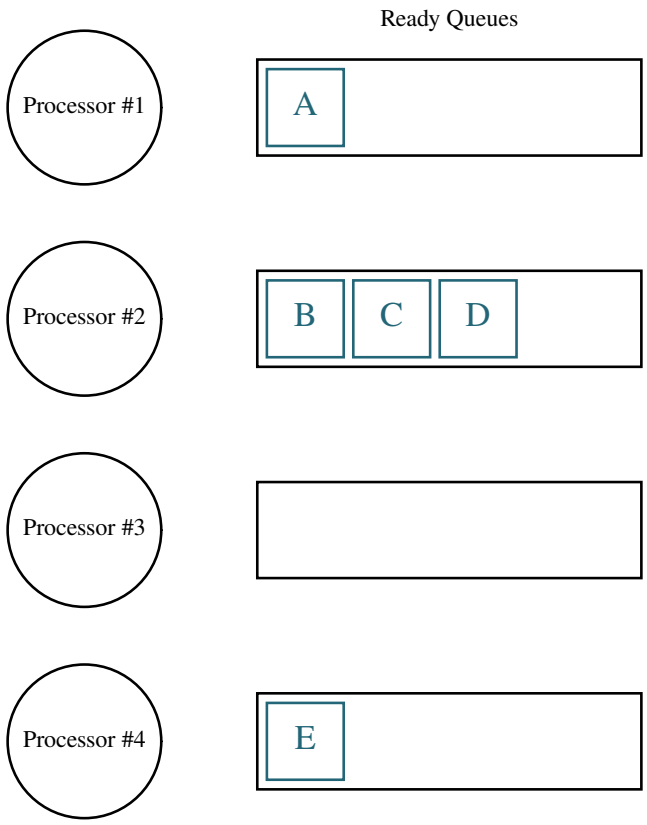
- در چند پردازندگی نامتقارن، یکی از پردازنده‌ها به عنوان پردازنده‌ی اصلی عملیات اصلی سیستم عامل را انجام می‌دهد مثل زمانبندی و عملیات ورودی و خروجی. به این پردازنده گاهی Master Server) گفته می‌شود.
- بقیه‌ی پردازنده‌ها فقط کد پردازه‌های کاربری را اجرا می‌کنند.



- در چند پردازندگی متقارن همه‌ی پردازنده‌ها برخی از عملیات سیستم عامل مثل زمانبندی و عملیات ورودی یا خروجی را انجام می‌دهند؛ بنابراین، هر یک از پردازنده‌ها یک صف زمانبندی خاص خودش را دارد که می‌تواند از یکی از الگوریتم‌هایی که دیده‌ایم استفاده کند.



- چه مشکلی در چندپردازندگی متقارن ممکن است رخ دهد؟



- در چندپردازندگی متقارن، چون چند صف زمانبندی وجود دارد ممکن است بار پردازشی پردازنده‌ها یکسان نباشد؛ برای مثال پردازنده‌ای به دلیل خالی بودن صف آماده‌باشش بیکار باشد ولی پردازنده‌ی دیگری مشغول اجرای پردازه‌ها باشد.
- در این شرایط مطلوبست با انتقال پردازه‌هایی از صف پردازنده‌هایی با بار بالا به صف سایر پردازنده‌ها بار پردازه‌ها را متوازن کرد.
- به این کار توازن بار (Load Balancing) گفته می‌شود.
- به انتقال پردازه‌ها از صف یک پردازنده به صف پردازنده‌ی دیگر برای توازن بار، مهاجرت (Migration) گفته می‌شود.

- مهاجرت به دو شکل انجام می‌شود.

  الف) مهاجرت Push

  الف) مهاجرت Pull

- در مهاجرت Push سیستم عامل به صورت متناوب در بازه‌های زمانی مشخصی وضعیت صف‌ها را بررسی می‌کند و از صف پردازنده‌های شلوغ پردازنده‌ها را به صف‌های خلوت انتقال می‌دهد.
- در مهاجرت Pull هر پردازنده صف خودش را بررسی می‌کند که اگر بار آن کم است از صف پردازنده‌های دیگر پردازه بر می‌دارد و به صف خودش اضافه می‌کند.
- از طرفی مهاجرت به دلیل از دست رفتن اطلاعات حافظه‌ی نهان پردازنده سربار دارد.
- برای جلوگیری از سربار پردازشی مهاجرت، می‌توان برای پردازه‌ها نزدیکی پردازنده (Processor Affinity) تعیین کرد.
- اگر برای پردازه‌ی A نزدیکی پردازنده به پردازنده‌ی P تعیین شود، به این مفهوم است سیستم عامل باید سعی کند پردازه‌ی A را در صف مربوط به پردازنده‌ی P قرار دهد.

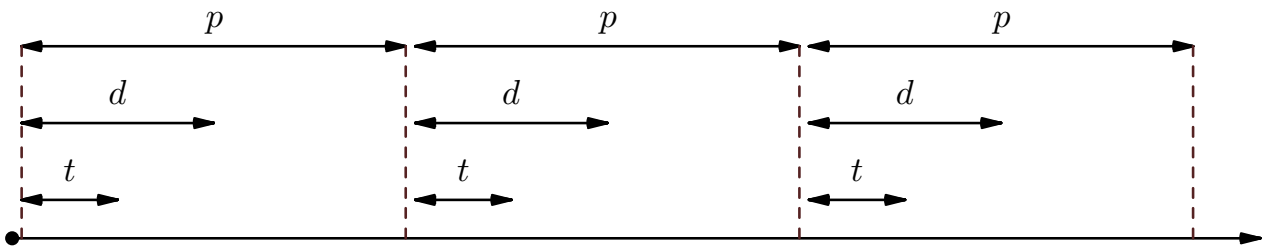
- قبلا با سیستم‌های بی‌درنگ آشنا شده‌ایم؛ در ادامه الگوریتم‌هایی برای زمانبندی در این سیستم‌ها خواهیم دید.
- سیستم‌های بی‌درنگ معمولا مبتنی بر رخداد هستند: برای نمونه فرا رسیدن زمان مشخص یا ایجاد یک وقفه‌ی سخت‌افزاری رخدادهایی هستند که پس از آنها پردازه‌ای با تأخیر کمی باید اجرا شود تا به این رخدادها پاسخ دهد.
- برای نمونه، در صورتی که یک خودرو با مانعی برخورد کند وقفه‌ای ایجاد می‌شود (توسط حسگر مناسب). پس از این رخداد، پردازه‌ای که کیسه‌ی هوا را مدیریت می‌کند باید در چند میلی‌ثانیه اجرا شود.
- سیستم‌های بی‌درنگ به دو دسته تقسیم می‌شوند:

  الف) سیستم‌های بی‌درنگ نرم (Soft Real-time)

  ب) سیستم‌های بی‌درنگ سخت (Hard Real-time)
- در زمانبندی سیستم‌های بی‌درنگ نرم تضمینی برای زمان اجرای یک پردازه که محدودیت دارد وجود ندارد. فقط این تضمین وجود دارد که پردازه‌هایی که باید به یک رخداد بی‌درنگ پاسخ دهند به پردازه‌های عادی سیستم عامل اولویت داده می‌شوند. بنابراین ممکن است با تأخیر پردازه‌های بی‌درنگ اجرا شوند.
- سیستم‌های بی‌درنگ سخت، یک پردازه‌ی بی‌درنگ باید در محدودیت زمانی خودش اجرا شود. برای نمونه، در صورتی که در یک سیستم ترمز ABS اگر چرخ لیز بخورد، که باید عکس‌العمل مناسب در سه تا پنج میلی‌ثانیه تولید شود. پس از این زمان پاسخ به رخداد فایده‌ای ندارد و غیر قابل قبول است.

- تأخیر رخداد (Event Latency): فاصله‌ی زمانی بروز رخداد تا پاسخ دادن به آن.
- تأخیر وقفه (Interrupt Latency): فاصله‌ی زمانی رخداد یک وقفه تا شروع اجرای روال پاسخ‌دهی به وقفه.
- تأخیر Dispatch (Dispatch Latency): فاصله‌ی زمانی متوقف کردن یک پردازه و اجرای یک پردازه‌ی دیگر (تعویض متن).
- بدیهی است که در یک سیستم بی‌درنگ بهتر است این تأخیرها کمتر باشند.

- در سیستم‌های عامل بی‌درنگ به پردازه‌های بی‌درنگ (که باید در محدوده‌ی زمانی مشخصی اجرا شوند) نسبت به سایر پردازه‌ها اولویت داده می‌شود.
- برای اینکه زمان تأخیر رخداد محدود باشد، لازم است زمانبندی در این سیستم‌های عامل به صورت Preemptive انجام شود.
- در ادامه فرض می‌کنیم پردازه‌های بی‌درنگ تناوبی (Periodic) هستند: هر یک از این پردازه‌ها در دوره‌های تناوب ثابتی (مثلا هر بیست میلی‌ثانیه) احتیاج به پردازش دارند.



- در هر دوره هر پردازه‌ی بی‌درنگ به زمان مشخصی پردازش احتیاج دارد.
- در هر دوره هر پردازه‌ی بی‌درنگ یک مهلت (Deadline) اجرا دارد که باید تا حداکثر این زمان پس از گذشت دوره، پردازش پردازه تمام شود.
- اگر دوره‌ی تناوب یک پردازه p، مهلت اجرای پردازه d و زمان پردازش پردازه t باشد، داریم:

$$0 \leq t \leq d \leq p$$

- نرخ (Rate) یک پردازه برابر $1/p$ است.

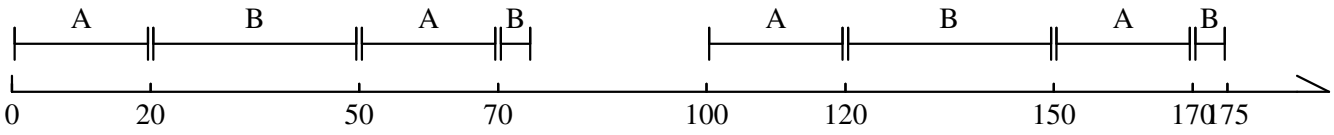
- اگر فقط به پردازه‌های بی‌درنگ اولویت بیشتری داده شود زمانبند سیستم عامل فقط برای شرایط بی‌درنگ نرم مناسب است.
- در زمانبندی برای سیستم‌های بی‌درنگ سخت، لازم است زمانبند مهلت اجرای پردازه‌ها را نیز در اختیار داشته باشد.
- در ادامه چند الگوریتم برای زمانبندی برای پردازه‌های بی‌درنگ سخت خواهیم دید.
- هر پردازه مهلت اجرایش را در اختیار زمانبند قرار می‌دهد؛ اگر زمانبند یا پردازه را با تضمین اجرای به موقعش می‌پذیرد یا آن را رد می‌کند.
- به این کار Admission-Control می‌گویند.

الگوریتم زمانبندی نرخ یکنواخت

- در الگوریتم زمانبندی نرخ یکنواخت (Rate-Monotonic) پردازه‌ها با توجه به اولویت زمانبندی می‌شوند، مشابه الگوریتم اولویت که در بخش قبل بررسی کردیم در حالت Preemptive.
- اولویت یک پردازه با توجه به عکس دوره‌ی تناوب آن تعیین می‌شود: پردازه‌ای اولویتش بیشتر است که دوره‌ی تناوب کوچک‌تری داشته باشد.
- در این الگوریتم فرض می‌کنیم زمان پردازش پردازه (CPU Burst) در هر دوره ثابت است.
- پردازه‌های زیر را فرض کنید (مهلت اجرای هر دوره تا آغاز دوره‌ی بعدی است).

Process	Period (p)	CPU Burst (t)	Deadline (d)
A	50ms	20ms	50ms
B	100ms	35ms	100ms

- در ابتدا با توجه به بار پردازش هر پردازه می‌توانیم بررسی کنیم که آیا امکان زمانبندی پردازه‌ها وجود دارد یا خیر.
- بار پردازشی پردازه‌ی اول ۴۰ درصد (۲۰ میلی‌ثانیه از هر ۱۰۰ میلی‌ثانیه) و بار پردازشی پردازه‌ی دوم ۳۵ درصد (۳۵ میلی‌ثانیه از هر ۱۰۰ میلی‌ثانیه) است. بنابراین بار پردازشی در مجموع ۷۵ درصد است و به نظر می‌رسد بتوان زمانبندی را انجام داد.
- نمودار زیر خروجی الگوریتم نرخ یکنواخت را نشان می‌دهد.



- دقت کنید که ممکن است بار پردازشی تعدادی پردازه کمتر از صد درصد باشد ولی توسط الگوریتم نرخ یکنواخت قابل زمانبندی نباشند.

الگوریتم زمانبندی نزدیک‌ترین مهلت اول

- در الگوریتم زمانبندی Earliest-Deadline-First (EDF) اولویت با پردازه‌ای است که زمان فرارسیدن مهلت آن از بقیه زودتر است.
- برای نمونه دو پردازه‌ی زیر را در نظر بگیرید.

Process	Period (p)	CPU Burst (t)	Deadline (d)
A	50ms	25ms	50ms
B	80ms	35ms	80ms

- نمودار زیر خروجی الگوریتم زمانبندی EDF را نشان می‌دهد.