

سیستم عامل جلسه پنجم

ادامه الگوریتم‌های زمان‌بندی

زمان‌بندی با اولویت (Priority)

الگوریتم SJF در حالت خاصی از الگوریتم زمان‌بندی با اولویت است. به هر فرایند یک اولویت نسبت داده می‌شود و پردازنده به فرایندی تخصیص می‌یابد که بالاترین اولویت را دارای باشد

فرایندهایی با اولویت یکسان، به ترتیب FCFS زمان‌بندی می‌شوند.

الگوریتم SJF یک الگوریتم با اولویت ساده است که در آن، اولویت (P)، معکوس انفجار تخمین زده شده بعدی پردازنده است. هر چه زمان انفجار پردازنده بیشتر باشد، اولویت کمتر است و برعکس.

بعضی از افراد برای اولویت‌های پایین از اعداد کوچک استفاده می‌کنند ولی بعضی دیگر برای اولویت پایین از اعداد بزرگ استفاده می‌کنند. این کار موجب سردرگمی است. فرض می‌کنیم که اعداد کوچک، اولویت بالا را نشان می‌دهد

فرایندهای زیر را در نظر بگیرید. به طوری که در زمان 0 به ترتیب p_1, p_2, p_3, \dots p5 رسیده‌اند و طول انفجار پردازنده بر حسب میلی ثانیه است:

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

با استفاده از زمانبندی با اولویت، این فرایندها بر اساس نمودار گانت (GANTT) زیر رسم می‌شوند:



میانگین زمان انتظار در این مثال، 8.2 میلی ثانیه است.

اولویت می‌تواند به طور داخلی یا خارجی تعریف شود. اولویت‌هایی که به طور داخلی تعریف می‌شوند، اولویت یک فرایند را با استفاده از کمیت‌های قابل اندازه‌گیری تعریف می‌کنند.

به عنوان مثال حدود زمانی، نیازمندی‌های حافظه، تعداد فایل‌های باز و نسبت میانگین انفجار i/o به میانگین انفجار پردازنده، در محاسبه اولویت‌های داخلی به کار می‌آیند.

اولویت‌های خارجی بر اساس معیارهایی تعیین می‌شوند که از نظر سیستم عامل، خارجی هستند، مثل اهمیت فرایند، نوع و میزان هزینه ای که برای استفاده از کامپیوتر پرداخته شده، میزان پشتیبانی موسسه از کار و سایر عوامل سیاست گذاری.

زمانبندی با اولویت می‌تواند با قبضه کردن یا بدون قبضه کردن باشد. وقتی فرایندی به صف آماده می‌رسد، اولویت آن با اولویت فرایند در حال اجرا مقایسه می‌شود. اگر فرایندی که تازه وارد صف شده بیشتر از اولویت فرایندی باشد که در حال اجرا است، الگوریتم زمانبندی با قبضه کردن (Preemptive)، پردازنده را در اختیار فرایند جدید قرار می‌دهد. اما در الگوریتم زمانبندی بدون قبضه کردن (Non Preemptive)، فرایند جدید بدون توجه به اولویتش در ابتدای صف آماده قرار می‌گیرد

مساله ی عمده در الگوریتم زمانبندی با اولویت، انسداد (indefinite Blocking) یا گرسنگی (قحطی) (Starvation) است.

Starvation or indefinite blocking is a phenomenon associated with the Priority scheduling algorithms. A process that is present in the ready state and has low priority keeps waiting for the CPU allocation because some other process with higher priority comes with due respect time.

فرایندی که آماده اجرا است ولی منتظر پردازنده باشد، مسدود در نظر گرفته می‌شود. الگوریتم زمانبندی با اولویت می‌تواند و منجر به این شود که فرایندهایی با اولویت پایین، به مدت نامحدودی منتظر پردازنده باشند. در یک سیستم کامپیوتری با بار زیاد،

فرایندهایی با اولویت بالا، مانع از این می‌شوند که پردازنده به فرایندهایی با اولویت پایین تعلق یابد. معمولاً یا سرانجام، فرایند با اولویت پایین اجرا می‌شود، یا سیستم کامپیوتری فرو می‌پاشد و همه فرایندهای با اولویت پایین که تمام نشده اند مفقود می‌شوند

راه حل این مساله ی انسداد نامحدود فرایندهای با اولویت پایین، سالمندی (Aging) است.

در این تکنیک اولویت فرایندی که مدت زیادی در سیستم منتظر مانده است، به تدریج افزایش می‌یابد.

اگر اولویت با مقادیری از 0 (اولویت بالا) تا 127 (اولویت پایین) مشخص شود، می‌توان هر 15 دقیقه، یک واحد به اولویت یک فرایند اضافه کنیم. سرانجام، حتی فرایندی که اولویت اولیه آن 127 است. اولویت بالایی در سیستم کسب می‌کند و می‌تواند اجرا شود. در واقع، برای اینکه فرایندی با اولویت 127 سالمند شود و اولویت 0 را بدست آورد بیش از 32 ساعت طول نمی‌کشد.

یادداشت:

P = 15 minute - period of each increment

X = 127 process index number

T = 127 total of indexes

$((T-1)*15)/60$

$((127-1)*15)/60$

= 31.5 h

زمانبندی نوبت گردشی “RR” (Round Robin)



الگوریتم نوبت گردشی RR مخصوص سیستم‌های اشتراک زمانی طراحی شده است. این الگوریتم شبیه FCFS است، با این تفاوت که در جابه جایی بین فرایندها، از زمان بندی با قبضه کردن (Preemptive) استفاده می‌شود. یک واحد زمانی کوچک، به نام کوانتوم (quantum) زمانی یا برهه زمانی (برش زمانی) تعریف می‌شود. کوانتوم زمانی معمولاً 10 – 100 میلی ثانیه است. صف آماده به صورت یک صف چرخشی در نظر گرفته می‌شود. زمانبند پردازنده در طول صف آماده جا به جا می‌شود و پردازنده را حداکثر به مدت یک کوانتوم زمانی به هر فرایند تخصیص می‌دهد.

برای پیاده سازی زمانبندی RR، صف آماده را به صورت یک صف FIFO (First in First Out) از فرایندها در نظر می‌گیریم، فرایندهای جدید به انتهای صف آماده اضافه می‌شود. زمانبندی پردازنده، اولین فرایند را از صف آماده انتخاب می‌کند و تایمر را طوری تنظیم می‌کند که پس از یک کوانتوم زمانی وقفه ای صادر شود و فرایند را روی پردازنده توزیع می‌کند.

دو حالت وجود دارد:

1. پردازنده کمتر از یک کوانتوم زمانی به فرایند اختصاص یابد. در این حالت، خود فرایند پردازنده را آزاد می کند و بدین ترتیب، پردازنده به فرایند بعدی موجود در صف آماده تخصیص می یابد.

2. اگر پردازنده بخواهد بیش از یک کوانتوم زمانی به فرایند در حال اجرا اختصاص یابد. تایمر خاموش می شود و وقفه ای را به سیستم عامل می فرستند. تعویض متن (Context switch) صورت می گیرد و فرایند به انهای صف آماده اضافه می شود سپس زمانبندی پردازنده، فرایند بعدی را از صف آماده انتخاب می کند

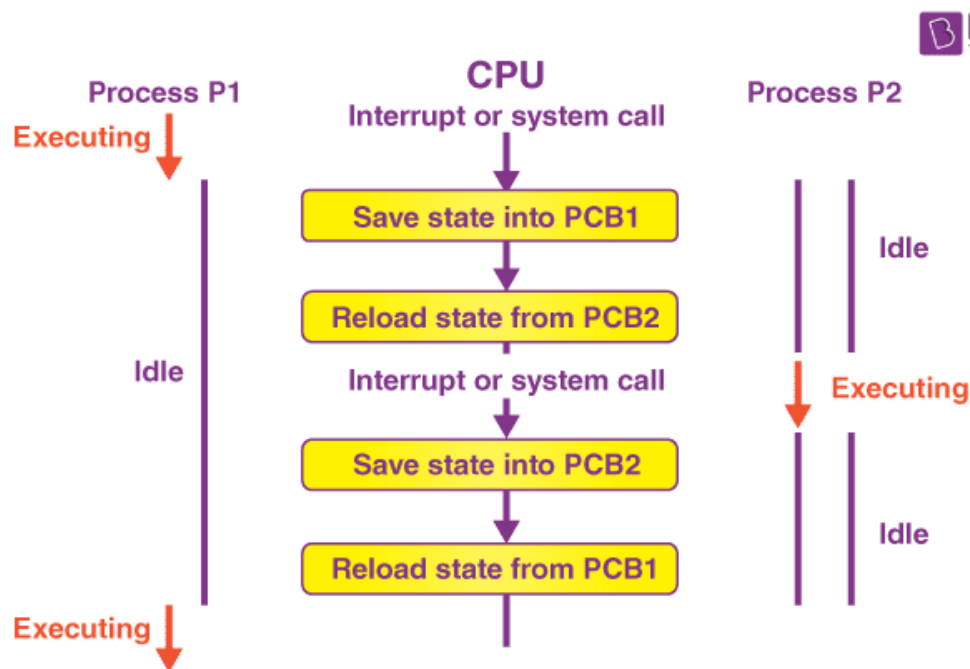
تعویض متن (تعویض بستر یا تعویض زمینه) (Context Switch):

وقفه (interrupt) موجب می شود سیستم عامل، پردازنده را از اجرای وظیفه فعلی به اجرای روال هسته ببرد. چنین عملیاتی غالباً در سیستم های همه منظوره رخ می دهد. وقتی وقفه ای رخ می دهد. لازم است سیستم، متن (وضعیت - status) فعلی فرایند را که در پردازنده در حال اجرا است، ذخیره کند، به طوری که پس از پردازش، آن متن را بازیابی می کند که موجب به تعویق افتادن فرایند و سپس از سرگیری آن می شود. متن، در PCB (Process control Block) مربوط به فرایند ذخیره می شود. متن شامل ثبات های پردازنده، حالت فرایند و اطلاعات مدیریت حافظه است.

PCB شامل وضعیت کامل هر فرایند می باشد از رجیستر ها تا اطلاعات پایه هر فرایند در واحد کنترل وظیفه (PCB) آن نوشته می شود

تعویض پردازنده به فرایند دیگر، نیازمند اجرای ذخیره حالت فعلی و بازیابی حالت فرایند دیگر است.

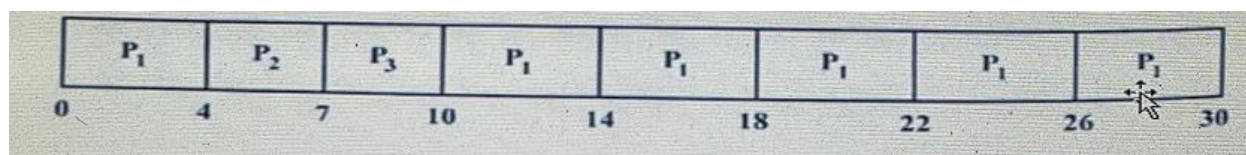
این کار، تعویض متن یا تعویض بستر یا (Context Switch) نام دارد. وقتی تعویض متن صورت می‌گیرد، هسته متن فرایند قبلی را در PCB (Process Control Block) آن ذخیره می‌کند و متن ذخیره شده فرایند جدید را که برای اجرا زمانبندی شد، بار می‌کند. زمان تعویض متن، سرباز محض است، زیرا هنگام تعویض، سیستم، کاری انجام نمی‌دهد. سرعت آن از ماشین به ماشین دیگر فرق می‌کند که به سرعت حافظه، تعداد ثبات‌هایی که باید کپی شوند و وجود دستورالعمل خاص (مثل تنها یک دستورالعمل برای باز کردن با ذخیره تمام ثبات‌ها) بستگی دارد. سرعت عای متدوال، چندین میلی ثانیه هستند.



میانگین زمان انتظار در الگوریتم RR ، اغلب زیاد است. فرایندهای زیر را در نظر بگیرید که به ترتیب P_1, P_2, P_3 در زمان O می‌رسند و زمان انفجار پردازنده بر حسب ثانیه است

| فرآیند | زمان انفجار (زمان اجرا) |
|--------|-------------------------|
| P_1 | 24 |
| P_2 | 3 |
| P_3 | 3 |

اگر از کوانتوم زمانی 4 میلی ثانیه استفاده کنیم، آن گاه فرایند P_1 ، 4 میلی ثانیه اول را می‌گیرد، چون به 20 میلی ثانیه دیگر نیاز دارد، پس از اولین کوانتوم زمانی قبضه می‌شود و پردازنده به فرایند بعدی موجود در صف، یعنی P_2 اختصاص می‌یابد. فرایند P_2 به 4 میلی ثانیه نیاز ندارد. به همین دلیل قبل از انقضای کوانتوم زمانی آن، خاتمه می‌یابد و سپس پردازنده به فرایند بعدی، یعنی P_3 تخصیص می‌یابد. پس از این که هر فرایند، پردازنده را، به اندازه یک کوانتوم زمانی در اختیار گرفت. پردازنده به فرایند P_1 بر می‌گردد تا کوانتوم زمانی دیگری در اختیارش باشد. زمانبندی RR مربوط به این مثال در نمودار GANTT گانت زیر آمده است:



میانگین زمان انتظار را براین این زمانبندی محاسبه می‌کنیم

P_1 به میزان $(4 - (4 + 3 + 3)) = 6$ میلی ثانیه منتظر می‌ماند.

P_2 به میزان 4 میلی ثانیه منتظر می‌ماند

P3 به میزان 7 میلی ثانیه منتظر می ماند

بنابراین در این مثال، میانگین زمان انتظار برابر با

$$6 = \text{<waiting time for p1>}$$

$$4 = \text{<waiting time for p2>}$$

$$7 = \text{<waiting time for p3>}$$

$$6 + 4 + 7 = 17$$

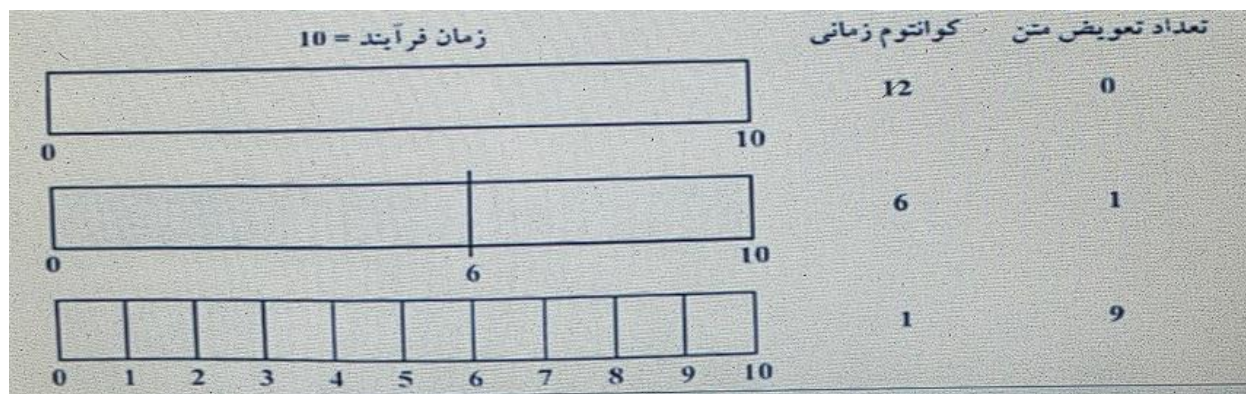
$$17 / 3 = 5.66 = \text{average waiting time}$$

در الگوریتم زمانبندی نوبت گردشی، پردازنده به هیچ فرایندی به طور متوالی بین از یک کوانتوم زمانی تخصیص نمی یابد (مگر این که تنها فرایند قابل اجرا باشد).

اگر زمان انفجار پردازنده فرایندی بیش از یک کوانتوم زمانی شود، آن فرایند قبضه می شود و در انتهای صف آماده قرار می گیرد.

الگوریتم RR، با قبضه کردن (Preemptive) همراه است

اگر n فرایند در صف آماده وجود داشته باشند و کوانتوم زمانی برابر با Q باشد. هر فرایند $1/n$ زمان پردازنده را حداکثر در واحد زمانی و در اختیار می گیرد. هر فرایند برای به دست گرفتن پردازنده در یک کوانتوم زمانی دیگر، نباید بیش از $q * (n-1)$ واحد زمانی منتظر بماند. به عنوان مثال، اگر 5 فرایند وجود داشته باشد و کوانتوم زمانی برابر با 20 میلی ثانیه باشد، هر فرایند حداکثر در هر 100 میلی ثانیه حداکثر 20 میلی ثانیه پردازنده را در اختیار می گیرد.



کارایی الگوریتم RR شدیداً به اندازه کوانتوم زمانی بستگی دارد. از یک طرف اگر کوانتوم زمانی بسیار بزرگ باشد، سیاست RR مثل FCFS خواهد بود. برعکس، اگر کوانتوم کوچک باشد (مثلاً 1 میلی ثانیه)، روش RR می‌تواند تعداد زیادی تعویض متن (Context Switch) ایجاد کند (سر بار زیاد). فرض می‌کنیم فقط یک فرآیند با 10 واحد زمانی داریم. اگر کوانتوم زمانی برابر با 12 واحد زمانی باشد، فرآیند در کمتر از 1 کوانتوم زمانی خاتمه می‌یابد و سربازی ندارد، اگر کوانتوم زمانی برابر با 6 واحد زمانی باشد فرآیند به 2 کوانتوم زمانی نیاز دارد و منجر به یک تعویض متن می‌شود. اگر کوانتوم زمانی برابر با 1 واحد زمان باشد، نیاز به 9 تعویض متن است. و بدین ترتیب، اجرای فرآیند کند می‌شود. بنابراین برای مقابله با تعداد تعویض متن، علاقه مند هستیم که کوانتوم زمانی بزرگ باشد، اگر زمان تعویض متن تقریباً 10 (1/10 - N/1) در صد کوانتوم زمانی باشد، نگاه 10 در صد از زمان پردازنده صرف تعویض متن می‌شود. در عمل اغلب سیستم‌های مدرن دارای زمان کوانتوم بین 10 تا 100 میلی ثانیه هستند. زمان مورد نیاز برای تعویض متن معمولاً کمتر از 10 میلی ثانیه است بنابراین، زمان تعویض متن، کسر کوچکی از کوانتوم زمانی است.

زمان برگشت (Turn Around Time) (زمان کل) نیز به اندازه کوانتوم بستگی دارد.
میانگین زمان برگشت مجموعه‌ای از فرایندها، با افزایش اندازه کوانتوم زمانی، الزاماً

بهبود نمی‌یابد. به طور کلی میانگین زمان برگشت در صورتی می‌تواند بهبود یابد که اغلب فرایندها انفجار بعدی پردازنده خودشان را فقط در یک کوانتوم زمانی به اتمام برسانند. به عنوان مثال:

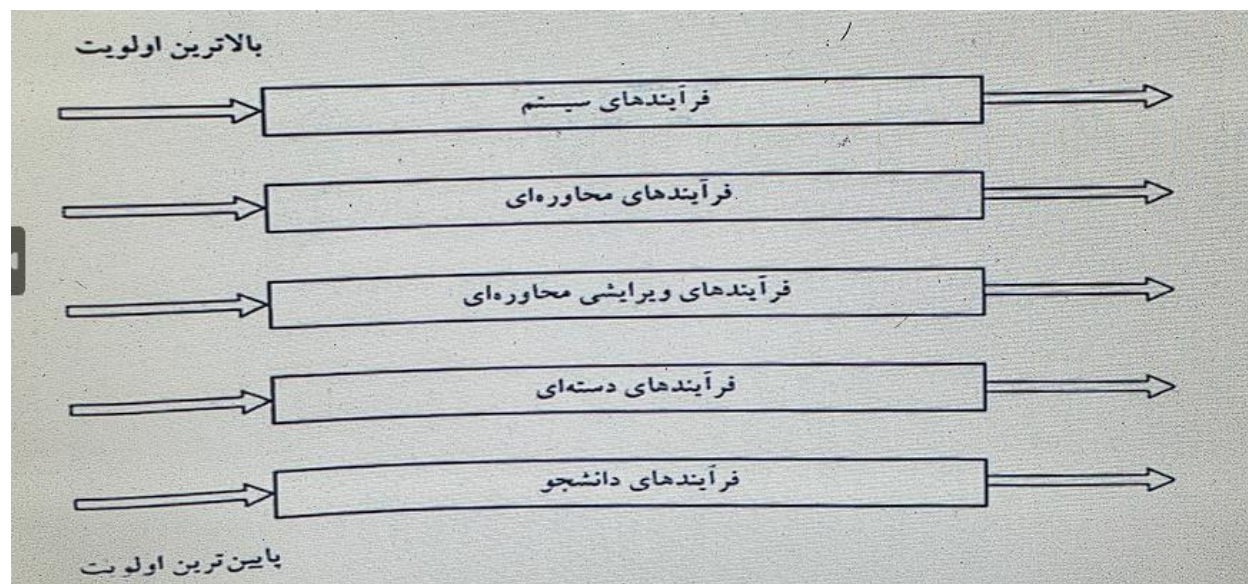
با سه فرایند که هر کدام به 10 واحد زمانی نیاز داشته باشند و کوانتوم زمانی برابر با 1 واحد زمانی باشد، میانگین زمان برگشت 29 است. اگر کوانتوم زمانی برابر با 10 باشد، میانگین زمان برگشت به 20 کاهش می‌یابد. با در نظر گرفتن زمان تعویض متن، هر چه کوانتوم زمانی کوچک‌تر باشد، میانگین زمان برگشت افزایش می‌یابد زیرا نیاز به تعویض متن بیشتری دارد.

گرچه کوانتوم زمانی باید در مقایسه با زمان تعویض متن بزرگ‌تر باشد. اما نیاز نیست بسیار بزرگ باشد، اگر کوانتوم زمانی بسیار بزرگ باشد زمانبندی RR به FCFS تبدیل می‌شود یک حساب سرانگشتی نشان می‌دهد که 80 درصد انفجارهای پردازنده باید کوتاه‌تر از زمان کوانتوم زمانی باشد

زمانبندی صف چند سطحی Multilevel queue

دسته دیگری از الگوریتم‌های زمانبندی برای وضعیت‌هایی ایجاد شدند که در آن‌ها، فرایندها می‌توانند به دو گروه تقسیم شوند. به عنوان مثال، یک تقسیم بندی متداول این است که فرایندها دو دسته اند: فرایندهای پیش زمینه (محوه ای) و پس زمینه (دسته ای). این دو نوع فرایند، زمان پاسخ زمان پاسخ متفاوتی دارند و در نتیجه باید زمانبندی

های متفاوتی داشته باشند. علاوه بر این ممکن است فرایندهای پیش زمینه اولویت بیشتری نسبت به فرایندهای پس زمینه داشته باشند



الگوریتم زمانبندی صف چند سطحی، صف آماده را به چند بخش مجزا تقسیم می‌کند.

هر فرایند بر اساس صفاتی که دارد در صفی قرار می‌گیرد. این صفات عبارت اند از:

اندازه حافظه، اولویت فرایند، نوع فرایند.

- 1.Process Type
- 2.Process priority
- 3.Process memory size

هر صف، الگوریتم زمانبندی خاص خودش را دارد. به عنوان مثال، ممکن است برای فرایندهای پیش زمینه و پس زمینه از صف‌های جداگانه ای استفاده شود و صف پیش زمینه بر اساس الگوریتم RR و صف پس زمینه بر اساس الگوریتم FCFS زمانبندی شود.

علاوه بر این، بین صف‌ها نیز باز زمانبندی وجود داشته باشد که بر اساس زمانبندی همراه با قبضه کردن (Preemptive) و با اولویت ثابت، پیاده سازی می‌شود.

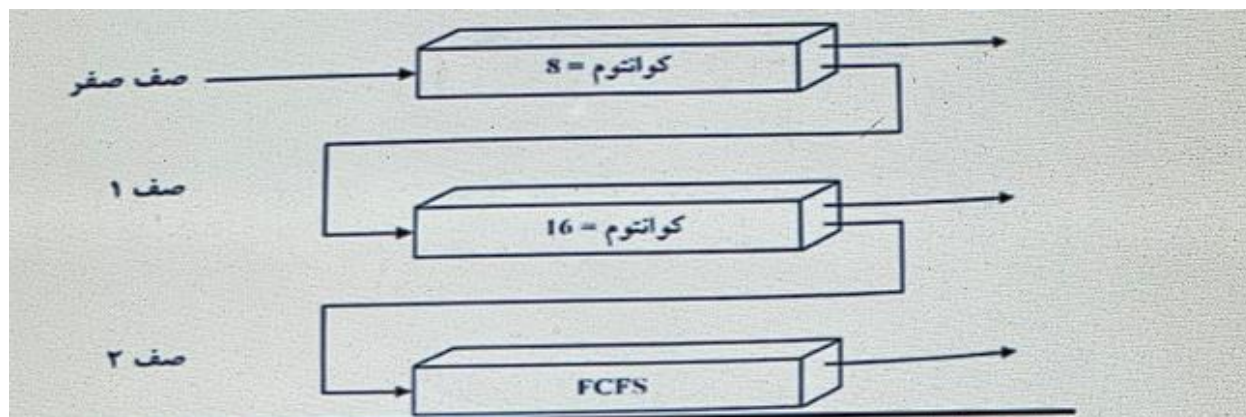
به عنوان مثال: صف پیش زمینه ممکن است نسبت به صف پس زمینه اولویت مطلق (بیشتری) داشته باشد

هر صف نسبت به صف‌های با اولویت پایین‌تر، اولویت مطلق دارد.

امکان دیگر، استفاده از برهه زمانی در بین صف‌هاست. هر صف بخشی از از زمان پردازنده را به خود اختصاص می‌دهد و می‌تواند بین فرایندهای مختلف خود زمانبندی کند. به عنوان نمونه در مثال مربوط به فرایندهای پیش زمینه و پس زمینه، صف پیش زمینه می‌تواند 80 درصد پردازنده را در اختیار گیرد و آن را به روش RR بین فرایندهایش زمانبندی کند، در حالی که صف پس زمینه 20 درصد وقت پردازنده را در اختیار می‌گیرد و آن را به روش FCFS بین فرایندهایش زمانبندی می‌کند.

زمانبندی صف چند سطحی بازخوردی (فیدبک) (feedback)

Multilevel feedback queue



معمولاً در الگوریتم زمانبندی صف چند سطحی، فرایندها هنگام ورود به سیستم در صفی قرار می‌گیرند. به طوری که از صفی به صف دیگر نمی‌رود. به عنوان مثال « اگر صف‌های جداگانه ای برای فرایندهای پیش زمینه و پس زمینه وجود داشته باشد، فرایندها از صفی به صف دیگر منتقل نمی‌شوند، زیرا ماهیت پیش زمینه ای و پس زمینه ای آن‌ها تغییر نمی‌کند. این کار موجب کاهش سربار زمانبندی می‌شود ولی قابلیت انعطاف کم می‌شود

اما، الگوریتم زمانبندی صف چند سطحی بازخوردی به فرایندها اجازه می‌دهد از صفی به صف دیگر منتقل شوند. فلسفه این کار این است که ویژگی‌های انفجارهای پردازنده فرایندها با یکدیگر متفاوت است. اگر فرایندی پردازنده را مدت زیادی در اختیار گیرد، به صفی با اولویت پایین‌تر منتقل می‌شود. بدین ترتیب، فرایندهای در تنگنای i/o و محاوره ای، در صف‌هایی با اولویت بالاتر قرار می‌گیرند. به طور مشابه، فرایندی که به مدت زیادی در صفی با اولویت پایین‌تر منتظر می‌ماند، ممکن است به صفی با اولویت بالاتر منتقل شود. در این شکل سالمندی (Aging) ، از مشکل گرسنگی (قحطی) (starving) جلوگیری می‌شود

به عنوان مثال، یک زمانبند صف چند سطحی بازخوردی، با سه صف را در نظر بگیرید که از صف تا 2 شماره گذاری شده اند.

زمانبند ابتدا تمام فرایندهای موجود در صف را اجرا می‌کند. وقتی صف صفر خالی باشد، فرایندهای صف 1 اجرا می‌شوند. به همین ترتیب، فرایندهای صف 2 وقتی اجرا می‌شوند که صف‌های 0 و 1 خالی باشند. فرایندی که برای صف 1 می‌آید. یک فرایند از صف 2

را قبضه می‌کند. به همین ترتیب، هر فرایند موجود در صف 1، توسط فرایندی که جدیداً وارد صف صفر می‌شود، قبضه (Preemptive) می‌شود

فرایندی که می‌خواهد به صف آماده وارد شود در صف صفر قرار می‌گیرد. به هر فرایند در صف صفر، کوانتوم زمانی 8 میلی ثانیه ای نسبت داده می‌شود. اگر فرایندی در این مدت زمان به اتمام نرسد، به انتهای صف 1 منتقل می‌شود. اگر صف صفر خالی باشد، به فرایند موجود در ابتدای صف 1، کوانتوم زمانی 16 میلی ثانیه ای تخصیص می‌یابد.

اگر اجرای آن در این مدت زمان کامل نشد، قبضه می‌شود و در صف 2 قرار داده می‌شود. در صورتی که هر یک از دو صف 0 و 1 خالی باشند؛ فرایندهای موجود در صف 2 بر اساس الگوریتم FCFS اجرا می‌شوند.

این الگوریتم زمانبندی، به فرایندی که انفجار پردازنده آن کمتر یا مساوی 8 میلی ثانیه ای باشد، بالاترین اولویت را می‌دهد. چنین فرایندی، سریعاً پردازنده را در اختیار می‌گیرد، انفجار پردازنده خودش را انجام می‌دهد و به انفجار بعدی i/o خود می‌رود فرایندهایی که به بیش از (8+16) کوانتوم زمانی نیاز داشته باشند،

گرچه اولویت کمتری نسبت به فرایندهای کوتاه‌تر دارند، ولی سریعاً اجرا می‌شوند. فرایندهای طولانی با زمان بیشتر از 24 کوانتوم زمانی، به طور خودکار به صف 2 می‌روند و به ترتیب FCFS اجرا می‌شوند

(با چرخه‌های پردازنده باقی مانده از صف 0 و 1)

زمانبند صف چند سطحی بازخوردی با پارامترهای زیر تعریف می‌شوند

- تعداد صف‌ها
- الگوریتم زمانبندی برای هر صف
- روشی که تعیین می‌کند چه هنگامی یک فرایند به صفی با اولویت بیشتر منتقل شود.
- روشی که تعیین می‌کند چه هنگامی یک فرایند به صفی با اولویت کمتر منتقل شود.
- روشی که تعیین می‌کند فرایندی که نیاز به خدمات دارد، به چه صفی وارد شود

زمانبند صف چند سطحی بازخوردی، متداول‌ترین الگوریتم زمانبندی پردازنده است.

این الگوریتم، پیچیده‌ترین الگوریتم است، زیرا تعریف بهترین زمان، نیاز به ابزارهایی برای انتخاب مقادیر برای تمام پارامترها دارد.