موسسه بابان

انتشارات بابان و انتشارات راهیان ارشد

درس و کنکور ارشد

سیستم عامل

(مدیریت فرآیندها و زمان‌بندی پردازنده)

ویژه‌ی داوطلبان کنکور کارشناسی ارشد مهندسی کامپیوتر و IT

براساس کتب مرجع

آبراهام سیلبرشاتز، ویلیام استالینگز و اندرو اس تنن‌بام

ارسطو خلیلی‌فر

۲ مدیریت فرآیند

مقدمه

تعاریف مختلفی برای فرآیند ارائه می‌شود، از جمله اینکه فرآیند، برنامه‌ی در حال اجراست. اما می‌دانیم فرآیند ممکن است همیشه در حال اجرا نباشد، مثلاً منتظر باشد. بنابراین به بیان تعریف دیگری می‌پردازیم:

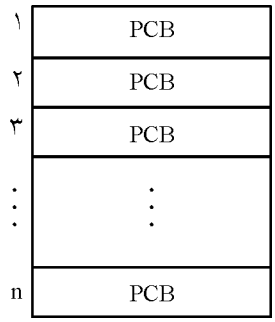
از لحظه‌ای که یک کار توسط زمانبند کار انتخاب و وارد گردونه‌ی مراحل مختلف می‌شود، تا لحظه‌ای که اجرای آن به طور کامل خاتمه یافته و از سیستم خارج می‌شود، **فرآیند (Process)** نام دارد.

نکته: یک برنامه به خودی خود فرآیند نیست. زیرا برنامه یک موجودیت غیر فعال (Passive) است که مثلاً در قالب یک فایل و بر روی دیسک ذخیره شده است. اما فرآیند یک موجودیت فعال (Active) می‌باشد که در حال اجراست.

نکته: یک فرآیند در واقع شامل اجزایی مانند رجیسترهای CPU (از جمله شمارنده‌ی برنامه (PC)، یک پشته در حافظه (جهت نگهداری داده‌های موقتی، متغیرهای محلی، پارامترهای توابع و زیرروال‌ها) و بخش داده‌ای (Data Segment) (شامل داده‌ها و متغیرهای سراسری) است.

توجه: شمارنده برنامه یا PC (program counter)، آدرس دستورالعمل بعدی فرآیند را نشان می‌دهد.

نکته: سیستم عامل اطلاعات مربوط به فرآیندها را در جدولی با عنوان Process Table درج و نگهداری می‌کند. در این جدول به هر فرآیند یک درایه (رکورد) اختصاص داده می‌شود که به آن PCB (Process Control Block) گویند.



جدول فرآیند

نکته: PCB در واقع یک ساختمان داده جهت نگهداری اطلاعات مربوط به فرآیندها نزد سیستم عامل است که در آن اطلاعاتی از جمله موارد زیر برای هر فرآیند ذخیره می‌شود:

ـ شناسه فرآیند (Process ID)

ـ شناسه‌ی کاربر فرآیند یا حتی شناسه‌ی گروهی که کاربر فرآیند عضو آن است (UID,GID)

ـ اولویت فرآیند

ـ رجیسترهای CPU (شامل PC ، PSW و مابقی رجیسترها)

ـ حالت و وضعیت فرآیند (مانند جدید، آماده، در حال اجرا، منتظر و معلق)

ـ اطلاعاتی راجع به منابع در اختیار فرآیند (فایل‌ها، حافظه، پردازنده)

ـ اطلاعاتی راجع به میزان مصرف منابع توسط فرآیند (مانند میزان پردازنده‌ی مصرفی)

ـ اطلاعات مربوط به زمانبندی

ـ اشاره‌گرهایی به قسمت کد، داده و پشته فرآیند (CS ، DS و SP)

ـ اطلاعات مربوط به مدیریت حافظه (جداول صفحه، قطعه و مسائل حفاظتی)

در واقع PCB مانند یک مخزن است که اطلاعاتی که از یک فرآیند به فرآیند دیگر متغیر است، در آن ذخیره می‌شود.

توجه: رجیستر PSW (program status word) : در این رجیستر مُد کاربر یا هسته مربوط به پردازنده نگهداری می‌شود. اگر صفر باشد، پردازنده در مُد هسته و اگر یک باشد، پردازنده در مُد کاربر قرار دارد.

سرگذشت فرآیند

یک فرآیند طی مدت زمانی که در سیستم وجود دارد، می‌تواند حالت‌های زیر را داشته باشد:

حالت اول) جدید (New): فرآیند در حال ایجاد شدن است.

حالت دوم) آماده (Ready): فرآیند درون حافظه قرار دارد و همه منابع مورد نیاز به جز CPU در اختیار آن است (در واقع منتظر است تا نوبت CPU به آن برسد).

حالت سوم) در حال اجرا (Running): فرآیند CPU را در اختیار دارد و در حال اجراست (بالطبع همه منابع موردنیاز دیگر نیز در اختیار آن می‌باشد).
حالت چهارم) در حال انتظار (Waiting): در این حالت فرآیند منتظر وقوع یک رویداد (Event) است (برای مثال منتظر اختصاص یک منبع (به جز CPU) یا تکمیل شدن یک عمل I/O). در واقع وقتی فرآیندی در حالت انتظار است، نیازی به CPU ندارد. به این حالت مسدود (Blocked) نیز می‌گویند.
حالت پنجم) خاتمه یافته (Terminated): اجرای فرآیند پایان یافته است.
نکته: در حالت‌های جدید، آماده و انتظار، صفی برای فرآیندها وجود دارد، اما در حالت اجرا صف نداریم چون در هر لحظه فقط یک فرآیند پردازنده را در اختیار دارد و در حال اجراست.

تغییر حالات ممکن برای یک فرآیند

۱ـ از New به Ready
اگر سیستم عامل آماده دریافت یک فرآیند باشد، فرآیند از حالت جدید به حالت آماده منتقل می‌شود. این انتقال توسط زمانبند کار (Job Scheduler) انجام می‌شود.

۲ـ از Ready به Running
در این حالت پردازنده به یک فرآیند تخصیص داده می‌شود تا اجرا گردد. این کار توسط زمانبند فرآیند (Procdess Scheduler) انجام می‌شود.

۳ـ از Running به Ready
در این حالت پردازنده از فرآیند گرفته شده و فرآیند دوباره به صف فرآیندهای آماده منتقل می‌شود.
نکته: CPU به یکی از دلایل زیر می‌تواند از یک فرآیند گرفته شود:
ـ فرآیند به طور اختیاری پردازنده را رها کند (مثلاً به I/O نیاز پیدا می‌کند).
ـ سهمیه فعلی اجرای فرآیند به پایان برسد.
ـ یک فرآیند با اولویت بالاتر به پردازنده نیاز داشته باشد.

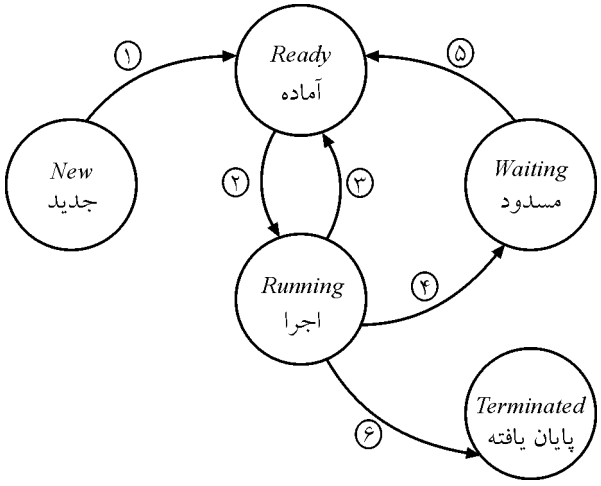
۴ـ از Running به Waiting (Blocked)
این تغییر حالت وقتی رخ می‌دهد که یک فرآیند در حال اجرا، منتظر وقوع یک رویداد است، برای مثال به I/O نیاز پیدا می‌کند.

۵ـ از Waiting (Blocked) به Ready
وقتی رویدادی که فرآیند منتظر آن است (و به خاطر آن مسدود شده است) رخ می‌دهد، فرآیند از حالت مسدود به آماده می‌رود.

۶ـ از Running به Terminated

وقتی فرآیندی به اتمام برسد، از حالت اجرا به پایان یافته منتقل می‌شود.

نکته: تغییر حالات ممکن برای یک فرآیند در شکل زیر نشان داده شده است (شماره‌های روی پیکان‌ها نشان دهنده تغییر حالت‌های مذکور می‌باشند):



تغییر حالت فرآیند ـ نمودار ۵ حالته

نکته: بیشتر سیستم عامل‌ها حالت‌های فرآیند را فقط محدود به پنج حالتِ جدید، اجرا، آماده، مسدود و پایان یافته نمی‌کنند، بلکه یک موقعیت جدید با عنوان معلق (Suspend) نیز برای فرآیند در نظر می‌گیرند. در حالت معلق فرآیند از حافظه به روی دیسک بیرون رانده شده است. به عنوان مثال فرض کنید همه فرآیندهای موجود در سیستم به حالت مسدود نقل مکان کنند (و همگی مثلاً منتظر اتمام ورودی ـ خروجی مختص به خود هستند)، بنابراین هیچ فرآیندی در صف آماده قرار ندارد. در این حالت سیستم عامل جهت بیکار نماندن CPU، می‌تواند یک فرآیند را که در حالت مسدود است، از حافظه به دیسک منتقل کرده و یک فرآیند جدید را وارد صف آماده کند.

نکته: با این توضیحات می‌توان دو حالت جدید برای فرآیند در نظر گرفت:

حالت ششم) منتظر و معلق (Suspend Wait) : یک فرآیند که در حالت منتظر (مسدود) قرار دارد از حافظه به دیسک منتقل می‌شود. در واقع در این حالت منبع حافظه نیز از فرآیند گرفته شده است. به این حالت مسدود و معلق (Suspend Blocked) نیز می‌گویند.

حالت هفتم) آماده و معلق (Suspend Ready): یک فرآیند که بر روی دیسک است، به محض برگشتن به حافظه، می‌تواند اجرا شود. یک فرآیند در دو صورت در این حالت به سر می‌برد:

الف ـ به علت کمبود حافظه یک فرآیند مستقیماً از حالت آماده، به روی دیسک برده شود.

ب ـ یک فرآیند که بر روی دیسک است و در حالت مسدود و معلق به سر می‌برد، حادثه موردنظرش

رخ دهد (مانند اتمام I/O).
با در نظرگرفتن حالات ۶ و ۷، تغییر حالت‌های دیگری را نیز می‌توان برای یک فرآیند در نظرگرفت.

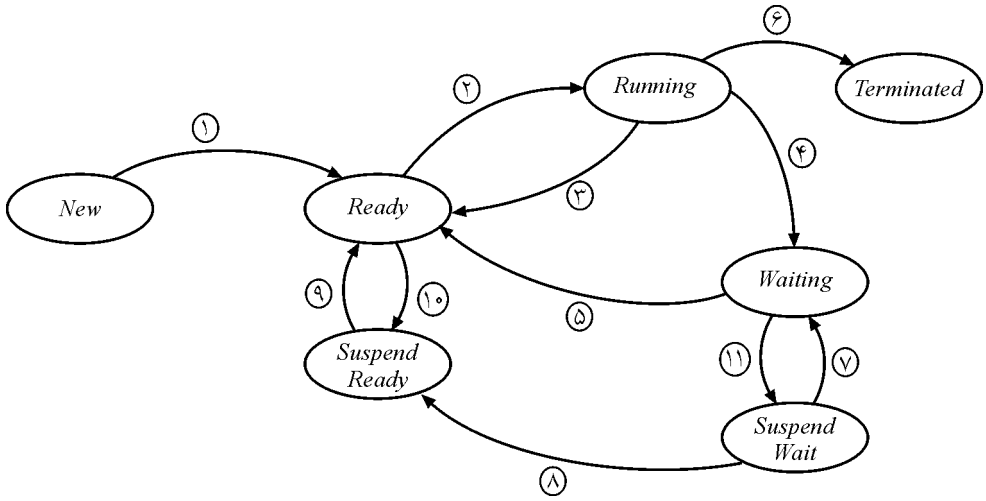
۷ـ از (Blocked) Waiting به (Suspend Blocked) Suspend Wait
هنگامی که هیچ فرآیندی، در صف آماده وجود نداشته باشد و همه فرآیندها مسدود (منتظر) باشند، سیستم عامل یک فرآیند را از صف مسدود انتخاب کرده و به روی دیسک منتقل می‌کند تا فضای موردنیاز برای فرآیند دیگری (جدید یا در حالت آماده و معلق) مهیا شود.

۸ـ از (Suspend Blocked) Suspend Wait به Suspend Ready
وقتی رویدادی که یک فرآیند مسدود و معلق منتظر آن بوده، رخ دهد (مانند تکمیل I/O) این فرآیند از حالت مسدود و معلق به حالت آماده و معلق منتقل می‌شود (دقت کنید فرآیند هنوز بر روی دیسک قرار دارد).

۹ـ از Suspend Ready به Ready
وقتی هیچ فرآیند آماده‌ای در سیستم موجود نباشد، اگر فرآیندی در حالت معلق و آماده وجود داشته باشد، سیستم عامل آن را به حافظه منتقل می‌کند.
از طرفی اگر اولویت فرآیندی که در حالت آماده و معلق قرار دارد از همه فرآیندهای آماده، بالاتر باشد، سیستم عامل این فرآیند را به حافظه منتقل کرده و به حالت آماده (و احتمالاً بعد از آن به حالت اجرا) می‌برد.

۱۰ـ از Ready به Suspend Ready
احتمال وقوع این تغییر حالت بسیار کم است اما اگر فرآیندی که در حال اجراست به فضای حافظه بیشتری نیاز داشته باشد، سیستم عامل جهت خالی شدن حافظه باید یک فرآیند را به دیسک منتقل کند، حال اگر هیچ فرآیندی در حالت مسدود نباشد، یک فرآیند آماده، برای این کار انتخاب و به دیسک منتقل می‌شود.

۱۱ـ از (Suspend Blocked) Suspend Wait به (Blocked) Waiting
اگر حافظه خالی شده و فرآیندی در حالت مسدود و معلق وجود داشته باشد، سیستم عامل فرآیندهای مسدود و معلقی را که بر روی دیسک هستند به حافظه منتقل می‌کند.
با این توضیحات، نمودار تغییر حالات یک فرآیند به صورت زیر است. (شماره‌های روی پیکان‌ها نشان‌دهنده تغییر حالت‌های مذکور می‌باشند):



تغییر حالت فرآیند ـ نمودار ۷ حالته

زمانبندها

با توجه به تغییر حالات فرآیندها، زمانبندها از یک دیدگاه به ۳ دسته تقسیم می‌شوند:

۱ـ زمانبند بلند مدت (Long Term Scheduler)

این زمانبند درجه چند برنامگی را مشخص می‌کند. در واقع از بین کارها، تعدادی را انتخاب کرده و به فرآیند تبدیل می‌کند. به این زمانبند، زمانبند کار (Job Scheduler) نیز می‌گویند.

۲ـ زمانبند میان مدت (Middle Term Scheduler)

این زمانبند بنا به دلایلی ممکن است فرآیندهایی را از روی حافظه، به دیسک و بالعکس منتقل کند. در واقع وظیفه این زمانبند جابه‌جا کردن فرآیندها بین حافظه و دیسک می‌باشد. به این کار مبادله (Swapping) گویند.

۳ـ زمانبند کوتاه مدت (Short Term Scheduler)

این زمانبند از بین فرآیندهای آماده در حافظه، یکی را جهت اجرا توسط پردازنده انتخاب می‌کند. از این رو به آن زمانبند فرآیند (Process Scheduler) نیز می‌گویند.

نکته: درکتاب پروفسور تنن باوم

به زمانبند بلند مدت، زمانبند پذیرش (Admission Scheduler)،

به زمانبند میان مدت، زمانبند حافظه (Memory Scheduler) و

به زمانبند کوتاه مدت، زمانبند پردازنده (CPU Scheduler) نیز اطلاق می‌شود.

نکته: زمانبند بلند مدت باید انتخاب‌های خود را بسیار دقیق و سنجیده انجام دهد. برای مثال باید

همواره ترکیب مناسبی از فرآیندهای CPU Limited و I/O Limited را به سیستم وارد کند تا تعادل برقرار باشد.

نکته: سوئیچ کردن پردازنده بین فرآیندهای مختلف، تعویض متن (Context Switch) نام دارد. این عمل نیازمند ذخیره کردن حالت فرآیند قبلی و بارگذاری حالت فرآیند فعلی است. از آن جا که از نظر اجرایی، کار مفیدی صورت نمی‌گیرد، این عمل یک زمان سربار به حساب می‌آید.

نکته: عمل تعویض متن توسط بخشی از سیستم عامل به نام Dispatcher صورت می‌گیرد. در واقع Dispatcher ماژولی است که کنترل CPU را به فرآیندی که توسط زمانبند کوتاه مدت انتخاب شده است، اعطا می‌کند.

زمانبندی فرآیندها

وظیفه زمانبند فرآیند، انتخاب یک فرآیند جهت در اختیار گرفتن پردازنده است (در واقع مشخص می‌کند که در لحظه حاضر، پردازنده در اختیار کدام فرآیند باشد).

اهداف الگوریتم‌های زمانبندی

الگوریتم‌های زمانبندی اهداف زیر را دنبال می‌کنند (در واقع موارد زیر معیارهایی جهت مقایسه عملکرد الگوریتم‌های مختلف می‌باشند):

عدالت (Fairness)

منظور از عدالت این است که فرآیندهای هم‌ارز باید از دید زمانبند، یکسان تلقی شوند. به عبارت دیگر یک فرآیند نباید از فرآیندی هم سطح خود سهم بیشتری از CPU دریافت کند.

البته این امر درباره فرآیندهایی که اولویت یکسان ندارند صادق نیست و منطقی است که فرآیندی با اولویت بالاتر، سهم بیشتری از CPU دریافت کند.

زمان پاسخ (Response)

یک الگوریتم زمانبندی باید زمان پاسخ فرآیندها را به حداقل برساند. منظور از زمان پاسخ، مدت زمان بین لحظه ورود کار و لحظه شروع پاسخ می‌باشد. این معیار در سیستم‌های محاوره‌ای و تعاملی اهمیت ویژه‌ای دارد.

زمان گردش (برگشت) کار (Turnaround Time)

یک الگوریتم زمانبندی باید زمان گردش کار فرآیندها را به حداقل برساند. منظور از زمان گردش کار، مدت زمان بین لحظه ورود کار و لحظه خروج کامل آن از سیستم می‌باشد.

نکته: به تفاوت بین زمان پاسخ و زمان گردش کار دقت کنید:

زمان پاسخ، مدت زمان بین صدور فرمان و تولید اولین پاسخ آن می‌باشد اما زمان گردش کار، مدت

زمان بین ورود کار و تکمیل نهایی آن است.

زمان انتظار (Waiting Time)

یک الگوریتم زمانبندی باید زمان انتظار فرآیندها را به حداقل برساند. منظور از زمان انتظار، مجموع زمان‌هایی است که یک فرآیند در صف فرآیندهای آماده، منتظر دریافت CPU می‌باشد.

واضح است که یک الگوریتم زمانبندی، بر روی مدت زمان اجرای یک فرآیند و مدت زمان ورودی ـ خروجی تأثیری ندارد بلکه فقط بر روی مدت زمان انتظار یک فرآیند در صف آماده مؤثر است.

بهره‌وری پردازنده (CPU Efficiency **یا** CPU Utilization)

یک الگوریتم زمانبندی باید بهره‌وری پردازنده را به حداکثر برساند، به این معنی که حتی‌الامکان در تمام زمان‌ها پردازنده مشغول باشد تا زمان‌های هدر رفته پردازنده به حداقل برسد.

توان گذردهی (توان عملیاتی) (Throughput)

یک الگوریتم زمانبندی باید توان گذردهی را به حداکثر برساند. منظور از گذردهی، تعداد فرآیندهایی است که در واحد زمان تکمیل می‌شوند.

نکته: علاوه بر معیارهای شش‌گانه ذکر شده، معیارهای زیر نیز کم و بیش برای مقایسه الگوریتم‌های زمانبندی به کار می‌روند:

ـ **تعادل (توازن) در استفاده از منابع**: در واقع الگوریتم زمانبندی باید باعث شود از تمام منابع سیستم به خوبی استفاده و بهره‌برداری شود. به عبارت دیگر منابع را مشغول نگه دارد.

ـ **پیش‌بینی‌پذیری**: الگوریتم زمانبندی باید به گونه‌ای عمل کند که در اجراهای مختلف و چندباره یک کار، به خصوص، زمان انتظار و زمان پاسخ تغییرات شدیدی نداشته باشند. این معیار برای کاربران اهمیت دارد.

ـ **رعایت اصول و اولویت‌ها**

ـ **رعایت ضرب‌العجل‌ها**

نکته: برخی از معیارها و اهداف فوق با هم محقق نمی‌شوند و ممکن است با یکدیگر در تضاد باشند. به عنوان مثال ممکن است جهت برقراری عدالت سیاستی اعمال شود که بهره‌وری CPU را کاهش دهد.

نکته: الگوریتم‌های زمانبندی به دو دسته تقسیم می‌شوند:

۱ـ Non Preemptive (غیرقابل پس‌گیری، انحصاری)

۲ـ Preemptive (قابل پس‌گیری، غیرانحصاری)

در زمانبندی انحصاری، هنگامی که پردازنده در اختیار فرآیندی قرار گیرد نمی‌توان پردازنده را از وی

پس گرفت، مگر این که خود داوطلبانه آن را آزاد کند و یا فرآیند خاتمه یابد اما در زمانبندی غیرانحصاری می‌توان پردازنده را از فرآیندی پس گرفت و به دیگری تحویل داد.

نکته: در سیستم‌های تعاملی، الگوریتم‌های غیرانحصاری (قابل پس‌گیری) تنها گزینه قابل قبول هستند، زیرا در غیر این صورت یک فرآیند CPU را به انحصار خود درآورده و به دیگران اجازه کار نمی‌دهد. شاید به نظر برسد برای سیستم‌های بلادرنگ تنها باید از الگوریتم‌های غیرانحصاری (قابل پس‌گیری) استفاده کرد، اما در این سیستم‌ها از هر دو نوع الگوریتم می‌توان بهره برد. از آنجا که فرآیندها در سیستم‌های بلادرنگ اغلب بسیار کوچک هستند و به سرعت اجرا شده و خاتمه می‌یابند (یا مسدود می‌شوند)، الگوریتم‌های انحصاری (غیرقابل پس‌گیری) نیز در این سیستم‌ها کاربرد دارند.

الگوریتم‌های زمانبندی پردازنده (CPU)

الگوریتم‌های متنوعی جهت زمانبندی پردازنده وجود دارند که برخی از آنها به قرار زیرند:

- FCFS
- SJF (یا SPN یا SPT)
- RR
- SRT (یا SRTF یا SRPT یا SRTN)
- HRRN یا HRN
- Priority
- MLQ
- MLFQ
- LPT
- Lottery
- Guaranteed
- FSS

نکته: برای مطرح کردن، تحلیل و بررسی مسائل مربوط به زمانبندی پردازنده از تعاریف زیر استفاده می‌کنیم:

۱ـ زمان ورود

لحظه‌ای که یک فرآیند به لیست فرآیندهای آماده سیستم اضافه می‌شود.

۲ـ زمان اجرا

مدت زمانی که یک فرآیند به پردازنده نیاز دارد که به آن زمان سرویس یا زمان انفجار محاسباتی

(CBT :CPU Burst Time) نیز می‌گویند.

۳ـ میانگین زمان انتظار

میانگین طول مدت زمانی که فرآیندها در صف قرار داشته، آماده‌اند و منتظر دریافت پردازنده هستند.

۴ـ میانگین زمان پاسخ

برابر میانگین زمان اجرای فرآیندها به علاوه زمان انتظار آنهاست.

نکته: در برخی از الگوریتم‌های زمانبندی، مشکلی با عنوان قحطی‌زدگی (گرسنگی) (Starvation) پدید می‌آید، به این ترتیب که ممکن است اجرای یک یا چند فرآیند متناوباً به تعویق بیفتد و این روند می‌تواند تا بی‌نهایت ادامه یابد یعنی ممکن است هیچ‌گاه نوبت به اجرای این فرآیندها نرسد.

الگوریتم FCFS (First Come First Served)

این الگوریتم ساده‌ترین الگوریتم زمانبندی پردازنده است. در این روش کارها با همان ترتیب ورود به سیستم، در یک صف قرار گرفته و از ابتدای صف به ترتیب، پردازنده را در اختیار می‌گیرند.

نکته: این الگوریتم، FIFO (First In First Out) نیز نامیده می‌شود.

نکته: FCFS یک الگوریتم **انحصاری (Non Preemptive)** است.

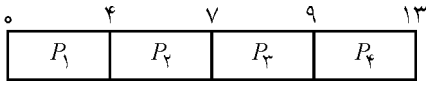
نکته: الگوریتم FIFO مشکل **قحطی‌زدگی ندارد**.

مثال: چهار فرآیند p_1، p_2، p_3 و p_4 را در نظر بگیرید که به ترتیب از p_1 تا p_4 وارد شده‌اند (همگی در لحظه ۰ و با اختلاف اندک وارد شده‌اند). با توجه به جدول زیر، میانگین زمان انتظار برای این چهار فرآیند چقدر است؟

فرآیند	مدت زمان اجرا (ms)
p_1	۴
p_2	۳
p_3	۲
p_4	۴

حل:

فرآیندها به ترتیب p_1، p_2، p_3 و p_4 وارد شده‌اند، بنابراین داریم:



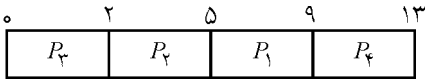
به نمودار فوق، نمودار گانت (Gantt Chart) گویند.

با توجه به نمودار فوق، فرآیند p_1 به محض ورود، پردازنده را در اختیار می‌گیرد، بنابراین زمان انتظار آن

برابر صفر است. فرآیند $p_۲$، ۴ میلی‌ثانیه، فرآیند $p_۳$، ۷ میلی‌ثانیه و فرآیند $p_۴$، ۹ میلی‌ثانیه منتظر می‌مانند. به این ترتیب میانگین زمان انتظار فرآیندها برابر است با:

$$\frac{۰ + ۴ + ۷ + ۹}{۴} = ۵ \text{ میلی ثانیه}$$

دقت کنید که اگر ترتیب ورود فرآیندها به صورت $p_۳$، $p_۲$، $p_۱$ و $p_۴$ بود مسئله به این صورت تبدیل می‌شد:



و میانگین زمان انتظار برابر بود با:

$$\frac{۰ + ۲ + ۵ + ۹}{۴} = ۴ \text{ میلی ثانیه}$$

الگوریتم SJF (Shortest Job First)

در این روش ابتدا کاری برای اجرا انتخاب می‌شود که از همه کوتاهتر باشد (زمان اجرای کمتری داشته باشد).

نکته: این الگوریتم، **SPN** (Shortest Process Next) و **SPT** (Shortest Processing Time) نیز نامیده می‌شود.

نکته: **SJF** یک الگوریتم **انحصاری** (Non Preemptive) است.

نکته: یک نقص عمده الگوریتم SJF این است که ممکن است باعث قحطی‌زدگی فرآیندهای طولانی شود. به این ترتیب که اگر همواره تعدادی فرآیند کوچک وارد سیستم شوند، اجرای فرآیندهای بزرگ به طور متناوب به تعویق می‌افتد. این روال حتی می‌تواند تا بی‌نهایت ادامه یابد و هیچگاه نوبت به اجرای فرآیندهای بزرگ نرسد!!!

نکته: در این روش اگر دو فرآیند مدت زمان اجرای برابر داشته باشند، براساس FCFS زمانبندی می‌شوند.

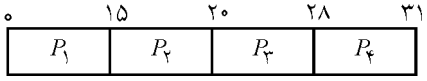
مثال: چهار فرآیند زیر را در نظر بگیرید.

فرآیند	CBT	زمان ورود
$p_۱$	۱۵	۰
$p_۲$	۵	۰
$p_۳$	۸	۰
$p_۴$	۳	۰

میانگین زمان انتظار و میانگین زمان پاسخ را برای این چهار فرآیند و با دو روش FCFS و SJF به دست

آورید (برای الگوریتم FCFS فرض کنید فرآیندها به ترتیب نامشان وارد می‌شوند).

حل: نمودار گانت برای الگوریتم FCFS به صورت زیر است:



بنابراین زمان انتظار فرآیندها در FCFS برابرند با:

p_1 = زمان انتظار ۰

p_2 = زمان انتظار ۱۵

p_3 = زمان انتظار ۲۰

p_4 = زمان انتظار ۲۸

و میانگین زمان انتظار در FCFS برابر است با:

$$\frac{۰ + ۱۵ + ۲۰ + ۲۸}{۴} = ۱۵/۷۵$$

زمان پاسخ فرآیندها در FCFS برابرند با:

p_1 = زمان پاسخ ۱۵

p_2 = زمان پاسخ ۲۰

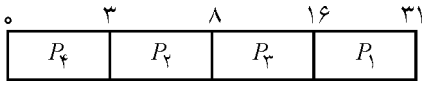
p_3 = زمان پاسخ ۲۸

p_4 = زمان پاسخ ۳۱

و میانگین زمان پاسخ در FCFS برابر است با:

$$\frac{۱۵ + ۲۰ + ۲۸ + ۳۱}{۴} = ۲۳/۵$$

همچنین برای الگوریتم SJF داریم:



بنابراین زمان انتظار فرآیندها در SJF برابرند با:

p_1 = زمان انتظار ۱۶

p_2 = زمان انتظار ۳

p_3 = زمان انتظار ۸

p_4 = زمان انتظار ۰

و میانگین زمان انتظار در SJF برابر است با:

$$\frac{۱۶ + ۳ + ۸ + ۰}{۴} = ۶/۷۵$$

زمان پاسخ فرآیندها در SJF برابر است با:

p_1 = زمان پاسخ ۳۱
p_2 = زمان پاسخ ۸
p_3 = زمان پاسخ ۱۶
p_4 = زمان پاسخ ۳

و میانگین زمان پاسخ در SJF برابر است با:

$$\frac{۳۱ + ۸ + ۱۶ + ۳}{۴} = ۱۴/۵$$

نکته: هدف الگوریتم SJF به حداقل رساندن میانگین زمان انتظار، میانگین زمان پاسخ و میانگین زمان گردش کار فرآیندهاست.

نکته: در عمل نمی‌توان الگوریتم SJF را پیاده سازی کرد، زیرا سیستم عامل زمان اجرای فرآیندها را از قبل نمی‌داند و تنها کاری که می‌تواند انجام دهد این است که زمان اجرای فرآیندها را فقط حدس زده و به طور تقریبی به دست آورد.

الگوریتم RR (Round Robin)

الگوریتم RR (نوبت چرخشی) یکی از پرکاربردترین الگوریتم‌ها در سیستم‌های اشتراک زمانی است. درباره‌ی نحوه‌ی عملکرد آن می‌توان گفت این الگوریتم نسخه‌ی **غیر انحصاری** (Preemptive) الگوریتم FCFS می‌باشد. در این الگوریتم زمان پردازنده را به برش‌های زمانی کوتاهی (Time Slice) تقسیم می‌کنیم. همانند الگوریتم FCFS، فرآیندهایی که به سیستم تحویل داده می‌شوند به انتهای یک صف وارد می‌شوند. سپس پردازنده از ابتدای صف شروع کرده و به هر فرآیند حداکثر به اندازه‌ی یک برش زمانی سرویس می‌دهد. در واقع پس از اینکه برش زمانی یک فرآیند به پایان رسید، پردازنده آن فرآیند را رها کرده و به سراغ فرآیند بعدی موجود در صف می‌رود. این عمل آنقدر تکرار می‌شود تا پردازنده به انتهای صف فرآیندهای آماده برسد.

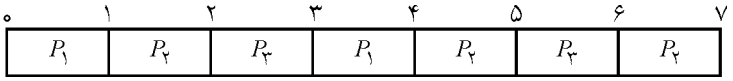
به عبارت دیگر فرآیندها در یک صف دایره‌ای شکل سازماندهی می‌شوند و پردازنده به صورت دوار در این صف حرکت کرده و به هر فرآیند فقط به اندازه‌ی حداکثر یک برش زمانی سرویس می‌دهد.

نکته: به برش زمانی، کوانتوم زمانی (Quantum Time) نیز می‌گویند.

مثال: سه فرآیند p_1، p_2 و p_3 را در نظر بگیرید. با استفاده از الگوریتم RR و با برش زمانی ۱ms، میانگین زمان انتظار و میانگین زمان پاسخ فرآیندها را به دست آورید (فرض کنید فرآیندها همه در لحظه‌ی صفر و به ترتیب نام وارد شده‌اند).

فرآیند	CBT	زمان ورود
$p_۱$	۲	۰
$p_۲$	۳	۰
$p_۳$	۲	۰

حل: با استفاده از الگوریتم RR و با برش زمانی ۱ ms، نمودار گانت به صورت زیر است:



بنابراین داریم:

فرآیند	زمان پاسخ	زمان انتظار
$p_۱$	۴	۲
$p_۲$	۷	۴
$p_۳$	۶	۴

$$\text{میانگین زمان انتظار} = \frac{۲+۴+۴}{۳} = ۳/۳$$

$$\text{میانگین زمان پاسخ} = \frac{۴+۷+۶}{۳} = ۵/۶$$

نکته: الگوریتم RR مشکل **قحطی زدگی ندارد**.

نکته: اگر حین اجرای الگوریتم RR، یک فرآیند تازه وارد و یک فرآیند قدیمی، هر دو به انتهای صف آماده برسند، فرآیندی که تازه وارد شده، جلوتر و فرآیند قدیمی، در انتها قرار می‌گیرد. **البته این مسأله در همه‌ی سیستم عامل‌ها رعایت نمی‌شود**.

نکته: یکی از اهدافی که الگوریتم RR کمابیش به آن نزدیک می‌شود، رعایت عدالت و مساوات است.

نکته: الگوریتم RR در سیستم‌های اشتراکی زمانی به خوبی استفاده می‌شود. در واقع با این الگوریتم کاربران تصور نمی‌کنند که همه‌ی فرآیندها به موازات هم در حال اجرا هستند!

نکته: برای به دست آوردن زمان پاسخ یک فرآیند، کافی است فاصله بین زمان خروج فرآیند و زمان ورود آن را به دست آورد و برای به دست آوردن زمان انتظار کافی است زمان اجرای یک فرآیند را از زمان پاسخ آن کم کرد.

زمان ورود – زمان خروج = زمان پاسخ

زمان اجرا – زمان پاسخ = زمان انتظار

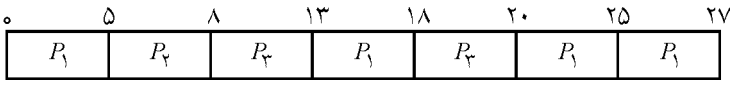
نکته: اگر کار فرآیندی پیش از پایان برش زمانی، به پایان برسد، داوطلبانه پردازنده را در نیمه‌ی برش

زمانی رها می‌کند و پردازنده در اختیار فرآیند بعدی قرار می‌گیرد.

مثال: سه فرآیند زیر را در نظر بگیرید. با استفاده از الگوریتم RR و با برش زمانی ۵ میلی‌ثانیه، میانگین زمان پاسخ و میانگین زمان انتظار را محاسبه کنید.

فرآیند	CBT	زمان ورود
$p_۱$	۱۷	۰
$p_۲$	۳	۱
$p_۳$	۷	۲

حل:



و برای محاسبه‌ی میانگین زمان پاسخ و میانگین زمان انتظار داریم:

فرآیند	زمان پاسخ	زمان انتظار
$p_۱$	۲۷−۰=۲۷	۲۷−۱۷=۱۰
$p_۲$	۸−۱=۷	۷−۳=۴
$p_۳$	۲۰−۲=۱۸	۱۸−۷=۱۱

$$\text{میانگین زمان پاسخ} = \frac{۲۷ + ۷ + ۱۸}{۳} = ۱۷/۳$$

$$\text{میانگین زمان انتظار} = \frac{۱۰ + ۴ + ۱۱}{۳} = ۸/۳$$

نکته: فرآیند تعویض متن (Context Switch) یک زمان تلف شده برای سیستم می‌باشد. به همین دلیل اندازه‌ی آن باید نسبت به طول برش زمانی بسیار کوچکتر باشد تا کارایی سیستم کاهش نیابد (در بیشتر مسائل زمان تعویض متن را نادیده می‌گیرند).

نکته: در الگوریتم RR، اگر برش زمانی بسیار بزرگ در نظر گرفته شود، این الگوریتم به FCFS تبدیل می‌شود.

نکته: افزایش اندازه‌ی برش زمانی بر روی میانگین زمان پاسخ فرآیندها تأثیر می‌گذارد. اما این تأثیر گاهی کاهش میانگین زمان پاسخ است و گاهی افزایش آن! بنابراین لزوماً **نمی‌توان گفت** افزایش اندازه‌ی برش زمانی، میانگین زمان پاسخ فرآیندها را افزایش یا کاهش می‌دهد، بلکه باید در هر مورد محاسبه

شود.

نکته: حد پایین یک برش زمانی توسط دو عامل مشخص می‌شود:

۱) طول مدت زمان تعویض متن، یک برش زمانی نباید آنقدر کوچک باشد که هزینه‌های تعویض متن بر کارایی سیستم غلبه کند.

۲) برش زمانی باید کمی بزرگتر از زمان لازم برای یک فعل و انفعال نوعی باشد، زیرا در غیر این صورت هر کاری احتیاج به حداقل دو برش زمانی خواهد داشت.

نکته: اگر اندازه‌ی برش زمانی، نسبت به مدت زمان اجرای فرآیندها بسیار کوچک باشد، این گونه به نظر می‌رسد که زمان CPU بین همه‌ی فرآیندها به طور مساوی به اشتراک گذاشته شده است و هر فرآیند کسری از CPU را در اختیار دارد.

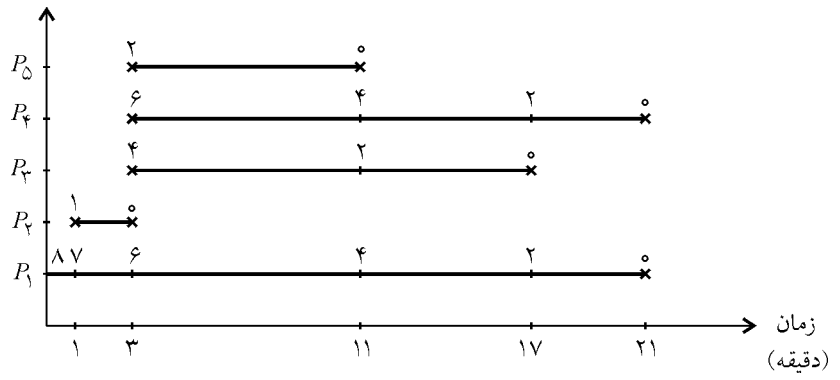
مثال: فرض کنید پنج فرآیند p_1 تا p_5 در یک سیستم وجود داشته باشند. اگر در این سیستم از روش RR با برش زمانی ۱ میلی ثانیه استفاده شود. میانگین زمان پاسخ چقدر است؟

فرآیند	p_1	p_2	p_3	p_4	p_5
CBT به دقیقه	۸	۱	۴	۶	۲
زمان ورود به دقیقه	۰	۱	۳	۳	۳

حل: همان طور که مشاهده می‌شود زمان‌های CBT در مقیاس دقیقه هستند و برش زمانی برابر با ۱ میلی ثانیه است.

در این روش اگر n فرآیند در سیستم موجود باشند، مانند این است که هر کدام یک پردازنده با سرعت $\frac{۱}{n}$ پردازنده‌ی سیستم برای خود دارند.

برای این مسأله نمودار ابداعی زیر را تحلیل می‌کنیم (اعدادی که بالای نمودار هر فرآیند درج شده، زمان باقیمانده‌ی هر فرآیند می‌باشد):



در لحظه‌ی صفر تنها فرایند p_1 در سیستم وجود دارد، پس پردازنده فقط به آن سرویس می‌دهد و تا دقیقه ۱، یک دقیقه از کار p_1 انجام شده و در دقیقه ۱ فرآیند p_2 نیز وارد می‌شود. بنابراین از این لحظه به بعد CPU بین این دو به اشتراک گذاشته می‌شود. پس تا دقیقه ۳، و طی دو دقیقه، از هر یک از فرآیندهای p_1 و p_2، یک دقیقه کم می‌شود. در دقیقه ۳، فرآیند p_2 خاتمه می‌یابد اما سه فرآیند جدید به سیستم اضافه می‌شوند و از این لحظه، CPU باید بین چهار فرآیند به اشتراک گذاشته شود. پس طی هشت دقیقه (از ۳ تا ۱۱) به هر فرآیند جمعاً دو دقیقه سرویس داده می‌شود و الی آخر.
بنابراین زمان پاسخ فرآیندها برابر است با:

فرآیند	زمان پاسخ (دقیقه)
p_1	۲۱−۰=۲۱
p_2	۳−۱=۲
p_3	۱۷−۳=۱۴
p_4	۲۱−۳=۱۸
p_5	۱۱−۳=۸

$$\text{میانگین زمان پاسخ} = \frac{۲۱ + ۲ + ۱۴ + ۱۸ + ۸}{۵} = ۱۲/۶$$

الگوریتم SRT (Shortest Remaining Time)

این الگوریتم نسخه **غیرانحصاری** (Preemetive) الگوریتم SJF است. در این الگوریتم اگر حین اجرای یک فرآیند، فرآیندی وارد شود که زمان اجرای کوتاه‌تری داشته باشد، پردازنده را در اختیار می‌گیرد.

نکته: این الگوریتم، **SRPT** (Shortest Remaining Processing Time)،
SRTF (Shortest Remaining Time First) و
SRTN (Shortest Remaining Time Next) نیز نامیده می‌شود.

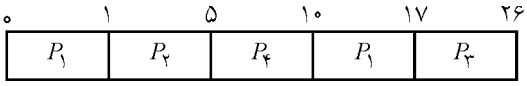
نکته: اگر لحظه ورود همه فرآیندها یکی باشد، الگوریتم SRT مشابه SJF عمل می‌کند.

نکته: در الگوریتم **SRT** نیز همانند الگوریتم SJF، احتمال وقوع **قحطی‌زدگی برای کارهای بزرگ** وجود دارد.

مثال: چهار فرآیند p_1 تا p_4 را در نظر بگیرید. با روش SRT، میانگین زمان انتظار برای فرآیندها چقدر است؟

فرآیند	$p_۱$	$p_۲$	$p_۳$	$p_۴$
CBT	۸	۴	۹	۵
زمان ورود	۰	۱	۲	۳

حل: با استفاده از روش SRT، نمودار گانت به صورت زیر است:



باید دقت کرد در لحظه ۱ که $p_۲$ وارد می‌شود چون زمان باقیمانده آن از زمان باقیمانده $p_۱$ کمتر است، پردازنده از $p_۱$ گرفته شده و به $p_۲$ داده می‌شود.

$$\text{میانگین زمان انتظار} = \frac{۹ + ۰ + ۱۵ + ۲}{۴} = \frac{۲۶}{۴} = ۶/۵$$

الگوریتم HRRN (Highest Response Ratio Next)

الگوریتم‌های SJF و SRT مشکل قحطی‌زدگی دارند. در این الگوریتم‌ها به فرآیندهای کوچک بیش از اندازه توجه می‌شود! برای رفع این مشکل الگوریتم HRRN مورد استفاده قرار می‌گیرد. در این الگوریتم اولویت فرآیندها برای اجرا، فقط اندازه آن‌ها نیست. برای تعیین اولویت یک فرآیند در الگوریتم HRRN از فرمول زیر استفاده می‌شود:

$$\text{اولویت} = \frac{\text{زمان انتظار + زمان اجرا}}{\text{زمان اجرا}}$$

در این فرمول، از آن جا که زمان اجرا در مخرج کسر قرار دارد، در نتیجه فرآیندهای کوچکتر اولویت بالاتری دارند اما از آن جا که زمان انتظار در صورت کسر قرار دارد، هر چه یک فرآیند بیشتر منتظر دریافت پردازنده بماند، اولویت بالاتری به دست می‌آورد. با این روش هم زمان اجرای یک فرآیند در تعیین اولویت آن تأثیر دارد و هم مدت زمانی که منتظر می‌ماند.

نکته: این الگوریتم HRN نیز نامیده می‌شود.

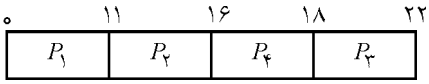
نکته: الگوریتم **HRRN** مشکل **قحطی‌زدگی ندارد**.

نکته: **HRRN** یک الگوریتم **انحصاری (Non Preemptive)** است.

مثال: میانگین زمان پاسخ و میانگین زمان انتظار را برای چهار فرآیند زیر به روش HRRN محاسبه کنید.

فرآیند	زمان ورود	زمان اجرا
$p_۱$	۰	۱۱
$p_۲$	۶	۵
$p_۳$	۸	۴
$p_۴$	۱۰	۲

حل: در لحظه صفر، فقط p_1 وارد می‌شود، بنابراین به صورت انحصاری پردازنده را در اختیار می‌گیرد و زمانی که اجرای آن به پایان رسید، برای تمام فرآیندهای موجود، اولویت محاسبه می‌شود و فرآیندی انتخاب می‌شود که از همه اولویت بالاتری داشته باشد. این روال به همین شکل تا انتها ادامه می‌یابد.



در لحظه ۱۱ اولویت‌ها به صورت زیر هستند:

$$pp_2 = \frac{5+5}{5} = 2$$

$$pp_3 = \frac{3+4}{4} = 1/75$$

$$pp_4 = \frac{1+2}{2} = 1/5$$

بنابراین در لحظه ۱۱ فرآیند p_2 انتخاب می‌شود.

در لحظه ۱۶ اولویت‌ها به صورت زیر هستند:

$$pp_3 = \frac{8+4}{4} = 3$$

$$pp_4 = \frac{6+2}{2} = 4$$

بنابراین در لحظه ۱۶ فرآیند p_4 انتخاب می‌شود و در لحظه ۱۸ نیز فقط فرآیند p_3 موجود می‌باشد.

در نهایت داریم:

فرآیند	زمان انتظار	زمان پاسخ
p_1	۰	۱۱
p_2	۱۱−۶=۵	۱۰
p_3	۱۸−۸=۱۰	۱۴
p_4	۱۶−۱۰=۶	۱۸

$$\text{میانگین زمان انتظار} = \frac{0+5+10+6}{4} = 5/25$$

$$\text{میانگین زمان پاسخ} = \frac{11+10+14+8}{4} = 10/75$$

الگوریتم‌های Priority (زمانبندی با اولویت)

در این روش زمانبندی، هر یک از فرآیندها اولویت مخصوص به خود را دارند. این اولویت معمولاً از خارج سیستم مشخص می‌شود. منطقی به نظر می‌رسد که اولویت فرآیندی مربوط به رئیس از اولویت فرآیندی مربوط به کارمند بالاتر باشد. ایده اصلی این الگوریتم بسیار ساده و مشخص است. هر فرآیند

باید یک اولویت داشته باشد و در هر لحظه فرآیندی اجرا می‌شود که بالاترین اولویت را دارد.

نکته: الگوریتم‌های **Priority** را می‌توان هم به صورت **انحصاری** و هم به صورت **غیرانحصاری** پیاده‌سازی کرد.

نکته: تنوع الگوریتم‌های زمانبندی با **اولویت** بسیار زیاد است و انواع مختلفی از آن وجود دارد که به عنوان مثال می‌توان به SJF، SRT و HRRN اشاره کرد. البته در این سه الگوریتم، اولویت فرآیندها در داخل سیستم و براساس شرایط مشخص می‌شود.

نکته: در الگوریتم‌های **Priority**، فرآیندهای با اولویت کمتر، دچار **قحطی زدگی** می‌شوند.

نکته: یک اولویت می‌تواند استاتیک یا دینامیک باشد:

ـ یک اولویت استاتیک هیچ‌گاه تغییر نمی‌کند، به همین دلیل پیاده‌سازی آن ساده است.

ـ یک اولویت دینامیک بر اثر تغییراتی که در محیط اتفاق می‌افتد تغییر می‌کند.

نکته: در الگوریتم‌های اولویت، جهت مقابله با مشکل قحطی‌زدگی برخی از فرآیندها، می‌توان از تکنیکی موسوم به **سالخوردگی (Aging)** استفاده کرد. در این تکنیک به تدریج اولویت پردازش‌هایی که مدت مدیدی در انتظار بوده‌اند، افزایش می‌یابد.

الگوریتم‌های MLQ (Multi Level Queues)

در این روش که به آن صفحات چندسطحی گویند، فرآیندها را به چند دسته تقسیم کرده و فرآیندهای موجود در هر دسته را در یک صف قرار می‌دهیم. در این حالت هر صف اولویت خاص خود را دارد و صف‌ها به ترتیب اولویت چیده می‌شوند. هر صف در داخل خود می‌تواند از الگوریتم زمانبندی جداگانه‌ای استفاده کند.

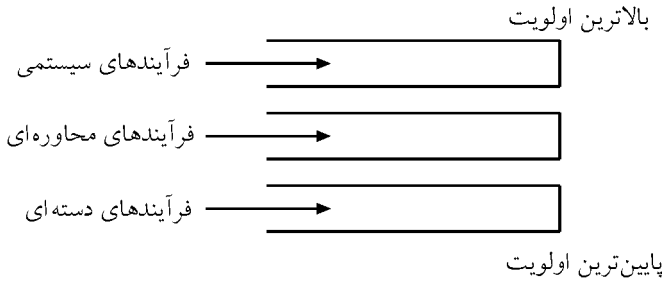
به عنوان مثال فرض کنید همه فرآیندهای سیستم در یکی از گونه‌های زیر قرار گیرند:

۱ـ فرآیندهای سیستمی

۲ـ فرآیندهای محاوره‌ای

۳ـ فرآیندهای دسته‌ای

در این حالت سیستم عامل برای هر گروه یک صف جداگانه تشکیل می‌دهد و فرآیندهای جدید را با توجه به نوعشان به صف موردنظر هدایت می‌کند. هر یک از صف‌ها اولویت خاص خود را دارد. برای مثال در این سیستم، صف فرآیندهای سیستمی از بالاترین اولویت برخوردار است.



مثالی از الگوریتم MLQ

در روش MLQ، هر صف الگوریتم زمانبندی خاص خود را دارد. برای مثال در صف فرآیندهای سیستمی می‌توان از روش SJF، در صف فرآیندهای محاوره‌ای از روش RR و در صف فرآیندهای دسته‌ای از روش FCFS استفاده کرد. نکته مهم در این الگوریتم این است که **نحوه تخصیص پردازنده بین صف‌ها** چگونه باشد؟ برای این منظور دو رویکرد وجود دارد، **انحصاری و غیرانحصاری**.

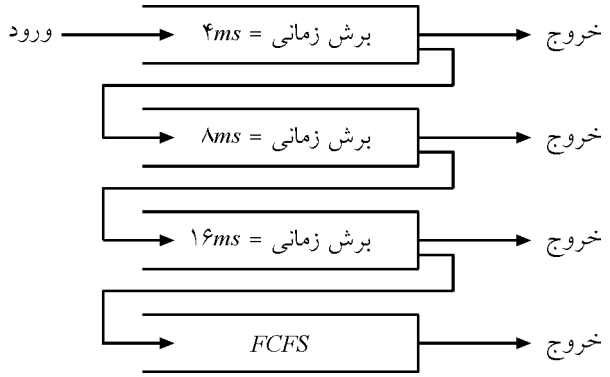
در رویکرد انحصاری، پردازنده به صورت غیرقابل پس‌گیری به صف‌ها داده می‌شود، بنابراین ابتدا به صف با بالاترین اولویت اختصاص می‌یابد و براساس الگوریتم مربوط به آن صف، به فرآیندهای آن سرویس می‌دهد. در این روش تا فرآیندهای موجود در یک صف با اولویت بالاتر به اتمام نرسیده باشند، پردازنده به صف با اولویت پایین‌تر تخصیص نمی‌یابد. از طرفی حتی اگر پردازنده مشغول سرویس دادن به صف سوم باشد و در این لحظه، یک فرآیند با اولویت بالاتر مثلاً برای صف اول یا دوم وارد شود، پردازنده صف سوم را رها کرده و به سراغ صف بالاتر می‌رود. واضح است که در این حالت احتمال وقوع قحطی‌زدگی وجود دارد.

اما در رویکرد غیرانحصاری، پردازنده با استفاده از روشی مشابه RR، بین صف‌ها حرکت می‌کند. مثلاً می‌توان ۵۰٪ زمان پردازنده را به صف اول، ۳۰٪ را به صف دوم و در نهایت ۲۰٪ را به صف سوم اختصاص داد.

الگوریتم MLFQ (Multi Level Feedback Queues)

روش MLFQ در واقع تغییریافته الگوریتم MLQ است. در الگوریتم MLQ یک فرآیند پس از ورود به یک صف خاص، تا پایان در همان صف باقی می‌ماند، اما ایده موجود در روش MLFQ این است که فرآیندها با توجه به رفتارشان بین صف‌ها حرکت کنند. به عنوان مثال فرآیندی که در یک صف با اولویت بالا قرار دارد و زمان پردازنده را زیاد مصرف می‌کند، به صف پایین‌تر منتقل می‌شود و یا فرآیندی که در یک صف با اولویت پایین بیش از حد منتظر دریافت پردازنده مانده است، به صف بالاتر منتقل می‌شود.

مثال: فرض کنید سیستمی با ۴ صف در اختیار داریم، در صف اول از روش RR با برش زمانی ۴ms، در صف دوم از روش RR با برش زمانی ۸ms، در صف سوم از روش RR با برش زمانی ۱۶ms و در صف آخر از روش FCFS استفاده می‌کنیم (شکل زیر). در این حالت فرآیندها ابتدا به صف اول وارد می‌شوند، اگر در این صف به پایان نرسیدند به انتهای صف دوم، اگر در این صف نیز کامل نشدند به انتهای صف سوم و اگر باز هم به پایان نرسیدند به انتهای صف چهارم وارد می‌شوند که از روش FCFS استفاده می‌کنند.



مثالی از الگوریتم MLFQ

الگوریتم LPT (Longest Processing Time)

این روش که **معمولاً به** صورت **انحصاری** پیاده‌سازی می‌شود، برعکس روش SJF می‌باشد، به این معنا که پردازنده از بین کارهای موجود، طولانی‌ترین کار را انتخاب می‌کند.

نکته: در الگوریتم **LPT** مشکل **قحطی‌زدگی برای کارهای کوچک** وجود دارد.

نکته: این الگوریتم میانگین زمان پاسخ و انتظار را حداکثر می‌کند!!!

الگوریتم Lottery

ایده موجود در این روش از ایده بخت‌آزمایی گرفته شده است. در این روش تعدادی بلیط بین فرآیندها تقسیم شده، سپس در ابتدای هر برش زمانی، یکی از بلیط‌ها به طور تصادفی برنده اعلام می‌شود و فرآیندی که آن بلیط را در اختیار دارد، پردازنده را **برای آن برش زمانی** در اختیار می‌گیرد.

در این روش فرآیندهای با اولویت بالاتر، بلیط‌های بیشتری در اختیار دارند، بنابراین شانس بیشتری برای در اختیارگرفتن پردازنده خواهند داشت.

نکته: الگوریتم **Lottery** مشکل **قحطی‌زدگی ندارد.**

نکته: فرآیندها می‌توانند بلیط‌های خود را با هم مبادله کنند. برای مثال فرآیندی که منتظر I/O است می‌تواند بلیط‌های خود را به فرآیندهای دیگر دهد.

نکته: دلیل **غیرانحصاری** بودن الگوریتم Lottery آن است که هنگامی که یک فرآیند پردازنده را در

اختیار دارد، پس از پایان برش زمانی پردازنده از وی گرفته شده و به فرآیند دیگری که بلیط اعلام شده بعدی را در اختیار دارد داده می‌شود.

الگوریتم Guaranteed (زمان‌بندی تضمین شده)

یکی از روش‌های متفاوت در زمـان‌بندی، Guaranteed Scheduling است. در ایـن روش ابـتدا بـا کاربران درباره‌ی سهمشان از پردازنده توافق شده، سپس سهم هر یک داده می‌شود.

یک مثال بسیار ساده این است که در یک سیستم چند کاربره با n فرآیند، اگر همه چیز را یکسان فرض کنیم، به هر فرآیند قول $\frac{۱}{n}$ زمان پردازنده را می‌دهیم و هر فرآیند باید $\frac{۱}{n}$ از زمان پردازنده را دریافت کند.

برای نیل به این هدف سیستم عامل باید برای هر فرآیند یک حساب باز کند! به این معنا که باید دقیقاً بداند هر فرآیند از ابتدا تا کنون چه مدت از پردازنده استفاده کرده و همچنین چه مقدار از سهم توافق شده باقی مانده است.

الگوریتم FSS (Fair Share Scheduling) (زمانبندی سهم عادلانه)

در الگوریتم FSS، قبل از زمانبندی فرآیندها این نکته باید در نظر گرفته شود که هر فرآیند متعلق به چه کسی است، در واقع در این حالت سهم هر **کاربر** مشخص می‌شود و زمانبند به گونه‌ای فـرآیـندها را انتخاب می‌کند که این سهم رعایت شود.

به عنوان مثال فرض کنید در یک سیستم دو کاربر A و B وجود دارند که کاربر A جمعاً ۹ فرآیند و کاربر B فقط ۱ فرآیند در حال اجرا دارد. در این حالت اگر مثلاً از الگوریتم RR استفاده کنیم، کاربر A جمعاً ۹۰٪ زمان پردازنده را از آن خود کرده است و به کاربر B فقط ۱۰٪ زمان پردازنده می‌رسد!

جهت جلوگیری از این وضعیت، الگوریتم FSS این نکته را در نظر می‌گیرد که فرآیندها متعلق به چه کسی هستند. مثلاً در سیستم قبل که دو کاربر A و B را داشتیم اگر به هر کدام ۵۰٪ از پردازنده را وعده داده باشیم، هر کدام از کاربرها بدون توجه به این که چه تعداد فرآیند دارند، سـهم خـود را دریـافت می‌کنند.

برای مثال فرض کنید کاربر A، چهار فرآیند $A_۱$، $A_۲$، $A_۳$ و $A_۴$ و کاربر B فقط یک فرآیند B را در اختیار دارد. اگر سهم هر کاربر از پردازنده ۵۰٪ باشد، با زمانبندی FSS روال تخصیص پردازنده به فرآیندها **می‌تواند** به صورت زیر باشد:

$A_۱ BA_۲ BA_۳ BA_۴ BA_۱ BA_۲ BA_۳ BA_۴ B...$

اما مثلاً اگر سهم کاربر A، ۷۵٪ و سهم کاربر B، ۲۵٪ باشد، دنباله زیر می‌تواند حاصل شود.

$A_۱ A_۲ A_۳ BA_۴ A_۱ A_۲ BA_۳ A_۴ A_۱ BA_۲ A_۳ A_۴ B...$