

سیستم عامل جلسه 3

حالت‌های فرآیند ها (Process Status)

وقتی فرایندی اجرا می‌شود، حالت آن تغییر می‌کند. حالت فرایند تا حدی توسط فعالیت فعلی آن تعریف می‌شود

In computing, the process status refers to the current state of a running program or process. The process status can provide information about whether the process is currently running, stopped, or terminated, as well as other details such as the amount of CPU time it has used, its priority level, and any signals that have been sent to it.

Common process states include:

- **Running**: The process is actively using the CPU.
- **Sleeping**: The process is waiting for some external event to occur (such as a user input or an I/O operation) before it can continue running.
- **Stopped**: The process has been suspended and is not currently executing any code.
- **Terminated**: The process has completed its execution and has been terminated by the operating system.

Process statuses can be viewed and managed using various tools and utilities provided by the operating system.

هر فرایند ممکن است در یکی از حالت‌های زیر باشد

جدید New : فرایند ایجاد می‌شود

اجرا Running : دستورات در حال اجرا هستند

انتظار Waiting : فرایند منتظر وقوع رویدادی است (مثل کامل شدن عمل i/o یا دریافت یک سیگنال)

پایان Terminate : اجرای فرایند خاتمه یافته است

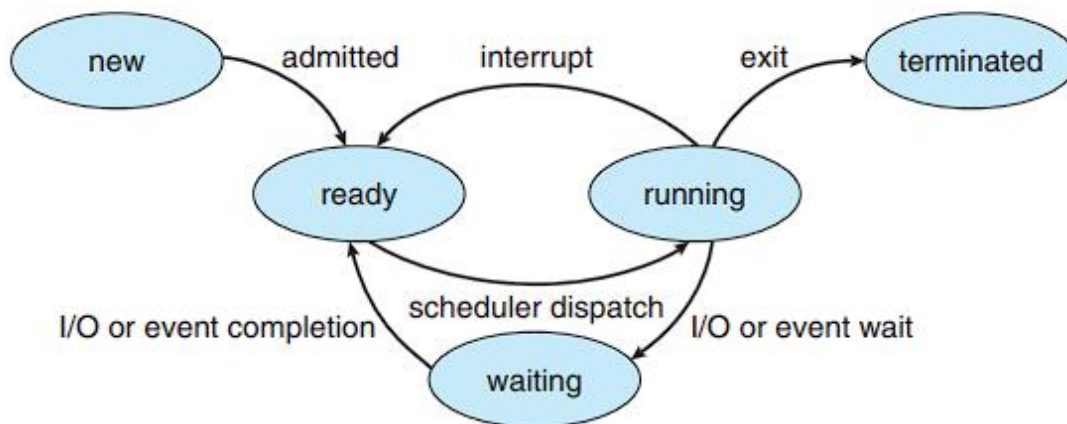
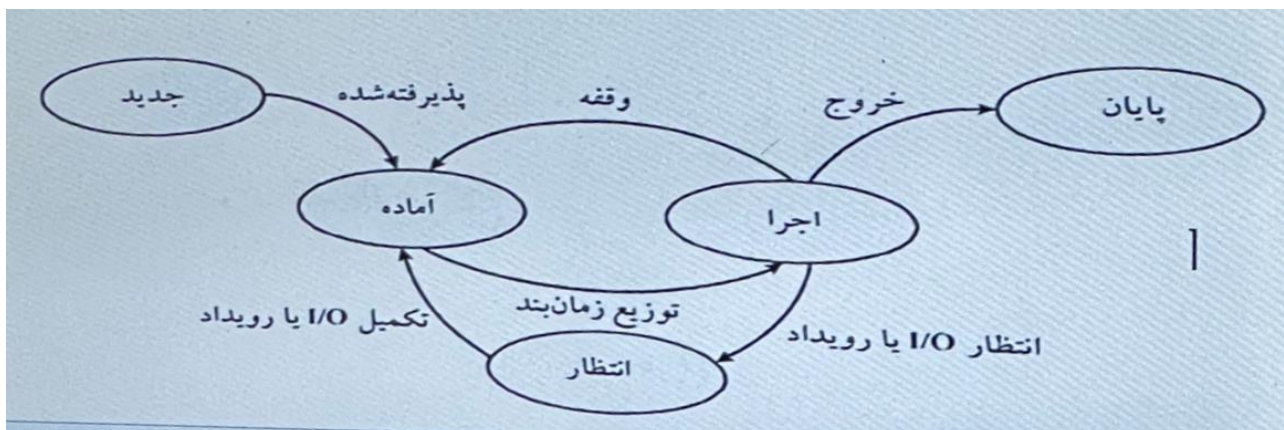


Figure 3.2 Diagram of process state.

مفاهیم اساسی زمانبندی (Scheduling)

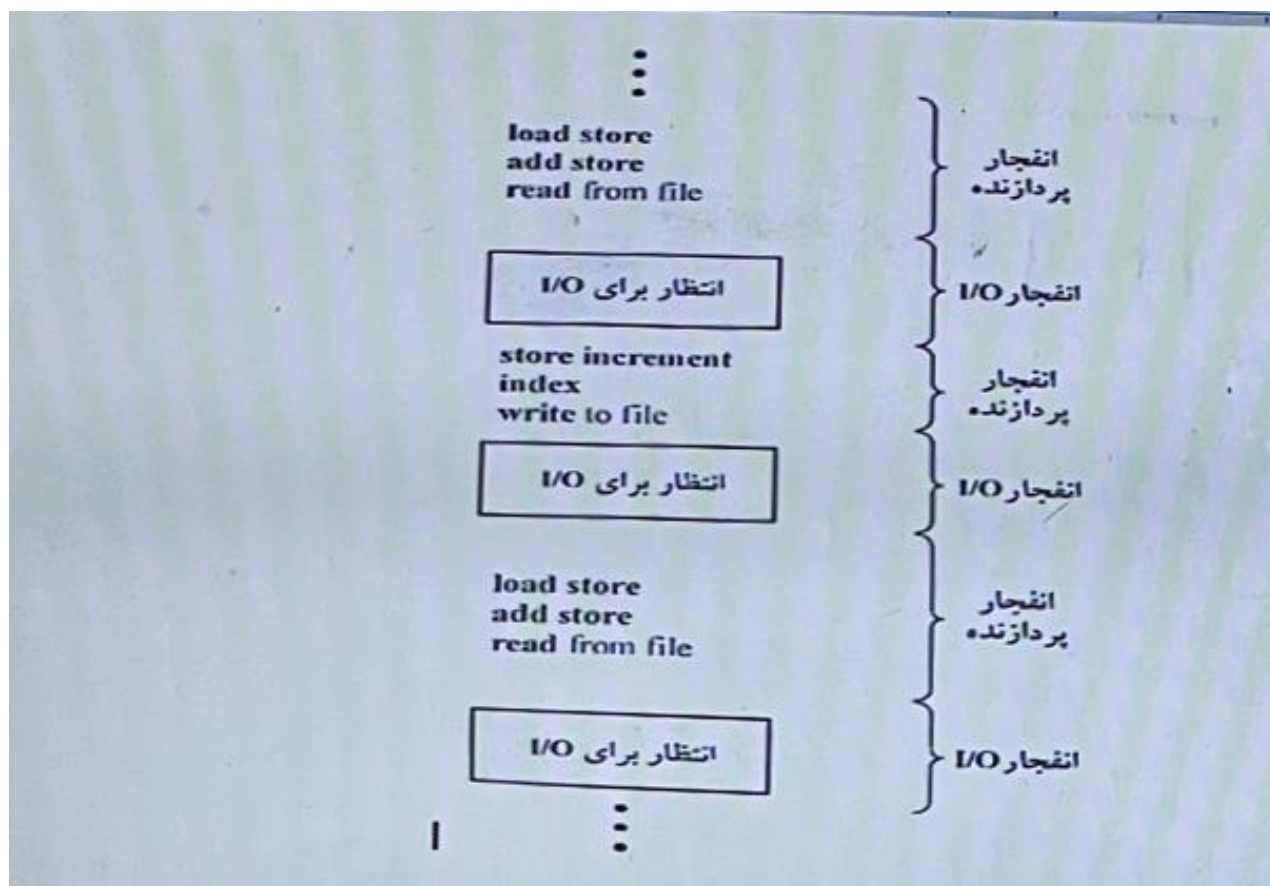
در یک سیستم تک پردازنده ای، در هر زمان فقط یک فرایند می تواند اجرا شود، بقیه باید منتظر بمانند تا پردازنده آزاد شود تا دوباره بتوانند زمانبندی شوند. هدف چند برنامه ای این است که همیشه چندین فرایند در حال اجرا باشند. تا بهره وری پردازنده به حداکثر برسد. هر فرایند به اجرایش ادامه می دهد تا برای یک عمل I/O در انتظار بماند در یک سیستم کامپیوتری ساده، در این مدت پردازنده باید بیکار بماند

در چندین برنامه ای، سعی می شود که از این زمان به نحوه احسن استفاده شود، در این سیستم، چند فرآیند به طور همزمان در حافظه نگهداری می شوند، وقتی یک فرایند به حالت انتظار می رود. سیستم عامل، پردازنده را از آن فرایند می گیرد و به فرایند دیگری می دهد. این روند ادامه می یابد، هر وقت که فرایندی به حالت انتظار رفت، فرایند دیگری از پردازنده استفاده می کند

تقریباً تمام منابع کامپیوتری، قبل از استفاده از زمانبندی می شوند. پردازنده یکی از منابع اصلی کامپیوتر است و در نتیجه زمانبندی آن موضوع اصلی طراحی سیستم عامل است

چرخه انفجار i/o و انفجار پردازنده (i/o and CPU Burst)

موفقیت زمانبندی پردازنده به این خاصیت فرایندها بستگی دارد، که اجرای فرایند شامل چرخه ای از اجرای پردازنده و انتظار برای i/o است. فرایندها بین این دو حالت سوییچ می کنند. اجرای فرایند با انفجار پردازنده شروع می شود. و به دنبال آن یک انفجار i/o قرار دارد که بعد از آن، انفجار دیگری از پردازنده و سپس انفجار دیگری از i/o قرار دارد و به همین ترتیب ادامه می یابد. سرانجام، آخرین انفجار پردازنده با درخواست سیستم برای خاتمه اجرا، پایان می پذیرد



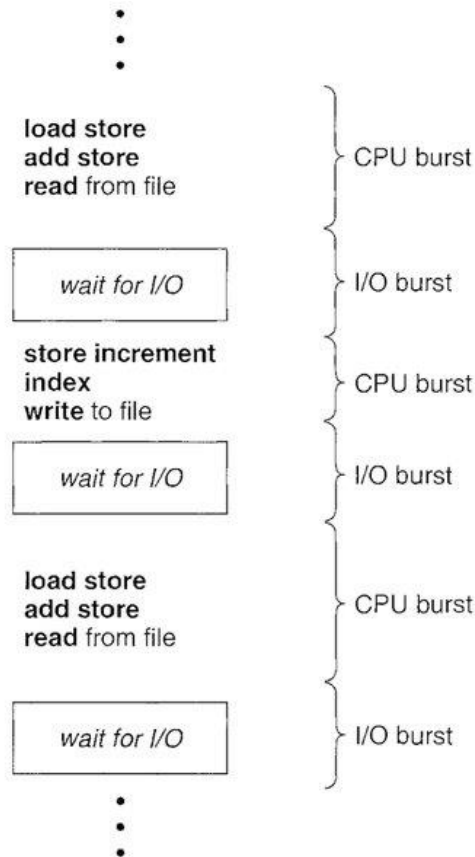


Figure 5.1 Alternating sequence of CPU and I/O bursts.

مدت این انفجارهای پردازنده از فرایندی به فرایند دیگری و از کامپیوتری به کامپیوتر دیگر متفاوت است. تعداد زیادی از انفجارهای کوتاه پردازنده و تعداد اندکی از انفجارهای بلند پردازنده وجود دارد. برنامه مقید به i/o (در تنگنای i/o) معمولاً دارای چند انفجار کوتاه پردازنده است. برنامه مقید به پردازنده (در تنگنای پردازنده) معمولاً چند انفجار بلند پردازنده دارد. این توزیع می‌تواند در انتخاب الگوریتم زمانبندی پردازنده مهم باشد.

انواع زمانبندی ها (Schedulers)

کلید چند برنامه‌گی زمانبندی است، زمانبندی بر روی کارایی سیستم اثر می‌گذارد زیرا مشخص می‌کند کدام فرایندها منتظر مانده و کدام فرایندها به جلو بروند

انواع زمانبندی برای پردازنده عبارت است از:

1. زمانبند بلند مدت (Long Term scheduler) | Job scheduler

تصمیم‌گیری در مورد افزودن به مجموعه‌ی فرایندها برای اجرا

2. زمانبند میان مدت (Middle Term scheduler) | Swap scheduler

تصمیم‌گیری در مورد افزودن به تعداد فرایندهایی که بخشی یا تمام آن‌ها در حافظه اصلی است.

3. زمانبند کوتاه مدت (Short Term Scheduler) | CPU scheduler

تصمیم‌گیری در مورد این که کدام یک از فرایندهای موجود در حافظه اصلی، برای اجرا توسط پردازنده انتخاب شوند

4. زمانبند ورودی - خروجی (i/o Scheduler)

تصمیم می‌گیرد که کدام درخواست i/o فرایندها به وسیله یک دستگاه i/o موجود انجام گیرد.

وظیفه فعال سازی و تعلیق فرایندها بر عهده زمانبند میان مدت است

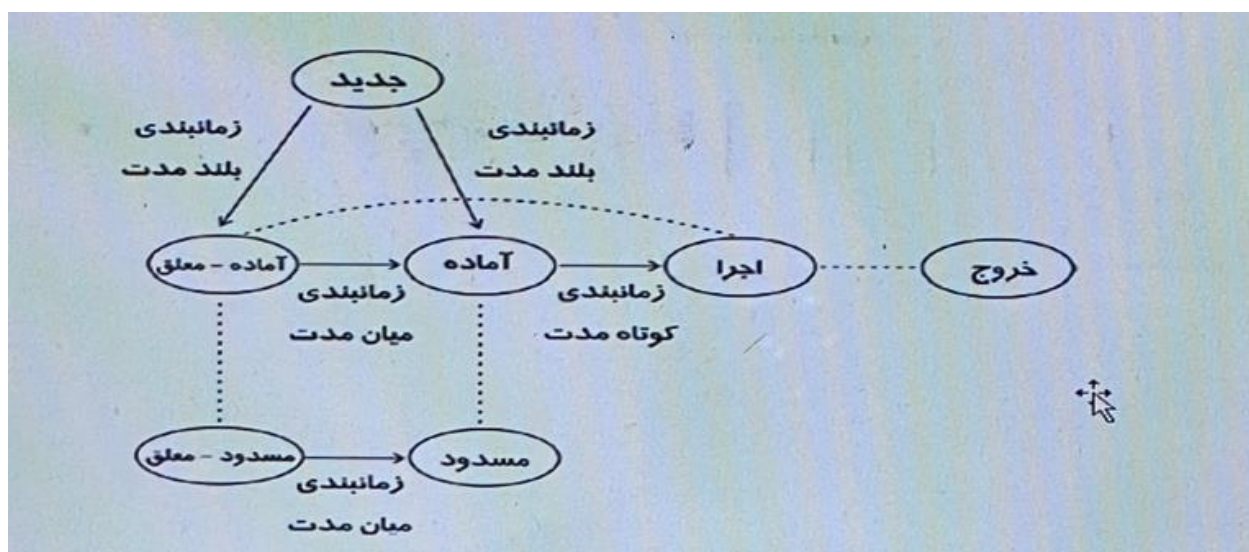
زمانبند میان مدت فرایندی را از حافظه اصلی حذف و به حافظه جانبی می‌برد. این فرایند بعداً می‌تواند به حافظه اصلی لود شود. این الگو را مبادله (Swapping) می‌گویند.

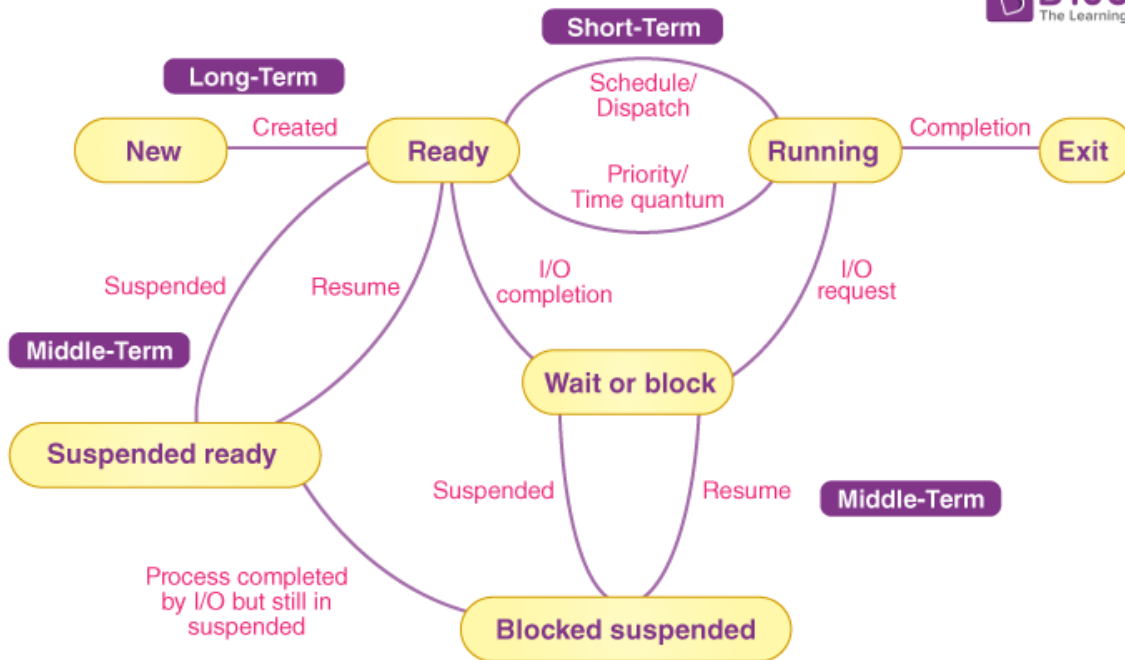
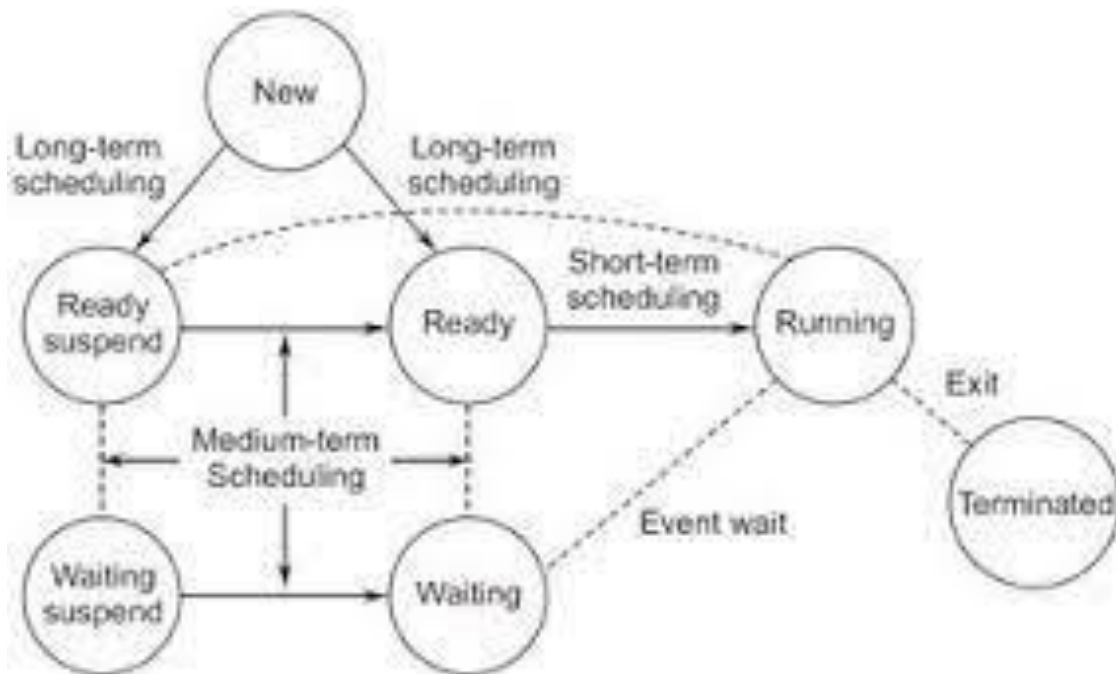
ایده اصلی زمانبندی میان مدت این است که می‌تواند فرایندی را از حافظه حذف کند و درجه چند برنامگی را کاهش دهد.

زمانبند بلند مدت ترکیب خوبی از فرایندهای i/o limited و CPU limited ، انتخاب می‌کند. نام دیگر زمانبند بلند مدت، زمانبند کار است. نام دیگر زمانبند کوتاه مدت، زمانبند پردازنده است.

زمانبند بلند مدت نسبتاً دفعات کمی اجرا می‌شود. زمانبند میان مدت نسبتاً تعداد دفعات بیشتری به اجرا در می‌آید و زمانبند کوتاه مدت بیشترین دفعات اجرا را دارد.

تذکر: اکثر سیستم‌های **اشتراک زمانی** فاقد زمانبند بلند مدت می‌باشند.



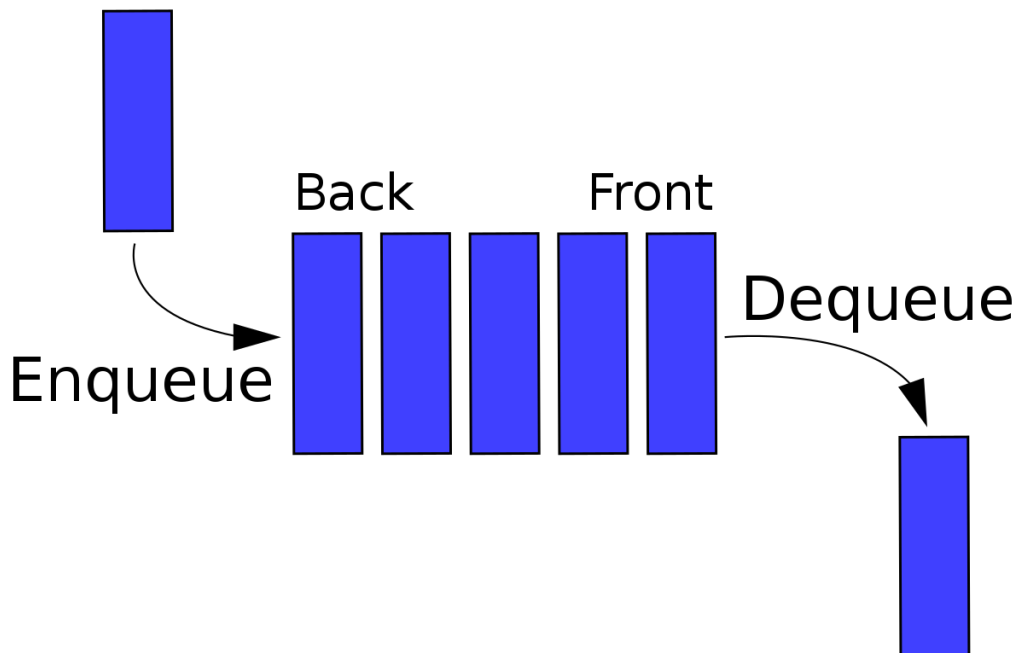


زمانبندی پردازنده (CPU scheduling)

هر وقت پردازنده بیکار می‌شود، سیستم عامل باید یکی از فرایندهای موجود در صف آماده را برای اجرا انتخاب کند. این انتخاب توسط زمانبند کوتاه مدت (زمان بند پردازنده) (Short scheduler – CPU scheduler) انجام می‌گیرد.

زمانبند، فرایندی را از بین فرایندهای موجود در حافظه که آمادگی اجرا دارند انتخاب می‌کند و پردازنده را به آن تخصیص می‌دهد.

صف الزاماً یک صف با ویژگی "خروج به ترتیب" (FIFO) (First in First Out) نیست. همان طور که هنگام بحث در مورد الگوریتم‌های زمانبندی خواهیم دید، صف آماده می‌تواند به صورت صف (FIFO)، صف اولویت، درخت (tree) یا یک لیست پیوندی (Linked list) نامرتب پیاده سازی شود. از نظر مفهومی تمام فرایندهای موجود در صف آماده، منتظر به دست آوردن پردازنده هستند تا اجرا شوند



زمان بندی با قبضه کردن (Preemptive) یا غیر انحصاری

یا انقطاع پذیر

فرایند در حال اجرا می تواند به وسیله سیستم عامل متوقف شود و به حالت آماده منتقل شود. در این الگوریتم ها در پایان برش زمانی و یا تغییر شرایط سیستم مدیر زمان بندی می تواند کنترل پردازنده را از یک "پردازهی (فرایند)" در حال اجرا گرفته و به "پردازهی (فرایند)" دیگری بدهد

به زبان دیگر زمانی که یک پردازش در حال اجراست اگر زمان بندی از نوع با قبضه کردن باشد سیستم در هر زمانی که بخواهد یا نیاز پیدا کند می تواند منابع را از پردازش گرفته و حالت آن را به انتظار تغییر دهد در این حالت پردازش حق انتخابی ندارد و باید منتظر تخصیص دوباره منابع بماند

تصمیمات زمان بندی پردازنده ممکن است تحت چهار شرط ریز اتخاذ شود:

1. وقتی که فرایندی از حالت اجرا به حالت انتظار می رود (مثل درخواست i/o ، یا فراخوانی `wait()` برای خاتمه ی یکی از فرایندهای فرزند)
2. وقتی فرایندی از حالت اجرا به حالت آماده می رود (مثل وقتی که رویدادی رخ می دهد)
3. وقتی که فرایندی از حالت انتظار به حالت آماده می رود (مثل تکمیل شدن i/o)
4. وقتی فرایندی خاتمه می یابد.

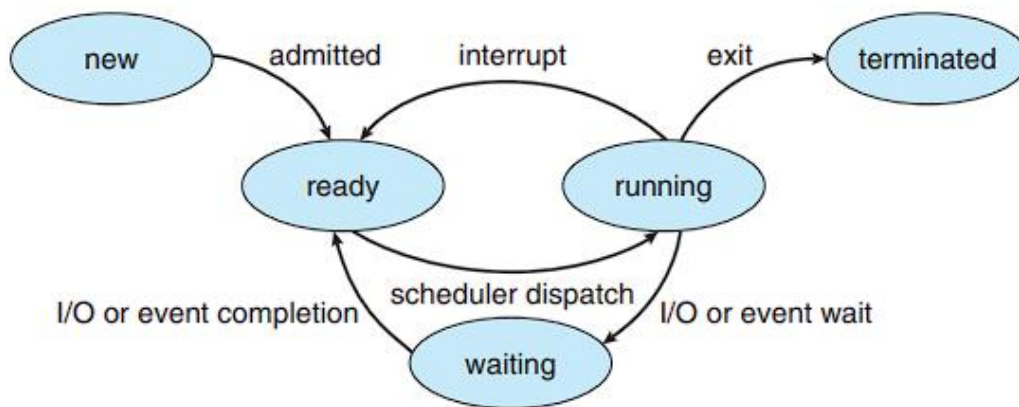
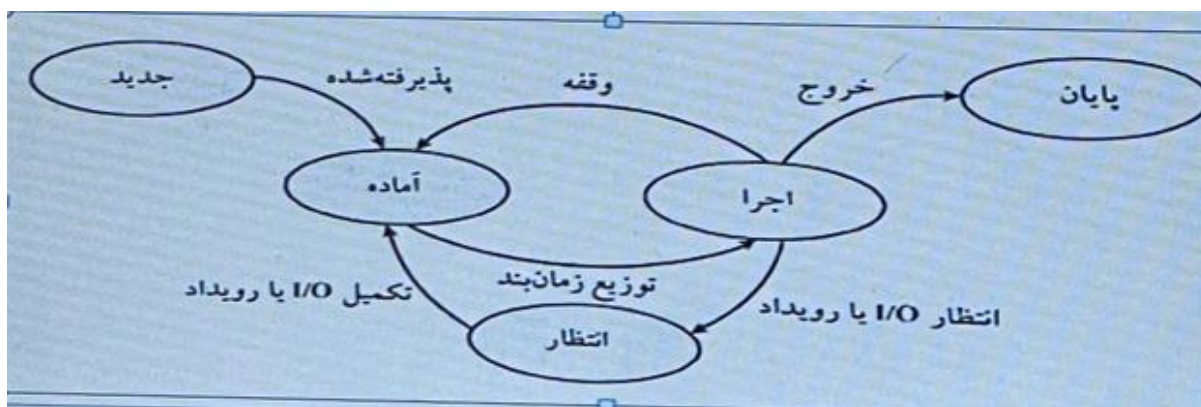


Figure 3.2 Diagram of process state.



برای شرط 1 و 4 انتخابی بر حسب زمانبندی وجود ندارد. در صورت وجود فرایندی در صف آماده، یک فرایند باید برای اجرا انتخاب شود. اما برای شرایط 2 و 3 امکان انتخاب وجود دارد.

وقتی زمانبندی فقط تحت شرایط 1 و 4 انجام می گیرد الگوی زمانبندی را بدون قبضه کردن (non preemptive) می نامیم و در غیر این صورت آن را با قبضه کردن (Preemptive) یا غیر انحصاری می نامیم.

انحصاری / بدون قبضه کردن (non preemptive)

در این الگوریتم‌ها تا زمان خاتمه پردازنده و یا زمان نیاز به عمل I/O نمی‌توان CPU را از پردازشی (Process) در حال اجرا گرفت و پردازشی دیگری داد

همین که یک فرایند در حالت اجرا قرار گرفت آن قدر به اجرا ادامه می‌دهد تا خاتمه یابد یا با این که خودش (داوطلبانه) برای انتظار I/O مسدود شود

به زبان دیگر وقتی پردازنده به پردازشی اختصاص یافت و زمانبندی از نوع بدون قبضه کردن بود سیستم عامل حق گرفتن منابع را از پردازشی ندارد و باید منتظر بماند تا پردازشی به حالت انتظار برای I/O برود یا پردازشی تمام شود یا به صورت داوطلبانه منابع را رها کند این حالت به صورت کلی حالت مخالف با قبضه کردن (Preemptive) است

یادداشت:

preemptive در کامپیوتر یعنی خالی کردن

مثلاً در سی پی یو اگر زمانبندی preemptive باشد یعنی ما این اجازه را داریم که سی پی یو را از یک برنامه ای که سی پی یو الان دستش و وقتش هم تمام نشده ازش بگیریم

برعکس این مدل non preemptive است و به این معناست که ما حق گرفتن سی پی یو از برنامه ای را نداریم و باید منتظر بمانیم تا سی پی یو خودش برنامه را رها کند

در زمانبندی بدون قبضه کردن، وقتی پردازنده به فرایندی اختصاص می‌یافت، آن قدر پردازنده را نگه می‌دارد تا این که خاتمه یابد یا به حالت انتظار برود که در این صورت پردازنده را آزاد می‌کند. این روش زمانبندی توسط ویندوز 3.x مورد استفاده قرار

می‌گرفت.. ویندوز 95، زمانبندی با قبضه کردن را معرفی کرد و تمام نسخه‌های بعدی سیستم عامل ویندوز از زمانبندی با قبضه کردن استفاده نمودند. سیستم عامل Mac OS X برای مکینتاش نیز از زمانبندی با قبضه کردن استفاده می‌کند، نسخه قبلی سیستم عامل مکینتاش مبتنی بر زمانبندی همکاری (Cooperative Scheduling) بود.

زمانبندی همکاری تنها روشی است که می‌تواند بر روی سکوهایی سخت افزار اجرا شود، زیرا به سخت افزار خاصی (مثلاً تایمر) که برای زمانبندی با قبضه کردن نیاز ندارد.

زمانبندی با قبضه کردن هزینه دارد. حالتی را در نظر بگیرید که دو فرایند داده‌های مشترکی دارند. یکی از آن‌ها ممکن است در زمان به هنگام ذخیره سازی داده‌ها قبضه شود و پردازنده‌ی دوم در حال اجرا باشد. فرایند دوم ممکن است تلاش کند داده‌هایی را بخواند که در حالت ناسازگار قرار دارد. بنابراین نیاز به راهکار جدیدی است تا دستیابی به داده‌های مشترک را هماهنگ کند

زمانبندی با قبضه کردن در طراحی هسته سیستم عامل موثر است. هنگام پردازش فراخوان سیستم، هسته ممکن است مشغول انجام فعالیتی در رابطه با یک فرایند باشد. این فعالیت‌ها ممکن است داده‌های مهم هسته را تغییر دهند (مثل صف i/o)

اگر فرایندی که در حال انجام این تغییرات است، قبضه شود و هسته (یا گرداننده دستگاه) نیاز به خوانده یا تغییر همان ساختار داشته باشد، چه اتفاقی می‌افتد؟

به نظر می‌رسد که هرج و مرج پیش می‌آید. بعضی از سیستم‌های عامل، از جمله اغلب نسخه‌های یونیکس، این مسئله را به این ترتیب حل می‌کنند. که قبل از تعویض متن، منتظر می‌مانند تا فراخوان سیستم کامل شود یا یک عمل i/o رخ دهد و این الگو

تضمین می کند که ساختار هسته ساده باشد، زیرا در حالی که ساختمان داده های هسته در وضعیت ناسازگاری وجود دارند ؛ هسته فرایندی را قبضه نمی کند.

متأسفانه این مدل (Preemptive) اجرای هسته برای پشتیبانی بی درنگ و پردازشی مناسب نیست.

چون وقفه ها در هر زمانی می توانند رخ دهند و چون هسته همیشه نمی تواند آن ها را نادیده بگیرد، بخش هایی از کد تحت تاثیر وقفه ها قرار دارند ، باید از استفاده ی همزمان در امان باشند و لازم است سیستم عامل وقفه ها را در تمام اوقات بپذیرد، وگرنه ممکن است ورودی مفقود شود و خروجی از بین برود. برای این که چندین فرایند به طور همزمان به این بخش های کد دسترسی نداشته باشند، هنگام ورود به آن ها وقفه را غیرفعال و هنگام خروج، وقفه را فعال می سازند. توجه به این نکته مهم است که بخش هایی از کد که وقفه ها را غیرفعال می کند زیاد نیستند و دستور العمل های زیادی ندارند.

وقفه (interrupt)

وقفه رخدادی در سیستم عامل است. که ترتیب اجرای دستورالعمل ها را توسط پردازشگر (CPU) تغییر می دهد. وقفه را سخت افزار کامپیوتر تولید می کند اما علت آن می تواند نرم افزاری باشد

یادداشت: در لینوکس وقفه های 1-18 وقفه های Non maskable هستند یعنی غیرقابل چشم پوشی و از 19 تا 31 برای intel رزرو شده است و از 32 تا 255 آزاد هستند یا Maskable

در زبان کامپیوتر، وقفه یا **Interrupt** به معنی سیگنالی است که برای پردازنده فرستاده می‌شود تا نشان دهد رخ دادن یک حادثه‌ی خاص الزامی است. این حادثه ممکن است توسط سخت‌افزار، نرم‌افزار، یا کاربر فراهم شود.

وقفه‌ها به سه دسته‌ی خارجی، داخلی و نرم‌افزاری تقسیم می‌شوند:

1. وقفه‌های خارج (External Interrupts): وقفه‌هایی هستند که توسط دستگاه‌های سخت‌افزاری مانند کارت شبکه، دستگاه‌های ورودی/خروجی و ... ایجاد می‌شوند. این وقفه‌ها باعث می‌شوند پردازنده در صورت لزوم به زودی به دستگاه مربوطه پاسخ دهد.

2. وقفه‌های داخلی (Internal Interrupts-Trap): وقفه‌هایی هستند که توسط خود پردازنده ایجاد می‌شوند. به عنوان مثال، وقفه‌ی حافظه پر شده که در صورت پر بودن حافظه به پردازنده اعلام می‌کند که باید داده‌ها را از حافظه خارج کند.

3. وقفه‌های نرم‌افزاری : ((Software Interrupts-SVC(Super visor call)) همچنین به آن‌ها سرویس‌های سیستمی گفته می‌شود و توسط برنامه‌های کاربری ایجاد می‌شوند. برای مثال، وقفه‌ی سرویس فایل که درخواست‌های مربوط به فایل‌ها را از برنامه‌های کاربری دریافت کرده و برای انجام با سیستم عامل تعامل دارد.

انواع وقفه ها

داخلی (Trap): بر اثر اجرای دستورات خود برنامه به صورت داخلی در CPU رخ می دهد. (از جنس نرم افزاری است و تولید کننده آن نرم افزار است)

خارجی: از دستگاه های خارجی مثل دستگاه های ورودی یا خروجی ، تایمر ، صفحه کلید و خطاهای سخت افزاری ناشی می شود.

نرم افزار (SVC - Supervisor Call): بر اثر فراخوانی توابع سیستمی توسط برنامه رخ می دهد (syscall)

توزیع کننده (Dispatcher)

مولفه ی دیگری که در زمانبندی پردازنده دخالت دارد، توزیع کننده (Dispatcher) است ؛ توزیع کننده ، پیمانه ای (module) است که کنترل را به پردازنده ای می دهد که زمانبند کوتاه مدت ، آن را انتخاب کرده است

این عمل شامل موارد زیر است:

- تعویض متن (تعویض بستر) (Context Switching)
- تغییر به حالت کاربر (user mode)
- پرش به محل مناسبی در برنامه ی کاربر و آغاز مجدد آن برنامه

توزیع کننده باید سرعت بالایی داشته باشد، زیرا هنگام تعویض هر فرایند فراخوانی می شود. مدت زمانی که توزیع کننده صرف می کند تا یک فرایند را متوقف و فرایند دیگری را شروع کند **تاخیر توزیع (Dispatch Latency)** نام دارد

به زبان دیگر حد فاصل متوقف کردن فرایند فعلی و اجرای یک فرایند دیگر توسط توزیع کننده را **dispatch latency** یا تاخیر توزیع می گویند

معیار های زمانبندی Scheduling Criteria

بهره وری پردازنده (CPU Performance/efficiency/Utilization) می خواهیم

تا آن جایی که ممکن است پردازنده مشغول باشد. درصد بهره وری پردازنده می تواند از 0 تا 100 باشد ، در یک سیستم واقعی ، باید از 40 (برای سیستم هایی با بار کم) تا 90 درصد (سیستم هایی با بار زیاد) باشد

توان عملیاتی (Throughput) (گذردهی): اگر پردازنده مشغول

اجرای فرآیند ها باشد، کاری که در حال انجام است. یک معیار کار، تعداد فرایندهایی

است که در واحد زمان کامل می شوند و توان عملیاتی نام دارد. برای فرایندهای

طولانی، ممکن است بر حسب ساعت سنجیده شود. برای تراکنش های کوتاه توان

عملیاتی ممکن است 10 فرایند در ثانیه باشد

زمان برگشت (Turnaround time) (یا زمان کل): از نقطه نظر یک

فرایند خاص، مهم ترین معیار، زمان اجرای یک فرایند است. فاصله ی زمانی از زمان

تحویل فرایند، تا زمان کامل شدن اجرای آن، زمان برگشت نام دارد. زمان بازگشت

مجموع زمان انتظار برای قرار گرفتن در حافظه، زمان انتظار برای قرار گرفتن در صف

آماده، اجرا در پردازنده و عمل i/o است.

زمان انتظار (Waiting Time): الگوریتم زمانبندی پردازنده، بر مدت

زمانی که در اثنای آن فرایندی که اجرا می شود یا عمل i/o انجام می دهد، تاثیر

ندارد. فقط بر مدت زمانی که فرایند در صف آماده منتظر می ماند موثر است، زمان انتظار

برابر با مجموع مدت های انتظار یک فرایند در صف آماده است.

زمان پاسخ (Response Time): در یک سیستم عامل محاوره ای،

زمان بازگشت ممکن است معیار خوبی نباشد. اغلب یک فرایند ممکن است بعضی از نتایج را زود تولید کند و هنگامی که این نتایج به خروجی می روند، محاسبات دیگری صورت می گیرد و نتایج دیگری به خروجی برود. لذا معیار دیگر، زمان تحویل یک درخواست تا دریافت اولین پاسخ است. این معیار که زمان پاسخ نام دارد، مدت زمانی که طول می کشد پاسخ آماده شود، نه مدت زمانی که طول می کشد تا آن پاسخ چاپ شود. زمان برگشت، به سرعت دستگاه خروجی بستگی دارد.

The criteria include the following:

- **CPU utilization:** The main objective of any CPU scheduling algorithm is to keep the CPU as busy as possible. Theoretically, CPU utilization can range from 0 to 100 but in a real-time system, it varies from 40 to 90 percent depending on the load upon the system.
- **Throughput:** A measure of the work done by the CPU is the number of processes being executed and completed per unit of

time. This is called throughput. The throughput may vary depending on the length or duration of the processes.

- **Turnaround time**: For a particular process, an important criterion is how long it takes to execute that process. The time elapsed from the time of submission of a process to the time of completion is known as the turnaround time. Turn-around time is the sum of times spent waiting to get into memory, waiting in the ready queue, executing in CPU, and waiting for I/O. The formula to calculate Turn Around Time = Completion Time – Arrival Time.
- **Waiting time**: A scheduling algorithm does not affect the time required to complete the process once it starts execution. It only affects the waiting time of a process i.e. time spent by a process waiting in the ready queue. The formula for calculating Waiting Time = Turnaround Time – Burst Time.

- **Response time:** In an interactive system, turn-around time is not the best criterion. A process may produce some output fairly early and continue computing new results while previous results are being output to the user. Thus another criterion is the time taken from submission of the process of the request until the first response is produced. This measure is called response time. The formula to calculate Response Time = CPU Allocation Time (when the CPU was allocated for the first) – Arrival Time
- **Completion time:** The completion time is the time when the process stops executing, which means that the process has completed its burst time and is completely executed.

Priority: If the operating system assigns priorities to processes, the scheduling mechanism should favor the higher-priority processes.

- **Predictability:** A given process always should run in about the same amount of time under a similar system load.