## فصل دوم

# پردازش و زمانبندی

## مفهوم پردازش

مهمترین مفهوک در هر سیستم عامل فرآیند پردازش (process)است. تمامی نرم افزار های کامپیوتر از جمله سیستم عامل به تعدادی از پروسس ها سازماندهی و تقسیم بندی می شوند. یک پردازش برنامه ای در حال اجراست. درواقع یک پروسس فقظ یک برنامه اجرایی است که علاوه بر کد برنامه( یا بخش متن text segment) شامل مقدار شمارنده برنامه، ریجیستر های CPU، پشته و بخش داده ها (Data segment)است. به عبارت دیگر می توان گفت که هر پروسس CPU مجازی خود را دارد. در سیستم چند برنامگی CPUاز یک پروسسی به پروسس دیگر سوِئیچ می کند و هر کدام و هر کدام را به مدت چند ده یا چند ضد میلی ثانیه به اجرا در می آورد.

باید دقت کرد که یک برنامه خودی خود یک پردازش نیست. برنامه الگوریتمی است که مثل محتویات یک فایل بر روی دیسک ذخیره شده است. به عبارتی دیگر برنامه یک نهاد غیرفعال (passive) است. در حالیکه پردازش یک نهاد فعال (active) می باشد که در حال اجراست. مثلا در یک کامپیوتر کاربران متعددی ممکن است در حال اجرای نسخه های متعددی از برنامه ویرایشگر باشند یا مثلا یک کاربر می تواند چند نسخه از برنامه ویرایشگر را همزمان اجرا کند، در این حال هر کدام از آنها یک پردازش جداگانه و اگر چه بخش متن شان(کدشان) یکسان است ولی بخش داده هایشان متفاوت است.

مشخضات پردازش ها در جدول پردازش (process table) نگهداری میشود.

در سیستم ها روشی مورد نیاز است تا در حین کار بتوان پروسس هایی را ایجاد کرد یا از بین برد در UNIX پروسس ها توسط فراخوان سیستمی fork پدید می آیند. این فراخوانی یک پردازش فرزند تولید می کند که نسخه ای دقیقا یکسان با پروسس پدر خواهد بود.

به همین ترتیب پردازش فرزند نیز میتواند fork را اجرا کرده و لذا سیستم میتواند درختی از پروسس ها داشته باشد. بدیهی است هر پروسس فقظ یک پدر دارد ولی میتواند صفر یا چندین فرزند داشته باشد.

به عنوان مثال نحوه عملکرد UNIX به صورت زیر است: در هنگام شروع سیسم عامل پروسس مخصوصی به نام init را اجرا می کند. این پردازش تعداد ترمینالهای موجود را مشخص می کند سپس توسط دستور fork به ازا هر ترمینال یک پروسس جدید تولید می کند. این پروسس ها منتظر می مانند تا شخصی به سیستم وارد شود (Login کند). در این هنگام پروسس Login یک پوسته (shell) را جهت پذیرش دستورات وی اجرا می کند. این دستورات نیز ممکن است

باعث شروع شدن پردازش های دیگر شوند. لذا تمام پروسس های درون سیستم به درختی تعلق دارند که ریشه آن init است.

زمانی که پردازشی یک زیر پردازش پدید می آورد، پروسس فرزند می تواند منابع مورد نیازش را مستقیما از سیستم عامل کسب کند یا محدود به استفاده از زیر مجموعه ای از منابع پروسس والدش گردد. پردازش والد نیز ممکن است مجبور شود منابعش را مابین فرزندانش قسمت کند یا ممکن است قادر باشد بعضی منابع(مثال حافظه یا فایل) را با فرزندانش به اشتراک گذارد. محدود کردن یک پردازش فرزند به استفاده از زیر مجموعه ای از منابع والدش، از زیاد بار شدن (overload) سیستم توسط ایجاد پروسس های بیشمار، پیشگیری می کند. وقتی پردازشی پردازش جدید را ایجاد می کند در مورد اجرا دو مکان وجود دارد. یا پردازش والد به اجرای همروند با فرزندنش ادامه می دهد و یا پردازش والد منتظر می ماند تا اجرای تعدادی یا تمام فرزندانش تمام شود. در یونیکس هر پردازش با شناسه ( PID=Process Identifier) که یک عدد صحیح یگانه است شناسایی می شود. یک پردازش جدید شامل یک کپی از فضای آدرس پردازش والد است و این مکانیزم اجازه می دهد پردازش والد با پردازش فرزندش به سادگی ارتباط داشته باشد.

در اکثر سیستم عامل ها (مثل UNIX) هنگامی که والدی تمام می شود کلیه فرزندانش نیز توسط سیستم عامل خاتمه داده می شوند. در این حال تمام پردازش ها میتواند به دو صورت باشد: یا اینکه در ابتدا پردازشهای فرزند ختم شوند و سپس پردازش پدر، که این روش ختم پی در پی یا Cascaded Trmination گویند، یا اینکه ابتدا پردازش پدر تمام شوده و سپس پردازش های فرزند ختم شوند.

تذکر: پردازش ها به طور کلی دو دسته هستند. 1- پردازش های مربوط به کاربر 2- پردازش های مربوط سیستم. به عبارتی دیگر زمانی که CPU می خواهد یک دستور را اجرا کند در دو حالت می تواند قرار گیرد.1- مد کاربر که دستورات معمولی اجرا می شوند. 2-مد سوپروایزر که می تواند تمام دستورات را اجرا کند. پردازش های سیستم در مد سوپروایزر اجرا گردد.

## زمانبند کار (Job scheduler)

زمانبند کار یا Job Scheduler روتینی است که بر اساس الگوریتمی خاص یکی از کارهای موجود در جدول ISPT(Input spool Table) را انتخاب کرده و جهت اجرا شدن آن را به حافظه می آورد. یک Job هنگامی تبدیل به پردازش می شود که تمامی منابع مورد نیاز (ازجمله حافظه) را در اختیار داشته باشد.

هنگامی که کارها در جدول ISPT منتظر اجرا شدن هستند اصطلاحا Hold شده اند و هنگامی که توسط زمانبند کار جهت اجرا انتخاب می شوند اصطلاحا به حالت Ready می روند. الگوریتم های زمانبندی کار یا به عبارت دیگر روشهای انتخاب Job عبارتند از:

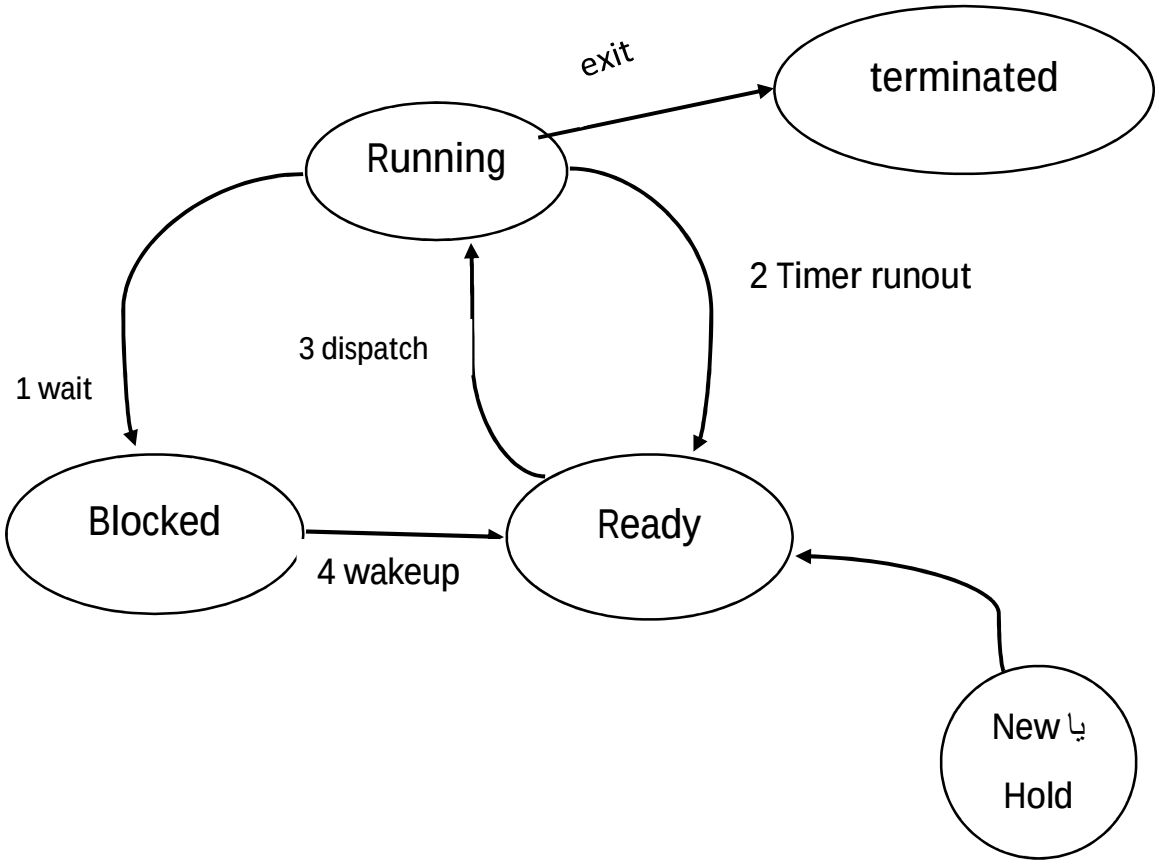
1-**روش FIFO**: در این الگوریتم که از اسم آن مشخص است(First In First out) اولین کاری که وارد ISPT شده انتخاب گردیده وتبدیل به پردازش (process) می گردد.

2-**روش SJF(Shortest Job First)**:در این الگوریتم کوتاه ترین کاری که در جدول ISPT وجود دارد پیدا شده جهت اجرا انتخاب می گردد.

3- **روش** Mixed: در این الگوریتم ترکیبی از کارهای CPU limitedو I/O جهت اجرا انتخاب می گردند.

## حالات یک پردازش

یک پروسس مطابق شکل زیر در زمان حیات خود می تواند در یکی از سه وضعیت اجرا (Running)، آماده (Ready)و بسته (Blocked) قرار گیرد.



هنگامی که دستور اجرا پردازشی صادر می گردد و یا Job scheduler کاری را جهت اجرا از جدول ISPT انتخاب می کند، این پردازش به وصف آماده وارد شده و منتظر CPU می ماند. انتقال 2 و3 توسط زمانبند پردازش( process scheduler) که بخشی از سیستم عامل است انجام می شود، بدون اینکه خود پروسس از آن اطلاعی داشته باشد. در انتقال 3 و CPU به پروسس داده می شود تا اجرا گردد. انتقال 2 زمانی صورت می گیرد که برش زمانی پردازش تمام شده و CPU باید از او گرفته شود، لذا پروسس از حالت Running (اجرا) به حالت Ready (آماده) می رود. در حالت Ready پردازش تمام منابع مورد نیاز را به جز CPU در اختیار دارد. زمانبندی یعنی اینکه در کدام زمان چه پردازشی جهت اجرا انتخاب گردد.

هنگامی که لازم باشد پردازش در حال اجرا برای بروز رخدادی (مثل تکمیل انتقال اطلاعات در دستورات I/O) صبر کند از آنگاه از طریق انتقال 1 از حالت Running به حالت Blocked (مسدود- بسته) می رود. پس از رفع علت انتظار (مثلا وقفه ای اعلام می کند که انتقال پایان یافته) پردازش از طریق انتقال 4 به حالت آماده رفته و در صف انتظاری CPU قرار می گیرد. گاهی اوقات به حالت Wait نیز می گویند.

این مدل پروسس، درک مسائل درون سیستم را ساده می کند. بعضی از پردازش ها برنامه های کاربران را اجرا میکند و برخی دیگر بخشی از خود سیستم عامل هستند. مثلا هنگامی که یک وقفه در رابطه یا دیسک رخ می دهد سیستم عامل تصمیم میگرد که پردازش در حال اجرا را قطع کرده و پروسس دیسک را اجرا کند. این پروسس قبلا جهت انتظار برای وقوع همین وقفه، بلوکه بوده است. بنابراین به جای پرداختن به وقفه ها ، به مسائلی مثل پروسس های کاربران، پروسس دیسک، پروسس ترمینال و غیره می اندیشیم که در حقیقت بلوکه شده اند تا یک اتفاق خارجی رخ دهد.

**تذکر**:در شکل حالات یک پردازش، گاهی اوقات بجای اصطلاح new از کلمه Hold استفاده میشود. بنابراین Jobی که در حالت Hold است یعنی در جدول ISPT روی دیسک قرار دارد و هنوز به پردازش تبدیل نشده است.

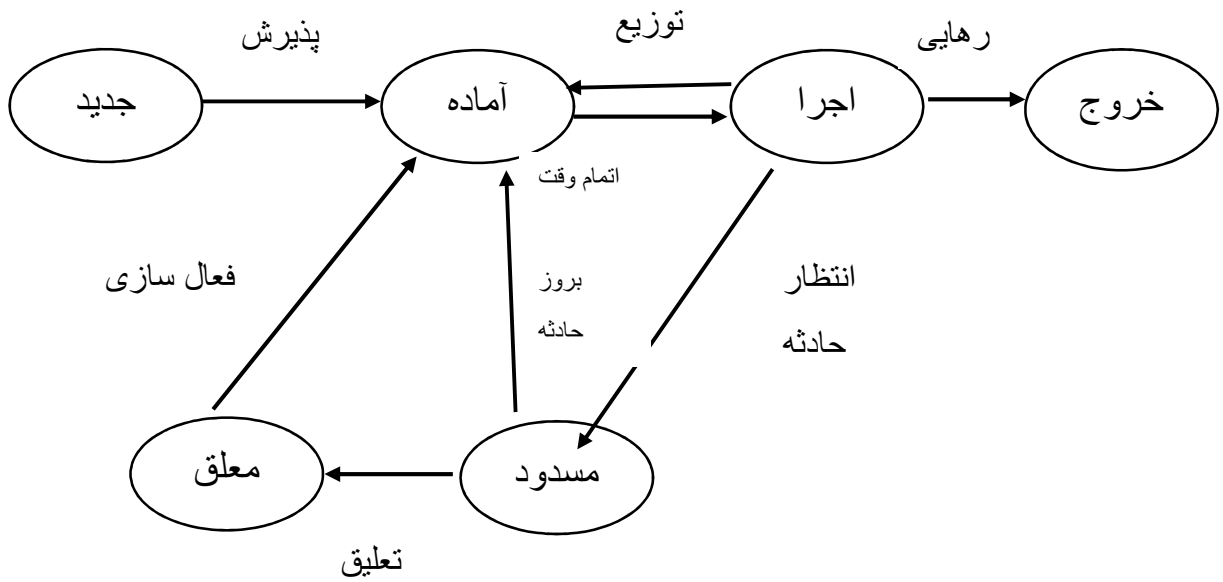
## حالت معلق (Suspend)

بسیاری از سیستم عامل های با سه حالت Blocked,Running,Ready پیاده سازی شده اند ولی اضافه کردن حالت معلق نیز در بسیاری موارد مناسب است. در سیستم قبلی

چند فرآیند در حافظه نگهداری می شوند و هنگامی که فرآیندی منتظر است، CPU میتواند به سراغ فرآیندی دیگر برود. اما غالبا CPU آنقدر سریع است که ممکن است تمام پردازش ها منتظر عمل I/O باشند. بنابراین حتی در سیستم چند برنامه ای نیز، CPU میتواند غالبا بیکار باشد.

برای رفع این مشکل می توان از تکنیک مبادله استفاده کرد، یعنی تمام یا بخشی از یک فرآیند را از حافظه اصلی به دیسک منتقل ساخت. هنگامی که هیچ یک از پردازش های موجود در حافظه اصلی در حالت اماده نیستند، در این حال سیستم عامل میتواند یکی از پردازش های مسدود را از حافظه اصلی خارج کرده و به صف فرآیند های معلق روی دیسک ببرد( مبادله کند) وقتی سیستم عامل عمل مبادله به خارج را انجام داد ، برای آوردن فرآیند بعدی به حافظه اصلی دو انتخاب دارد. یا می توان فرآیندی که به تازگی ایجاد شده است را بپذیرد یا فرآیندی که قبلا معلق بوده را بیاورد.

باتوجه به توضیحات فوق حالات یک پردازش را می توان به صورت زیر تقسیم کرد( کتاب استالینگ)



در این جا دو مفهوم مستقل وجود دارد:

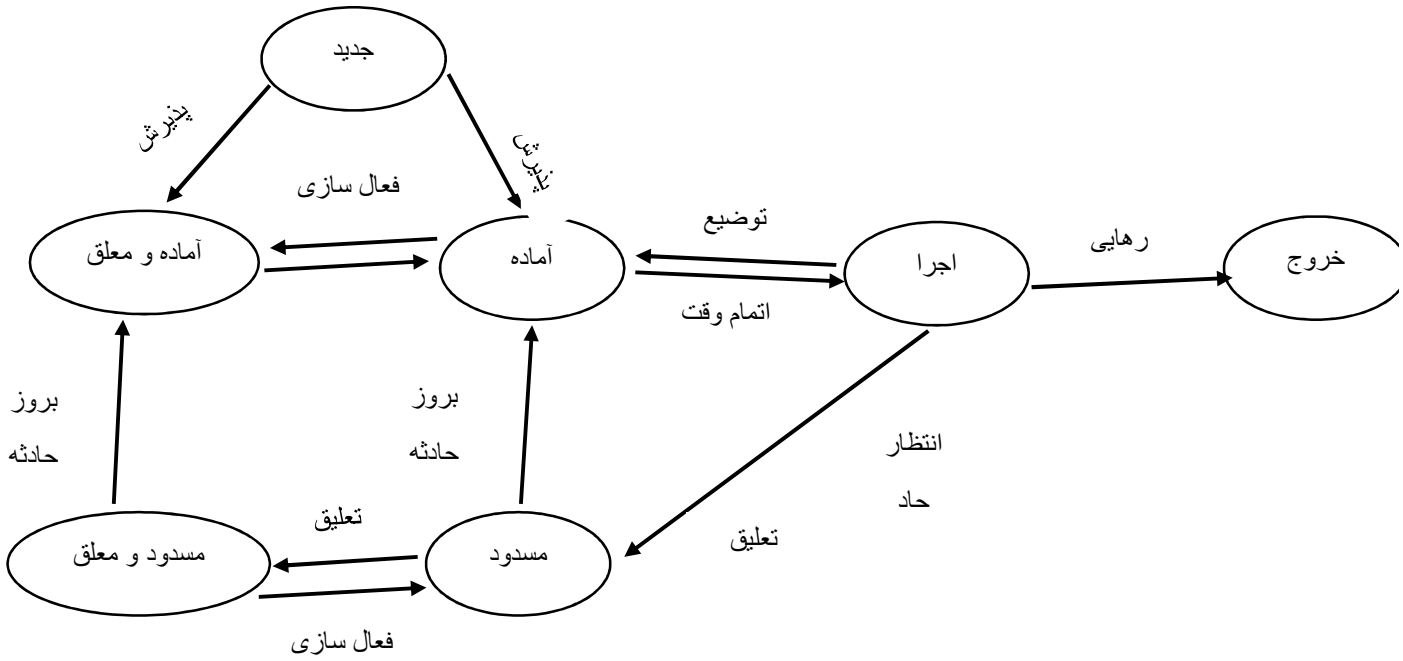
الف) فرآیند منتظر حادثه ای هست یا خیر( مسدود است یا خیر)

ب) فرآیند از حافظه اصلی بیرون برده شده است یا خیر( معلق است یا خیر)

بنابراین در یک بحث کاملتر باید گفت چهار حالت زیر امکان پذیر است:

1- آماده (ready) فرآیند در حافظهاصلی بوده و برای اجرا آماده است.
2- مسدود( waitیا blocked) : فرآیند در حافظه اصلی بوده است و منتظر بروز حادثه ای است.
3- مسدود و معلق ( suspend wait) : فرآیند در حافظه ثانویه بوده و منتظر بروز حادثه ای است . اگر تعداد فرآیند های مسدود شده در حافظه زیاد گردد ممکن است حافظه کم بیاید در این وضعیت سیسم عامل میتواند بعضی از فرآیند های منتظر را از حافظه به دیسک ببرد و بدین ترتیب این حالت مسدود و معلق پدید می اید.
4- آماده و معلق ( suspend ready) : فرآیند مورد نظر در حافظه ثانویه بوده و به محض بار شدن در حافظه اصلی آماده اجرا است. اگر در حالت ready، پردازش آماده باشد ولی حافظه نداشته باشیم به این حالت می رود.

با در نظر گرفتن حالت فوق ، نمودار حالت یک پردازش را به صورت کامل زیر می توان ترسیم کرد:



خط چین ها در شکل فوق نمایانگر آن است که تغییر حالت ممکن است ولی لازم نیست.

تذکر 1: به طور کلی می توان گفت هر پردازش سه مرحله ایجاد، گذراندان چرخه حالات و خاتمه را دارد و بخشی از سیستم عامل به نام ((مدیر پردازش )) سه مرحله ذکر شده را مدیریت میکند.

تذکر 2: بعضی از کتابها حالتی را به نام ((کامل)) معرفی کرده اند و منظور آنها هنگامی است که پردازش کارش تمام شده ولی هنوز از لیست پردازش های سیستم خارج نشده است.

## بلوک کنترل پردازش (PCB)

همانطور که قبلا گفتیم پردازش برنامه در حال اجراست. به بیانی دیگر پردازش ها فعالیتهایی با سرعت متفاوت بوده و وظیفه اصلی آنها اجرا کردن یک برنامه است. ولی از دید سیستم عامل می توان گفت پردازش یکسری ساختمان داده است. هر پردازش در سیستم عامل توسط یک ساختمان داده به نام بلوک کنترل پردازش یا PCB (Process Control block) نشان داده می شود. PCB شامل اطلاعات زیادی در مورد یک پردازش است. این اطلاعات مثلا هنگامیکه پروسس از ((حالت اجرا)) به حالت ((آماده)) می رود لازم است ذخیره شود که اگر پروسس خواست به حالت اجرا برگردد از همان نقطه ای که قطع شده بود، به درستی ادامه یابد. این اطلاعات عبارتند از:

- حالت جاری پردازش : که می تواند، آماده ، اجرا یا بسته باشد.
- شماره شناسایی پردازش (PID)
- شمارنده برنامه: (Program counter-PC) که آدرس دستورالعمل بعدی قابل اجرای پردازش می هد.
- محل حفظ ثباتها: هنگام وقوع وقفه یا سوئیچ کردن بین پردازشها ثباتهای پردازش جاری می بایست در PCB مربوط ذخیره شوند تا بعدا دوباره بازیابی شوند.
- اطلاعات زمانبندی CPU: مثل اولویت پردازش ، اشاره گرها به صف زمانبندی و غیره.
- اطلاعات مدیریت حافظه: مثل محل قرار گیری پردازش در حافظه و مسائل حفاظتی آن.
- نشانی محل برنامه پردازش
- اطلاعات وضعیت I/O : شامل لیستی از وسایل I/O تخصیص یافته به پردازش ، لیست فایلها باز شده برای پردازش و غیره. به عبارت دیگر نشانی سایر منابع پردازش .
- اطلاعات حسابرسی : مثل زمانبندی CPU مصرف شده برای پردازش ، شماره حساب ، شماره پردازش و غیره.

وقتیکه سیتم عامل CPU را به پردازش دیگر می دهد با استفاده از FCB تمام اطلاعاتی که جهت راه اندازی مجدد پردازش قبل لازم دارد را حفظ می کند. به این علملیات تعویض متن Context Swith گویند. تعویض متن به وسیله بخشی از سیستم عامل به نام Dispatcher انجام می پذیرد. از آنجا که سیستم عامل خیلی با PCB سرو کار دارد، در بسیاری از کامپیوترها ثباتی سخت افزاری که همیشه به PCB پردازش در حال اجرا اشاره می کند. دستوراتی نیز وجود دارند که خیلی سریع اطلاعات را در PCB بار می کنند. عملیات تعویض متن الزاما سربار اضافی (overhead) روی کامپیوتر ایجاد کرده و قدری از وقت CPU را جهت این کار به هدر می دهد، البته این زمان آنقدر زیاد نیست که بر مزیت چند برنامگی فلبه کند. زمان تعویض متن تابع سخت افزاری می باشد و به طور نمونه ای از این زمان از 1 تا 1000 میکرو ثانیه متغیر است.

## انواع زمانبندی ها

از یک جنبه زمانبندیهای پردازش به سه دسته:

الف- دراز مدت (Long term scheduler )

ب- کوتاه مدت (short term scheduler)

ج- میان مدت

تقسیم بندی میشوند.

در یک سیستم دسته ای پردازشهای بیشتری نسبت به آنچه فورا میتواندند اجرا شوند تحویل داده میشوند. این پردازشها در دیسک نگهداری میشوند. زمانبند دراز مدت (یا زمانبند کار job scheduler **یا** macro scheduler ) کارهای را انتخاب کرده و انها را برای اجرا از دیسک به حافظه اصلی می آورد.

زمانبند کوتاه مدت ( یا زمانبند cpu, زمانبند فرایند ها یا micro scheduler) از بین پروسسهای موجود در حافظه اصلی که اماده اجرا هستند یکی را انتخاب کرده و cpu را به آن اختصاص می دهد. غالبا زمانبند کوتاه مدت هر صد میلی ثانیه یک بار اجرا میشوند ولی زمانبد دراز مدت ممکن است هر چند دقیقه یک بار اجراء شود. در واقع زمانبند دراز مدت درجه چند برنامگی (degree of multiprogramming) ینی تعداد پردازشهای موجود در حافظه را کنترل میکند. زمانبند دراز مدت وفت زیادی برای تصمیم گیری دارد ولی زمانبند کوتاه مدت میبایست خیلی سریع تصمیم گیری کند. زمانبند دراز مدت میبایست مخلوط مناسبی از پردازشهای cpu-limited و I/O limited را جهت قرارگیری در حافظه انتخاب

کند تا کارایی cpu و وسایل I/O بهینه شود. در بعضی سیستم ها مثل اغلب سیستم های اشتراک زمانی زمانبند دراز مدت وجود ندارد, چرا که هر پردازش جدید جهت زمانبند CPU در حافظه گذاشته میشود تا زمان پاسخدهی به برنامه مناسب باشد.

البته بعضی سیستم عاملها از زمانبند میان مدت (medium-time) نیز استفاده میکنند بدین ترتیب که گاهی پروسهایی از حافظه و در واقع از رقابت جهت دریافت cpu حذف شده وبه دیسک برده میشوند

(swap out). بدین ترتیب درجه چند برنامگی کاهش میابد. سپس در زمانی دیگر پردازش مذکور مجددا به حافظه اورده شده(swap in) و اجرایش از همان نقطه قبلی ادامه میابد,این عملیات به نام مبادله (swapping) معروف است پس فرایند ها توسط زمانبند متوسط – مدت به بیرون یا درون حافظه و دیسک مبادله می شوند.

تکنیک مبادله را بعدا در فصل مدیریت حافظه بیشتر شرح میدهیم.

ما در این فقط به زمانبند کوتاه مدت یا زمانبند cpu می پردازیم.

زمانبند cpu به طور کلی میتواند انحصاری (غیر قابل پس گرفتن non preemptive) یا غیر انحصاری (قابل پس گرفتن preemptive) باشد.

در سیستم انحصاری فقط هنگامی cpu از پردازش در حال اجرا گرفته میشود که جهت عملیات I/O یا اتمام پردازش فرزند یا رخداد دیگری بلوک شود بنابر این مفهوم و پیاده سازی الگوریتم زمانبندی انحصاری ساده است. ولی ممکن است پردازشی برای مدتی طولانی CPU را جهت محاسبات در اختیار بگیرد. در این حال پردازشهای دیگر برای مدتی طولانی انتظار خواهند کشید و این موضوع مخصوصا برای سیستم های اشتراک زمانی نامناسب است. لذا در اغلب سیستم ها از یک زمانسنج (timer) داخلی برای ایجاد وقفه های متناوب سخت افزاری جهت گرفتن cpu استفاده می شود. در هر وقفه ساعت. سیستم عامل اجرا میشود تا تصمیم بگیرد که ایا به پروسس در حال اجرا اجازه ادامه کار را بدهد یا اینکه چون پروسس به اندازه کافی از زمان cpu استفاده کرده ان را معلق نماید تا cpu به پروسس دیگری تخصیص داده شود فرکانس این وقفه های ساعت بین 50 تا 60 بار در ثانیه است. این نوع زمانبندی که در ان پس از تمام شدن برش زمانی معین, cpu از پردازش گرفته می شود زمانبندی غیر انحصاری نام دارد.

حد پایین برش زمانی را دو پارامتر معین میکند.

1- هزینه های ثابت تویض task ها. مقدار زمان برش زمانی نباید کمتر از زمان تعویض برنامه ها باشد.

2- زمان اجرای نمونه ای (typical) یک فعل و انفعال .اگر برش زمانی کوچک باشد آنگاه هر کار کوچکی نیار به حداقل دو برش زمانی خواهد داشت.

به طور کلی میتوان گفت در اثر استفاده از برش زمانی ، زمان پاسخ برای فعل و انفعال های کوتاه تقلیل می یابد و این به قیمت بالا رفتن پاسخ فعل و انفعالاتی تمام میشود که احتیاج به چندین برش زمانی دارند.

عموما در سیستم غیر انحصاری برنامه ها نسبت به سیستم انحصاری موازی تر (نرم تر) اجرا میشوند. به عبارت دیگر در روش غیر انحصاری سعی می شود زمان پاسخ بهتری برای کاربران فراهم گردد. ولی این روش نسبت به تکنیک انحصاری هزینه های اضافی همراه خواهد داشت: 1) هزینه تعوض پردازش، یعنی زمانی که صرف جا به جایی cpu بین پردازش ها می شود. 2) هزینه حافظه بیشتر، چرا که تمام برنامه های در حال اجرا باید در حافظه قرار گیرند. بدیهی است که در سیستم ((یک کار در هر زمان )) حافظه کمتری مورد نیاز است. به عنوان مثال ویندوز 3,1 به صورت انحصاری و ویندوز NT و 98 به صورت غیر انحصاری عمل می کنند. هر چند که معمولا سیستم غیر انحصاری بهتر از سیستم انحصاری است ولی در بعضی موارد استفاده از مدل انحصاری معقول است. به عنوان نمونه در یک سیستم ویژه مثل خدمتگذار پایگاه داده (Data base server) پروسس اصلی می تواند از نوع انحصاری باشد که پروسس فرزندش را برای انجام یک درخواست آغاز می کند و به او اجازه می دهد که اجرا شود تا اینکه به پایان برسد یا بلوکه شود. در اینجا کلیه پروسس ها تحت کنترل پروسس اصلی پایگاه داده است که می داند هر یک از فرزندهایش قصد انجام چه کاری را دارد و اجرای آنها حدودا چه قدر طول می کشد.

**تذکر**: در یک سیستم شرط پایداری این است که میانگین نرخ ایجاد پردازش ها با میانگین نرخ خروج پردازشها برابر باشد.

## معیارهای زمانبندی

قبل از پرداختن به الگوریتم های زمانبندی لازم است معیار های انتخاب و مقایسه این الگوریتم ها مشخص میگردد. این معیار ها می تواند موارد زیر باشد:

1-عدالت (fairness) یعنی اطمینان از اینکه هر پروسس سهم عادلانه و منصفانه ای از cpu را دریافت کند.

2-کارایی یا بهره وری cpu (utilization-efficency) یعنی اینکه cpu در تمام زمانها (حتی الامکان) مشغول باشد.

3-زمان پاسخ(response time) یعنی به حداقل رساندن زمان پاسخ برای فرمان های محاوره ای کاربر. این زمان معمولابا سرعت ابزار خروجی محدود می شود.

4-زمان برگشت یا بازگشت (یا گردش کار turnaround) یعنی به حداقل رساندن زمانی که کاربران دسته ای باید منتظر بمانند تا خروجی آنها پدید آید. فاصله زمانی از لحظه تحویل کار تا لحظه تکمیل کار را زمان برگشت می نامند ولی زمان پاسخ مدت زمانی است که از صدور یک تقاضا تا تولید اولین پاسخ آن طول می کشد (نه زمان خروجی کل برنامه).

بدیهی است که همواره: ( زمان برگشت $\geq$ زمان پاسخ) میباشد.

زمان بارگذاری در حافظه + زمان عملیات I/O + زمان اجرا + زمان انتظار در صف اماده = زمان گردش کار

5-توان عملیاتی یا گذردهی (throughput) به تعداد پردازش هایی که در واحد زمان تکمیل میشوند توان عملیاتی میگویند. الگوریتم زمانبندی باید به گونه ای باشد که این معیار را افزایش دهد.

6-زمان انتظار (waiting time) الگوریتم زمانبندی cpu، بر میزان زمان اجرا پردازش یا اعمال I/O اثر نمی کند بلکه فقط در زمان صرف جهت انتظار در صف آماده اثر می گذارد. زمان انتظار، مجموع پریودهای زمانی صرف شده در صف آماده می باشد.

البته ممکن است تعدادی از اهداف فوق با هم در تضاد باشند.

به عبارتی دیگر میتوان گفت شرایط یک الگوریتم زمانبند عبارتند از: عادل بودن،کاهش هزینه های سیستم، استفاده متعادل از سیستم و منابع، قابل پیش بینی بودن عملکرد،توجه به اولویتهای برنامه، کاهش بار سیستم با عمل مبادله، عدم تعویض نامحدود یک پردازش، بالا بردن میزان کاربرد CPU فراهم اوردن زمان پاسخ مناسب به برنامه ها و انجام کار در فاصله زمانی معینی که کاربر ممکن است قید کند.

نکته مهم دیگر در زمان بندی ها رعلیت اولویت هاست. اولویتها میتوانند بصورت اتوماتیک توسط سیستم نسبت داده شوند و یا از خارج سیستم تعیین گردند. مثلا ممکن است یک کاربر کار فوری داشته باشد و حاضر باشد به خاطر بدست اوردن سرویس بالاتر هزینه بیشتری بپردازد، یعنی اولویت را بخرد یک اولویت ممکن است استاتیک باشد یا دینامیک. اولویت استاتیک تغییر نمیکند و بنابراین پیاده سازی آن ساده تر است. ولی این نوع اولویت در مقابل تغییرات محیطی تغییر میکند، مثلا ممکن است در آغاز یک برنامه اولویت پایینی داشته باشد ولی به تدریج اولوبت آن بهبود یابد.

در اینجا تعدادی از الگوریتم های زمانبندی مهم را بررسی خواهیم کرد. این زمانبندیها عبارتند از:

1- زمانبندیFCFS 2- Round Robin 3- اول کوتاه ترین کار(SJF) 4-SRT 5-HRRN 6- زمانبندی اولویت 7- زمانبندی صفهای چندگانه 8- زمانبندی صفهای چندگانه با فید بک 9- زمانبندی شانسی 10- زمانبندی LPT.

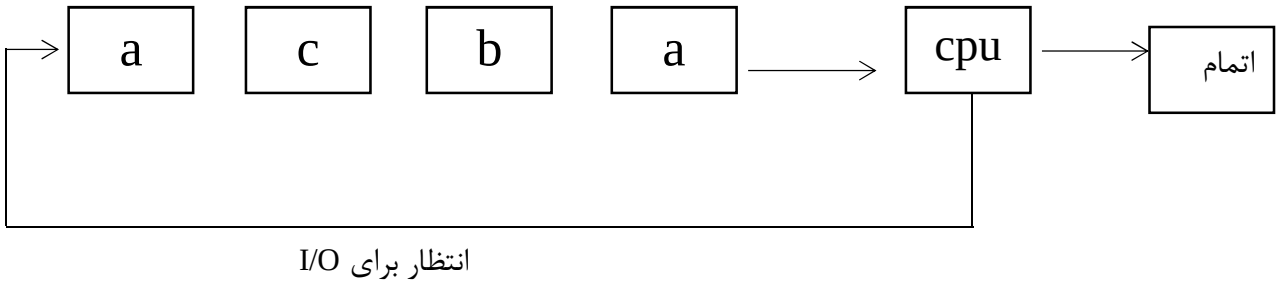
اجرای یک پردازش شامل سیکلی از اجرای cpu و سیکلی برای اجرای انتظار O/I است و پردازش مابین این دو حالت تغییر میکند. سیکلی را که پردازش cpu را در اختیار میگیرد انفجار محاسباتی (cpu burst cycle) و سیکلی را که پردازش در انتظار یک عمل I/O میباشد را انفجار I/O (I/O burst cycle) میگویند. به cpu burst cycle گاهی اوقات cpu burst time یا cbt نیز گفته می شود

در بررسی الگوریتم های زیر معمولا معیار مقایسه را میانگین زمان انتظار میگیریم.

همچنین جهت سهولت مثالها، تنها یک پریود انفجاری cpu را به ازاء هر پروسس و بر حسب میلی ثانیه در نظر خواهیم گرفت ولی دقت کنید که یک پروسس شامل چندین پریود انفجاری cpu است که هر یک ممکن است زمانهای متفاوتی داشته باشند.

## 1- زمانبندی FCFS

ساده ترین الگوریتم زمانبندی cpu، الگوریتم اول آمده- اول سرویس شده = first served – (first come FCFS ) می باشد. گاهی اوقات به این روش FIFO (first in first out) نیز میگویند. در این شیوه هر پردازشی که اول درخواست cpuرا صادر کند، اولین پروسسی خواهد بود که آن را به دست می آورد. این روش از نوع انحصاری ( non – preemptive) است که به سادگی توسط یه صف FIFO پیاده سازی میشود.



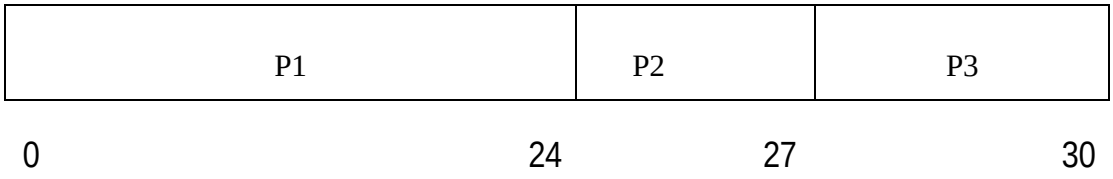

هنگامی که پردازش cpu را به دست گرفت آن را ره نمی کند مگر اینکه تمام شود یا جهت انجام عملیات I/O به حالت بسته برود.

متوسط زمان انتظار برای روش FIFO غالبا طولانی است این موضوع را با مثال نشان میدهیم:

**مثال1**: سه پردازش با زمان انفجار cpu زیر در زمان صفر وارد سیستم شده اند:

| پردازشها | P1 | P2 | P3 |
|---|---|---|---|
| زمان انفجار | 24 | 3 | 3 |

اگر پردازشها به ترتیب p3,p2,p1 وارد شوند بر طبق الگوریتم FCFS نمودار گانت (gantt chart) به صورت زیر خواهد بود:




یعنی زمان انتظار p1 برابر 0 میلی ثانیه، p2،24 میلی ثانیه و p3 برابر 27 میلی ثانیه است. لذا میانگین زمان انتظار (0+24+27)/3=17 میلی ثانیه می شود ولی اگر پردازشها به ترتیب p1,p3,p2 وارد سیستم شوند نمودار گانت به صورت زیر در می آید:


در این حالت میانگین زمان انتظار (0+3+6) ÷3=3 میلی ثانیه خواهد بود. پس میانگین زمان انتظار در روش PCFS معمولاً کمینه نیست و بر حسب تغییرات زمانی انفجار CPU تغییر می کند.

این شیوه باعق اثر اسکورت (Convoy effect) می شود. به این معنا که اگر یک پردازش CPU Limired (با زمان انفجار CPU نسبتاً زیاد) با پردازشهای دیگری (که زمان انفجار CPU کمتری دارند) چند برنامگی شود، برنامه های با زمان انفجار CPU اندک می بایست زمان زیادی در صف انتظار CPU باقی بمانند تا کار پردازش CPU Limited تمام

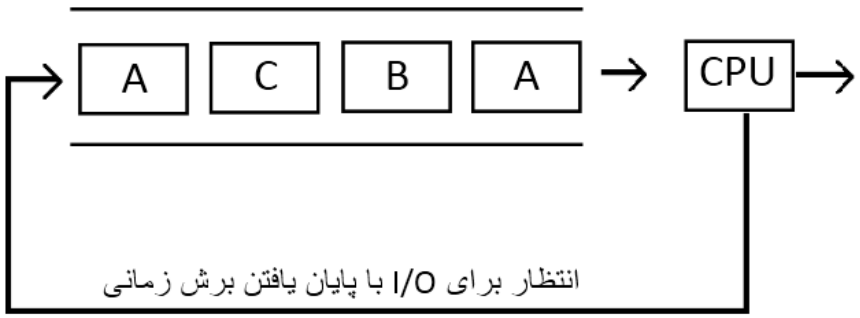
شود. این الگوریتم جهت سیستم های اشتراک زمانی می بایست به صورت محاوره ای باشند و جواب هر پردازش سریعاً داده شود اصلاً مناسب نیست.

معمولاً این روش همراه روش های دیگر استفاده می شود، مثلاً ممکن است برنامه ها بر اساس اولویت آنها زمانبندی شوند ولی برنامه هایی که دارای اولیت یکسان هستند طبق روش FCFS زمانبندی می گردند. مزیت FCFS سادگی پیاده سازی آن است و از طرف دیگر مطمئن هستیم پردازشی که در صف Ready قرار گرفته حتماً اجرا می شود.

## 2- زمانبندی نوبت گردشی (Round Robin = RR)

این زمانبندی یکی از قدیمی ترین، ساده ترین، عادلانه ترین و رایجترین الگوریتم های زمانبندی است و از نوع غیرانحصاری (preemptive) می باشد، این الگوریتم شبیه FCFS است ولی به هر پردازش حداکثر به میزان زمانی مضخصی CPU داده می شود. به عبارتی دیگر (یک واحد کوچک زمانی به نام کوانتوم زمانی time quantom) یا برش زمانی (time slice) تعریف می شود که معمولاً بین 10 تا 100 میلی ثانیه است و هر پروسس حداکثر به این میزان می تواند CPU را در اختیار بگیرد.) هنگامی که پردازشی CPU را در اختیار دارد دو حالت ممکن است رخ دهد؛ یا انفجار محساباتی جاری کمتر از یک کوانتوم زمانی است که در این حالت پردازش داوطلبانه CPU را رها می کند و منتظر اتمام عملیات I/O می شود (مانند FCFS) و یا اینکه انفجار محاسباتی بیشتر از یک کوانتوم زمانی است که در این جالت تایمر یک وقفه به سیستم عامل می دهد و سیستم عامل با تعویض متن (Context switch) CPU را از پردازش جاری گرفته و آن را به ته صف آماده می فرستد، سپس از ابتدای صف آماده، پردازش های دیگری را جهت اجرا انتخاب می کند (مطابق شکل زیر):

**تذکر**: اگر یک فرآیند تازه وارد و یک فرآیند قدیمی هر دو در یک زمان به انتهای صف برسند، فرآیندی که تازه وارد شده، در صف جلوتر قرار می گیرد.



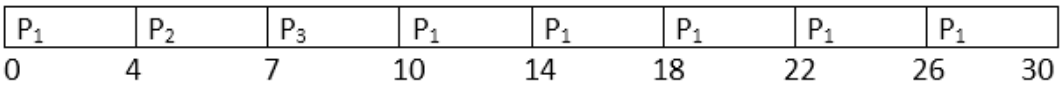

از این روش در سیستم های اشتراک زمانی Ineractive استفاده شده تا زمانهای پاسخ برای کاربران محاوره ای بصورت مناسب گارانتی شود.

میانگین زمان انتظار برای الگوریتم RR غالباً طولانی است و مثال زیر این موضوع را نشان می دهد.

**مثال** 2: سه پردازش با زمانهای انفجاری زیر در لحظه 0 وارد سیستم می شوند. میانگین زمان انتظار آنها را در سیستم RR بدست آورید، اگر کوانتوم زمانی 4 میلی ثانیه باشد.

| پردازشها | $P_1$ | $P_2$ | $P_3$ |
|---|---|---|---|
| زمان انفجار محاسباتی | 24 | 3 | 3 |

حل : نمودار گانت پردازشها به صورت زیر خواهد بود:




دقت کنید هنگامی که CPU به $P_1$ داده شد پس از کوانتوم زمانی از آن گرفته شده و سپس به $P_2$ داده می شود. ولی $P_2$ قبل از اتمام مهلت زمانی 4 میلی ثانیه، کاملاً تمام می شود و پردازش $P_3$، CPU را در اختیار می گیرد.

متوسط زمان پاسخ $= \frac{30+(7-0)+(10-0)}{3} = \frac{47}{3} = 15.66$

زمان انتظار مجموع پریودهای صرف شده در حالت انتظار در صف آماده است لذا برای $P_1$ داریم:

زمان انتظار $P_1$ = 10 – 4 = 6

میانگین زمان انتظار $= \frac{6+4+7}{3} = \frac{17}{3} = 5.66$ . متوسط زمان اجرا = $= \frac{3+3+24}{3} = \frac{30}{3} = 10$

در مثال فوق داریم: 15.66=10 + 5.66

یعنی در حالت کلی فرمول زیر برقرار است:

**متوسط زمان انتظار + متوسط زمان اجرا = متوسط زمان پاسخ (برگشت)**

**تذکر**1: عموماً منظور از مان پاسخ در تست ها، همان زمان برگشت است که از فرمول زیر به دست می آید:

**زمان برگشت پردازش = O – A**

O زمان خروج پردازش از سیستم و A زمان ورود پردازش به سیستم است که اگر A را ندهند آن را صفر در نظر می گیریم.

**تذکر** 2: در حالت کلی برای محاسبه زمان انتظار پردازش دلخواه می توان از فرمول زیر استفاده کرد:

$$W = O - A - S$$

O زمان خروج پردازش از سیستم است که معادل آخرین نقطه ای است که آن پردازش در نمودار گانت دیده می شود. A زمان ورود پردازش به سیستم است که اگر در مسئله داده نشود آن را صفر می گیریم. S طول اجرای پردازش است که عموماً در صورت مسئله داده می شود و برابر جمع قطعاتی از نمودار گانت است که متعلق به این پردازش می باشد.

در مثال فوق زمان انتظار $P_1$ یعنی $W_1$ را از راه $10 - 4 = 6$ به دست آوردین ولی طبق فرمول فوق این زمان را می توانیم از رابطه زیر نیز به دست آوریم:

$$W_1 = O - A - S = 30 - 0 - 24 = 6$$

کارایی الگوریتم RR به طور کامل به مقدار برش زمانی بستگی دارد. هنگامی که برش زمانی بی نهایت فرض شود در واقع الگوریتم RR تبدیل به الگوریتم FCFS می شود. به طور کلی کوانتوم زمانی نسبت به زمان سوئیچ متن باید بزرگ باشد چراکه در طی تعویض متن هیچ عمل مفیدی انجام نمی شود. مثلاً اگر هر برش زمانی 10 میلی ثانیه و زمان تعویض متن 1 میلی ثانیه باشد آنگاه 10% وقت CPU به هدر می رود که مقدار زیادی است.

حد بالای برش زمانی از فرمول زیر به دست می آید:

زمان پاسخ مناسب * N = برش زمانی

که N تعداد پردازشهاست.

مثلاً اگر در هنگام وارد کردن خطوط یک برنامه زمان پاسخ حدود 1 ثانیه است ولی در هنگام کامپایل کردن برنامه ها زمان پاسخ می تواند بیشتر باشد.

پس حد بالای کوانتوم زمانی باید به قدری باشد که زمان پاسخ دهی مناسبی داشته باشیم. حد پائین برش زمانی توسط دو عامل تعیین می شود یکی اینکه باید این برش خیلی بزرگتر از زمان تعویق متن باشد مثلاً هزاران برابر. دیگر آنکه مقدار برش زمانی بایستی کمی بزرگتر از زکان لازم برای یک فعل و انفعال نوعی باشد. چرا که در غیر اینصورت هر کار کوچکی نیاز به چندین برش زمانی خواهد داشت و کارایی سیستم به علت تعویض متنهای متعدد کم می شود. یک قاعده سرانگشتی این است که 80 درصد انفجارهای محاسباتی به علت باید کوتاه تر از کوانتوم زمانی باشند و در عمل برای این امر برش زمانی را حدود 100 میلی ثانیه در نظر می گیرند.

با توجه به توضیحات فوق مشخص می شود که اصولاً الگوریتم RR برای کاهش زمان پاسخ به کار می رود. زمان برگشت نیز به اندازه برش زمانی بستگی دارد و مثال زیر این موضوع را نشان می دهد.

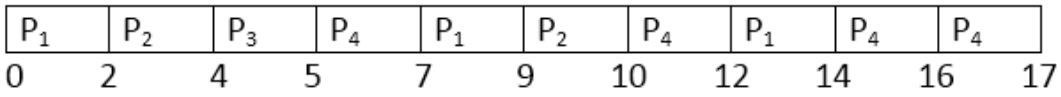
**مثال 3**: پردازش را با زمانهای انفجار محاسباتی زیر در نظر بگیرید که در زمان صفر وارد سیستم شده اند. برای کوانتوم های 1 تا 7 میلی ثانیه میانگین زمان برگشت را محاسبه کنید.

(زمان برگشت = مدت زمان تحویل پردازش به سیستم تا تمام شدن آن)

| پردازشها | $P_1$ | $P_2$ | $P_3$ | $P_4$ |
|---|---|---|---|---|
| زمان انفجار محاسباتی | 6 | 3 | 1 | 7 |

حل: ما در اینجا مسئله را برای کوانتوم 2 حل می کنیم و بقیه حالات را شما بدست آورد:

برای کوانتوم 2:




متوسط زمان برگشت یا پاسخ = $\frac{14+10+5+17}{4} = \frac{46}{4}$ 11.5

متوسط زمان اجرا = $\frac{7+1+3+6}{4}$ = 4.25

برای محاسبه متوسط زمان انتظار اگر از فرمول W = O – A - S استفاده کنیم، داریم:

متوسط زمان انتظار = $\frac{(14-6)+(10-3)+(5-1)+(17-7)}{4} = \frac{8+7+4+10}{4} = \frac{29}{4}$ = 7.25

اگر جمع زمان هایی در نمودار گانت را در نظر بگیریم که پردازش مورد نظر در سیستم حضور داشته ولی CPU را در اختیار ندارد، خواهیم داشت:

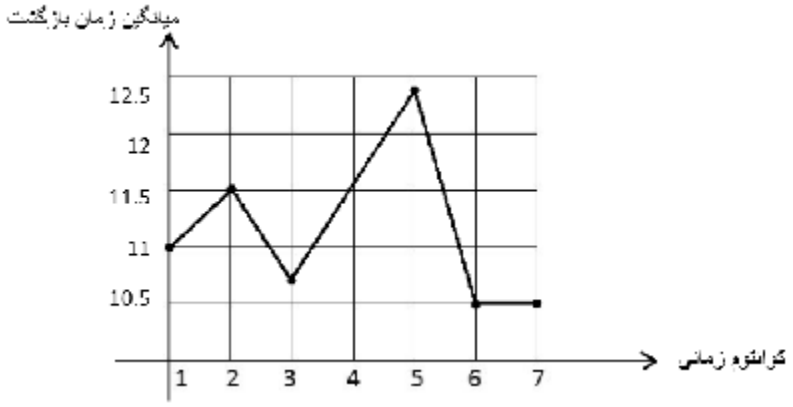
متوسط زمان انتظار = $\frac{[(7-2)+)12-9)+(2-0)+(9-4)]+[(4-0)]+[(5-0)+(10-7)+(14-12)]}{4}$

$= \frac{8+7+4+10}{4} = \frac{29}{4}$ = 7.25

با توجه به فرمول زیر داریم:

متوسط زمان پاسخ = متوسط زمان اجرا + متوسط زمان انتظار = 11.5 = 2.25 + 7.25

به همین ترتیب هنگامی که این متوسط زمان برگشت را برای زمانهای 3 تا 6 بدست می آوریم شکل زیر حاصل می شود:




این شکل نشان می دهد که میانگین زمان برگشت یک مجموعه پردازش الزاماً در اثر افزایش کوانتوم کم نمی شود. ولی به طور کلی اگر اکثر پرداشها انفجارهای محاسباتی خود را در یک کوانتوم زمانی به اتمام برسانند زمان برگشت بهبود می یابد (کم می شود).

## 3- زمانبندی اول کوتاهترین کار (SJF)

در الگوریتم SJF (Shortest Job First) که روشی انحصاری است CPU به پردازش داده می شود که کوچکترین انفجار محاسباتی بعدی را دارد. البته اصطلاح مناسبتر، »کوتاهترین انفجار محاسباتی بعدی« می باشد. زیرا این زمانبندی براساس طول مدت انفجار CPU بعدی عمل می کند و نه براساس طول کل پردازش. اگر دو پردازش مدت انفجار محساباتی یکسانی داشته باشند براساس FCFS زمانبندی می شوند. این الگوریتم می تواند انحصاری و غیرانحصاری باشد.

**مثال 4**: چهار پردازش با زمانهای انفجاری زیر را در نظر بگیرید. اگر الگوریتم زمانبندی SJF انحصاری باشد زمان انتظار میانگین را بدست آورید. همین زمان را در الگوریتم FCFS به شرط ترتیب ورودی $P_1$ تا $P_4$ بدست آورید.

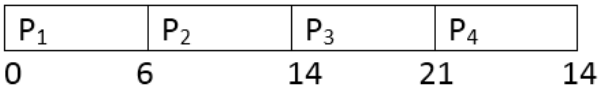
| پردازشها | $P_1$ | $P_2$ | $P_3$ | $P_4$ |
|---|---|---|---|---|
| زمان انفجار محاسباتی | 6 | 8 | 7 | 3 |

حل : نمودار گانت SJF به صورت زیر می شود:

| $P_1$ | $P_1$ | $P_3$ | $P_2$ |
|---|---|---|---|
| 0 | 3 | 9 | 16 24 |

میانگین زمان انتظار = $\frac{0+3+9+16}{4} = \frac{28}{4} = 7$

نمودار گانت SCFS به صورت زیر است :




میانگین زمان انتظار = $\frac{0+6+14+21}{4} = \frac{41}{4} = 10.25$

همانطور که مشاهده می شود این الگوریتم زمان انتظار کمتری نسبت به روش FCFS دارد.

می توان ثابت کرد در حالت کلی این زمانبندی SJF از نظر میانگین زمان انتظار یا میانگین زمان برگشت بهینه (کمترین) حالت است. فرض کنید چهار کار با زمانهای اجرای d, c, b, a موجودند. اولین کار در زمان a دومین کار در زمان a+b سومی در زمان a+b+c و چهارمی در زمان a+b+c+d تمام می شوند. در این حالت متوسط زمان برگشت از رابطه زیر بدست می آید:

$$\frac{a+(a+b)+(a+b+c)+(a+b+c+d)}{4} = \frac{(4a+3b+2c+d)}{4}$$

از آنجاکه a بیشترین اثر را در زمان متوسط بازگشت دارد، در نتیجه باید کوتاهترین کار باشد و به همین ترتیب d, c, b.

این الگوریتم مخصوصاً برای کارهای دسته ای که از قبل زمان اجرای آن کارها، مشخص و معین باشد به کار می رود. مهمترین مشکل در SJF آگاهی از طول درخواست بعدی CPU می باشد. هیچ راهی که طول انفجار محساباتی بعدی را برای ما مشخص سازد وجود نداردو لذا درصورت لزوم مجبوریم آن را پیش بینی کنیم. یعنی انتظار داشته باشیم که طول انفجار بعدی خیلی شبیه طول انفجارهای قبلی باشد. برای این تخمین معمولاً از فرمول میانگین تمانی یعنی:

$$S_{a+1} = (1-a)S_n + aT_n$$

استفاده می شود که در آن $0 \leq a \leq 1$ است. $T_a$ زمان اجرای فرایند برای n ُمین نوبت (نوبت اخیر) است. $S_a$ جمع زمانهای گذشته را نشان می دهد. مثلا اگر $a = \frac{1}{2}$ باشد و زمان اولین انفجار محاسباتی $T_0$ باشد آنگاه تخمین های بعدی به صورت زیر خواهد بود:

$$T_0, \left(\frac{T0}{2}\right) + \frac{T1}{2}, \left(\frac{T0}{4} + \frac{T1}{4}\right) + \frac{T2}{2}, \left(\frac{T0}{8} + \frac{T1}{8} \quad \frac{T2}{4}\right) \quad \frac{T3}{2}, \ldots$$

این فرمول نشان می دهد که برای تخمین زمان انفجار بعدی از زمانهای واقعی تمام انفجار های قبلی استفاده کرده ایم. ولی اثر زمان انفجار اخیر بیشتر از زمانهای دورتر است. مثلاً در فرمول پس از سه مرحله اثر $T_0$ به $\frac{1}{8}$ تعدیل یافته است.

مطلب فوق نشان میدهد که aدر واقع وزن نسبی مشاهدات اخیر را تعیین میکندو مثلاً اگر a=1 باشد تنها زمان انفجار قبلی ملاک است. اگر a=0.8 باشد تقریباً تنها چهار مشاهده اخیر مهم است. اگر A=0.2 باشد حدود هشت مشاهده اخیر مهم است. محاسبه تخمین زمان انفجار بعدی با فرمول میانگین تمانی محاسبات کمی دارد. لذا سربار کمی بر روی کامپیوتر می گذارد مخصوصاً برای $a = \frac{1}{2}$ کافی است مقدار جدید را به جمع تخمینهای قبلی اضافه کنیم و حاصل را بر 2 تقسیم کنیم (یک بیت شیفتبه راست دهیم). این تکنیک تخمین زدن را معمولاً رشد سالمندی (Aging) می نامند.

مشکل دیگر استفاده از SJF امکان گرسنگی یا قحطی زدگی (Starvation) فرایندهای طولانی در صورت ورود دائم فرایندهای کوچکتر است. چرا که در این صورت مرتباً پردازشهای کوتاه اجراء شده و نوبت به فرایندهای طولانی نمی رسد.

**تذکر**: نام دیگر این الگوریتم SPN (Shortest Proccess Next) می باشد.

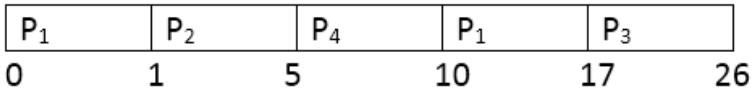
## 4- زمانبندی کوتاه ترین زمان باقی مانده (SRT)

زمانبندی (Shortest Remaining Time = SRT) شبیه SJF ولی از نوع غیرانحصاری (preemtive) است. در SRT برنامه ای که احتیاج به کمترین زمان جهت تکمیل دارد ابتدا اجراء می شود. در هنگام انتخاب یک برنامه کارهایی که تازه به صف آماده وارد می شوند هم در نظر گرفته می شوند. در این حالت ممکن است CPU از یک برنامه در حال اجراء توسط برنامه جدیدی که نیاز به زمان کمتری جهت تکمیل دارد گرفته شود. در این زمان روش هم مانند SJF نیاز به تخمین آینده داریم و احتمال بروز مشکل گرسنگی نیز وجود دارد. SRT در مقایسه با SJF زمان کل بهتری را ارائه می کند زیرا ه کار کوتاه نسبت به کار بلند در حال اجراء، اولویت می دهد. ولی از طرف دیگر به خاطر همین پس گرفتنهای CPU و مقایسه های زمانی بیشتر، روش SRT از SJF پرهزینه تر است.

**مثال 5**: چهار پردازش زیر را در نظر گرفته و میانگین زمان انتظار را برای حالت SRT و SJF بدست آوید.

| پردازشها | $P_1$ | $P_2$ | $P_3$ | $P_4$ |
|---|---|---|---|---|
| زمان انفجار | 8 | 4 | 9 | 5 |
| زمان ورود | 0 | 1 | 2 | 3 |

حل : نمودار گانت SRT به صورت زیر می شود:



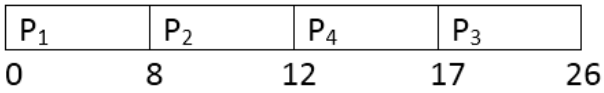

پردازش $P_1$ در زمان صفر شروع می شود چون تنها پردازش موجود در صف است. هنگامی که $P_1$ در زمان وارد می شود زمان باقی مانده $P_1$ برابر 7 است و زمان $P_2$ (برابر 1) بیشتر می باشد لذا CPU به $P_2$ می شود و الی آخر.

زمان انتظار زمانی است که پردازش جهت اجراء در صف آماده انتظار کشیده است.

$P_1$ زمان انتظار = 10 – 1 = 9 $P_2$ زمان انتظار = 1 – 1 =0 , .....

میانگین زمان انتظار = $\frac{(10-1)+(1-1)+(17-2)+(5-3)}{4} = \frac{26}{4} = 6.5$

برای زمان بندی SJF نمودار گانت به شکل زیر در می آید:




متوسط زمان انتظار = $\frac{(0-0)+(8-1)+(12-3)+(17-2)}{4} = 7.75$

**تذکر**1: به روش SRT الگوریتم SRTF (Shortest Remaining Time First) نیز می گویند.

**تذکر** 2: اگر همه فرآیندها با هم وارد سیستم شوند همان روش SJF خواهد شد.

## 5- زمانبندی بالاترین نسبت پاسخ (HRRN)

زمان بندی HRRN (Highest Response Ratio Next) نوعی زمانبندی انحصاری است که بعضی از مشکلات SJF را برطرف می سازد. در SJF نظر افراطی خوبی نسبت به کارهای کوتاه و برعکس نظر افراطی بدی نسبت به کارهای طولانی وجود دارد به طوری که ممکن است مشکل قحطی زدگی رخ دهد. در این زمانبندی اولویت ها دینامیک بوده و از فرمول زیر محاسبه می شود:

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

چون زمان سرویس در مخرج است پس کارهای کوتاهتر اولویت بیشتری داشته و زودتر اجراء می شوند. ولی از طرف دیگر چون در صورت زمان انتظار را داریم کارهای طولانی نیز که مدت زیادی در صف انتظار بوده اند اولویت بیشتری کسب کرده و بالأخره در یک زمان معین اجراء می شوند. بدین ترتیب مشکل قحطی زدگی برطرف می شود. صورت کسر همان زمان پاسخ سیستم است.

**تذکر**: نام دیگر این الگورتیم، HRN می باشد.

## 6- زمانبندی اولویت (priority scheduling)

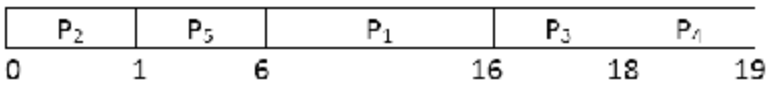
در بسیاری از سازمانها اولویت کارها با همدیگر فرق میکنند. مثلاً در کامپیوتر چندکاربره دانشگاه اولویت رئیس دانشگاه از همه بیشتر است، سپس اساتید و بعد دانشجویان قرار دارند. لذا کارهای اساتید باید زودتر از دانشجویان اجراء شود. حتی در یک کامپیوتر شخصی نیز ممکن است اولویت پروسس ها با هم فرق داشته باشد. مثلاً اولویت پردازش نمایش فیلم بیشتر از پست الکترونیکی و دریافت fax است. اولویتها هم می توانند توسط اپراتور به صورت خارجی برای سیستم تعریف شوند و هم خود سیستم عامل به صورت داخلی بنابر فاکتورهایی اولویتها را مشخص می سازد.

زمانبندی اولویت می تواند انحصاری و غیرانحصاری باشد. ولی اغلب به صورت انحصاری پیاده سازی می شود.

وقتی پردازشی وارد صف آماده می شود، برتری او با برتری پردازش در حال اجراء مقایسه می شود. اگر اولویت پروسس جدید بیشتر از پردازش در حال اجرا باشد در سیستم غیرانحصاری CPU را پس خواهد گرفت ولی در سیستم انحصاری پردازش مذکور در جلوی صف آماده قرار می گیرد.

**مثال** 6: در یک سیستم انحصاری پردازشهای زیر موجودند. در الگوریتم اولویت، میانگین زمان انتظار را به دست آورید (عدد کوچکتر نمایانگر اولویت بیشتر است).

| پردازشها | $P_1$ | $P_2$ | $P_3$ | $P_4$ | $P_5$ |
|---|---|---|---|---|---|
| زمان انفجار | 10 | 1 | 2 | 1 | 5 |
| زمان ورود | 3 | 1 | 3 | 4 | 2 |




حل : نمودار گانت به صورت زیر خواهد بود:

میانگین زمان انتظار = $\frac{0+1+6+16+18}{5}$ = 8.2

به عنوان مثالی دیگر سیستم UNIX دستوری به نام nice دارد که هر کاربر می تواند به طور ارادی اولویت پروسس خود را کاهش دهد تا وضع کاربران دیگر بهتر شود (البته کسی از این دستور استفاده نمی کند).