وقفه‌های داخلی (trap) که بر اثر اجرای دستورات خود برنامه به صورت داخلی در CPU رخ می‌هند.

وقفه‌های خارجی که از دستگاههای خارجی مثل دستگاههای ورودی یاخروجی ,DMA,تایمرها , صفحه کلید و خطاهای سخت افزاری ناشی می‌شوند.

وقفه‌های نرم افزاری(یاهمان SVC) که بر اثر فراخوانی توابع سیستمی توسط برنامه رخ می‌دهند.

با اجرای مجدد برنامه , وقفه‌های داخلی به همان صورت قبلی دوباره رخ می‌دهند ولی وقفه‌های خارجی مستقل از دستورات برنامه و ناهمگام با برنامه می‌باشند.

اگر چند منبع همزمان تقاضاهایشان را از طریق یک خط وقفه به CPU اعلام کنند , آنگاه CPU با روش همه پرسی یا سرکشی(polling) منبع وقفه دهنده را تشخیص خواهد داد.)

فراخوانی سیستمی

فراخوان های سیستمی رابط ما بین سیستم عامل و برنامه‌های کاربردی می‌باشند. در زبان سطح بالایC)) و ((پاسکال مستقیماً می‌توان این فراخوان‌های سیستمی را به کار برد. از فراخوانهای سیستمی عبارتند از:

مدیریت پردازشها: مانند ایجاد و اتمام پردازش , بارگذاری و اجرای پردازش در سیستم عامل , تخصیص و آزاد کردن حافظه و غیره.

مدیریت فایلها و فهرستها: ایجاد و حذف فایل , باز وبسته کردن فایل , خواندن و نوشتن , تغییر صفات فایل و غیره

مدیریت وسایل: درخواست و رهاسازی وسیله,خواندن و نوشتن در وسیله وغیره

بدست آوردن اطلاعات : خواندن و تنظیم تاریخ و زمان, خواندن زمان استفاده از سیستم توسط کاربر,تعداد کاربران , میزان فضای آزاد حافظه یا دیسک , نسخه سیستم عامل وغیره.

اکثر سیستم عامل (مثل UNIX,DOS) به وسایل I/O مشابه فایلها نگاه می‌کنند و ابزارهای I/Oبا نامهای فایلهای ویژه شناخته می‌شوند. در این حال برای کار با وسایل I/O می‌توان از همان دستورات readو write فایلها استفاده کرد.

اگر چند منبع در یک لحظه همزمان سیگنال وقفه را ارسال کنند,CPUبا تکنیکهایی آنها را اولویت بندی کرده و سپس بر اساس اولویت سرویس می‌دهد.

CPU به جای تکنیک وقفه در سیستم عامل می‌تواند با تکنیک سرکشی (polling) متوجه شودکه کدام وسیله به سرویس دهی نیاز دارد. دراینحالت ثباتهای کنترلی دستگاههای جانبی مرتباً چک می‌شود تا نیاز سرویس دهی آنها مشخص گردد. این روش باعث اتلاف وقت CPU می‌گردد.

انوع سیستم عامل از نظر ساختار:

تکنیک سیستم یکپارچه

سیستمهای تجاری زیادی وجود دارند که ساختار خوش تعریفی ندارند. اغلب این سیستم عاملهابه عنوان سیستم های کوچک و محدودی شروع شده‌اند وسپس به تدریج ورای دید اولیه طراحان گسترش یافته‌اند .

سیستم عامل DOS از این دسته می‌باشد.

سیستم عامل به صورت یک مجموعه از رویه‌ها نوشته شده است که هر یک از آنها می‌توانند دیگری را به هنگام نیاز فراخوانی کنند . برای مخفی کردن اطلاعات امکاناتی وجود ندارد و هر رویه برای دیگر رویه‌ها کاملا قابل مشاهده است .

مثلاً در MS-DOS واسطه‌ها و سطوح عملیاتی به خوبی مجزا نشده‌اند و مطابق شکل زیر برنامه‌های کاربردی می‌توانند مستقیماً به توابع ROM BIOS ویا حتی پورت دستگاههای مختلف (مثل هارد دیسک) دسترسی پیدا کنند, لذا به راحتی می‌توان برنامه‌های مخرب زیادی تحت DOS پدید آورد.

اکثر CPU ها دارای دو مد کاری هستند مد هسته که مخصوص سیستم عامل است و در آن تمامی دستورالعملها مجاز می‌باشد و دیگری مد کاربر است که مخصوص برنامه‌های کاربران بوده ودر آن دستورات I/O و دستورالعملهای معین دیگری مجاز نمی‌باشند.

سیستم عامل DOS توسط سخت افزار زمان خود «پردازنده ۸۰۸۸» محدود بوده است چرا که این پردازنده فقط در یک مد کار می‌کند و تمام دستورات در آن مجاز می‌باشد ولی پردازنده۳۸۶ دارای مدهای مختلفی است که سیستم عامل ویندوز از آن به خوبی استفاده می‌کند.

برنامه‌ای کاربردی یکی از فراخوانهای سیستمی (توابع سیستم عامل) را صدا می‌زند . در این حال ماشین از مد کاربر (user mode)به مد هسته (kernel mode) تغییر حالت می‌دهد و کنترل به سیستم عامل سپرده می‌شود .سیستم عامل با توجه به پارامترهای تابع مذکور تعیین می‌کند کدام فراخوان سیستمی باید اجراء شود سپس سیستم عامل به جدولی رجوع می‌کند که در ردیف k ام آن جدول یک اشاره‌گر به رویه اجراء کننده فراخوان سیستمی وجود دارد..سپس آن روتین اجراء شده و در انتها کنترل به برنامه کاربر بر می‌گردد.

تکنیک سیستم لایه ای

در روش لایه‌ای سیستم عامل به تعدادی سطح یا لایه تقسیم می‌شود که هر کدام در بالای لایه پائین تر قرار می‌گیرند. مزیت مهم این روش پیمانه‌ای (modularity) بودن آن است . یعنی لایه‌ها به گونه‌ای تقسیم بندی می‌شوند که هر لایه فقط توابع و سرویس های لایه پائین تر را استفاده می‌کند. بدین ترتیب هر لایه را می‌توان مستقل از لایه‌های دیگر طراحی کرد , بسط دادو خطایابی کرد.

هر سطح با استفاده از اعمال لایه‌های پایین تر پیاده سازی می‌شود ولی آن سطح نمی‌داند که اعمال سطح پایین چگونه پیاده شده‌اند و فقط باید بداند که آن اعمال چه می‌کنند. بدین ترتیب هر لایه مسائلی را از لایه‌های بالاتر مخفی می‌سازد.

اولین سیستم لایه‌ای , سیستم THE با ۶ لایه بود: لایه صفر مسائل زمانبندی(scheduling) پردازنده را انجام می‌دهد یعنی اینکه در هر لحظه CPU در اختیار کدام برنامه باشد . لایه یک مدیریت حافظه (اصلی و جانبی) را بر عهده دارد . لایه دو ارتباط بین هر پروسس و کنسول اپراتور را برقرار می‌سازد.

لایه سه مدیریت دستگاههای I/O و بافر کردن اطلاعات را برعهده دارد . در بالای این لایه هر پروسس به جای دستگاههای I/O حقیقی و پیچیده با دستگاههای ساده و مجازی I/O سرو کار دارد . در لایه چهار برنامه‌های کاربران اجراء می‌شوند که هیچ نگرانی در مورد مدیریت پروسس ,حافظه, کنسول و I/O ندارند. در لایه پنجم پروسس اپراتور سیستم قرار می‌گیرد.

مشکل اصلی در روش لایه‌لایه, تعریف مناسب لایه‌های مناسب است . از آنجا که یک لایه فقط می‌تواند لایه‌های پایین تر را به کار برد برای طراحی آن باید دقت زیادی به خرج داد. مشکل دیگر این ساختار این است که نسبت به انواع دیگربازدهی کمتری دارند .

هنگامی که دستورات از لایه بالا به سمت پایین حرکت می‌کنند, در هر لایه پارامترهای دستور ممکن است , از نظر صحت بررسی شده و یا تغییر یابند .لذا هر لایه قدری بار سر (overhead)به سیستم اضافه می‌کند و در نتیجه فراخوانی سیستمی نسبت به سیستم غیر لایه‌ای بیشتر طول می‌کشد. لذا در سالهای اخیر سعی شده است لایه‌های کمتری با قابلیت عمل بیشتری طراحی شود .

به عنوان مثال محصول اولیه windows NT با لایه‌های زیاد, کارایی کمتری نسبت به ویندوز ۹۵ داشت . در NT4.0 سعی شد لایه ها به همدیگر نزدیکتر و مجتمع تر شوند تا کارایی بیشتر گردد.

سیستم MULTICS به جای لایه‌ها به صورت یکسری حلقه‌ها متحدالمرکز سازماندهی شده است بطوریکه هر حلقه داخلی از امتیازات بالاتری نسبت به حلقه خارجی خود بهره مند می‌باشد.اگر یک رویه از حلقه خارجی بخواهد یک رویه از حلقه داخلی را صدا بزند .

بایدیکی را فراخوان‌های سیستمی را اجراء کندو اعتبار پارامترهای این دستورالعمل قبل از اجراء به دقت بررسی می‌شود. مثلاً یک استاد برنامه گرفتن امتحان ونمره دادن را در حلقه n می‌نویسد و برنامه دانشجویانش در حلقه n+1 اجراء می‌شود, بدین ترتیب دانشجویان نمی‌توانند نمره خود را تغییر دهند

سیستم مجازی در سیستم عامل

سیستم عامل VM بر روی سیستمهای IBM بهترین مثال از مفهوم ماشین مجازی است . قلب سیستم که به مانیتور ماشین مجازی(Virual Machine Monitor) معروف است , بر روی سخت افزار عریانی اجراء شده و چند برنامگی را پدید می‌آورد, این مانیتور مجازی را در لایه بالاتر فراهم می‌سازد.

این ماشین های مجازی برای کاربران مشابه یک نسخه از سخت افزار عریان هستند که دارای مودهای کابر و هسته , I/O , وقفه‌ها و چیزهای دیگر «ماشین حقیقی» می‌باشند.

به هر کاربر ماشین مجازی خودش داده می‌شود و او می‌تواند هر یک از سیستم عامل‌ها یا بسته‌های نرم افزاری موجود را روی ماشین خودش اجراء کند.

هر کاربر یک برنامه(Conversational Monitor System) مخصوص به خود را دارد که یک سیستم عامل تک کاربره محاوره‌ای است .

مزایای این ماشین مجازی عبارتند از:

در این سیستم دو وظیفه اصلی چند برنامگی و ایجاد واسطه راحت(مستقل از سخت افزار) از یکدیگر مجزا شده‌اند . مانیتور ماشین مجازی وظیفه چند برنامگی را بر عهده داردو لایه بالای آن وظیفه ایجاد واسطه کاربر با سخت افزار را بر عهده دارد. لذا هریک از این بخشها ساده‌تر شده و از قابلیت انعطاف بیشتری برخوردارند.

هر ماشین مجازی از سایر ماشین ها کاملاً جداست . بنابراین هیچ مشکل امنیتی وجود نخواهد داشت و برنامه‌های کاربران تداخلی با همدیگر ندارند.

از آنجا که هر ماشین مجازی کاملاً مشابه سخت افزار واقعی است , هر یک از آنها می‌توانند هر سیستم عاملی را مستقلاً اجراء کند. این امر همچنین باعث می‌شود مراحل تحقیق و توسعه سیستم عاملها راحت تر صورت بگیرد, چرا که دیگر سازندگان سیستم عامل برای تست کردن سیستم عامل تولیدی جدید لازم نیست کل کامپیوتر را در اختیار داشته باشند.

ایده ماشین‌های مجازی امروزه نیز جهت رفع مشکلات عدم سازگاری گسترش زیادی یافته است . به عنوان مثال شرکتهای میکروسیستم یا شرکتDEC که کامپیوترهای غیر intel را می‌سازند مایلند که مشتریهایشان بتوانند برنامه های DOS (تحت intel) را نیز اجراء کنند. برای این کار یک ماشین مجازی اینتل بر روی پردازنده خودپدید می‌آورند.

در این حال ماشین مجازی دستورات اینتل را به دستورات پردازنده جدید تبدیل می‌کند. یا مثلاً کامپیوترpower PC شامل ماشین مجازی Motorola 6800 می‌باشد . مثال دیگر اجراء شدن DOS تحت محیط ویندوز است,پردازنده‌های ۳۸۶ به بعد دارای یک مد مجازی هستند. که می‌توانند چندین برنامه تحت DOS تحت ویندوز نیز اجراء شوند (البته به شرطی که دستورالعملهای عادی را اجراء کنندو مستقیماً با پورتهای مهم سر و کار نداشته باشند.)

مثال دیگر از این مفهوم ماشین مجازی زبان جاوا«Java) می‌باشد. کامپایلر زبان جاوا توسط شرکت sun طراحی شده است یک خروجی بایت کد(byte code) تولید می‌کند. این بایت کدها دستوراتی

هستند که بر روی ماشین مجازی جاوا(JVM) اجراء می‌شوند. جهت اجرای برنامه‌های جاوا در یک ماشین , آن کامپیوتر می‌بایست دارای یک JVM باشد.

امروزه JVM بر روی بسیاری از انواع کامپیوترها (PC,مکینتاش, SUN مینی کامپیوترها و مین فریم ها) وجود دارد . JVM همچنین در Microsoft Explorer ویندوز پیاده سازی شده است . بدنی ترتیب برنامه‌هایی که به زبان java نوشته شده‌اند به راحتی بر روی انواع کامپیوترها اجراء می‌شوند. فقط کافی است بایت کدها را روی آن ماشین کامپایل کرد. بدیهی است به علت نیاز به کامپایل شدن بایت کدها , برنامه های جاوا سرعت کمتری نسبت به برنامه‌هایی نظیر c دارد .

برنامه های cتوسط کامپایلر بومی یک کامپیوتر , برای یک بار تبدیل به زبان ماشین آن کامپیوتر می‌گردد. پس خروجی زبان ماشین کامپایلر c از یک نوع کامپیوتر به کامپیوتر دیگر متفاوت است ولی بایت کدهای خروجی جاوا برای همه ماشین ها یکسان است.

سیستم مشتری –خدمتگزار

سیستم عامل VM با جابجا کردن بخش زیادی از کد سیستم عامل به لایه بالاتر (یعنی CMS) باعث ساده شدن هسته اصلی یعنی مانیتور ماشین مجازی شد. با این همه هنوز هم VM یک برنامه پیچیده می‌باشد.

روند طراحی سیستم عاملهای جدید همواره با این ایده همراه بوده که تا جایی که ممکن است کدها به عامل را در سطح کاربر و مشابه پردازشهای کاربران پیاده سازی می‌کنند.

مثلاً برای خواندن یک بلوک از فایل پروسس کاربر (پروسس مشتری client) یک در خواست به پروسس خدمتگزار (server) ارسال می‌کند و از آن می‌خواهد که کارش را انجام داده و جواب برگرداند. در این مدل تنها کاری که هسته انجام می‌دهد این است که ارتباط بین مشتریها و خدمتگزارها را از طریق پیامها بر قرار می‌سازد (مشابه شکل زیر):

مزایای این مدل عبارتند از :

از آنجا که سیستم عامل به چند بخش تقسیم شده که هر یک فقط یکی از وظایف سیستم عامل را انجام می‌دهند, بنابراین سیستم عامل را می‌توان ساده تر طراحی و پیاده سازی کرد.

به علت اجرای کلیه پروسس های خدمتگزار در مد کاربر (و نه مد هسته) هیچکدام از آنها دسترسی مستقیم به سخت افزار ندارند. لذا اگر اشکالی مثلاُ در خدمتگزار فایل ایجاد شود فقط موجب اختلال در خدمات فایل خواهد شد و به ندرت موجب خراب شدن کل سیستم می‌شود.

فایده دیگر این مدل سازگاری آن برای استفاده در سیستمهای توزیع شده است . از آنجا که یک مشتری به وسیله ارسال پیام‌هایش با یک خدمتگزار ارتباط برقرار می:ند, مشتی نیاز ندارد که بداند آیا به پیغام وی به صورت ملی در ماشین خودش رسیدگی می‌شود و یا اینکه پیغام از طریق یک شبکه به یک ماشین دور ارسال می‌شود

زبان های پیاده سازی سیستم عامل

سیستم عاملهای اولیه به زبان اسمبلی نوشته می‌شدند ولی امروز, اکثر سیستم عاملها به زبان C))(یا C++)نوشته می‌شوند . سیستم عامل ((OS/2, UNIX و ویندوز بیشتر به زبان C نوشته شده‌اند و قسمت اندکی از آنها به زبان اسمبلی است.

مهمترین مزیت استفاده از زبان سطح بالا برای پیاده سازی سیستم عامل قابلیت حمل آن بر روی انواع کامپیوترها و سادگی پیاده سازی , تغییر و بسط دادن سیستم عامل می‌باشد.

ممکن است ادعا شود پیاده سازی سیستم عامل به زبان C باعث کاهش سرعت و افزایش مصرف حافظه می گردد . اگر چه یک برنامه نویس ماهر زبان اسمبلی , می‌تواند برنامه‌های کوچک و بسیار بهینه بنویسد ولی برای برنامه‌های بزرگ یک کامپایلر خوب, می‌تواند تحلیل پیچیده تری نسبت به مغز انسان ماهر انجام داده و بهینه سازی‌های کاملی را انجام دهد.

لذا در عمل برنامه‌های بزرگ C کد اسمبلی بهینه تر و کمتری را تولید می کنند, نسبت به حالتی که برنامه‌نویس بخواهد همان کاری به زبان اسمبلی انجام دهد . از طرف دیگر در عمل کارایی اصلی نتیجه ساختمان داده و الگوریتم‌های بهتر است نه نتیجه نوشتن برنامه به زبان اسمبلی .

همچنین اگر چه سیستم عاملها برنامه‌های بزرگی هستند ولی تنها بخش کوچکی از کد آنها, نسبت به کارایی , بحرانی(Critical) می‌باشد مثل مدیریت حافظه و زمانبندی CPU .

لذا پس از آنکه سیستم عامل به زبان سطح بالا نوشته شد و به درستی عمل کرد می توان روتین های گلوگاه (bottleneck) و مهم را شناسایی کرد و سپس آنها را با روتین‌های معادل زبان اسمبلی جایگزین نمود.

پردازش و زمابندی:

پردازش در سیستم عامل

مهمترین مفهوم در هر سیستم عامل فرآیند یا پردازش (process) است . تمامی نرم افزارهای کامپیوتر از جمله سیستم عامل به تعدادی از پروسس ها سازماندهی و تقسیم بندی می‌شوند.

یک پردازش برنامه‌ای در حال اجراست . در واقع یک پروسس فقط یک برنامه اجرایی است که علاوه بر کد برنامه (یا بخش متن text segment) شامل مقدار شمارنده برنامه, رجیسترهای CPU ,پشته و بخش داده‌ها(Data segment) است . به عبارتی دیگر می‌توان گفت که هر پروسس CPU مجازی خود را دارد . در سیستم چند برنامگی CPU از یک پروسس به پروسسی دیگر سوئیچ می‌کند و هر کدام را به مدت چند ده یا چند صد میلی ثانیه به اجرا در می‌آورد.

باید دقت کرد که یک برنامه به خودی خود یک پردازش نیست .برنامه الگوریتمی است که محتویات یک فایل بر روی دیسک ذخیره شده است . به عبارتی دیگر برنامه یک نهاد غیر فعال (passive) است .

در حالیکه پردازش یک نهاد فعال(active) می‌باشد که در حال اجراست .

مثلاً در یک کامپیوتر کاربران متعددی ممکن است در حال اجرای نسخه‌های متعددی از برنامه ویرایشگر باشند یا مثلاً یک کاربر می‌تواند چند نسخه از برنامه ویرایشگر را همزمان اجراء کند , در این حال هر کدام از آنها یک پردازش جداگانه‌اند و اگر چه بخش متن شان (کدشان)یکسان است ولی بخش داده‌هایشان متفاوت می‌باشد .

در سیستمها روشی مورد نیاز است تا در حین کار بتوان پروسس‌هایی را ایجاد کرد یا از بین برد در UNIXو پروسس‌ها توسط فراخوان سیستمی fork پدید می‌آیند, این فراخوانی یک پردازش فرزند تولید می‌کند که نسخه‌ای دقیقاً یکسان با پروسس پدر خواهد بود.

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

حالات یک پردازش

پردازش برنامه در حال اجراست . ولی از دید سیستم عامل می‌توان گفت پردازش در سیستم عامل یکسری ساختمان داده است .

هر پردازش در سیستم عامل توسط یک ساختمان داده به نام بلوک کنترل پردازش در سیستم عامل یا (process Control Block)PCB نشان داده می‌شود. PCB شامل اطلاعات زیادی در مورد یک پردازش در سیستم عامل است . این اطلاعات مثلاً هنگامیکه پروسس از «حالت اجرا» به حالت «آماده » می‌رود لازم است ذخیره شود که اگر دوباره پروسس خواست به حالت اجرا برگردد از همان نقطه ای که قطع شده‌بود, به درستی ادامه یابد . این اطلاعات عبارتند از :

حالت جاری پردازش در سیستم عامل : که می‌تواند, آماده,اجراء یا بسته باشد.

شمارنده برنامه:(program Conuter=PC) که آدرس دستورالعمل بعدی قابل اجرای پردازش در سیستم عامل را نشان می‌دهد .

محل حفظ ثباتها: هنگام وقوع یا سوئیچ کردن بین پردازشهای پردازش در سیستم عامل جاری می‌بایست در PCB مربوط ذخیره شوند تابعداً دوباره بازیابی شوند .

اطلاعات زمانبندی CPU : مثل اولویت پردازش در سیستم عامل , اشاره‌گرها به صف‌های زمانبندی و غیره

اطلاعات مدیریت حافظه: مثل محل قرار گیری پردازش در سیستم عامل در حافظه و مسائل حفاظتی آن.

اطلاعات وضعیت I/O :شامل لیستی از وسایل I/O تخصیص یافته به پردازش در سیستم عامل , لیست فایلهای باز شده برای پردازش در سیستم عامل و غیره

اطلاعات حسابرسی : مثل میزان زمان CPU مصرف شده برای پردازش در سیستم عامل , شماره حساب, شماره پردازش در سیستم عامل و غیره.

وقتیکه سیستم عامل CPU را به پردازش در سیستم عامل دیگر می‌دهد با استفاده از PCB تمام اطلاعاتی که جهت راه اندازی مجدد پردازش در سیستم عامل قبل لازم دارد را حفظ می‌کند. به این عملیات تعویض متن Context Switch انجام می‌پذیرد .

.تعویض متن بوسیله بخشی از سیستم عامل به نام Dispatcher انجام می‌پذیرد. از آنجا که سیستم عامل خیلی با PCB سرو کار دارد, در بسیاری از کامپیوترها ثباتی سخت افزاری وجود دارد که همیشه PCB پردازش در سیستم عامل در حال اجزا اشاره می‌کند .

دستوراتی نیز وجود دارند که خیلی سریع اطلاعات را در PCB بار می‌کنند . عملیات تعویض متن الزاماً سربار اضافی (overhead)روی کامپیوتر ایجاد کرده و قدر از وقت CPU را جهت این کار به هدر می‌دهد, البته این زمان آنقدر زیاد نیستکه بر مزیت چند برنامگی غلبه کند .

زمان تعویض متن تابع سخت افزار می‌باشد و به طور نمونه‌ای این زمان از ۱تا۱۰۰میکرو ثانیه متغیر است .

بلوک کنترلی پردازش

هدف چند برنامگی این است که در همه اوقات ,پردازشی در حالت اجرا وجود داشته باشد تا بهره وری CPU ما بین پردازش ها به قدر مکرر , سوئیچ نماید که کاربران با برنامه در حال اجرا محاوره داشته باشند.

زمانی که بیش از یک پروسس قابل اجرا باشد سیستم عامل باید تصمیم بگیرد که کدامیک اول اجرا شود. بخشی از سیستم عامل که این تصمیم گیری را انجام می دهد زمانبندی (Scheduler) نامیده می‌شود. پردازش‌هایی که در حافظه اصلی قرار دارند و منتظر اجرا شدن هستند در صفی به نام صف آماده (ready queue) قرار می‌گیرند.

این صف معمولا به شکل یک لیست پیوندی (linked list) پیاده سازی می‌شود. سرایند صف (header queue) شامل اشاره‌گرهایی به اولین و آخرین PCB های لیست می‌باشد:

البته در سیستم صفهای دیگری نیز وجود دارند, مثل صف وسیله(I/O queue) که مشخص می‌سازد هر وسیله توسط چه پردازشهایی مورد نیاز است . هر وسیله صف مخصوص به خود را دارد.

پردازش در حال اجرا بنا به دلایل زیر می‌تواند به صف آماده برود تا زمانبندی مجدد شود:

پردازش می‌تواند یک درخواست I/O را صادر نماید و سپس دریک صف I/O منتظر بماند تا به آن سرویس داده شود.

پردازش می‌تواند یک پردازش جدید (فرزند) ایجاد نموده و برای اتمام آن صبر کند.

پردازش به علت تمام شدن برش زمانی (time slice) از CPU جدا می‌شود تا این امکان به بقیه پردازشها نیز داده شود که از CPU استفاده کنند.

زمان بندی در سیستم عامل

هدف چند برنامگی این است که در همه اوقات ,پردازشی در حالت اجرا وجود داشته باشد تا بهره وری CPU ما بین پردازش ها به قدر مکرر , سوئیچ نماید که کاربران با برنامه در حال اجرا محاوره داشته باشند.

زمانی که بیش از یک پروسس قابل اجرا باشد سیستم عامل باید تصمیم بگیرد که کدامیک اول اجرا شود. بخشی از سیستم عامل که این تصمیم گیری را انجام می دهد زمانبندی (Scheduler) نامیده می‌شود. پردازش‌هایی که در حافظه اصلی قرار دارند و منتظر اجرا شدن هستند در صفی به نام صف آماده (ready queue) قرار می‌گیرند.

این صف معمولا به شکل یک لیست پیوندی (linked list) پیاده سازی می‌شود. سرایند صف (header queue) شامل اشاره‌گرهایی به اولین و آخرین PCB های لیست می‌باشد:

البته در سیستم صفهای دیگری نیز وجود دارند, مثل صف وسیله(I/O queue) که مشخص می‌سازد هر وسیله توسط چه پردازشهایی مورد نیاز است . هر وسیله صف مخصوص به خود را دارد.

پردازش در حال اجرا بنا به دلایل زیر می‌تواند به صف آماده برود تا زمانبندی مجدد شود:

پردازش می‌تواند یک درخواست I/O را صادر نماید و سپس دریک صف I/O منتظر بماند تا به آن سرویس داده شود.

پردازش می‌تواند یک پردازش جدید (فرزند) ایجاد نموده و برای اتمام آن صبر کند.

پردازش به علت تمام شدن برش زمانی (time slice) از CPU جدا می‌شود تا این امکان به بقیه پردازشها نیز داده شود که از CPU استفاده کنند.

انواع زمانبند ها در سیستم عامل

از یک جنبه زمانبندهای پردازش در سیستم عامل به سه دسته الف- دراز مدت (Long term scheduler) ب– کوتاه مدت(Short term scheduler) ج – میان مدت, تقسیم بندی می‌شوند.

در یک سیستم دسته‌ای پردازشهای بیشتری نسبت به آنچه فوراً می‌توانند اجرا شوند تحویل داده می‌شوند . این پردازشها در دیسک نگهداری می‌شوند .زمانبندی دراز مدت (یازمانبندی کار sheduler Job)پروسسهایی را انتخاب کرده و آنها را برای اجرا از دیسک به حافظه اصلی می‌آورد.

زمانبند کوتاه مدت (یا زمانبند CPU) از بین پروسسهای موجود در حافظه اصلی که آماده اجرا هستند یک را انتخاب کرده و CPU را به آن اختصاص می‌دهد. غالبا زمانبند کوتاه مدت هر صد میلی ثانیه یک بار اجراء می‌شود ولی زمانبند دراز مدت ممکن است هر چند دقیقه یک بار اجرا شود. در واقع زمانبند دراز مدت در جه چند برنامگی (degree of multiprogramming) یعنی تعداد پردازشهای موجود در حافظه را کنترل می کند .

زمانبند دراز مدت وقت زایدی برای تصمیم گیری دارد ولی زمانبند کوتاه مدت می‌بایست خیلی سریع تصمیمی گیری کند. زمانبند دراز مدت می‌بایست مخلوط مناسبی از پردازشهای CPU-limiter و I/O limited را جهت قرار گیری در حافظه انتخاب کند تا کارایی CPU و وسایل I/O بهینه شود. در بعضی سیستمها مثل اغلب سیستم های اشتراک زمانی زمانبند دراز مدت وجود ندارد, چرا که هر پردازش در سیستم عامل جدید جهت زمانبند CPU در حافظه گذاشته می‌شود تا زمان پاسخ دهی به برنامه مناسب باشد.

البته بعضی سیستم عاملها از زمانبند میان مدت نیز استفاده می کنند. بدین ترتیب که گاهی پروسس هایی از حافظه و در واقع از رقابت جهت دریافت CPU حذف شده و به دیسک برده می‌شوند (swap Out) .بدین ترتیب درجه چند برنامگی کاهش می‌یابد . سپس در زمانی دیگر پردازش در سیستم عامل مذکور مجددا به حافظه آورده شده (swap in) و اجرایش از همان نقطه قبلی ادامه می‌یابد, این عملیات به نام مبادله (swapping)معروف است .

زمانبندی CPU به طوری کلی می تواند انحصاری (غیر قابل پس گرفتن non preemptive) یا غیر انحصاری (قابل پس گرفتن preemptive) باشد.

در سیستم انحصاری فقط هنگامی CPU ازپردازش در حال اجراء گرفته می‌شود که جهت عملیات I/O یا اتمام پردازش در سیستم عامل فرزند را رخداد دیگری بلوکه شود. بنابراین مفهوم و پیاده سازی الگوریتم زمانبندی انحصاری ساده است .ولی ممکن است پردازشی برای مدت طولانی CPU را جهت محاسبات در اختیار بگیرد.

رد این حال پردازشهای دیگر برای مدتی طولانی انتظار خواهند کشید و این موضوع مخصوصاً برای سیستم‌های اشتراک زمانی نامناسب است .لذا در اغلب سیستمها از یک زمان سنج(Timer) داخلی برای ایجاد وقفه‌های متناوب سخت افزاری جهت گرفتن CPUاستفاده می‌شود.

در هر وقفه در سیستم عامل ساعت, سیستم عامل اجرا می‌شود تا تصمیم بگیرد که آیا به پروسس در حال اجرا اجازه ادامه کار را بدهد یا اینکه چون پروسس به اندازه کافی از زمان CPU استفاده کرده آن را معلق نماید تا CPU به پروسس دیگری تخصیص داده شود. فرکانس این وقفه در سیستم عامل‌های ساعت معمولا بین ۵۰تا۶۰ بار در ثانیه است . این نوع زمانبندی که در آن پس از تمام شدن برش زمانی معین , CPU از گرفته می‌شود زمانبندی غیر انحصاری نام دارد.

معیار های زمانبندی در سیستم عامل

عدالت (fairness) یعنی اطمینان از اینکه هر پروسس سهم عادلانه و منصفانه‌ای از CPU را دریافت کند.

کارایی یا بهره وری (CPU utilization- Efficiency) یعنی اینکه CPU در تمام زمانها (حتی الامکان) مشغول باشد

زمان پاسخ (Response Time) یعنی به حداقل رساندن زمان پاسخ برای فرمانهای محاوره‌ای کاربر. این زمان معمولاً با سرعت ابزار خروجی محدود می‌شود.

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

زمان بارگذاری در حافظه +زمان عملیات I/O +زمان اجراء+ زمان انتظاردر صف آماده = زمان گردش کار

توان عملیاتی یا گذردهی (throughput) به تعداد پردازشهایی که در واحد زمان تکمیل می‌شوند توان عملیاتی می‌گویند. الگوریتم زمانبندی باید به گونه‌ای باشد که این معیار را افزایش دهد .

زمان انتظار (waiting time) الگوریتم زمانبندی CPU, بر میزان زمان اجرای پردازش یا اعمال I/O اثر نمی‌کند, بلکه فقط در زمان صرف شده جهت انتظار در صف آماده اثر می‌گذارد. زمان انتظار , مجموع پریودهای زمانی صرف شده در صف آماده می‌باشد.

انواع زمانبندی ها:

اولویتها می‌توانند بصورت اتوماتیک توسط سیستم نسبت داده شوند و یا از خارج سیستم تعیین گردند, مثلاً ممکن است یک کاربر کار فوری داشته باشدو حاضر باشد به خاطر بدست آوردن سرویس بالاتر هزینه بیشتری بپردازد , یعنی اولویت را بخرد . یک اولویت ممکن است استاتیک باشد یا دینامیک . اولویت استاتیک تغییر نمی‌کندو بنابراین پیاده سازی آن ساده است .

ولی این نوع اولویت در مقابل تغییرات محیطی عکس العملی نشان نمی‌دهد . برعکس اولویت دینامیک بر اثر تغییرات محیطی تغییر می‌کند مثلاً ممکن است در آغاز یک برنامه اولویت پائینی داشته باشد ولی به تدریج اولویت آن بهبود یابد.

اول آمده–اول سرویس شده

ساده ترین الگوریتم زمانبندی CPU ,الگوریتم اول آمده, اول سرویس شده (first come-first served =FCFS) می‌باشد . گاهی اوقات به این روش (FIFO(first In First Out نیز می‌گویند. در این روش هر پردازش در سیستم عاملی که اولین در خواست CPU را صادر کند , اولین پروسسی خواهد بود که آن را به دست می‌آورد .

این روش از نوع انحصاری (non- preemptive) است که به سادگی توسط یک صف FIFO پیاده سازی می‌شود.

هنگامی که پردازش در سیستم عامل CPU را به دست گرفت آن را رها نمی‌کند مگر اینکه تمام شود یا جهت انجام عملیات I/O به حالت بسته برود.

زمانبندی نوبت گردشی

این زمانبندی یکی از قدیمیم ترین , ساده ترین , عادلانه ترین و رایجترین الگوریتم های زمانبندی است و از نوع غیر انحصاری (preemptive) می‌باشد. این الگوریتم شبیه FCFS است ولی به هر پردازش حداکثر به میزان زمانی مشخصی CPU داده می‌شود.

به عبارتی دیگر یک واحد کوچک زمانی به نام کوانتوم زمانی (time quantum) با برش زمانی (time slice) تعریف می‌شود که معمولاً بین ۱۰ تا ۱۰۰میلی ثانیه است و هر پروسس حداکثر به این میزان می‌تواند CPU را در اختیار بگیرد. هنگامی که پردازشی CPU را در اختیار دارد دوحالت ممکن است رخ دهد .

یا انفجار محاسباتی جاری کمتر از یک کوانتوم زمانی است که در این حالت پردازش داوطلبانه CPU را رها می‌کند و منتظر اتمام عملیات I/O می‌شود (مانند FCFS) و یا اینکه انفجار محاسباتی بیشتر از یک کوانتوم زمانی است که در این حالت تایمر یک وقفه به سیستم عامل می‌دهد و سیستم عامل با تعویض متن (Context switch) CPU را از پردازش جاری گرفته و آن را به ته صفآماده می‌فرستد, سپس از ابتدای صف آماده, پردازش دیگری را جهت اجرا انتخاب می‌کند :

از این روش در سیستمهای اشتراک زمانی استفاده شده تا زمانهای پاسخ برای کاربران محاوره‌ای بصورت مناسب گارانتی شود.

حد بالای کوانتوم زمانی بایدبه قدری باشد که زمان پاسخ دهی مناسبی داشته باشیم.

حد پایین برش زمانی توسط دو عامل تعیین می‌شود یکی اینکه باید این برش خیلی بزرگتر از زمان تعویض متن باشد مثلاً هزاران برابر.

دیگر آنکه مقدار برش زمانی بایستی کمی بزرگتر از زمان 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 می‌شود.

یک قاعده سرانگشتی این است که go درصد انفجارهای محاسباتی باید کوتاه‌تر از کوانتوم زمانی باشند و در عمل برا یاین امر برش زمانی را حدود ۱۰۰ میلی ثانیه در نظر می‌گیرند.

اول کوتاه ترین زمان

در الگوریتم (Shortest Job First) که روشی انحصاری است CPU به پردازش داده می‌شود که کوچکترین انفجار محاسباتی بعدی را دارد.

البته اصطلاح مناسبتر , «کوتاهترین انفجار محاسباتی بعدی»می‌باشد. زیرا این زمانبندی بر اساس طول مدت انفجار CPU بعدی عمل می کند و نه بر اساس طول کل پردازش در سیستم عامل . اگر دو پردازش در سیستم عامل مدت انفجار محاسباتی یکسانی داشته باشد براساس FCFS زمانبندی می‌شوند. این الگوریتم می تواند انحصاری و غیر انحصاری باشد.

این الگوریتم مخصوصاً برای کارهای دسته‌ای که از قبل زمان اجرای آن کارها , مشخص و معین باشد به کار می‌رود .

مهمترین مشکل در SJF آگاهی از طول درخواست بعدی CPU می‌باشد. هیچ راهی که طول انفجار محاسباتی بعدی را برای ما مشخص سازد وجود ندارد.

لذا در صورت لزوم مجبوریم آن را پیش بینی کنیم . یعنی انتظار داشته باشیم که طول انفجار بعدی خیلی شبیه طول انفجارهای قبلی باشد.

کوتاه ترین زمان باقی مانده

بالا ترین نسبت پاسخ

دادن اولویت به پردازش

صفهای چند گانه MQ

صفهای چند گانه با فید بک

بلا درنگReal time

در سیستم بلادرنگ سخت , پردازش در سیستم عامل ها می‌بایست در یک زمان تخمین شده اجراء و اتمام شوند., مانند سیستم کنترل موشک . چنین تضمینی در یک سیستم با حافظه ثانویه یا حافظه مجازی غیر ممکن است . در سیستم بلادرنگ نرم (مانند پخش موسیقی) زمان پاسخگویی به پردازش در سیستم عامل مهم است ولی مانند بلادرنگ سخت , حیاتی نیست .

اتفاقاتی که سیستم بلادرنگ باید به آنها پاسخ دهد به دو دسته متناوب و غیر متناوب تقسیم می‌شوند. وقایع متناوب در فواصل زمانی مساوی اتفاق می‌افتند ولی وقایع متناوب به صورت تصادفی و تصادفی بوده و غیر قابل پیش بینی می‌باشند.

روشهای زمانبندی بلادرنگ به دو دسته کلی پویا و ایستا تقسیم می‌شوند. در حالت ایستا قبل از شروع سیستم , تصمیمات زمانبندی گرفته می‌شود ولی در حالت پویا تصمیمات زمانبندی در زمان اجرای سیستم انجام می‌پذیرد . سه روش زمانبندی بلا درنگ پویا عبارتند از:

الگوریتم نرخ یکنواخت (Rate monotonic) : در این الگوریتم به هر پردازش در سیستم عامل اولویتی متناسب با فرکانس رخداد آن واقعه نسبت داده می‌شود. مثلاً به پردازشی که هر ۲۰ میلی ثانیه تکرار می‌شود, اولویت ۵۰ و به پردازشی که هر ۱۰۰ میلی ثانیه تکرار می‌شود, اولیت ۱۰ داده می‌شود. این الگوریتم از نوع غیرانحصاری است . می‌توان اثبات کرد که این الگوریتم بهینه است.

الگوریتم ابتدا زودترین مهلت (Earliest deadline first) در این الگوریتم پردازش در سیستم عاملی ابتدا اجراء می‌شود که فرصتش از همه کمتر است یعنی نزدیکترین مهلت را دارد . این مهلت برای وقایع متناوب برابر زمان رخداد واقعه بعدی می‌باشد.

الگوریتم کمترین سستی (least laxity) زمان سستی یک پردازش در سیستم عامل زمانی است که می‌تواند آماده باقی مانده و اجراء نشود. مثلاً اگر یک پردازش در سیستم عامل به ۲۰۰ میلی ثانیه وقت CPU احتیاج داشته باشد. و ۲۵۰ میلی ثانیه نیز مهلت داشته باشد که کارش را تمام کند, زمان سستی او برابر ۲۵۰-۲۰۰=۵۰ میلی ثانیه می‌باشد. در این الگوریتم پردازشی ابتدا اجراء می‌گردد که کوچکترین زمان سستی را دارد.

انواع سیستم عامل ها:

Windows

Linux

Unix

DOS

OS/2

Solaris