موسسه آموزش عالی آزاد

پارسه

مفاهیم سیستم عامل

خلاصه کتاب دکتر فهیمی

تهیه و گردآوری

مهندس سلامی - مهندس اکبرپور

« مبحث : کارآیی سیستم های ورودی و خروجی »

یکی از قابلیت های هر سیستم کامپیوتر توانایی ارتباط برقرار کردن آن سیستم با استفاده کننده اش ، توسط دستگاه های ورودی و خروجی است. کارآیی کلی سیستم کامپیوتر به مقدار زیاد بستگی به عملکرد دستگاه های جنبی ورودی/خروجی دارد. بطور کلی یک سیستم کامپیوتر باید بتواند انواع گوناگون کارها را اجرا کند و هدف بایستی این باشد که از تمامی منابع سیستم به بهترین نحو در هر شرایطی استفاده شود. بنابراین ، به تکنیکهای موثر جهت کنترل عملیات دستگاه های جنبی نیاز است.

« اصول ساده چند برنامگی »

در ساده ترین روش، ممکن است بخواهیم حالتی را ایجاد کنیم که فقط یک برنامه ***CPU-limited*** با یک برنامه ***I/O-limited*** چند برنامه ایی شده و هر دو با هم در حافظه کامپیوتر نگاه داشته شوند. حال برنامه ***I/O-limited*** آنقدر اجرا می شود تا لازم گردد برای تکمیل یک انتقال اطلاعات ***(data transfer)*** صبر کند. در این لحظه برنامه ***CPU-limited*** به اجرا در می آید تا از زمان اضافی ***CPU*** استفاده کند. بمحض پایان یافتن انتقال اطلاعات از دستگاه جنبی ، برنامه ***I/O-limited*** به جای این برنامه به اجرا در می آید. ترتیب و زمانبندی اجرای این برنامه در شکل زیر نشان داده شده است.

سلسله عملیات در چندبرنامگی

تکنیک چند برنامگی، که برای بهبود کارآیی مورد استفاده قرار می گیرد از یک نقطه ضعف برخوردار است و آن این است که کارهای از نوع ***CPU-limited*** هم به هرحال احتیاج به ورود و خروج اطلاعات دارند. و این مورد واضح است که یک دستگاه جنبی نمی تواند توسط بیش از یک برنامه در ماشین بکار آید زیرا در غیر این صورت اطلاعات ورودی یا خروجی کارها با هم مخلوط می شوند. بنابراین برای اینکه این تکنیک موثر عمل کند لازم است تعداد بیشتری دستگاه های جنبی بکار برد.

« استفاده از وقفه ها »

در سیستمی که از تکنیک چندبرنامگی استفاده می کند، مشکل ترین مسئله در هنگام کنترل دستگاه های جنبی این است که تعیین شود در چه هنگامی انتقال اطلاعات بین دستگاه های جنبی و کامپیوتر کامل شده است. در ساده ترین تکنیک که نمونه برداری ***(Polling)*** نام دارد، ثبات های کنترلی ***(control register)*** مربوط به دستگاه ، مرتباً امتحان می شوند ، ولی این مورد می تواند باعث اتلاف وقت ***CPU*** گردد. حال مکانیزم وقفه جهت غلبه بر این مشکل طراحی شده

است. در این حالت تمام دستگاه های جنبی دارای یک سیگنال کنترل مخصوص هستند که به *CPU* راه دارد. هنگامی که یک دستگاه انتقالی را تمام می کند، مثلاً زمانی که یک دستگاه ورودی کاراکتری جهت دستیابی توسط برنامه دارد و یا دستگاه خروجی چاپ یک کاراکتر را تمام کرده است ، در چنین زمانی دستگاه جنبی سیگنال کنترل را صادر می کند. و بدین صورت دستگاه جنبی به *CPU* اعلام می کند که انتقال کامل شده است.

برای اینکه چندبرنامگی کارهای از نوع *I/O-limited* و *CPU* موثر باشد وجود یک مکانیزم وقفه ضروری است. به این ترتیب کارهایی که از نوع *I/O-limited* است دستگاهای جنبی را کنترل می کند.

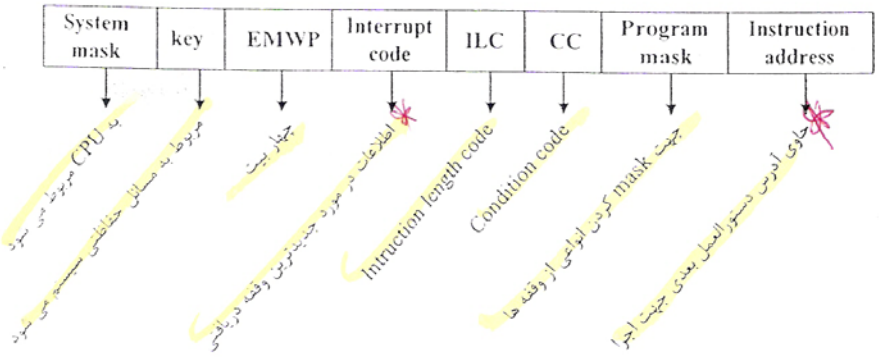
هنگامی که دستگاه در حال انتقال است این کار متوقف می شود و وقوع وقفه علامت یا سیگنالی است که باعث می شود با پایان یافتن انتقال ، کنترل دوباره از کار *CPU-limited* به کار *I/O-limited* برگردد.

« حافظه های میانگیر »

برای یکنواخت کردن اختلاف بین عرضه و تقاضا. عبارتست از یک حافظه میانگیر و یا بافر یعنی اینکه یک خط از اطلاعات ورودی ممکن است در حافظه انباشته شود پیش از آنکه مورد نیاز واقع شده باشد. هنگامیکه برنامه به این خط احتیاج دارد، می تواند آن را با سرعتی متناسب با سرعت حافظه و نه متناسب با سرعت دستگاه جنبی پردازش کند.

هنگامیکه وقفه رخ می دهد ابتدا سیستم عامل وضعیت کامل برنامه در حال اجرا را حفظ می کند، سپس سیستم عامل، وقفه را بررسی می کند و کنترل را به یک روال وقفه گیر *(Interrupt Handler)* مناسب تحویل می دهد و تمام این موارد به سرعت انجام می شود. بهمین جهت رویتینهای مربوط به وقفه ها معمولاً به زبان ماشین و بطور بهینه نوشته می شوند.

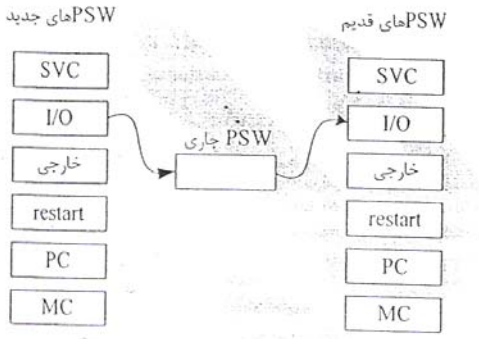
پیش از آنکه سیستم عامل کنترل را به یک روال وقفه گیر بخصوص رد کند وضعیت پردازش جاری را در محلی حفظ می کند تا بتواند بعداً آنرا ادامه دهد و سپس بطرف روال وقفه گیر می رود. به این جریان، تعویض متن *(context switch)* می گویند و رکن اصلی تعویض متن ثباتهای داخلی *program status word* یا *PSW* هستند. که ترتیب اجرای دستورالعمل ها را کنترل می کنند و حاوی اطلاعات متنوعی مطابق شکل زیر هستند.



« ثبات *PSW* »

در ماشین های *IBM* ، سه نوع *PSW* وجود دارند : جاری ، قدیمی و جدید.

PSW های جدید حاوی آدرس روالهای وقفه گیر هستند و به اصطلاح بردار وقفه *(Interrupt vector)* را می سازند ، *PSW* جاری همیشه آدرس دستورالعمل بعدی جهت اجرا را می دهد زمانی که وقفه رخ می دهد محتوای *PSW* جاری در *PSW* قدیم مخصوص وقفه ای که رخ داده است حفظ می شود و *PSW* جاری با محتوای جدیدش پر می شود.

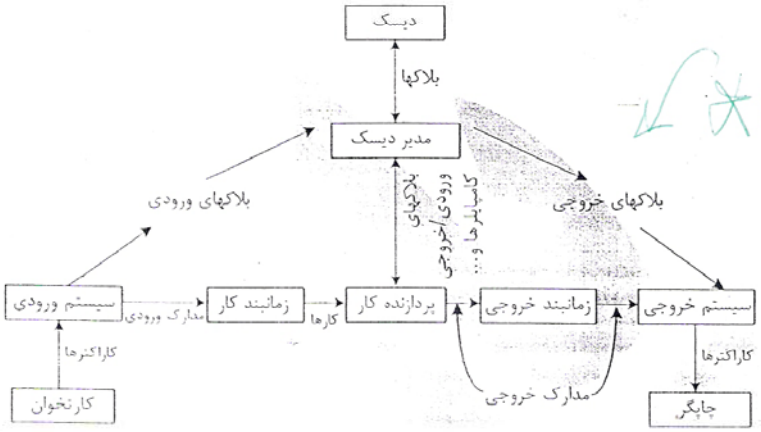


تعویض PSW

« مبحث : سیستم های *Spooling* »

« طراحی یک سیستم ساده *Spooling* »

در داخل این سیستم عامل یک سری وظایف ضروری را می توان تشخیص داد بنابراین اجراء این سیستم مطابق شکل زیر می باشد:



سیستم ورودی : کار این سیستم با عملیات دستگاه کارتخوان همگام*(synchronized)* شده است کاراکترهایی که توسط این دستگاه خوانده می شوند در بلاک هایی جمع آوری می گردند و هنگامیکه بلاک ها پر شدند. آنگاه مدیر دیسک آنها را روی دیسک می نویسد. در انتهای هر مدرک ورودی اطلاعاتی در مورد آن مدرک نظیر محل آن بر روی دیسک ، اولویت ، اسم کار و اسم استفاده کننده به قسمت زمانبند کار می دهد.

زمانبند کار : این زمانبند یک لیست از کارهای موجود در ماشین و اطلاعات لازم در مورد مدارک ورودی مورد نیاز هریک را نگاه می دارد .

پردازنده کار : زمانبندکار به پردازنده کار می گوید که کدام کار بعدی را اجرا کند برای این منظور لازم است اطلاعاتی در مورد محل کار و مدارک ورودی آن بر روی دیسک به پردازنده کار داده شود و در ضمن این پردازنده از محل کامپایلر و سایر نرم افزار سیستم بر روی دسک مطلع است.

زمانبند خروجی : این بخش لیستی از مدارکی که باید چاپ شوند را نگه می دارد. هنگامی دستگاه چاپ آزاد می شود، زمانبند خروجی مدرک بعدی را برای چاپ انتخاب می کند و به سیستم خروجی ، محل مدرک بر روی دیسک را می گوید.

سیستم خروجی : بلاک کاراکترهای خروجی را از روی دیسک باز می یابد *(Retrieve)* و دستگاه چاپگر را مطلع می کند.

مدیر دیسک : ارتباط بین مدیر دیسک و پردازش های دیگر در سیستم در سطح نسبتاً ساده ای است وظایف اصلی مدیر دیسک عبارتند از :

۱. خواندن یک بلاک از دیسک
۲. نوشتن یک بلاک بر روی دیسک
۳. تخصیص یک بلاک خالی بر روی دیسک
۴. برگرداندن یک بلاک به مجموعه فضای آزاد بر روی دیسک

هماهنگ کننده (Coordinator)

یک جزء از این سیستم در شکل ترسیم نشده است که هماهنگ کننده است و مسئول زمانبندی پردازش های سیستم و فراهم آوردن عملیاتی که جهت همگام کردن بکار می آیند می باشند. این عملیات توسط دو روال انجام می شود:

۱. *wait* که پردازش جاری را متوقف می کند و دوباره وارد زمانبند می شود.
۲. *free* که یک پردازش بخصوص را جهت زمانبندی شدن آماده می سازد .

« مبحث : سیستم های اشتراک زمانی »

در سیستم های *Spooling* توانایی فعل و انفعال *(interact)* با برنامه وجود ندارند ، یک نوع مرسوم از سیستم عامل همان سیستم دسترسی چندتایی *(Multi Access)* و یا اشتراک زمانی است که در آن ظاهراً تعداد زیادی استفاده کننده می توانند بصورت همزمان از یک کامپیوتر بهره برند.

مهمترین خاصیت یک سیستم دسترسی چندتایی این است که در آن کامپیوتر نسبت به محرک هایی از دستگاه هایی که به ماشین متصل هستند عکس العمل نشان می دهد. این دستگاه ها ممکن است ترمینال ها و محرک ممکن است توسط استفاده کننده ای که روی صفحه کلید تایپ می کند تولید شود.

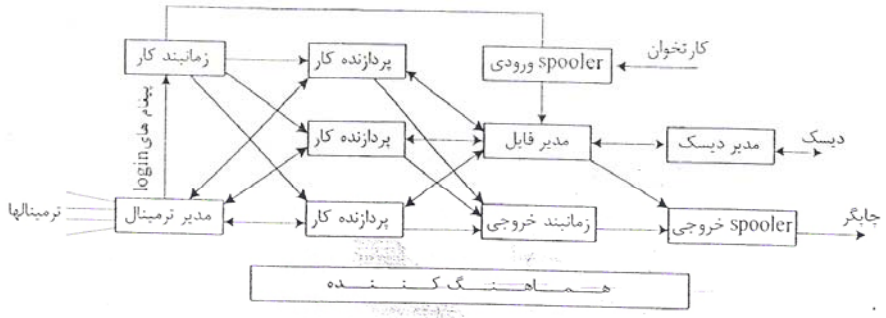
می توان سیستم های دسترسی چندتایی را بر طبق کاربرد آنها تقسیم بندی کرد و بدین طریق سه نوع سیستم به ترتیب زیر حاصل می شود:

۱. سیستم های بلادرنگ *(Real Time)*
۲. سیتم های خاص – منظور پردازش تراکنش ها
۳. سیستم های همه – منظوره *(General purpose)* و اشتراک زمانی

- ***سیستم های بلادرنگ :*** مهمترین کاربرد سیستم های بلادرنگ در رابطه با عملیات کنترل پردازش است. خاصیت مهم سیستم های بلادرنگ این است که هر فعل و انفعال با کامپیوتر بایستی یک پاسخ در مدت زمانی که از قبل تعیین شده است دریافت دارد . سیستم بایستی بتواند این زمان پاسخ را گارانتی کند.
- ***سیستم های خاص – منظور پردازش تراکنشها*** : سیستمی که در رزرواسیون شرکت هواپیمایی بکار می آید نمونه ای از این نوع سیستم است. در اینجا استفاده کنندگان با بسته های نرم افزاریی که از قبل نوشته شده اند تماس برقرار می کنند تا برای مثال معاملاتی را در سیستم وارد کنند و یا اینکه از بانک اطلاعات سوالاتی بکنند، باز هم ، سیستم باید یک زمان پاسخ منطقی را گارانتی کند اما مسئله زمان پاسخ در این سیستم ها به جادی سیستم های بلادرنگ نمی باشد.
- ***سیستم های همه – منظوره و اشتراک زمانی*** : در این سیستم ها کاربران می توانند بصورت مستقیم برنامه خود را بسازند و با آن ارتباط داشته باشند ، میزان وسایلی که در چنین سیستم هایی فراهم می آید ممکن است بین سیستم های مختلف تفاوت داشته باشد ولی در تمام موارد این نوع سیستم ها برای استفادکنندگان آنها به مراتب راحت تر از سیستم های ساده *Spooling* هستند .

« طراحی یک سیستم اشتراک زمانی »

طراحی یک سیستم اشتراک زمانی همه منظوره تا حد زیادی بر اساس سیستم ساده *Spooling* می باشد . با اینکه در سیستم اشتراک زمانی از ترمینال ها برای برقرار کردن ارتباط بین استفاده کننده و کامپیوتر استفاده می شود ولی هنوز ممکن است امکاناتی جهت چاپ کردن مدارک بر روی دستگاه چاپ و یا وارد کردن یک فایل و یا ورود یک کار توسط کارتخوان فراهم شوند. در نتیجه اجزاء *spooler* ورودی و *spooler* خروجی در طرح سیستم اشتراک زمانی هم وجود خواهند داشت. طرح این سیستم مطابق شکل زیر است :



طرح یک سیستم اشتراک زمانی ساده

ویژگی اصلی این سیستم آن است که باید بتواند تعدادی کار استفاده کنندگان را بصورت همزمان اجرا کند البته به صورت ظاهری ، باید به همان صورت که استفاده کنندگان تقاضای فعل و انفعال با کار خود را می نمایند، ***CPU*** از یک پردازنده کار به دیگری منتقل شود. هرچند کپی های متعددی از این پردازنده کار وجود دارد و هر یک دارای فضای داده ای (***Data space***) مربوط به خود ، پشته (***stack***) و کپی هایی از ثبات ها می باشند ولی در عمل کدهایی که در درون این پردازنده ها هستند همگی یکسان می باشند. تفاوت دیگری که بین این نوع سیستم ها و ***Spooling*** وجود دارد این است که در این حالت امکاناتی جهت حفظ فایل ها و راندمان ترمینال های ورودی/خروجی فراهم می آیند. بدین صورت وجود یک سیستم فایل ضروری است بدلیل آنکه نمی توان در هر بار اجرای کار مدرک بزرگی توسط ترمینال ها وارد کامپیوتر کرد. مدیر دیسک وظیفه توزیع فضای خالی روی دیسک و انجام انتقالات را بر عهده دارد ولی مدیر فایل وظیفه مدیریت فهرست های راهنمای فایل ها و امنیت فایل ها را عهده دار است.

وظایف مدیر ترمینال با آن مورد که ***Spooler*** ها در هنگام کنترل سایر دستگاه های ورودی/خروجی انجام می دهند متفاوت است، برای مثال یک ***Spooler*** ورودی ، یک کار کامل را بر روی دیسک تشکیل می دهد و سپس اطلاعات مربوط به آنرا به زمانبند کار ارسال می کند. حال مدیر ترمینال تنها یک کاراکتر و یا یک خط از اطلاعات ورودی را در حافظه اصلی بافر می کند. پس هنگامی که استفاده کننده اولین خط را تایپ کرد، این خط از طرف مدیر ترمینال به زمانبندکار فرستاده می شود. معمولاً از خط اول جهت معرفی کردن استفاده کننده و دادن کلمه رمز و سایر اطلاعاتی که برای تولید صورتحساب ***charging*** لازم است استفاده می شود که به این کار ***LOGIN*** می گویند. زمانبندکار یک پردازنده کار را به این ترمینال بخصوص منسوب می کند و مدیر ترمینال را هم از این عمل مطلع می گرداند. تمام ورود و خروج های بعدی مستقیماً بین مدیر ترمینال و پردازنده کار صورت می گیرد.

از موارد ذکر شده می توانیم به طور خلاصه تعاریف زیر را در نظر بگیریم :

۱. **سیستم های تک برنامگی** : در این حالت یک پردازنده داریم و فقط قادر به اجرای تنها یک برنامه استفاده کننده تا اتمام آن است

۲. **سیستم های چند برنامگی** : در این حالت فقط یک پردازنده وجود دارد و به کمک مکانیزم وقفه بین کارهای *CPU-limited* و *I/O-limited* سوئیچ می شود و به ظاهر اجرای این برنامه ها بطور موازی و همزمان صورت می گیرد.

۳. **سیستم های اشتراک زمانی** : در این حالت فقط یک پردازنده وجود دارد که توسط مکانیزم های زمانبندی مثل برش زمانی بین برنامه های مختلف کاربران سوئیچ می گردد و در هر لحظه به ظاهر چند برنامه استفاده کننده در حال اجرا هستند.

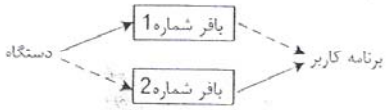
۴. **سیستم های چند تکلیفی** : این سیستم ها ، همان سیستم های اشتراک زمانی هستند که در آنها سیستم عامل به برنامه کاربر اجازه می دهد برنامه های دیگری بسازد و به هر کدام اجزای قسمتی از یک برنامه را بدهد و به این ترتیب به ظاهر یک برنامه در چند بخش اما بصورت موازی در حال اجراست .

۵. **سیستم های پردازش چندتایی** : در این حالت چند پردازنده وجود دارد. تقسیم برنامه به چند پردازش مجزا و اجرای هر پردازش توسط یک پردازنده ما را به پردازش موازی واقعی نزدیک می کند.

« مبحث : تکنیک های بافر کردن »

« سیستم بافر کننده دوبل »

بعد از آنکه دستگاه یک بافر را پر نمود می توان به برنامه استفاده کننده اجازه داد که اطلاعات را از آن بافر بخواند . در همین حال یک بافر دیگر توسط دستگاه در حال پر شدن است که این حالت مطابق شکل زیر است:



در این حالت از بافرها یکی در میان استفاده می شود یعنی وقتیکه *Buffer 1* پر می شود *Buffer 2* در حال خالی شدن است و برعکس. قبل از استفاده از این تکنیک بافر، باید به موارد زیر توجه نمود :

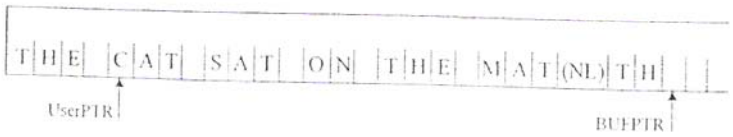
۱) در یک سیستم ساده ، سرعت پردازش اطلاعات با سرعت آهسته ترین دستگاه، همگام می شود و بنابراین معمولاً بهتر است که از بافر دوبل فقط در رابطه با آهسته ترین دستگاه استفاده شود.

۲) اگر کاری که دستگاه را کنترل می کند *CPU-limited* باشد، دیگر ارزشی ندارد که از روش بافر دوبل استفاده شود.

۳) اگر توقف یک دستگاه ایجاد اشکال می کند و یا گران تمام می شود، در اینصورت باید همیشه از بافر دوبل استفاده شود.

«سیستم بافر چرخه ای »

تمام روش های بافر سعی می کنند اطلاعاتی که برنامه استفاده کننده نیاز دارد مثلاً به اندازه یک خط را در یک جا جمع کنند مشکل این است که طول یک خط به میزان زیادی تغییر می کند و اگر بافر را به اندازه طولانی ترین خط ممکن در نظر بگیریم در بیشتر مواقع مقدار زیادی از فضای حافظه تلف می شود. یک راه حل این است که تنها از یک بافر نسبتاً بزرگ که می تواند یک یا چند خط را در خود نگه دارد استفاده شود، بطور کلی پردازش بافر کردن ، کاراکترها را در یک سر بافر قرار می دهد در حالتی که پردازش استفاده کننده ، کاراکترها را از سر دیگر بافر استخراج می کند. در عمل پردازش استفاده کننده بطور مداوم پردازش بافر کننده را بطرف انتهای باز تعقیب می کند. این موارد در شکل زیر موجود است که در این حالت منظور از *NL* همان *New Line* و یا علامتی می باشد که در آخر هر خط می باشد.



زمانی که هر یک از دو نشانگر به انتهای بافر می رسند ، دوباره به ابتدای بافر بازگردانده می شوند. به طور کلی در اینجا یک مشکل جلو زدن یک نشانگر از دیگری موجود است ، این موارد ممکن است در شرایط زیر ایجاد شود :

۱. ورود اطلاعات سریعتر از پردازش اطلاعات باشد. در این حالت *BUFPTR* ممکن است از *UserPTR* جلو بزند و بدین ترتیب اطلاعات جدید بر روی کاراکتر هایی که هنوز توسط برنامه استفاده کننده خوانده نشده بودند ، نوشته می شود.

۲. اگر سرعت پردازش اطلاعات بیشتر از سرعت ورود آنها باشد. در این حالت *UserPTR* ممکن است از *BUFPTR* جلو بزند بدین ترتیب برنامه استفاده کننده سعی در خواندن کاراکترهایی می کند که هنوز وارد بافر نشده اند.

یکی از موارد کاربرد تکنیک بافر چرخه ای در کنترل ترمینال ها توسط سیستم عامل می باشد.

« شرایط استفاده از بافر چرخه ای در کنترل ترمینال ها »

۱) پردازش استفاده کننده فقط زمانی صدا زده می شود که یک خط کامل از اطلاعات ورودی جهت پردازش آماده باشد. زیرا زمانبندی کردن یک پردازش نیاز به یک زمان سربار عملیاتی ***Overhead*** دارد .

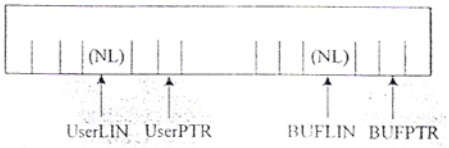
۲) استفاده کننده باید قادر باشد خطوط کامل را تایپ کند مثلاً ماکزیمم ۸۰ کاراکتر

۳) استفاده کننده باید بتواند غلط های تایپی خود در خطی که مشغول تایپ کردن می باشد را اصلاح کند برای این مورد پردازش بافر کننده باید با دیدن کاراکتر ***Backspace*** در بافر به عقب برگردد.

۴) در صورتیکه سیستم خیلی مشغول باشد و نتواند بلافاصله اطلاعات ورودی را پردازش کند، استفاده کننده باید بتواند بیش از یک خط را تایپ کند.

۵) پردازش استفاده کننده باید قادر به دوباره خواندن خطی که مشغول خواندن آن است ، باشد. بدین صورت عملیات تکراری که در ویرایشگرها وجود دارند را می توان به سادگی پیاده سازی کرد.

۶) اگر سیستم منتظر اطلاعات ورودی است و استفاده کننده هنوز شروع به تایپ کردن خط بعدی نکرده است، در آن صورت یک علامت ***Prompt*** باید صادر شود.



« نحوه پیاده سازی بافر چرخه ای »

شکل بالا یک سیستم بافر چرخه ای را نمایش می دهد این سیستم باید دارای خواص زیر باشد :

۱) حداقل طول بافر باید ۸۰ کاراکتر باشد یعنی اندازه یک خط

۲) کاراکترها توسط پردازش بافر کننده در نقطه ***BUFPTR*** وارد بافر می شوند.

۳) کاراکتر ها توسط پردازش استفاده کننده در نقطه ***USERPTR*** از بافر استخراج می شوند.

۴) هرگاه یک کاراکتر ***NL*** در بافر وارد شود ، پردازش استفاده کننده صدا زده می شود.

۵) نشانگر ***BUFLIN*** محل آخرین کاراکتر ***NL*** است که در بافر وارد شده است. این نشانگر برای دو حالت زیر به کار می رود:

الف) وقتی که ***USERPTR*** به ***BUFLIN*** می رسد پردازش استفاده کننده متوقف می شود و اگر استفاده کننده هنوز چیزی تایپ نکرده باشد یک ***Prompt*** صادر می شود یعنی زمانی که ***BUFLIN = BUFPTR***

ب) ***Backspace*** تنها تا ***BUFLIN*** عمل می کند ، بدین صورت استفاده کننده نمی تواند کاراکترهایی را که قبلاً توسط پردازش خوانده شده حذف نماید.

۶) *USERLIN* نشان دهنده محل آخرین کاراکتر *NL* است که توسط پردازش استفاده کننده خوانده شده است نشانگر *BUFPTR* حق ندارد از این نقطه جلوتر برود ، بدین ترتیب پردازش استفاده کننده همیشه قادر خواهد بود خط جاری را دوباره بخواند.

« مبحث : اصول زمانبندی »

✓ **اهداف زمانبندی :**

یک روش زمانبندی مناسب می تواند دارای برخی از خواص زیر باشد: (مهمترین اهداف)

۱) فراهم آوردن زمان پاسخ یا زمان گردش بهینه

۲) انجام یک کار در یک فاصله زمانی که معمولاً از طرف کاربر معین شده است .

۳) بالا بردن بهره وری *CPU*

۴) استفاده از سایر منابع سیستم با بهره وری بالا

✓ **سطوح زمانبندی:**

سه سطح زمانبندی به ترتیب زیر مطرح می شود:

۱) **زمانبندکار یا زمانبند سطح بالایی :** این زمانبند تصمیم می گیرد که کدام کار به سیستم وارد شود.

۲) **زمانبند پردازش یا زمانبند سطح میانی :** این زمانبند اولویت برنامه ها را تنظیم می کند و عملیات برش زمانی را سازمان دهی می کند.

۳) **هماهنگ کننده یا زمانبند سطح پایینی :** این زمانبند عملیات همگام کردن منطقی برنامه ها را انجام می دهد.

✓ **تخمین زمان پاسخ :**

شرایطی را به این صورت درنظر می گیریم که در آن استفاده کننده ، مثلاً از دستور ویرایش استفاده می کند که با اندازه C واحد از زمان پردازنده را احتیاج دارد. استفاده کننده در هر T ثانیه یک دستور را تایپ می کند و سپس باید به اندازه زمان R صبر کند تا پاسخی را بر روی ترمینال مشاهده کند ، بنابراین استفاده کننده نیاز به C واحد زمان پردازنده در هر T+R ثانیه دارد. اگر فرض کنید که N استفاده کننده وجود داشته باشند که عملیات مشابهی را بر روی ماشین انجام دهند . هر کدام از آنها یک دستور در هر T+R ثانیه اجرا می کنند در نتیجه خواهیم داشت $N \times C \leq T + R$

مثال) اگر بیست استفاده کننده وجود داشته باشند و هر کدام عملیات مشابهی همچون ویرایش را انجام دهند که نیاز به ۱۵ ثانیه از زمان CPU داشته باشد و هر یک از استفاده کنندگان به اندازه ۵ ثانیه از هر فعل و انفعال را صرف فکر کردن و تایپ کردن بکند ، زمان متوسط پاسخ چه اندازه است ؟

$20\times0/5=5+R \rightarrow R=5$ ثانیه

این مدل را می توان به نحوی توسعه داد تا دربرگیرنده حالاتی باشد که در آنها استفاده کنندگان همگی عملیات مشابهی را انجام نمی دهند.

برای مثال برای دانشجویانی که در یک کلاس عملی برنامه نویسی می کنند و ممکن است ۱۹ فعل و انفعال کوتاه جهت ویرایش کردن یک فایل لازم آید و در مقابل یک فعل و انفعال طولانی لازم است ، تا آخر کامپایل کنند.

اگر کامپایل کردن کردن ۵ ثانیه طول بکشد ، آنگاه هر ۲۰ فعل و انفعال مقادیر زیر زمان مصرف می کنند :

$19+0/5$ ثانیه زمان CPU برای ویرایش ، 1×5 ثانیه زمان CPU برای کامپایل

در صورتی که پردازش ها تحت برش زمانی قرار نگیرند ، در ان صورت زمان هر فعل و انفعال هنوز برابر با T+R ثانیه است. در طی ۲۰ فعل و انفعال می توان زمان طی شده را برابر با زمان CPU که عرضه شده دانست ، در نتیجه فرمول زیر نتیجه می شود :

$$20\times(5+R) = (19\times 0/5 + 1\times 5)\times N$$

که برای ۲۰ استفاده کننده زمان پاسخ برابر با ۹/۵ می شود $R=9/5$

اگر برنامه ها را تحت برش زمانی قرار دهیم ، عملکرد این نوع سیستم کاملاً متفاوت می شود. زمان لازم برای انجام یک فعل و انفعال ساده برابر با T+R می باشد، البته در این صورت R عبارتست از زمانی که یک برنامه صبر می کند تا یک برش زمانی به آن داده شود ، بعلاوه زمانی جهت انجام آن عمل لازم است. برای فعل و انفعال های طولانی تر ، زمان پاسخ عبارتست از $T+n\times R$ که n تعداد برش زمانی است که برای تکمیل فعل و انفعال لازم است. در مثال قبلی اگر مدت هر برش زمانی برابر با ۰/۵ ثانیه باشد ، دستورهای ویرایش در یک برش زمانی اجرا می شوند و کامپایل کردن در ده برش زمانی که $n=10$ باشد به صورت زیر محاسبه می شود :

زمان طی شده $=19\times(5+R) + 1\times(5+10R)$ ثانیه

که در طی آن هنوز بایستی زمان CPU زیر را تقسیم کرد :

$19\times 0/5 + 1\times 5$ ثانیه برای برنامه

اگر ۲۰ استفاده کننده باشند آنگاه داریم :

$$19\times(5+R) + 1\times(5+10R) = (19\times 0/5 + 1\times 5)\times 20$$

$R=6/5$ ثانیه

« مبحث : زمانبندی – الگوریتم ها »

روشی که باید در هنگام زمانبندی انتخاب شود ، به بارکاری سیستم و بویژه به این موضوع بستگی دارد که سیستم قطعی (deterministic) است یا غیر قطعی (non – deterministic). سیستم قطعی شامل سیستم های کنترل بلادرنگ می باشد در این از سیستم ها تعداد فعل و انفعالات و زمان پردازنده مورد نیاز هر یک از قبل معلوم است. سیستم غیر قطعی معمول تر و رایج تر است و شامل سیستم های همه منظوره اشتراک زمانی می شود ، این سیستم ها عملکرد استفاده کننده متغیر و گوناگون دارند و به جهت به حداکثر رساندن کارآیی تخمینی سیستم بایستی از توزیعات احتمالی مناسب برای زمانهای ورود و اجرا بهره مند گردند.

«زمانبندی غیر انحصاری برای تکالیف مستقل »

در مواردی که تکالیف مستقل هستند و در ضمن گرفتن CPU از تکالیف هم مجاز است به سادگی می توان طول زمانبندی را کمینه ساخت . برای این کار از الگوریتمی به نام bin packing استفاده می شود ، کمترین طول زمانبندی برای هر سیستم تکلیفی دارای طولی است که به کمک رابطه زیر مشخص می شود:

$$W_{OPT} = \text{Max}\ \{ {}^{1}/_{M} \sum \tau_i , \text{Max}\ \{ \tau_i \} \}$$

(M تعداد پردازنده ها است و τ_i زمان اجرای تکلیف i است) زیرا حداقل نیاز به زمانی برابر با $\text{Max}_i\ \{ \tau_i \}$ (یعنی مقدار ماکسیمم بین مقادیر مختلف τ_i) جهت اجرای طولانی ترین تکلیف داریم و حداقل نیاز به $\sum \tau_i$ زمان پردازنده است. الگوریتم bin packing نیازی به مرتب کردن تکالیف ندارد و تکالیف را می توان به هر ترتیبی انتخاب نمود و به یکی از پردازنده ها منسوب کرد تا اینکه به این پردازنده به اندازه W_{OPT} کار داده شده باشد.

به عنوان مثال در سیستم تکالیف زیر :

$$\{ \tau_i \} = \{ 13 , 8 , 7 , 6 , 4 , 2 , 2 , 1 \}$$

برای تعداد پردازنده های مختلف زیر زمانبندی های مربوطه را می توان به راحتی مشاهده کرد.

الف) تعداد پردازنده ها (n) برابر با ۲ است .

$$W_{OPT} = \text{Max}\ \{ \tfrac{1}{2} (13 + 8 + 7 + 6 + 4 + 2 + 2 + 1) , \text{Max}\ (13 , 8 , 7 , 6 , 4 , 2 , 2 , 1) \}$$
$$= \text{Max}\ \{ 21/5 , 13 \} = 21/5$$

P۱	T۱/۱۳			T۲/۸		T۳/$\frac{۱}{۲}$
P۲	T۳/۶$\frac{۱}{۲}$	T۴/۶	T۵/۴	T۶/۲	T۷/۲	T۸/۱

ب) تعداد پردازنده ها (n) برابر ۳ است .

$W_{OPT} = 14\frac{1}{3}$

P۱	T۱/۱۳				T۲/۱$\frac{۱}{۳}$
P۲	T۲/۶$\frac{۲}{۳}$	T۳/۷			T۴/$\frac{۲}{۳}$
P۳	T۴/۵$\frac{۱}{۳}$	T۵/۴	T۶/۲	T۷/۲	T۸/۱

پ) تعداد پردازنده ها برابر با ۴ است .

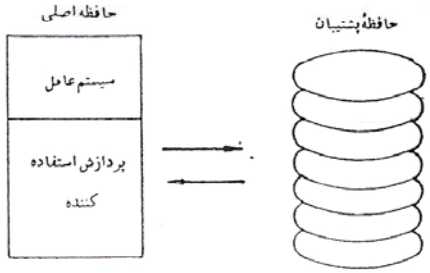
$W_{OPT} = 13$

P۱	T۱/۱۳				
P۲	T۲/۸			T۳/۵	
P۳	T۳/۲	T۴/۶		T۵/۴	T۶/۱
P٤	T۶/۱	T۷/۲	T۸/۱		

تعداد توقف ها و از دست دادن CPU کمتر یا مساوی با (n-1) است. البته این مورد الزاماً کمترین تعداد از دست دادن های CPU نیست.

« مبحث : مدیریت حافظه – اصول اولیه »

• سیستم مبادله ساده : ساده ترین حالت سیستمی است که در آن آنقدر فضا وجود دارد که سیستم عامل تنها همراه یک پردازش استفاده کننده در حافظه اصلی قرار می گیرد. هر پردازش دیگری که به آن برش زمانی تعلق گیرد باید به داخل این تنها حافظه آورده شود که به این مورد Swap – in می گویند که شکل زیر این مورد را نشان می دهد :



معیارهایی که جهت سنجش مبادله بکار گرفته می شوند یعنی مقادیر مورد تقاضای حافظه و سرعت نشان می دهد که:

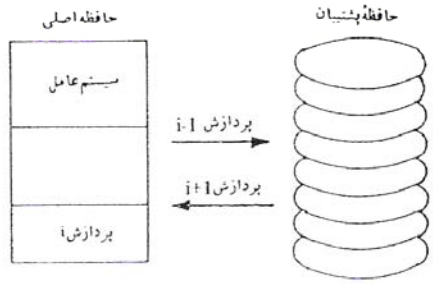
سیستم عامل + یک پردازش = اندازه حافظه اصلی

زمان مبادله خارج + زمان CPU + زمان مبادله داخل = زمان هر فعل و انفعال

زمان CPU + زمان مبادله × ۲ =

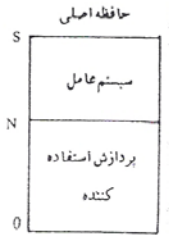
$$\text{بکارگیری CPU} = \frac{\text{زمان CPU}}{\text{زمان مبادله} + \text{زمان CPU} \times 2} \times 100$$

• سیستم مبادله پیچیده تر : اشکال روش مبادله ساده این است که وقتی یک پردازش به داخل و یا خارج از حافظه اصلی می رود CPU بیکار می ماند و زمانی که کارها CPU–limited هستند و طول تناوب هر برش زمان هم زیاد است باز هم زمان مبادله غالب است. برای برطرف شدن این مشکل باید سعی شود مقداری از مبادله به صورت همزمان با محاسبه در پردازش استفاده کننده دیگر انجام گیرد. برای این مورد سیستم باید قادر باشد بیش از یک پردازش را در هر زمان در حافظه اصلی نگه دارد مانند شکل زیر :



در این سیستم پردازش i در حالی اجرا می شود که پردازش قبلی (i - 1) به خارج از حافظه برده شود و پردازش بعدی(i + 1) به داخل حافظه آورده می شود.

- حفاظت حافظه : در زمانی که تعدادی پردازش در سیستم وجود دارند ، سیستم عامل باید از هر یک در مقابل دخالت های دیگران حفاظت بعمل آورد. در شکل زیر پردازش استفاده کننده در حافظه در محل صفر قرار داده شده و سیستم عامل در بالای حافظه است. زمانی که برنامه استفاده کننده در حال اجرا است ، این برنامه باید بتواند فقط به محل های صفر تا N دست پیدا کند و این در حالی است که در هنگام اجرای سیستم عامل تمامی حافظه یعنی از صفر تا S در دسترس است.



اعتبار آدرس ها باید توسط سخت افزار بررسی شود برای انجام این کار دو نکته را باید در نظر گرفت :

۱) محل شروع سیستم عامل ، یعنی N از یک ثبات ویژه استفاده می کند تا این اطلاعات را نگهداری کند به این ثبات، ثبات حد (Limit Register) گفته می شود.

۲) در هر زمان ، باید بدانیم کدامیک از برنامه های استفاده کننده یا سیستم عامل در حال اجرا می باشند ، برای پیاده سازی این حالت باید دو وجه اجرا فراهم آورده شود : ۱- وجه استفاده کننده (User) ۲- وجه ممتاز (Privileged). در وجه استفاده کننده فقط آدرس های صفر تا N معتبر است و در وجه ممتاز آدرس های صفر تا S مجاز هستند و انتقال از وجه استفاده کننده به ممتاز در حالتهای زیر اتفاق می افتد :

۱) زمانی که یک وقفه رخ می دهد مقدار شمارشگر برنامه برابر با آدرس یکی از نقاط ورودی کد سیستم عامل قرار داده می شود.

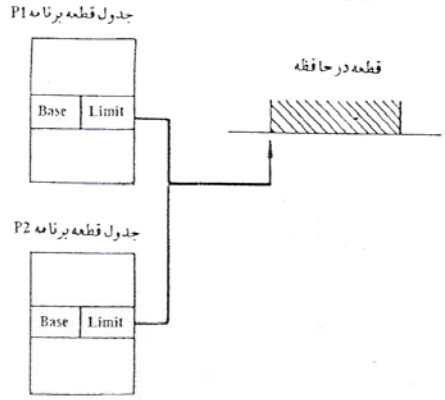
۲) در نتیجه یک دستورالعمل ویژه مثل Enter Operating System بدین صورت تنها در هنگام اجرای کد سیستم عامل می توان در وجه ممتاز قرار گرفت.

« مبحث مدیریت حافظه – قطعه بندی »

- قطعات اشتراکی : برنامه ها می توانند از یک یا چند قطعه بخصوص بصورت اشتراکی استفاده کنند ، که روشهای مختلفی برای این کار موجود است اما مهمترین این روشها عبارتند از :

۱) همه مستقیم ۲) یکی مستقیم بقیه غیر مستقیم ۳) همه غیر مستقیم

۱) همه مستقیم : در این روش هر یک از برنامه هایی که بصورت اشتراکی از یک قطعه استفاده می کنند در جدول قطعه خود دارای یک کپی از مقادیر پایه – حد مربوط به قطعه اشتراکی می باشند در این روش اگر قرار باشد این قطعه حرکت کند به طور مثال داخل و یا خارج از حافظه مبادله شود باید تمام جداول قطعه ای که به این قطعه اشاره می کنند ، اصلاح شوند که این کار زمان بر است .



۲) یکی مستقیم ، بقیه غیر مستقیم : در این روش ، کاربر اصلی قطعه اشتراکی دارای یک اشاره گر مستقیم از جدول قطعه خودش به قطعه مورد نظر در حافظه می باشد. سایر برنامه ها دارای اشاره گرهای غیر مستقیم هستند که از جدول قطعه آنها به درون جدول قطعه برنامه اصلی است و به این صورت به شکل غیر مستقیم به قطعه مورد نظر در حافظه دسترسی دارند.

در این روش باید به نکات زیر توجه کرد :

۱) برای کلیه ورودی های جدول قطعه لازم است یک بیت که به این بیت، بیت غیر مستقیم می گویند ایجاد شود تا به این صورت بتوان تشخیص داد که آیا یک ورودی مستقیماً به یک قطعه اشاره می کند و یا اینکه به یک ورودی در یک جدول قطعه دیگر اشاره می کند.

۲) این روش فقط زمانی می تواند مناسب باشد که مالک اصلی یک قطعه اشتراکی به صورت دائمی باشد اما اگر جدول قطعه این برنامه اصلی حرکت نماید و یا اینکه برنامه مالک تمام شود این روش نمی تواند مناسب باشد.