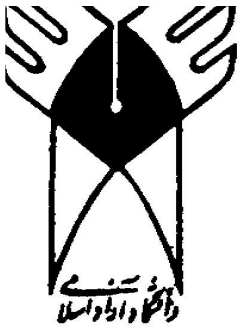

دانشگاه آزاد اسلامی

.....

بررسی:

INPUT/ OUTPUT MANAGEMENT

استاد:

.....

یکی از عملکردهای بسیار مهم و اصلی سیستم عامل کنترل و مدیریت دستگاههای ورودی/ خروجی (I/O) در کامپیوتر است. سیستم عامل باید فرمانهایی را به این دستگاهها بفرستد، وقفه ها را بگیرد و خطاها را اداره کند. و همچنین باید یک رابط بین دستگاهها و بقیه سیستم به منظور استفاده ساده تر از آنها فراهم نماید.

اصول سخت افزاری I/O

دید افراد مختلف نسبت به سخت افزار I/O متفاوت است. مهندسین برق و الکترونیک آن را بصورت مدارهای مجتمع، مدار چاپی، منابع تغذیه، موتورها و دیگر اجزاء فیزیکی تشکیل دهنده سخت افزار می بینند. برنامه نویسان آن رابه شکل واسطه ارائه شده به نرم افزار، فرامین مورد قبول سخت افزار، توابع قابل اجرا و خطاهای احتمالی و گزارش آنها می بینند. از یک دیدگاه، دستگاههای ورودی/ خروجی بطور کلی به دو دسته تقسیم می شوند.

۱- دستگاههای بلوکی BLOCK DEVICE

۲- دستگاههای کارکتری CHARACTER DEVICE

یک دستگاه بلوکی وسیله ای است که اطلاعات را در یک بلوک با اندازه معین ذخیره می کند که هر کدام با آدرس خودشان مشخص شده اند. حدود اندازه بلوک‌های معمولی از ۵۱۲ بایت تا ۳۲۷۶۸ بایت می باشند خاصیت اساسی یک دستگاه بلوکی این است که خواندن و نوشتن هر بلوک را بطور مستقل از بقیه ممکن می سازد دیسکها از متداولترین دستگاههای بلوکی هستند.

نوع دیگری از دستگاههای ورودی/ خروجی، دستگاههای کارکتری است. یک دستگاه کارکتری یک جریان از کارکترها را بدون توجه به هیچ ساختار بلوکی، دریافت نموده و یا تحویل می دهد، بنابراین قابلیت آدرس دهی و جستجو در آن وجود ندارد. چاپگرها، واسطه های شبکه، موشواره ها و دیگر دستگاههایی که شبیه دیسک نیستند، به عنوان دستگاه کارکتری در نظر گرفته می شوند. از دیدگاه دیگر دستگاههای ورودی/ خروجی به سه طبقه تقسیم می شوند:

۱- قابل خواندن توسط انسان:

مناسب برای ارتباط با کاربر، مانند پایانه های نمایش دهنده تصویر، صفحه کلید و...

۲- خواندن توسط ماشین:

مناسب برای ارتباط با وسائل الکترونیکی، مانند گردانندهای دیسک و نوار، حس کننده‌ها، کنترل کننده ها و محرکها.

۳- ارتباطات:

مناسب برای ارتباط با دستگاههای دور مانند، گرداننده های دیجیتالی خط و مودمها.

تفاوتهای زیادی در بین هر طبقه وجود دارد که مهمترین آنها عبارتند از:

نرخ انتقال، کاربرد، پیچیدگی کنترل، واحد انتقال، نمایش داده ها، شرایط خطا.

سازمان عمل ورودی/ خروجی

الف) ورودی/ خروجی برنامه ریزی شده: پردازنده یک فرمان ورودی/ خروجی را از جانب فرایندی به یک مولفه ورودی/ خروجی صادر می کند. سپس آن فرایند قبل از ادامه، تا کامل شدن عمل ورودی/ خروجی به انتظار می ماند.

ب) ورودی/ خروجی مبتنی بر وقفه: پردازنده یک فرمان ورودی/ خروجی را از جانب فرایند صادر می کند، سپس به اجرای دستورالعملهای بعدی ادامه می دهد و با کامل شدن عمل ورودی/ خروجی یا وقفه مولفه ورودی/ خروجی مواجه می شود. دستورالعملهای بعدی می توانند از همان فرایند باشند البته به شرطی که فرایند نیازی به انتظار برای تکمیل ورودی/ خروجی نداشته باشد. در غیر این صورت فرایند در انتار وقفه معلق می گردد و کار دیگری انجام می گیرد.

پ) دسترسی مستقیم به حافظه DMA: مولفه DMA تبادل داده ها بین حافظه اصلی و مولفه ورودی/ خروجی را کنترل می کند. پردازنده تقاضایی برای انتقال یک بلوک از داده ها را به مولفه DMA می فرستد و فقط پس از انتقال کل بلوک مورد وقفه قرار می گیرد.

تکامل عمل ورودی/ خروجی

می توان گامهای تکاملی را بصورت زیر خلاصه نمود:

۱- کنترل دستگاه جانبی بطور مستقیم.

۲- یک کنترل کننده یا مولفه ورودی/ خروجی افزوده شده است. پردازنده از ورودی/ خروجی برنامه ریزی شده بدون وقفه استفاده می کند. با این گام پردازنده از جزئیات اختصاصی واسط های دستگاه خارجی جدا می گردد.

۳- همان پیکربندی گام دوم، ولی این بار از وقفه استفاده شده است. دیگر وقت پردازنده تلف نمی شود.

٤- از طریق DMA کنترل مستقیم حافظه به مولفه ورودی/ خروجی داده شده است. حال می توان بلوکی از داده را بدون دخالت پردازنده به/ از حافظه انتقال داد.

۵- مولفه ورودی/ خروجی تا حد پردازنده ای مجزا، با مجموعه ای دستورالعملهای خاصی برای ورودی/ خروجی ارتقاء یافته است. پردازنده مرکزی (CPU)، پردازنده ورودی/ خروجی را برای اجرای برنامه ورودی/ خروجی از حافظه هدایت می کند، پردازنده ورودی/ خروجی این دستورالعملها را بدون دخالت پردازنده مرکزی واکنشی و اجرا می کند.

٦- مولفه ورودی/ خروجی دارای حافظه محلی اختصاصی و در واقع به نوبه خود یک کامپیوتر است. با این معماری، مجموعه بزرگی از دستگاههای ورودی/ خروجی با حداقل دخالت پردازنده مرکزی می تواند کنترل شود.

دسترسی مستقیم به حافظه

اساساً مولفه DMA باید فقط زمانی از گذرگاه استفاده کند که پردازنده به آن نیازی ندارد، یا باید پردازنده را وادار کند که بطور موقت عملیات خود را معلق نماید. روش

اخیر بیشتر معمول است و به آن ربودن چرخه می گویند. روش DMA به این صورت کار می کند که، هرگاه پردازنده در صدد خواندن یا نوشتن بلوکی از داده ها برآید، فرمانی به مولفه DMA می دهد و از این طریق اطلاعات زیر را به آن مولفه می‌فرستد.

- اینکه درخواست خواندن یا نوشتن.
- آدرس دستگاه ورودی/ خروجی درگیر.
- آدرس مکان شروع خواندن یا نوشتن در حافظه.
- تعداد کلماتی که باید خوانده یا نوشته شود.

سپس پردازنده به کارهای دیگر پرداخته و عمل ورودی/ خروجی را به مولفه DMA محول می کند. مولفه DMA همه بلوک داده ها را به صورت یک کلمه در هر زمان به طور مستقیم و بدون عبور از پردازنده به/ از حافظه انتقال می دهد. هنگامیکه انتقال کامل شده مولفه DMA یک علامت وقفه به پردازنده ارسال می کند. بنابراین پردازنده فقط در آغاز و پایان انتقال درگیر می باشد.

اصول نرم افزاری I/O

هدف اصلی نرم افزار I/O، سازماندهی نرم افزار بصورت یک سری از لایه ها است که کار لایه های پایینی پنهان ساختن ویژگی های پیچیده سخت افزار از لایه های بالاتر و کار لایه های بالایی نشان دادن یک واسطه منظم، ساده و شفاف برای استفاده کننده است. از این اهداف می توان به موارد زیر اشاره نمود.

– برنامه ها تا حد ممکن باید از دستگاه مستقل باشند. یعنی باید بتوانیم برنامه هایی بنویسیم که قادر باشد فایلهای روی یک دیسک نرم، دیسک سخت و یا CD-ROM را بخواند.

– یکی دیگر از اهداف نرم افزار I/O که ارتباط نزدیکی با مفهوم نرم افزار مستقل دستگاه دارد، نامگذاری یکسان می باشد. نام یک فایل یا یک دستگاه بطور ساده باید یک رشته یا یک عدد صحیح باشد و به هیچ وجه به دستگاه بستگی نداشته باشد.

– دیگر اینکه نرم افزار I/O بتواند خطاها را اداره کند. بطور کلی خطاها باید در حد امکان نزدیک به سخت افزار اداره شوند. اگر کنترل کننده یک خطای خواندن را کشف کند باید سعی کند که اگر می تواند خودش خطا را اصلاح کند و اگر نمی تواند این کار را به گرداننده دستگاه (لایه بعدی) بسپارد.

– هدف دیگر، که یک موضوع کلیدی است، انتقال همگام (بلوکه کردن)در مقابل انتقال های ناهمگام (مبتنی بر وقفه) می باشد.

– مورد دیگر، دستگاههای قابل بهره وری مشترک در مقابل دستگاههای انحصاری می باشند. بعضی از دستگاههای I/O از قبیل دیسکها می توانند توسط چندین کاربر بصورت همزمان استفاده شوند. هیچ مشکلی از این بابت که کاربرهای متعدد، فایلها را روی یک دیسک و در یک زمان باز کنند بوجود نمی آید. دستگاههای دیگر از قبیل نوار گردان باید فقط به یک کاربر اختصاص یابند و تاپایان کار آن کاربر، در اختیار روی باقی بماند.

اهدافی که در بالا بیان شد. با روشی قابل درک و موثر با ساختن نـرم افـزار I/O در چهار لایه بدست می آید:

الف) اداره کننده های وقفه

ب) گرداننده های دستگاه

پ) نرم افزار سیستم عامل مستقیم از دستگاه

ج) نرم افزار سطح کاربر

از طرفـی، کـارآیی و عمومیـت در طراحـی امکانـات ورودی/ خروجـی از اهـداف طراحی سیستم عامل است. چون اغلب اوقات عملیـات ورودی/ خروجـی در سیسـتم کامپیوتری موجب تنگنا می شوند. بیشتر دستگاههای ورودی/ خروجـی در مقایسـه بـا حافظه اصلی و پردازنده بسـیار کنـد هسـتند یکـی از راههـای مقابلـه بـا ایـن مسـئله بکارگیری چند برنامگی است که اجازه می دهد در حـالی کـه چنـد فراینـد در انتظـار عملیات ورودی/ خروجی هستند، فرایند دیگری اجرا گردد.

هدف اصلی دیگر عمومیت است. به خاطر سادگی و رهایی از خطا، مطلـوب اسـت این است که تمام دستگاهها به صورتی یکنواخت هم در چگونگی نگرش فرایندها بـه دستگاههای ورودی/ خروجـی، و هـم در نحـوه اداره دسـتگاههای ورودی/ خروجـی توسط سیستم عامل بکار می روند. بدسـت آوردن عمومیـت واقعـی در عمـل مشـکل است. آنچه می توان انجام داد، استفاده از رویکـرد سلسـله مراتبـی و مولفـه ای بـرای طراحی اعمال ورودی/ خروجی است.

بن بست

در سیستم های کامپیوتری منابع بسیاری وجود دارد که در هر لحظه فقط توسط یک پروسس مورد استفاده قرار می گیرند. مثالهایی از این منابع، پلاترها و یا پرینترها، CD-ROM،... می باشند. اگر دو فرایند بطور همزمان بر روی یک چاپگر بنویسند، نتایج نادرست بدست خواهد آمد. استفاده همزمان دو فرآیند از یک درآیه جدول فرآیند موجب سقوط و از کار افتادن سیستم خواهد شد بنابراین در کلیه سیستم عاملها توانایی واگذاری انحصاری یک منبع خاص جهت دستیابی یک فرآیند (بطور موقت) وجود دارد.

بن بست زمانی اتفاق می افتد که فرایندها می توانند بطور انحصاری به دستگاهها، فایلها و دیگر اشیاء دست یابند. بطور کلی به هر یک از این اشیاء منبع گفته می‌شود. منابع می توانند سخت افزاری و یا یک رکورد از یک بانک اطلاعاتی باشد. منابع به دو گروه تقسیم می شوند.

الف) قابل پس گرفتن PREEMPTABLE

به منبعی قابل پس گرفتن می گویند که بتوان آن را بدون اینکه تاثیر منفی بر جای گذارد از فرایند صاحب آن باز پس گرفت (بعد از پایان کار).

ب) غیر قابل پس گرفتن NON PREEMPTABLE

به منبعی غیرقابل پس گرفتن می گویند که بدون اینکه باعث خراب شدن محاسبات آن شویم نمی توان آن را از صاحبش که گویی آن را قبضه کرده است، باز پس گرفت.

با توجه به مطالب ارائه شده در بالا می توان بن بست را بدین گونه تعریف نمود.

یک مجموعه از فرایندها در صورتی منجر به بن بست می شوند که هر یک از فرایندهای درون مجموعه منتظر رویدادی است که فقط فرآیند دیگر از همین مجموعه می تواند باعث ایجاد آن شود. این رویداد رهاسازی منابع توسط فرآیند قبلی است.

شرایط بن بست

از دید کافمن، برای پدید آمدن بن بست ٤ شرط باید بوجود آید.

۱- **انحصار متقابل**: در هر زمان تنها یک فرآیند می تواند از یک منبع استفاده کند.

۲- **نگهداری و انتظار**: در حالیکه فرایند منبع تخصیص داده شده را نگه می‌دارد، منتظر تخصیص منبع دیگر است.

۳- **قبضه نکردن**: منبعی که قبلاً به فرآیند واگذار شده را نتوان به زور از آن باز پس گرفت.

٤- **انتظار مدور**: زنجیر بسته ای از فرآیندها وجود دارد، به گونه ای که هر یک، حداقل یک منبع مورد نیاز فرایند بعد در زنجیر را نگه دارد.

برای برخورد با بن بست الگوریتم های مختلفی وجود دارد. اما بطور کلی سه راه برای اداره بن بست وجود دارد.

۱- کشف بن بست و ترمیم

۲- اجتناب از بن بست

۳- پیشگیری از بن بست

نکته: عده ای یک مورد به موارد بالا می افزایند و آن اینکه بن بست را نادیده بگیریم و در بعضی سیستم عاملها حتی برای کشف آن اقدامی بعمل نمی آید مانند UNIX.

کشف بن بست

راهبرد کشف بن بست، دسترسی به منابع را محدود نمی کند بلکه هر جا که ممکن باشد منابع درخواست شده در اختیار فرایندها قرار می گیرد و سیستم بجز نظارت بر فرایند درخواستها و رهاسازی منابع کار دیگری انجام نمی دهد.

ترمیم

پس از کشف بن بست، برای ترمیم راهبردهایی وجود دارد. در زیر رویکردهای ممکن به ترتیب پیچیدگی فهرست شده اند.

الف) قطع تمام فرایندهای بن بست

ب) برگشت هر یک از فرایندهای بن بست به آغاز و شروع مجدد هر یک

پ) قطع پی‌درپی فرایندهای بن بست تاجایی که دیگر بن بست وجود نداشته باشد

ت) قبضه کردن پی در پی منابع تا جایی که دیگر بن بست وجود نداشته باشد

اجتناب از بن بست

یک رویکرد در حل مسئله بن بست، اجتناب از بن بست است. در اینجا در مورد درخواستهای منبع به گونه ای عمل می کنیم که حداقل یکی از ٤ شرط بن بست اتفاق نیفتد برای اینکار به دو روش می توان عمل نمود.

الف) عدم شروع فرایندی که ممکن است درخواستهایش منجر به بن بست شود.

ب) عدم پاسخ به درخواستهای منبع اضافی از فرایندی که با این واگذاری ممکن است بن بست پیش آید عدم تخصیص منبع یکی از راه حل ها می باشد و الگوریتم بانکداران که توسط DIJKSTRA ارائه شد به این مقوله می پردازد.

پیشگیری از بن بست

راهبرد پیشگیری از بن بست این است که طرح سیستم به گونه ای باشد که از قبل امکان بن بست از بین برده شود روش پیشگیری دو گروهند. روش غیر مستقیم که پیشگیری از بروز یکی از شرایط ۱ تا ۳ است و روش مستقیم که پیشگیری از بروز انتظار مدور است.

انحصار متقابل

در کل شرط اول را نمی توان رد کرد. اگر دسترسی به منبعی نیازمند انحصار متقابل باشد در این صورت سیستم عامل باید از انحصار متقابل حمایت نماید. به عنوان مثال برای خواندن از پرونده توسط چندین فرایند نباید مشکل پیش آید اما برای نوشتن باید انحصاراً در اختیار یک فرایند قرار گیرد.

نگهداری و انتظار

در این روش همه فرایندها را وادار می کنیم که منابع مورد نیاز خود را قبل از اینکه شروع به اجرا کنند را درخواست نمایند. در صورتی که همه چیز مهیا بود فرایند شروع شود در غیر این صورت فرایند فقط باید صبر کند. این رویکرد از دو جهت ناکار آمد است اول اینکه ممکن است فرایندی برای مدت طولانی در انتظار تکمیل تمام منابع مورد درخواستش باقی بماند، درحالیکه حتی با بعضی از آن منابع هم می توانست پیش برود. دوم اینکه ممکن است برای مدت قابل ملاحظه ای منابعی که به یک فرایند تخصیص داده شده است بی استفاده بماند، در حالیکه درخواستهای فرایندهای دیگر رد می شوند.

قبضه نکردن

از راههای متعددی می توان از بروز قبضه نکردن پیشگیری کرد. یک راه این است که اگر فرایندی منابعی را در اختیار دارد، درخواست جدیدش مورد قبول قرار نگیرد، چنین فرایندی باید منابع قبلی خود را رها کند و در صورت نیاز، مجدداً با منبع اضافی، درخواست نماید.

راه دیگر این است که اگر فرایندی منبعی را درخواست نماید که فرایند قبلی آنرا نگهداشته است، احیاناً سیستم عامل آن فرایند را قبضه کرده و منابعش را آزاد نماید. طرح اخیر تنها هنگامی از بن بست پیشگیری می کند که فرایندها از اولویت یکسان برخوردار نباشند.

انتظار مدور

با تعریف یک ترتیب خطی از انواع منابع، می توان از بروز شرایط انتظار مدور پیشگیری کرد. اگر منبعی از نوع R، به یک فرایند تخصیص داده شده است، در ادامه تنها منابعی را می تواند درخواست کند که (در این ترتیب خطی) نوع آنها بعد از R قرار گرفته باشد.

میانگیری ورودی/ خروجی

فرض کنید یک فرایند کاربر می خواهد بلوکهای داده ها را یکی یکی از نوار بخواند و طول هر بلوک ۱۰۰ بایت است. داده ها باید در یک ناحیه داده ای در فرایند کاربر در آدرس ۱۰۰۰ تا ۱۰۹۹ خوانده شوند. ساده ترین روش، اجرای یک فرمان ورودی/ خروجی برای واحد نوار و سپس انتظار برای آماده شدن داده هاست. این انتظار می‌تواند انتظار مشغولی یا عملی تر، بصورت معلق شدن فرایند بر روی وقفه باشد.

این رویکرد با دو مسئله همراه است. اولاً، برنامه در انتظار تکمیل ورودی/ خروجی نسبتاً کند معلق شده است. ثانیاً این رویکرد به ورودی/ خروجی، با تصمیمات مبادله توسط سیستم عامل تداخل دارد. مکانهای مجازی ۱۰۰۰ تا ۱۰۹۹ باید در طول انتقال بلوک در حافظه اصلی بمانند.

در غیر این صورت ممکن است برخی از داده ها از دست بروند. اگر از صفحه‌بندی استفاده شده باشد، حداقل صفحه ای که شامل مکانهای مورد نظر است، باید در حافظه

اصلی قفل شود. پس، اگرچه ممکن است بخشهایی به دیسک منتقل شوند، ولی مبادله کامل فرایند به خارج ممکن نیست، حتی اگر این امر مورد نظر سیستم عامل باشد.

همچنین توجه کنید که خطر بن بست یک فرایند واحد وجود دارد. اگر فرایندی یک رمان ورودی/ خروجی صادر کند و در انتظار نتیجه معلق شود و قبل از شروع عمل به خارج مبادله گردد، این فرایند در انتظار رخداد ورودی/ خروجی، در حالت مسدود قرار می گیرد و آن عمل ورودی/ خروجی هم در انتظار مبادله شدن آن فرایند به داخل قرار می گیرد. برای پرهیز از این بن بست، حافظه کاربر که درگیر عمل ورودی/ خروجی است، باید بلافاصله بعد از صدور تقاضای ورودی/ خروجی در حافظه اصلی قفل شود.

برای اجتناب از این سربارها و عدم کاراییها، برخی اوقات راحتتر است که انتقالهای ورودی پیش از درخواست انجام شود و انتقالهای خروجی مدتی بعد از درخواست صورت گیرد. این روش بنام میانگیری شناخته می شود.

در بحث رویکردهای متفاوت به میانگیری، گاهی تمایز قائل شدن بین دو نوع دستگاه ورودی/ خروجی بلوکی و جریانی مهم است. دستگاههای بلوکی اطلاعات را در بلوکهایی که معمولاً دارای اندازه ثابت هستند ذخیره می کنند و انتقالها بصورت بلوک بلوک انجام می گیرد. معمولاً مراجعه به داده ها با شماره بلوک امکانپذیر است. دیسکها و نوارها این گونه اند. دستگاههای جریانی انتقال داده ها به خارج/ داخل را به

صورت جریانی از بایتها انجام می دهند پایانه ها، چاپگرها، درگاههای ارتباطات، موس،... این گونه اند. اکثر دستگاههایی که حافظه ثانوی نیستند، جریانی می‌باشند.

تک میانگیر

ساده ترین پشتیبانی سیستم عامل، ارائه میانگیر واحد است. وقتی فرایند کاربر یک درخواست ورودی/ خروجی صادر کند، سیستم عامل یک میانگیر در بخش سیستمی حافظه اصلی به عمل مزبور تخصیص می دهد. برای دستگاههای بلوکی، طرح تک میانگیر به صورت زیر توصیف می شود.

انتقالهای ورودی به میانگیر سیستم صورت می گیرند. با کامل شدن انتقال، فرایند مزبور بلوک را به داخل فضای کاربر منتقل کرده و بلافاصله بلوک دیگری را تقاضا می‌کند. این رویکرد نسبت به سیستمی که میانگیری ندارد، عموماً موجب افزایش سرعت می شود. فرایند کاربر می تواند مشغول پردازش یک بلوک از داده ها باشد، در حالیکه بلوک بعدی در حال خوانده شدن است. از انواع دیگر میانگیر می توان دو میانگیر و میانگیر مدور را نام برد.

ورودی/ خروجی در UNIX

در UNIX هر واحد دستگاه ورودی/ خروجی به پرونده خاصی مربوط شده است. اینها توسط سیستم پرونده ها مدیریت می شوند. به همان صورت پرونده های داده‌های کاربران در/ از آنها نوشته یا خوانده می شود. به این ترتیب واسط یکنواخت و روشنی

برای کاربران و فرایندها فراهم می گردد. برای خواندن از یک دستگاه یا نوشتن در آن، درخواستها به پرونده مخصوص همان دستگاه داده می شود.

در UNIX دو نوع ورودی/ خروجی با میانگیر و بدون میانگیر وجود دارد. ورودی/ خروجی با میانگیر از میانگیرهای سیستم می گذرد، در حالیکه در ورودی/ خروجی بدون میانگیر انتقال بین مولفه ورودی/ خروجی و ناحیه ورودی/ خروجی فرایند، بطور مستقیم انجام می شود. برای ورودی/ خروجی با میانگیر، دو نوع میانگیر بکار رفته است: حافظه های پنهان برای میانگیر سیستم و صفهای نویسه ها.

حافظه پنهان میانگیر

حافظه پنهان میانگیر در UNIX در واقع یک حافظه پنهان دیسک است. عملیات ورودی/ خروجی با دیسک، از طریق حافظه پنهان میانگیر انجام می شود. انتقال داده‌ها بین حافظه پنهان میانگیر و فضای فرایند کاربر همواره با به کارگیری DMA رخ می‌دهد. از آنجا که هم حافظه پنهان میانگیر و هم فضای فرایند در حافظه اصلی هستند، در این مورد امکانات DMA برای کپی کردن حافظه به حافظه استفاده می‌گردد. این عمل از چرخه های پردازنده استفاده نمی کند ولی چرخه های گذرگاه را به کار می گیرد.

برای مدیریت حافظه پنهان میانگیر، سه بست نگهداری می شود.

لیست آزاد، لیست دستگاه، صف ورودی/ خروجی گرداننده.

صف نویسه ها

دستگاههای بلوکی، مانند دیسک و نوار می توانند به طور موثری از حافظه پنهان میانگیر استفاده کنند. برای دستگاههای نویسه ای مانند پایانه ها، چاپگرها، صورت دیگری از میانگیر مناسب است. صف نویسه ها یا توسط دستگاه ورودی/ خروجی نوشته و به دستگاه خوانده می شود. در هر صورت، مدل تولید کننده/ مصرف کننده بکار می رود. پس صف نویسه ها فقط یک بار می تواند خوانده شود، یعنی با خواندن هر نویسه، عملاً آن نویسه نابود می شود. این بر خلاف حافظه پنهان میانگیر است که می‌تواند چندین بار خوانده شود، در نتیجه از مدل خوانندگان/ نویسندگان پیروی می‌کند. UNIX پنج نوع دستگاه را می شناسد.

۱- دیسکها ۲- نوارها ۳- پایانه ها ٤- خطوط ارتباطی ٥- چاپگر

ورودی/ خروجی در WINDOWS NT

مدیریت ورودی/ خروجی در WINDOWS NT مسئول تمام عملیات ورودی/ خروجی برای سیستم عامل است و واسط یکنواختی بوجود می آورد که انواع گرداننده‌ها می توانند آن را فراخوانی کنند.

مولفه های پایه ای ورودی/ خروجی

مدیریت ورودی/ خروجی از چهار مولفه تشکیل شده است.

الف) مدیر حافظه پنهان: مدیر حافظه پنهان برای تمام سیستم ورودی/ خروجی، عمل استفاده از حافظه پنهان را اداره می کند. مدیر حافظه پنهان خدمت استفاده از

حافظه پنهان در داخل حافظه اصلی را برای تمام سیستمهای پرونده ها و اجـزاء شـبکه فراهم می کند. با تغییر مقدار حافظه فیزیکی موجود، مـی تـوان بصـورت پویـا انـدازه حافظه پنهان اختصاص یافته به یک فعالیت را کم یا زیاد نمود.

ب) گردانندە های سیستم پرونده: مدیریت سیستم ورودی/ خروجی همان رفتاری را با گردانندە سیستم پرونده می کند که با هر دستگاه دیگری انجـام مـی دهـد و پیـام برای جلدهای مشخص را به نرم افزار گردانندە مناسب وفق دهندە آن دستگاه هـدایت می‌کند.

پ) گردانندە های شبکه: WINDOWS NT شامل مجموعه قابلیتهای شبکه سازی است و از کاربردهای توزیعی پشتیبانی می کند.

ت) گردانندگان سخت افزار: از طریق نقاط ورود به کتابخانـه هـای پیونـد پویـای مجری WINDOWS NT، این گردانندە ها به ثباتهای سخت افزار دست می یابند.

ورودی/ خروجی همگام و ناهمگام

WINDOWS NT دو حالت ورودی/ خروجی ناهمگام و همگام را ارائه مـی کنـد. حالت ناهمگام در جایی بکار می رود کـه بتـوان کـارایی کـاربرد را بهینـه سـاخت. در ورودی/ خروجی ناهمگام، کار عدد، عمل ورودی/ خروجـی را آغـاز مـی نمایـد و در حالیکه درخواست ورودی/ خروجی تکمیل می شود، به پردازش ادامه می دهد.

در ورودی/ خروجی همگام، تا زمانی کـه عمـل ورودی/خروجـی کامـل شـود، آن کاربرد مسدود است. WINDOWS NT چهار روش مختلـف را بـرای اعـلام خاتمـه ورودی/ خروجی ارائه می کند.

الف) علامت دهی یکسانی دستگاه هسته

ب) علامت دهی یکسانی رویداد هسته

پ) ورودی/ خروجی قابل هشدار

ت) درگاههای تکمیل ورودی/ خروجی

منابع:

سیستمهای عامل اندرو اس تتباوم، آلبرت وودهال

سیستمهای عامل ویلیام استالینگ