فصل آخر

مدیریت I/O

دستگاه‌های I/O از کندترین مؤلفه‌های یک سیستم کامپیوتری به شمار می‌روند، به این ترتیب هرگونه تغییری که باعث بهبود سرعت کار آن‌ها با کامپیوتر گردد، مورد توجه بسیار قرار خواهد گرفت. دراین راستا نقش سیستم‌عامل در I/O، مدیریت و کنترل عملیات دستگاه‌های I/O می‌باشد. چون دستگاه‌های I/O از نظر عملکرد و سرعت متنوع‌اند، برای کنترل آن‌ها از روش‌های متعددی استفاده می‌شود، این روش‌ها زیرسیستم I/O مربوط به هسته را تشکیل می‌دهند که بقیهٔ هسته را از پیچیدگی مدیریت دستگاه‌های I/O تفکیک می‌کند.

سخت‌افزار I/O

هر دستگاه با ارسال و دریافت سیگنال‌هایی، با سیستم کامپیوتری ارتباط برقرار می‌کند. به نقطهٔ اتصال هر دستگاه با کامپیوتر، پورت گفته می‌شود. اگر یک یا چند دستگاه از مجموعهٔ مشترکی از سیم‌ها استفاده کنند به این اتصال باس گفته می‌شود. به عبارت دیگر باس، مجموعه‌ای از سیم‌ها و پروتکل‌های تعریف شده است که ارسال پیام (دیتا) را به همه‌ی دستگاه‌های متصل به آن به صورت همزمان، میسر می‌کند. کنترل کنندهٔ دستگاه، مجموعه‌ای از قطعات الکترونیکی است که می‌توانند بر روی پورت، باس یا دستگاه

عمل کند. هر کنترل کننده برای کنترل، نیاز به ثبات‌هایی دارد. این ثبات‌ها یکی از چهار نوع: وضعیت، کنترلی، ورودی یا خروجی می‌تواند باشد. ثبات‌های وضعیت، حالت فعلی دستگاه را مشخص می‌کنند و ثبات‌های کنترلی برای فرمان دادن به دستگاه مورد استفاده قرار می‌گیرند.

با توجه به اینکه تنوع دستگاه‌های I/O زیاد است، سیستم‌عامل باید کاری کند که دستگاه‌ها برای برنامه‌های کاربردی به شکل استاندارد و یکنواخت باشند. سیستم‌عامل برای رسیدن به چنین هدفی به صورت لایه‌ای عمل می‌کند. لایهٔ اول گردانندهٔ دستگاه[1] است، هدف این لایه، مخفی کردن تفاوت‌های بین کنترل‌کننده‌ها از زیرسیستم I/O هسته می‌باشد. لایهٔ بعدی فراخوان‌های سیستمی I/O هستند. این فراخوان‌ها دستگاه‌ها را در کلاس‌هایی کلی بسته‌بندی می‌کنند تا تفاوت‌های سخت‌افزار را از برنامه‌های کاربردی پنهان نمایند. با این روش لایه‌بندی در صورتی که دستگاه جدیدی به سیستم اضافه شود، کافی است گردانندهٔ آن به لایهٔ گرداننده‌ها اضافه شود و نیازی به تغییر سیستم‌عامل نیست.

وظایف زیرسیستم I/O

به طور کلی می‌توان وظایف زیرسیستم I/O را به صورت زیر دسته‌بندی کرد:

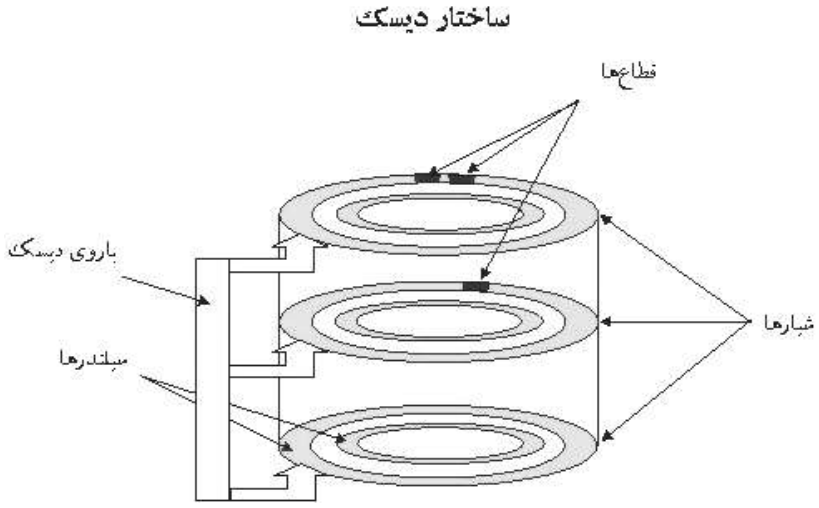
- مدیریت بر فضای نام فایل‌ها و دستگاه‌ها
- کنترل دستیابی به فایل‌ها و دستگاه‌ها
- پیکربندی گرداننده دستگاه‌ها و مقدار دهی اولیه آن‌ها
- کنترل عملیات دستگاه‌ها
- تخصیص فضای سیستم فایل
- تخصیص دستگاه
- میانگیری، حافظهٔ پنهان و اسپولینگ
- نظارت بر وضعیت دستگاه، پردازش خطا و تصحیح آن
- زمانبندی I/O

ساختمان حافظهٔ ثانویه

دیسک‌ها مهمترین حافظهٔ ثانویه در کامپیوترهای مدرن هستند. در شکل زیر، یک دیسک و قسمت‌های مختلف آن نمایش داده شده است. یکی از مسؤلیت‌های سیستم‌عامل، زمانبندی بهینه دسترسی به دیسک است. زمان دستیابی به دیسک شامل سه بخش است: زمان پیگرد: زمان لازم برای حرکت بازوی دیسک،

1 Device Deriver

جهت انتقال هد به سیلندر حاوی قطاع مطلوب (همهٔ هدها با هم حرکت می‌کنند). تأخیر چرخشی: زمان لازم برای چرخش دیسک تا رسیدن قطاع مطلوب به زیر هد (نوک خواندن). زمان انتقال: زمان مورد نیاز برای انتقال بایت‌های خوانده شده به حافظه. در صورتی که سرعت انتقال داده‌ها کمتر از سرعت خواندن بلوک توسط کنترل کننده باشد، آنگاه برای خواندن دو بلوک متوالی، دیسک باید دو دور بزند. (به علت پایین بودن سرعت انتقال، امکان خواندن دو بلوک متوالی وجود ندارد) برای حل این مشکل، شماره‌های بلوک‌ها را یکی در میان قرار می‌دهند تا زمان کافی برای انتقال داده‌ها وجود داشته باشد. به این روش Interleaving گفته می‌شود.



شکل ۱۰- ساختار دیسک

زمانبندی دیسک

زمانبندی FIFO(First In First Out)

ساده‌ترین زمانبندی دیسک است و به تقاضاهای رسیده به ترتیب ورود سرویس می‌دهد. به عنوان مثال اگر هد دیسک ابتدا در سیلندر ۵۳ باشد و صف دیسک شامل درخواست‌های I/O در سیلندرهای

۹۸و۱۸۳و۳۷و۱۲۲و۱۴و۱۲۴و۶۵و۶۷

باشد. زمانبندی دیسک برای رسیدگی به درخواست‌ها به صورت زیر است:

۵۳و۹۸و۱۸۳و۳۷و۱۲۲و۱۴و۱۲۴و۶۵و۶۷

بنابراین برای رسیدگی به درخواست‌ها، ۶۴۰سیلندر را باید طی کند.

زمانبندی SSTF(Shortest Seek Time First)

برای کارایی بهتر نسبت یه الگوریتم FIFO بهتر است ابتدا به درخواست‌های نزدیک به محل فعلی هد رسیدگی شود. الگوریتم SSTF، خدمات را بر حسب کوتاهترین زمان پیگرد (نزدیکترین سیلندر) انجام می‌دهد. به عنوان مثال اگر هد در سیلندر ۵۳ باشد و صف دیسک شامل درخواست‌های زیر باشد:

لیست تقاضاها:۹۸و۱۸۳و۳۷و۱۲۲و۱۴و۱۲۴و۶۵و۶۷

آنگاه نزدیکترین سیلندر، سیلندر ۶۵ است و به همین ترتیب به درخواست‌ها رسیدگی می‌شود.

ترتیب رسیدگی به تقاضاها:۵۳و۶۵و۶۷و۳۷و۱۴و۹۸و۱۲۲و۱۲۴و۱۸۳

با این روش تعداد سیلندرهایی که باید طی شوند به ۲۳۶ سیلندر می‌رسد.

زمانبندی SCAN یا آسانسور

در این الگوریتم بازوی دیسک از اولین سیلندر شروع و به درخواست‌ها پاسخ می‌دهد تا به آخرین سیلندر برسد. سپس از آخرین سیلندر شروع و برمی‌گردد. در موقع برگشت نیز به درخواست‌ها رسیدگی می‌شود. در این الگوریتم برای بدست آوردن ترتیب رسیدگی به تقاضاها، علاوه بر داشتن موقعیت فعلی هد، باید جهت حرکت هد را نیز بدانیم. به عنوان مثال اگر هد در سیلندر ۵۳ باشد و هد به سمت سیلندر صفر در حال حرکت باشد آنگاه ترتیب رسیدگی به درخواست‌های مثال قبلی به صورت زیر است:

ترتیب رسیدگی به تقاضاها:۵۳و۳۷و۱۴و۶۵و۶۷و۹۸و۱۲۲و۱۲۴و۱۸۳

زمانبندی پیمایش حلقوی C_SCAN

در الگوریتم SCAN، زمان انتظار به صورت یکنواخت توزیع نشده است. برای توزیع یکنواخت از الگوریتم حلقوی استفاده می‌شود. در این روش هد از سیلندر صفر شروع می‌کند و به درخواست‌ها رسیدگی می‌کند تا به آخرین سیلندر برسد، سپس فوراً به سیلندر صفر برمی‌گردد و این کار را تکرار می‌کند (در زمان بازگشت به هیچ درخواستی رسیدگی نمی‌کند).

الگوریتم LOOK

در دو الگوریتم SCAN و C_SCAN، بازوی دیسک در کل دیسک حرکت می‌کند ولی در الگوریتم LOOK هد از سیلندر صفر شروع می‌کند ولی حداکثر تا آخرین درخواست پیش می‌رود و بعد از رسیدن به آخرین سیلندر درخواستی، مسیر حرکت را عوض می‌کند و در مسیر برگشتی شروع به رسیدگی می‌کند.

الگوریتم LOOK حلقوی یا C_LOOK

این الگوریتم ترکیبی از دو روش LOOK و C_SCAN است. به این صورت که از سیلندر صفر شروع و تا آخرین سیلندر درخواستی پیش می‌رود، سپس سریعاً به سیلندر صفر برمی‌گردد.

مدیریت فضای مبادله

مدیریت فضای مبادله، وظیفهٔ سطح پایین سیستم‌عامل است. حافظهٔ مجازی از فضای دیسک به عنوان توسعه‌ای از حافظهٔ اصلی استفاده می‌کند. این فضا به روش‌های گوناگون مورد استفاده قرار می‌گیرد که به پیاده‌سازی الگوریتم‌های مدیریت حافظه بستگی دارد. فضای مبادله می‌تواند در دو محل قرار گیرد: در یک فایل بزرگ از سیستم فایل یا در یک پارتیشن جداگانه از دیسک.

پیاده‌سازی فضای مبادله در یک فایل بزرگ، ساده ولی ناکارآمد است. مرور ساختار دایرکتوری و ساختمان دادهٔ دیسک، مستلزم زمان و دستیابی به دیسک است، همچنین تکه تکه شدن خارجی نیز در آن تأثیر منفی دارد.

(Redundant Array Independent Disk)RAID

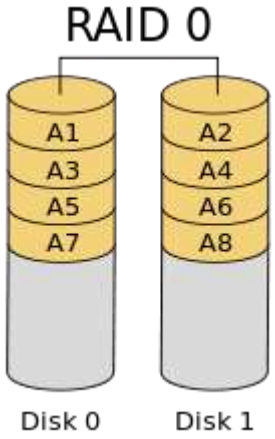
استفاده از چندین دیسک مستقل و موازی باعث افزایش کارایی می‌شود. در چنین سیستمی‌درخواست‌های I/O می‌توانند تا جایی که داده‌های مورد نیاز روی دیسک‌های جداگانه قرار داشته باشند به صورت موازی به اجرا درآیند.

با استفاده از دیسک‌های چندگانه، راه‌های مختلفی برای سازمان‌دهی داده‌ها و اضافه کردن افزونگی برای بهبود قابلیت اطمینان وجود دارد. RAID طرحی است که برای به‌کارگیری دیسک‌های چندگانه استاندارد شده است و شامل هفت سطح می‌باشد. این سطوح مشخص‌کنندهٔ معماری‌های مختلفی هستند که در سه خصوصیت مشترکند:

۱- RAID مجموعه‌ای از گرداننده‌های دیسک فیزیکی است که به عنوان یک گردانندهٔ منطقی واحد از طرف سیستم‌عامل دیده می‌شود.
۲- داده‌ها بر روی این دیسک‌ها توزیع شده‌اند.
۳- ظرفیت افزونگی دیسک، برای ذخیره‌سازی اطلاعات توازن به کار گرفته می‌شود. این اطلاعات قابلیت بازیابی داده‌ها در شرایط شکست دیسک را تضمین می‌نمایند.

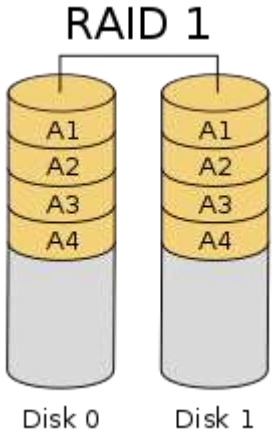
RAID سطح صفر

شامل افزونگی نمی‌باشد و با دسترسی موازی به دیسک‌های چندگانه، کارایی را بالا می‌برد. یعنی در این حالت، رایانه دارای دو دیسک سخت است که هر کدام نصفی از اطلاعات کل را در خود دارند. بدین ترتیب در زمان خوانده شدن دیتا، می‌توان نصفی را از دیسک اول و نصفی را از دیسک دوم به صورت همزمان خواند و سرعت خوانده شدن دیتای حجیم را بالا برد. شکل زیر، تصویری از نحوه‌ی ذخیره‌ی دیتای کل، در دو دیسک است.



RAID سطح یک

در این سطح افزونگی به صورت تکرار می‌باشد. این روش نیازمند دو برابر فضای دیسک عادی می‌باشد. ترمیم خرابی یک دیسک، ساده می‌باشد. در صورتی که بتوان تقاضاهای خواندن را دو نیمه کرد به طوری که هر دو دیسک مشارکت نمایند، آنگاه سرعت خواندن نیز دو برابر می‌شود.

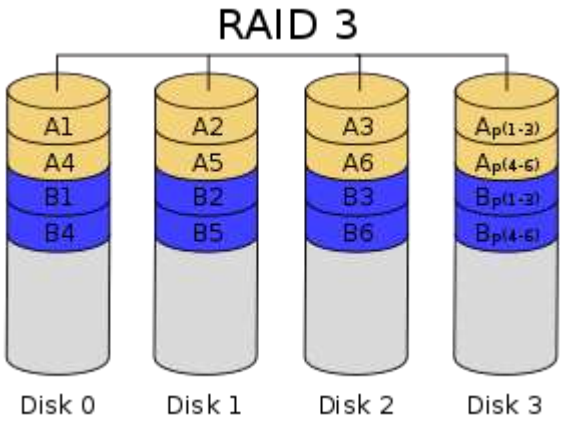


RAID سطح دو

در این سطح از چند دیسک برای افزونگی استفاده می‌شود که در آن‌ها کدهای تصحیح خطا برای هر بایت یا کلمهٔ اطلاعات، ذخیره می‌شود. با استفاده از این افزونگی، می‌توان خطاهای یک بیتی را تصحیح و خطاهای دو بیتی را تشخیص داد. در این سطح با دسترسی موازی به دیسک‌ها، کارایی بالا می‌رود.

RAID سطح سه

شبیه به سطح دو می‌باشد ولی فقط به یک دیسک افزونگی نیاز دارد و به جای یک کد تصحیح خطا از یک بیت توازن ساده استفاده می‌شود. بدین معنی که مثلا در شکل زیر، دیتا بر سه دیسک توزیع می‌شود و دیسک چهارم نقش چک بیت توازن به ازای سه دیسک قبلی را دارد و هرگاه یکی از دیسک‌ها خراب شود، می‌توان با قرار دادن یک دیسک سالم خالی، آن را از روی سه دیسک دیگر بازسازی کرد. پس چنین سیستمی در مقابل خرابی یک دیسک، مقاوم است و عملکرد آن با خرابی یک دیسک، مختل نمی‌شود.

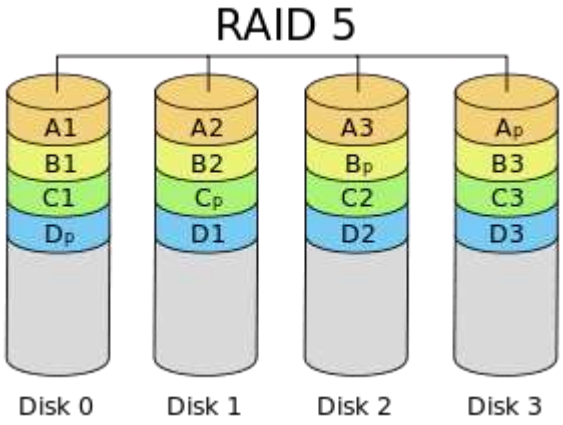


RAID سطح چهار

در این سطح به دیسک‌ها به صورت مستقل می‌توان دسترسی داشت. بنابراین تعدادی درخواست‌های I/O مجزا را می‌توان به صورت موازی اجرا کرد. از یک دیسک مجزا برای افزونگی بیت توازن استفاده می‌شود. در این سطح، چون افزونگی روی یک دیسک است و به ازای نوشتن در هر یک از دیسک‌ها، دیسک توازن را باید بازسازی کرد، بنابراین دیسک توازن به صورت یک گلوگاه می‌شود.

RAID سطح پنج

این سطح، شبیه سطح چهار است با این تفاوت که برای از بین بردن گلوگاه دیسک توازن، اطلاعات توازن را روی همهٔ دیسک‌ها توزیع می‌کنند.



RAID سطح شش

سدر این سطح، افزونگی به دو روش مختلف ساخته می‌شود بنابراین نیاز به دو دیسک اضافی دارد ولی در عوض در صورتی که حتی دو دیسک حاوی داده‌ها دچار شکست شود، امکان بازسازی وجود دارد. در این روش نیز اطلاعات افزونگی روی دیسک‌ها توزیع می‌شود.

نمونه سوال محاسباتی:

۱- دیسکی با ۲۰۰ شیار از شماره ۰ تا ۱۹۹ را درنظربگیرید که هد از شیار ۱۲۵ به شیار ۱۴۳ رسیده و صف تقاضاها به ترتیب FIFO چنین است :

۱۳۰و۱۷۵و۱۰۲و۱۵۰و۹۴و۱۷۷و۹۱و۱۴۷و۸۶ :(از چپ به راست)

a) حرکتهای هد موردنیاز برای رسیدگی به تقاضاها در زمانبندی FCFS را لیست کنید :

143 -> 86 -> 147 -> 91 -> 177 -> 94 -> 150 -> 102 -> 175 -> 130

b) حرکتهای هد موردنیاز برای رسیدگی به تقاضاها در زمانبندی SSTF را لیست کنید :

143 -> 147 -> 150 -> 130 -> 102 -> 94 -> 91 -> 86 -> 175 -> 177

c) حرکتهای هد موردنیاز برای رسیدگی به تقاضاها در زمانبندی LOOK را لیست کنید :

86 -> 91 -> 94 -> 102 -> 130 -> 177 -> 175 -> 150 -> 147 -> 143