# سازمان سما

## وابسته به دانشگاه آزاد اسلامی

## دانشگاه سما واحد حاجی آباد

# سیستم عامل

منبع : سیستم عامل دکتر شیر افکن

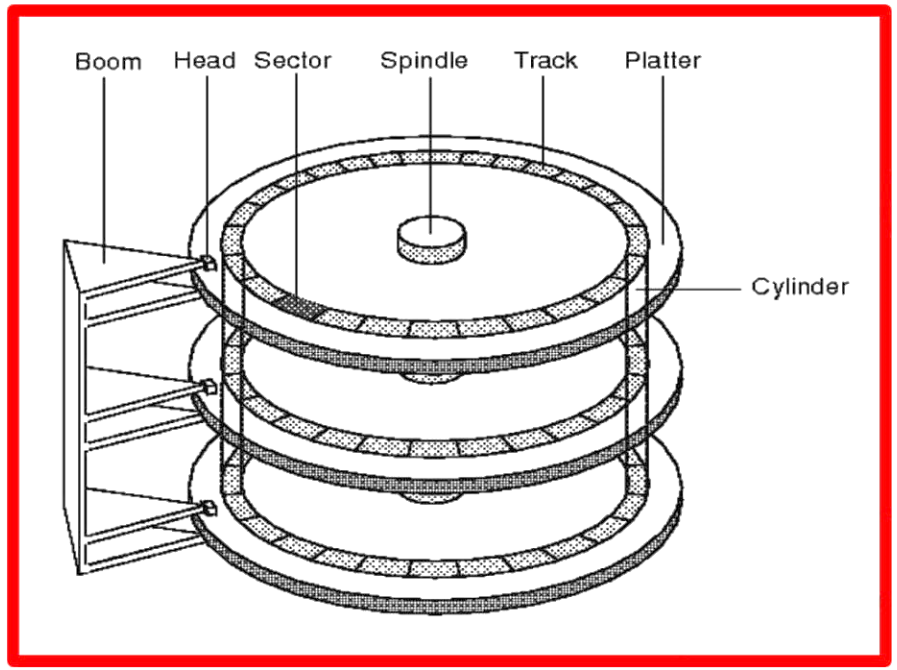
حمیدرضا رضاپور

WWW.HREZAPOUR.IR

# مدیریت دیسک

یکی از وظایف سیستم عامل، مدیریت دیسک می باشد.

# دیسک مغناطیسی




رسانه ای گردان با امکان دستیابی مستقیم به داده های ذخیره شده می باشد.

دیسک از صفحه ای مدور و مغناطیس شونده که حول یک محور عمودی می چرخد، تشکیل شده است. رویه‌های این صفحه از غشاء فرو مغناطیسی پوشیده شده که بر روی آنها شیارهایی به صورت دایره‌های متحدالمرکز وجود دارد.

شیارها از بیرون به درون با شروع از صفر شماره گذاری شده اند.

تمام شیارهای هم شعاع، تشکیل یک استوانه را می‌دهند.

# زمان استوانه جویی (Seek time)

زمانی که طول می کشد تا نوک خواندن/نوشتن به استوانه ای که داده مورد نظر در آن قرار دارد برسد.

متوسط این زمان را با S نمایش می‌دهند و واحد آن میلی ثانیه است.

# زمان درنگ دوران ( Rotational latency time )

زمانی که طول می کشد تا ابتدای داده مورد نظر در اثر دوران دیسک به زیر نوک R /W برسد. که واحد آن میلی ثانیه است.

زمان یک دور کامل چرخش دیسک (2r) :

$$2r = \frac{60000}{rpm}$$

rpm : سرعت چرخش دیسک (دور در دقیقه)

متوسط زمان درنگ دورانی :

$$r = \frac{30000}{rpm}$$

زمان دستیابی به دیسک = زمان استوانه جویی + زمان درنگ دورانی + زمان انتقال

# الگوریتم های

# زمان بندی بازوی دیسک

# الگوریتم های زمان بندی بازوی دیسک

## ۱- خروج به ترتیب ورود FCFS (First Come First Serviced)

در خواست ها به ترتیب ورود به صف، اجرا می‌شوند.

## ۲- ابتدا کوتاهترین زمان جستجو SSTF (Shortest Seek Time First)

بازو به سمت درخواستی حرکت می‌کند که نزدیک ترین درخواست بعدی به مکان فعلی باشد.

## ۳- مرور(آسانسور ) :Scan

در ابتدا بازو به جهتی حرکت می‌کند که کوتاهترین زمان استوانه جویی را برای دستیابی نیاز دارد.

اگر در جهت انتخاب شده به همه درخواستها پاسخ داده شد، جهت حرکت عوض می‌شود.

نام دیگر روش SCAN ، روش LOOK می باشد.

## ۴- مرور مدور: C – SCAN

مانند روش Scan است، با این تفاوت که پس از پاسخ به آخرین درخواست در یک جهت(مثلا رو به بالا)، بازو بلافاصله به پایین ترین شماره سیلندر رفته و به سمت بالا حرکت می کند.

# مثال

صف درخواستهای سیلندر به صورت ۱۴، ۲۰، ۹، ۵، ۱۲ می‌باشد و هد بر روی سیلندر ۱۰ قرار دارد.

**FCFS : 10 , 12 , 5 , 9 , 20 , 14**

**SSTF : 10 , 9 , 12 , 14 , 20 , 5**

**SCAN : 10 , 9 , 5 , 12 , 14 , 20**

**C-SCAN : 10 , 9 , 5 , 20 , 14 , 12**

# مثال

در یک دیسک سخت، نوک I/O HEAD روی سیلندر۲۰ قرار دارد. اگر تقاضا برای خواندن سیلندرهای به ترتیب ۱۰، ۲۲، ۲۰، ۲، ۴۰، ۶ و ۳۸ به Driver آن وارد شود و چنانچه حرکت هد I/O بین دو سیلندر مجاور ۶ میلی ثانیه طول بکشد، در صورت استفاده از الگوریتم SSTF برای خواندن سیلندرها، کل seek time مورد نیاز چقدر خواهد بود؟

$$20 \rightarrow 22 \rightarrow 10 \rightarrow 6 \rightarrow 2 \rightarrow 38 \rightarrow 40$$

$$2 + 12 + 4 + 4 + 36 + 2 = 60$$

چون هر حرکت ۶ میلی ثانیه طول می‌کشد، پس در کل ۶ × ۶۰ میلی ثانیه طول خواهد کشید.

# مثال

یک دیسک را در نظر بگیرید که شامل ۱۰۰ سیلندر است(۰ تا ۹۹) .

زمان لازم برای عبور هد از یک سیلندر به سیلندر مجاور یک واحد زمانی است.

در زمان صفر هد بر روی سیلندر صفر است و درخواستی از گذشته وجود ندارد.

شش درخواست در زمان های مختلف مطابق جدول زیر وارد می شوند.

در زمان حرکت هد به سمت یک سیلندر، ورود درخواست جدید تاثیری بر حرکت ندارد.

ترتیب اجرای درخواست ها برای الگوریتم SCAN (آسانسور) چیست؟

| زمان ورود درخواست | 0 | 10 | 20 | 70 | 80 | 90 |
|---|---|---|---|---|---|---|
| سیلندر درخواست شده | 21 | 75 | 16 | 68 | 2 | 17 |

$$0 \rightarrow 21 \rightarrow 75 \rightarrow 68 \rightarrow 16 \rightarrow 2 \rightarrow 17$$

در لحظه صفر هد بر روی سیلندر صفر قرار دارد و تنها درخواست موجود در این لحظه، درخواست برای سیلندر ۲۱ می باشد. در ثانیه ۲۱، درخواست برای سیلندر ۷۵و۱۶ رسیده، که بر طبق الگوریتم **scan** ، هد به سمت سیلندر ۷۵ می رود .

زمانی که هد بر روی سیلندر ۷۵ قرار دارد، درخواست برای سیلندرهای ۶۸ و ۱۶ وجود دارد که هد به سیلندر ۶۸ می رود .

سپس به سیلندر ۱۶ می رود. زمانی که بر روی سیلندر ۱۶ قرار دارد، درخواست سیلندر ۱۷ نرسیده است، بنابراین هد به سیلندر ۲ می رود. در نهایت هد به سیلندر ۱۷ می رود.

| زمان ورود درخواست | 0 | 10 | 20 | 70 | 80 | 90 |
|---|---|---|---|---|---|---|
| سیلندر درخواست شده | 21 | 75 | 16 | 68 | 2 | 17 |

- ✓ عدالت فقط در روش FCFS رعایت می شود.
- ✓ روشهای FCFS ، Scan و C-Scan ، بدون قحطی هستند.
- ✓ کارایی (میانگین زمان جستجو) در FCFS پایین است.

# روش های تخصیص

# فضای دیسک به فایل

# تخصیص فضای دیسک به فایل

فضای دیسک به سه روش زیر می تواند به فایل تخصیص داده شود.

۱- پیوسته

۲- پیوندی

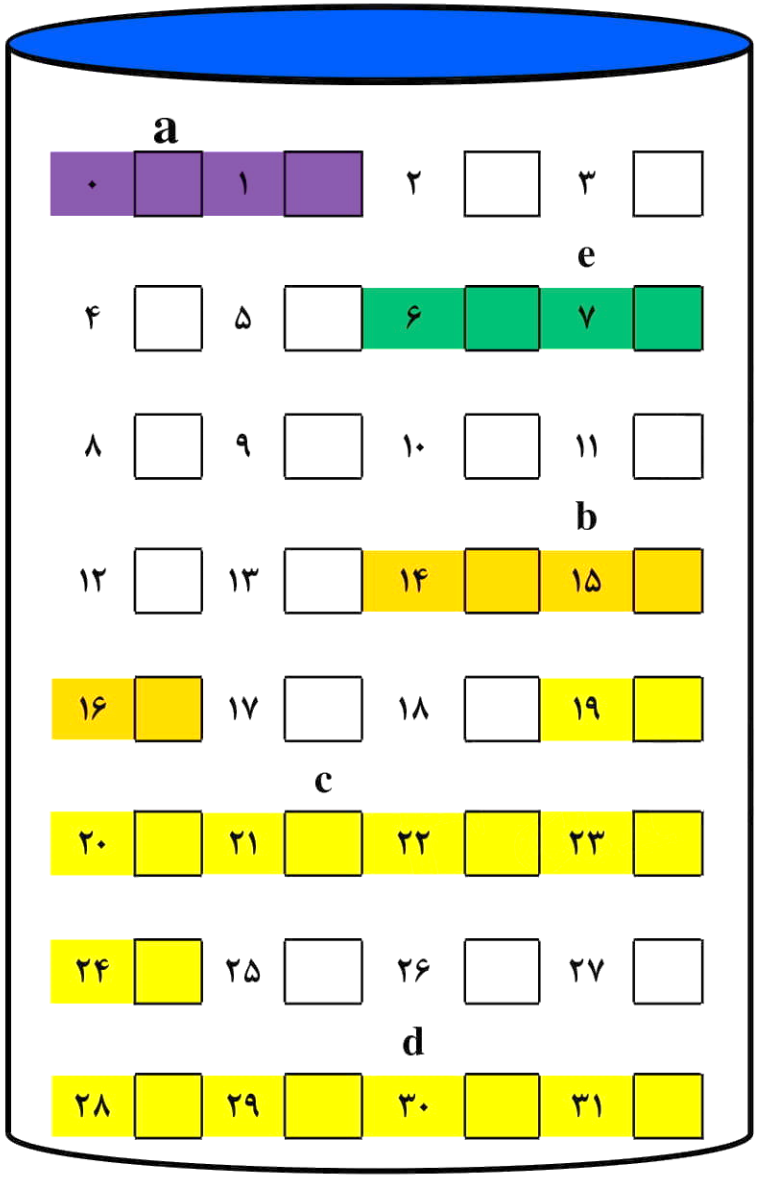
۳- شاخصی

هدف روش های مختلف، رسیدن به حالتی است که هم از فضای دیسک به خوبی استفاده شود و هم دستیابی به فایل به سرعت صورت گیرد.

# تخصیص پیوسته

در این روش هر فایل تعدادی بلاک پیوسته روی دیسک را اشغال می کند و کافی است که شماره بلاک اول روی دیسک و تعداد بلاک های فایل را ذخیره کرد.

# تخصیص پیوسته :




دایرکتوری

| فایل | شروع | طول |
|---|---|---|
| a | 0 | 2 |
| b | 14 | 3 |
| c | 19 | 6 |
| d | 28 | 4 |
| e | 6 | 2 |

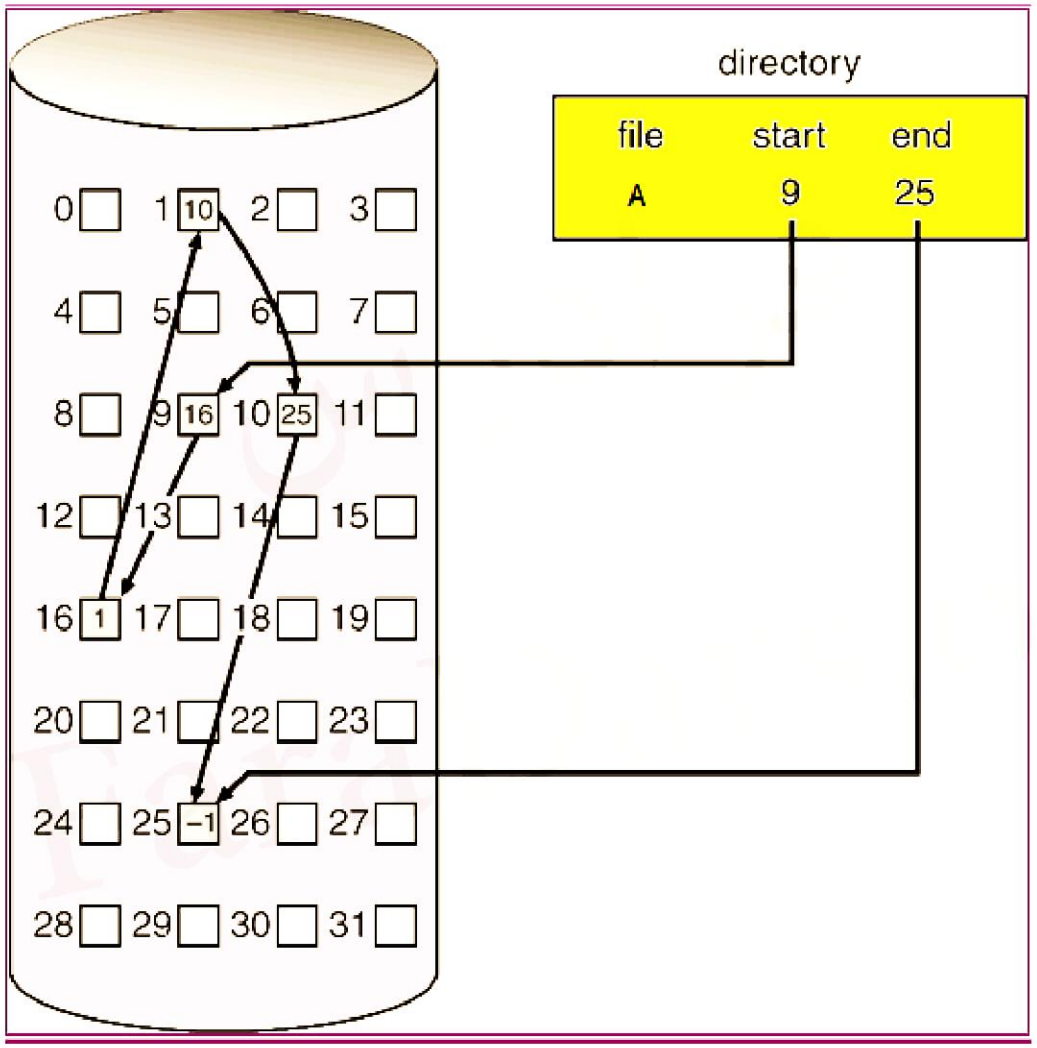
# تخصیص پیوندی

در این روش هر فایل، یک لیست پیوندی از بلاک های روی دیسک است.

بلاک ها ممکن است در هر کجای دیسک پراکنده باشند.

در فهرست راهنما برای هر فایل اشاره گری به اولین بلاک فایل قرار دارد.

# مثال از تخصیص پیوندی



# تخصیص شاخصی

اشاره گرها به بلاکهای فایل روی دیسک در یک مکان(بلاک شاخص) ، جمع آوری می شوند.

هر فایل دارای بلاک شاخص خود است که یک ماتریس از آدرسهای بلاکها است.

ورودی I ام در بلاک شاخص به بلاک I ام فایل اشاره می کند.

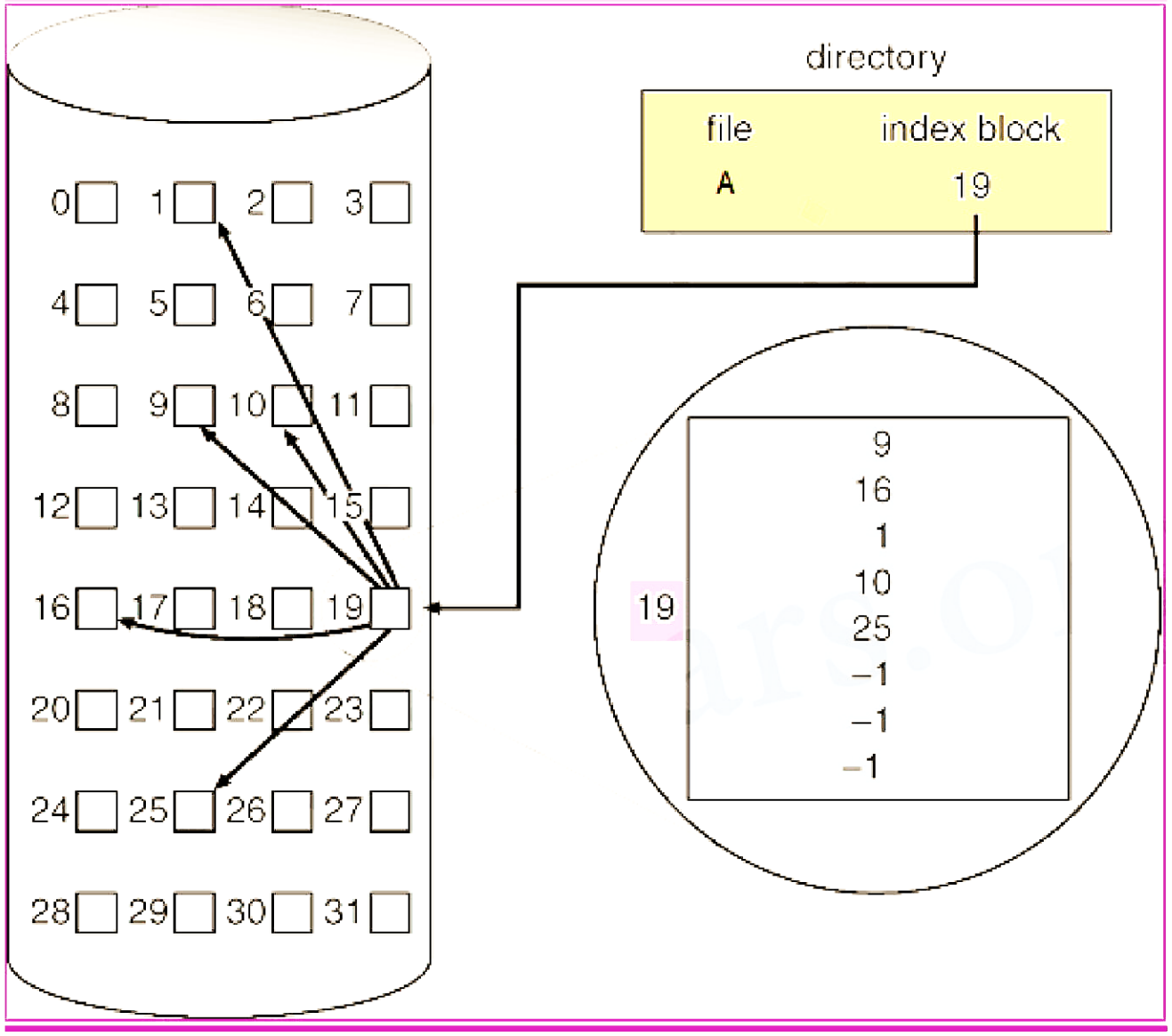
در فهرست راهنما تنها آدرس بلاک شاخص حفظ می شود.

در این روش براحتی می توان دستیابی مستقیم را حمایت کرد.

البته فضای بلاک شاخص تلف می شود و در بسیاری از مواقع به تمامی بلاک شاخص نیاز نمی باشد.

علت پر طرفدار بودن روش شاخصی، رابطه نزدیک آن با مدیریت حافظه قطعه بندی–صفحه بندی شده است. بلاک شاخص می تواند یک جدول صفحه باشد و بلاک های فایل همان صفحات فایل هستند.

# مثال از تخصیص شاخصی



# مثال

فایلی دارای ۵ بلوک از شماره ۱ تا ۵ می باشد.

می خواهیم بلوک شماره ۴ را حذف کنیم.

تعداد کل نقل و انتقال دیسک در سه حالت تخصیص دیسک به فایل را مشخص کنید.

(در ابتدا فایل باز است.)

**الف- پیوسته :** چون همه بلوک های فایل به صورت پشت سر هم قرار دارند، بلوک ۵ خوانده شده و بر روی بلوک ۴ نوشته می شود. بنابراین به ۲ دسترسی دیسک نیاز است.

**ب- پیوندی :** چون هر بلوک فایل حاوی اشاره گری است که آدرس بلوک بعدی را مشخص می کند، از بلوک ۱ تا ۴ خوانده تا آدرس بلوک ۵ را به دست آوریم. سپس این آدرس را جایگزین آدرس بلوک ۳ می کنیم. بنابراین به ۵ دسترسی دیسک نیاز است.

**ج- شاخصی(اندیسی) :** چون آدرس همه بلوک ها در یک جدول شاخص بر روی دیسک ذخیره شده، بلوک حاوی اندیس ها را خوانده و آدرس بلوک ۴ را حذف کرده و بعد از به روز رسانی، بر روی دیسک باز نویسی می کنیم. بنابراین به ۲ دسترسی دیسک نیاز دارد.

# مشکلات تخصیص پیوسته

۱- یافتن فضای خالی برای یک فایل جدید.

(برای یک فایل n بلاکی باید به دنبال n بلاک آزاد پشت سرهم گشت. )

۲- تعیین مقدار فضای مورد نیاز یک فایل

# مشکلات تخصیص پیوندی

## ۱- عدم حمایت از دستیابی مستقیم

این روش فقط در رابطه با دستیابی ترتیبی خوب عمل می کند. زیرا برای رسیدن به بلاک i ، باید بلاکهای قبل از آن دستیابی شوند که هر کدام از اینها به یک خواندن از دیسک نیاز دارد.

## ۲- اتلاف فضا توسط پیوند ها

## ۳- عدم قابلیت اطمینان سیستم

با از بین رفتن تنها یکی از اشاره گرها، صدمات جدی به فایل و فضای روی دیسک وارد می گردد.