

۱- **سکتور** کوچک‌ترین واحد قابل آدرس‌دهی روی یک دیسک سخت یا دیسک SSD است. هر سکتور معمولاً حجمی معادل **512 بایت** یا در دیسک‌های جدیدتر، **4096 بایت (۴ کیلوبایت)** دارد.

در خروجی دستور **fdisk**، ستون **"sectors"** نشان می‌دهد که هر پارتیشن از کدام سکتور شروع می‌شود و در کدام سکتور به پایان می‌رسد. به عبارت دیگر، این ستون موقعیت مکانی و اندازه پارتیشن‌ها را بر اساس تعداد سکتورها نمایش می‌دهد.

-۲

1. Linux Desktop Systems (سیستم‌های دسکتاپ لینوکس):

پیشنهاد پارتیشن‌بندی:


- /: روت اصلی سیستم حداقل ۲۰GB
- home/: برای فایل‌های کاربر نسبت به نیاز، مثلاً ۱۰۰GB یا بیشتر
- swap: بسته به میزان RAM معمولاً نصف تا دو برابر RAM
- boot/: اگر نیاز به بوت خاص یا dual boot باشد حدود ۱GB

2. Linux Servers for Databases or Web Services with Extensive Logging (سرورهای پایگاه‌داده یا (

(سرورهای وب با لاگ‌گیری گسترده):

پیشنهاد پارتیشن‌بندی:


- /: برای فایل‌های سیستمی حداقل ۱۵-۲۰GB
- var/: برای لاگ‌ها و فایل‌های پایگاه‌داده بسیار مهم، بسته به نوع سیستم شاید ۱۰۰GB یا بیشتر
- tmp/: برای فایل‌های موقتی مثلاً ۵-۱۰GB
- swap: بسته به RAM معمولاً برابر یا نصف RAM
- Home/: اگر کاربران مدیریتی خاصی داشته باشد (اختیاری)

 **دلیل:** دایرکتوری **var** محلی است که لاگ‌ها، کش‌ها و دیتابیس‌ها ذخیره می‌شوند. اگر با / یکی باشد، ممکن است کل فضای سیستم پر شود و سیستم از کار بیفتد.

3. Linux Servers in University Labs or Staging Environments (سرورهای آزمایشگاهی با کاربران)

(متعدد):

پیشنهاد پارتیشن‌بندی:

- / : سیستم عامل حدود 20GB
 - / home : برای دایرکتوری‌های شخصی کاربران (بسیار مهم، بسته به تعداد کاربران)
 - swap : مطابق با RAM
 - / var : اگر لاگ‌گیری زیاد است یا سرویس‌هایی مثل mail یا log در حال اجراست
-  **دلیل:** کاربران نیاز به فضای ذخیره‌سازی جداگانه دارند، بنابراین داشتن / home مجزا برای کنترل دسترسی و پشتیبان‌گیری اهمیت دارد.

۳-

نحوه زمان‌بندی پردازش‌ها روی CPU :

- سیستم عامل با استفاده از الگوریتم‌های زمان‌بندی مثل Round Robin یا Priority Scheduling تصمیم می‌گیرد که هر پردازش چه زمانی و به چه مدت روی CPU اجرا شود.
- هر پردازش فقط برای بازه‌های کوتاه روی CPU قرار می‌گیرد، به همین دلیل چند پردازش می‌توانند به‌نظر هم‌زمان اجرا شوند در واقع CPU بین آن‌ها سوئیچ می‌کند.

نحوه بارگذاری داده‌ی پردازش‌ها در RAM :

- هنگام اجرای یک برنامه، بخش‌هایی از آن (کد، داده‌ها و استک) به صورت page به RAM منتقل می‌شوند.
- سیستم عامل فقط بخش‌هایی از برنامه را که فعلاً مورد نیاز هستند بارگذاری می‌کند (نه کل آن).

وقتی RAM کافی نباشد:

- سیستم عامل از بخشی از دیسک به عنوان حافظه‌ی موقت (swap space) استفاده می‌کند.
- داده‌های بلااستفاده یا قدیمی از RAM به این فضای swap منتقل می‌شوند تا برای پردازش جدید جا باز شود.

استفاده از دیسک و paging و swapping :

- **Paging:** انتقال صفحات حافظه بین RAM و دیسک فقط در صورت نیاز.
- **Swapping:** انتقال کامل یک پردازش (یا بخش بزرگی از آن) از RAM به دیسک یا بالعکس.
- این فرایندها زمانی انجام می‌شود که سیستم تحت فشار حافظه باشد.

استراتژی‌های مدیریت حافظه Demand Paging :

- فقط زمانی یک page از حافظه به RAM منتقل می‌شود که واقعاً به آن نیاز باشد (اولین دسترسی).
- این روش باعث صرفه‌جویی در RAM و افزایش کارایی سیستم می‌شود.

آیا همه‌ی صفحات یک پردازش باید در RAM باشند؟

- **خیر**، فقط صفحاتی که در لحظه مورد نیاز هستند باید در RAM باشند.
- این امکان اجرای برنامه‌های بزرگ‌تر از حافظه‌ی فیزیکی را فراهم می‌کند.

-۴

Translation Lookaside Buffer (TLB) یک حافظه کش کوچک و بسیار سریع در داخل CPU است که برای افزایش سرعت ترجمه آدرس‌های مجازی به آدرس‌های فیزیکی در سیستم‌های دارای حافظه مجازی استفاده می‌شود. در زمان اجرای یک برنامه، سیستم‌عامل از جدول صفحات برای نگاشت آدرس‌های مجازی به فیزیکی استفاده می‌کند که این فرآیند ممکن است زمان‌بر باشد. TLB با نگهداری نتایج آخرین ترجمه‌های انجام‌شده، در صورت تکرار درخواست به همان آدرس، از مراجعه مجدد به جدول صفحات جلوگیری کرده و باعث افزایش سرعت اجرای برنامه می‌شود. در صورتی که آدرس مورد نظر در TLB موجود نباشد (TLB miss)، سیستم مجبور به مراجعه به جدول صفحه در RAM می‌شود که کندتر است. بنابراین، TLB نقش مهمی در بهینه‌سازی عملکرد سیستم و کاهش بار روی حافظه اصلی دارد.

-۵

یک **page** صفحه واحد پایه‌ای حافظه در سیستم‌های دارای مدیریت حافظه مجازی است و معمولاً اندازه‌ای ثابت دارد (مانند ۴ کیلوبایت). یک **virtual page** صفحه مجازی به صفحه‌ای در فضای آدرس‌دهی مجازی اشاره دارد که ممکن است در RAM یا در فضای **swap** (دیسک) قرار داشته باشد. هنگام اجرای یک پردازش، سیستم‌عامل آدرس‌های مجازی آن را به آدرس‌های فیزیکی متناظر نگاشت می‌کند و داده‌ها را به صورت **page** بارگذاری یا جایگزین می‌کند.

Context switch

تغییر زمینه زمانی رخ می‌دهد که CPU از اجرای یک پردازش به پردازش دیگری تغییر می‌کند و باید وضعیت (context) فعلی را ذخیره و وضعیت پردازش جدید را بازیابی کند. افزایش فضای **swap** می‌تواند منجر به کند شدن **context switch** شود، زیرا در

صورتی که داده‌ها یا صفحات مورد نیاز پردازنده در RAM نباشند و مجبور به خواندن آن‌ها از دیسک شوند، تأخیر بالایی ایجاد می‌شود. همچنین اندازه **page** نیز در این فرآیند مؤثر است: صفحات بزرگ‌تر باعث کاهش تعداد **context switch** های وابسته به صفحه (**page faults**) می‌شوند اما می‌توانند باعث هدررفت حافظه (**internal fragmentation**) شوند، در حالی که صفحات کوچک‌تر مدیریت حافظه دقیق‌تری را ممکن می‌سازند ولی ممکن است موجب افزایش تعداد **page faults** و تأخیر در عملکرد شوند.

-۶-

Huge page به صفحات حافظه‌ای با اندازه بزرگ‌تر از اندازه پیش‌فرض مثلاً ۲ MB یا حتی ۱ GB به جای ۴ KB گفته می‌شود که توسط سیستم‌عامل و CPU پشتیبانی می‌شود. هدف اصلی استفاده از **huge pages** کاهش سرشار ناشی از مدیریت تعداد زیاد صفحات کوچک در حافظه است. زمانی که برنامه‌ها یا پایگاه‌های داده‌ی بزرگ نیاز به اختصاص حجم زیادی از حافظه دارند، استفاده از **huge pages** باعث کاهش تعداد ورودی‌های موجود در **Translation Lookaside Buffer (TLB)** می‌شود و در نتیجه سرعت دسترسی به حافظه افزایش می‌یابد. همچنین، این صفحات بزرگ باعث کاهش میزان **page faults** و بهبود عملکرد کلی سیستم در بارهای کاری سنگین مانند سیستم‌های مجازی‌سازی، پایگاه‌های داده بزرگ مثل **Oracle** یا **PostgreSQL** و اپلیکیشن‌هایی که نیاز به حافظه پیوسته و زیاد دارند، می‌شوند. به طور کلی، **huge pages** در سناریوهایی کاربرد دارند که مصرف حافظه زیاد و دسترسی‌های متناوب به داده وجود دارد و عملکرد سیستم حیاتی است.

-۷-

Memory fragmentation یا تکه‌تکه شدن حافظه در RAM به حالتی گفته می‌شود که فضای حافظه به بخش‌های غیرپیوسته تقسیم شده و در نتیجه، اختصاص دادن یک بلوک حافظه‌ی بزرگ به یک برنامه یا فرآیند جدید دشوار یا غیرممکن می‌شود، حتی اگر مجموع فضای آزاد کافی وجود داشته باشد. این پدیده معمولاً در سیستم‌هایی رخ می‌دهد که به‌طور مکرر حافظه را آزاد و تخصیص می‌دهند، مانند سیستم‌های چندوظیفه‌ای یا سرورهایی با بارکاری پویا **Fragmentation**. می‌تواند به دو صورت باشد: **external fragmentation** که در آن بلوک‌های آزاد در نقاط مختلف حافظه پخش شده‌اند و نمی‌توان یک بلوک پیوسته بزرگ ایجاد کرد، و **internal fragmentation** که در آن بخشی از حافظه اختصاص داده شده بلااستفاده باقی می‌ماند. نتیجه این مشکل می‌تواند کاهش کارایی سیستم، افزایش زمان پاسخ‌دهی، و حتی ناتوانی در اجرای برنامه‌های جدید باشد، به‌خصوص در برنامه‌هایی که به حافظه پیوسته نیاز دارند. در سیستم‌های حیاتی، این موضوع ممکن است منجر به **crash** یا توقف اجرای سرویس‌ها شود.