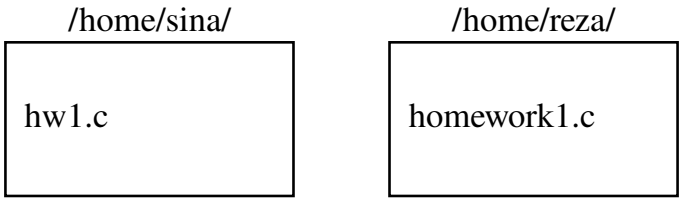
یادداشت‌های درس سیستم‌های عامل - بخش دوم

در جلسه‌های گذشته دلیل اهمیت سیستم عامل و لزوم وجود آن را مطالعه کردیم. با چهار وظیفه‌ی مهم سیستم عامل آشنا شدیم. دیدیم چگونه پردازنده در چند حالت اجرا می‌شود تا دسترسی پردازه‌ها به سخت‌افزار را محدود کنند و چگونه پردازه‌ها با کمک فراخوانی‌های سیستمی از سیستم عامل درخواست می‌کنند. با سخت‌افزار کامپیوتر، شیوه‌ی بررسی رخدادها و روال راه‌اندازی سیستم عامل آشنا شدیم. روش‌های انتقال اطلاعات بین دستگاه‌های ورودی و خروجی و حافظه‌ی اصلی را مرور کردیم.

در ادامه با برخی از وظایف سیستم عامل به صورت دقیق‌تر آشنا خواهیم شد.

حفاظت (Protection) یعنی مدیریت دسترسی کاربران و پردازه‌ها در یک سیستم عامل.



- اهمیت در گذشته: در کامپیوترهایی که چند صد یا چند هزار کاربر داشتند
- اهمیت در کامپیوترهای تک‌کاربره‌ی امروزی

دو هدف در حفاظت

در سیستم عامل شما به دنبال مدیریت دسترسی پردازه‌ها هستید تا

الف) از دسترسی‌های غیر مجاز جلوگیری کنید و

ب) از اختلال به دلیل اشکال یک پردازه جلوگیری کنید.

- فرض کنید یک برنامه‌ی ساده مثل یک بازی را از اینترنت دریافت کرده‌اید و آن را اجرا می‌کنید. اطلاعات شخصی شما (برای مثال اطلاعات کارت بانکی که در فایلی ذخیره شده است) نباید در اختیار این برنامه قرار گیرد تا از راه شبکه برای یک مقصد ارسال شود. اگر این برنامه دچار مشکل شود، نباید به اطلاعات شما یا سایر پردازه‌های در حال اجرا آسیب برساند (برای مثال فایل‌های فایل سیستم را حذف کند).

حفاظت در Unix - کاربران

بسیاری از سیستم‌های عامل امروزی از روش حافظت فایل‌ها در سیستم عامل Unix استفاده می‌کنند (در کنار سایر روش‌ها).

- به هر کاربر در سیستم عامل Unix یک شناسه تخصیص می‌یابد (User ID).
- تعدادی گروه کاربری نیز تعریف شده‌اند و هر گروه نیز شناسه‌ی خاص خودش را دارد (Group ID). هر کاربر می‌تواند عضو تعدادی از این گروه‌ها باشد.

User	User ID	Groups
reza	101	students, audio, video, disk, cdrom
omid	102	students
sina	103	students, disk, cdrom

- هر فایل در فایل‌سیستم یک صاحب دارد و به یک گروه تعلق دارد.
- صاحب فایل دسترسی افراد به آن فایل را مشخص می‌کند. این دسترسی برای سه دسته انجام می‌شود: صاحب فایل، افراد عضو گروه و سایرین.
- صاحب فایل برای هر یک از این سه دسته مشخص می‌کند که اجازه‌ی خواندن فایل، اجازه‌ی نوشتن فایل و اجازه‌ی اجرای فایل را دارند یا خیر. این دسترسی‌ها به صورت یک عدد نه بیتی که به آن Access Mode می‌گویند نمایش داده می‌شود.
- هر پردازه‌ای در Unix متعلق به یک کاربر است و فقط فایل‌هایی را می‌تواند بخواند که به آن اجازه‌ی دسترسی دارد.

در مثال زیر، فایل abc.txt متعلق به کاربر sina و گروه students است.

- کاربر sina (و پردازه‌های متعلق به او) می‌توانند فایل را بخوانند، به آن بنویسند و آن را اجرا کنند.
- افراد عضو گروه students فقط می‌توانند فایل را بخوانند.
- سایرین نمی‌توانند هیچ یک از این سه عمل را روی این فایل انجام دهند.

File	**/home/sina/abc.txt**
Owner	sina
Group	students
Access Mode	owner: rwx, group: r--, others: ---

- دسترسی یک فایل معمولا به صورت یک عدد سه رقمی در مبنای هشت بیان می‌شود، مثل 740.

سایر روش‌ها

روش‌های متنوع دیگری برای حفاظت در سیستم‌های عامل استفاده می‌شود.

- بسیاری از سیستم عامل‌های کاربران یا گروه‌های ویژه‌ای برای دسترسی به سخت‌افزار (مثلا پخش موسیقی) یا اجرای فایل‌های خارجی دارند.
- یکی از این روش‌های قدیمی Chroot Jail هست که قسمت قابل دسترسی فایل سیستم را برای یک پردازه محدود می‌کند. در واقع پردازه فقط به دسترسی به محتویات یک شاخه (مثل یک زندان) محدود می‌شود.
- اصطلاحا به اجرای پردازه‌ها به صورتی که دسترسی آنها به کل سیستم محدود شود Sandbox می‌گویند. ماشین‌ها مجازی یکی از این روش‌ها هستند که در قسمت‌های بعدی درس مطالعه می‌شوند.

امنیت (Security) یعنی مقابله با حمله‌های داخلی و خارجی به سیستم عامل.

- مثال‌هایی از این حمله‌ها، ویروس‌ها، کرم‌ها (Worms) و حمله‌های DoS (Denial of Service) هستند.
- در حمله‌های DoS درخواست‌های غیر واقعی زیادی به سیستم عامل فرستاده می‌شود به طوری که سیستم عامل نتواند به درخواست‌های واقعی پاسخ دهد.

فایل سیستم شیوه‌ی ذخیره‌سازی و بازیابی اطلاعات در دستگاه‌های ذخیره‌سازی را مشخص می‌کند.

- در بیشتر سیستم‌های عامل، اطلاعات کاربر در یک ساختار درختی ذخیره می‌شود (درخت فایل سیستم).
- در این درخت، رأس‌ها شاخه‌ها و فایل‌ها هستند و هر شاخه می‌تواند تعدادی شاخه و فایل را به عنوان فرزند نگه داشته باشد.
- شما با مفهوم درخت فایل سیستم آشنا هستند و به کمک آن اطلاعاتتان را در یک سیستم عامل به سادگی مدیریت می‌کنید.

سیستم عامل این رابط سطح بالای مبتنی بر درخت را در اختیار کاربران قرار می‌دهد و جزئیات ذخیره‌سازی اطلاعات را پنهان می‌کند (به پنهان کردن جزئیات پیاده‌سازی اصطلاحا انتزاع یا Abstraction می‌گویند).

- فایل سیستم مشخص می‌کند این درخت چگونه در یک دستگاه ذخیره‌سازی اطلاعات مثل دیسک قرار می‌گیرد.
- فایل سیستم‌های متنوعی موجود هستند، مثل fat، ext4 یا btrfs.

همه‌ی عملیاتی که روی فایل‌ها انجام می‌شود توسط فراخوانی‌های سیستمی پیاده‌سازی می‌گردند و پیاده‌سازی فراخوانی‌های سیستمی کاملا در اختیار سیستم عامل قرار دارد. بنابراین، سیستم عامل این توانایی را دارد که فایل‌هایی را در اختیار کاربر قرار دهد که وجود خارجی ندارند و در دیسک ذخیره نمی‌شود.

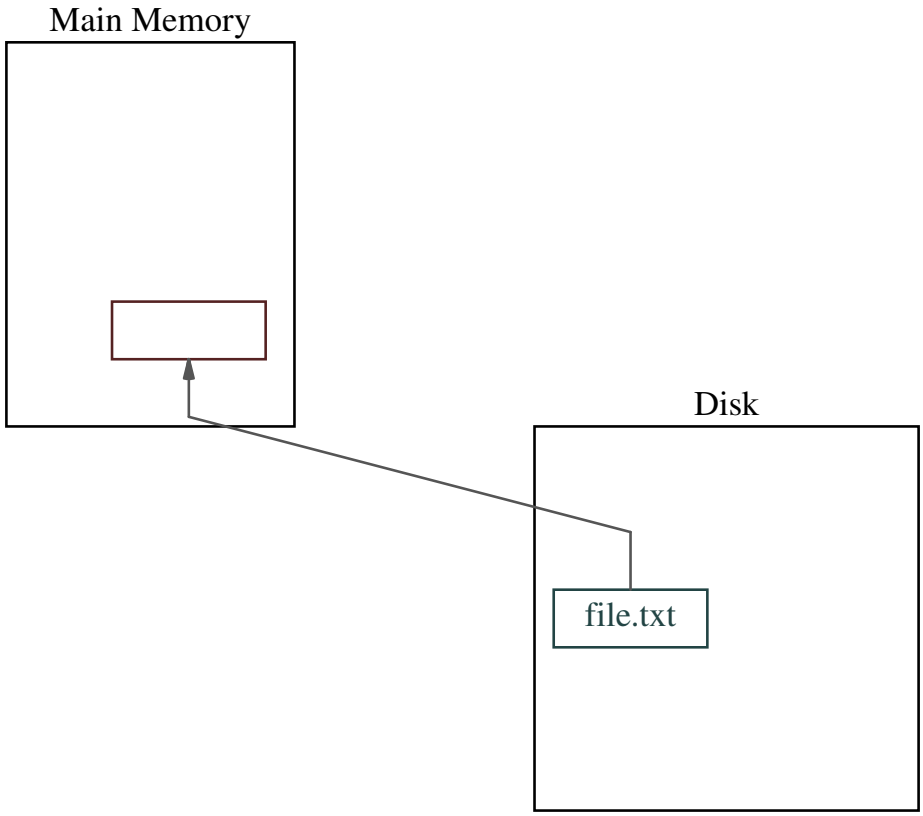
```
/dev/null
/dev/full
/dev/empty
/dev/zero
/dev/random
```

- مثال‌های زیادی از این فایل‌ها وجود دارند. برای نمونه در بسیاری از سیستم‌های عامل مشابه Unix، در شاخه‌ی /dev/ فایل‌هایی وجود دارند که برای دستگاه‌های ورودی و خروجی مثل کارت صدا، کارت گرافیک و چاپگر هستند. اگر روی این فایل‌های چیزی نوشته شود در دیسک ذخیره نمی‌شود و به جای آن به دستگاه ورودی و خروجی فرستاده می‌شود (مثلا صدایی از کارت صدا تولید می‌گردد).
- در /dev/ فایل‌های غیر واقعی دیگری مثل فایل /dev/null (که هر چه به آن نوشته شود دور انداخته می‌شود)، فایل /dev/empty (که همیشه خالی است)، فایل /dev/full (که همیشه پر است)، فایل /dev/zero (رشته‌ای بی پایان از بایت‌های صفر از آن خوانده می‌شود) و فایل /dev/random (که محتویات آن یک رشته‌ی تصادفی است).
- همچنین در شاخه‌ی /proc/، به ازای هر پردازه‌ی موجود در سیستم عامل یک زیر شاخه وجود دارد که اطلاعات مربوط به آن پردازه را در فایل‌هایی بیان می‌کند.

در فایل سیستم‌های امروزی مسائل زیادی در مورد بهبود عملکرد یا بازیابی از خطا (مثلا پس از رفتن برق) در نظر گرفته می‌شوند.

- یکی از ویژگی‌های جالب برخی از فایل سیستم‌ها برگرداندن فایل‌سیستم به یک وضعیت گذشته است. برای مثال می‌توان دید محتویات یک فایل در شنبه‌ی گذشته ساعت ده چه بوده است (به این فایل سیستم‌ها دارای ویژگی Snapshot یا Archival گفته می‌شود).
- یک نمونه‌ی قدیمی از فایل سیستم‌هایی که این امکان را دارد فایل سیستم Fossil هست که در سیستم عامل Plan 9 پیاده‌سازی شده است.
- برخی از فایل‌سیستم‌ها امکان رمزگذاری محتویات را دارند.

- یکی از کارهایی که سیستم عامل برای بهبود عملکرد فایل‌سیستم انجام می‌دهد Caching است.



- چون سرعت دیسک کمتر از سرعت حافظه‌ی اصلی است، وقتی محتویات یک فایل از دیسک خوانده می‌شود، سیستم عامل یک کپی از آن را در حافظه نگه می‌دارد تا دسترسی‌های بعدی به آن فایل سریع‌تر انجام شود.

سیستم‌های عامل در محیط‌ها و کاربردهای متفاوتی استفاده می‌شوند. با برخی از این محیط‌ها آشنا می‌شویم.

کامپیوترهای شخصی

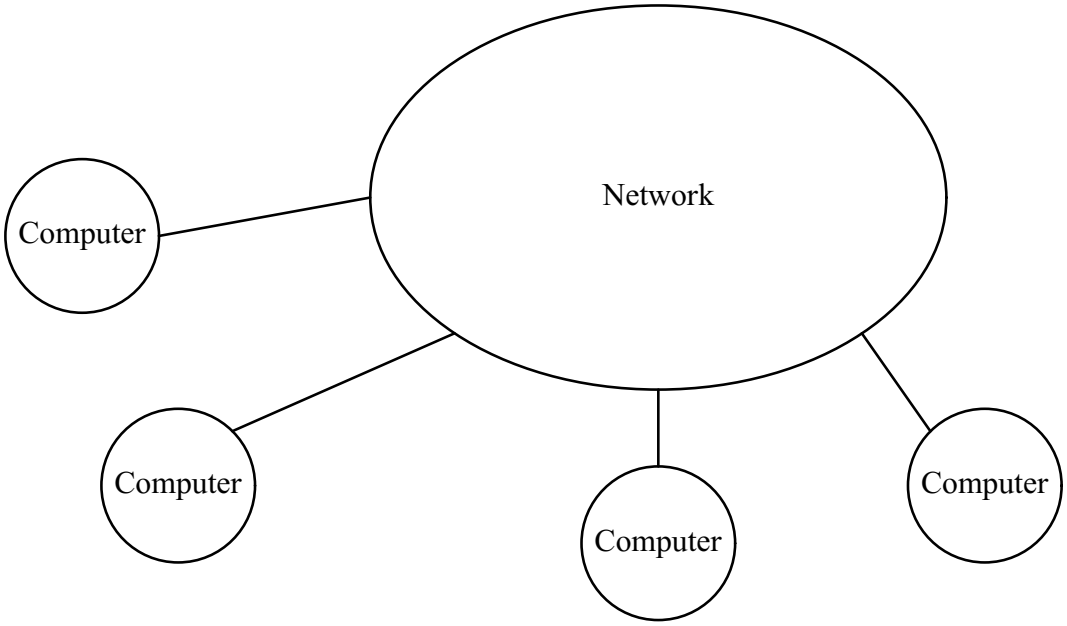
کامپیوترهای شخصی کامپیوترهای تنهایی هستند که برای کاربردهایی مثل ویرایش متن و استفاده از اینترنت استفاده می‌شوند. گاهی این کامپیوترها به شبکه نیز متصل می‌شود.

کامپیوترهای همراه

کامپیوترهای سیار امروزه بسیار گسترش یافته‌اند.

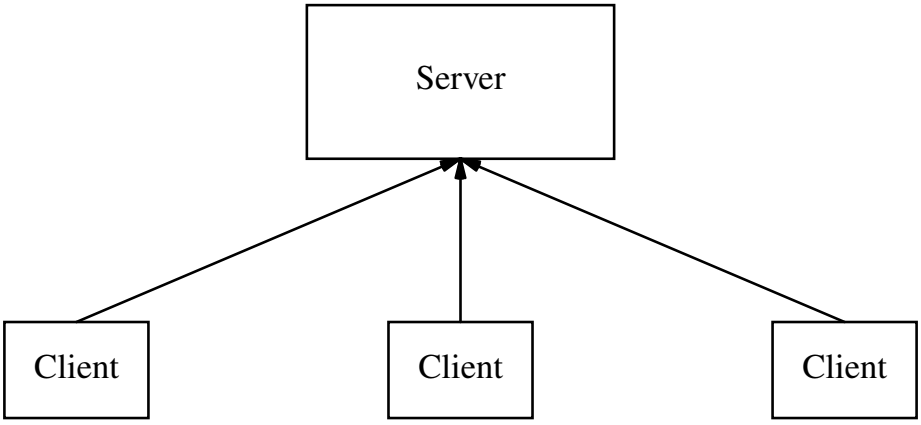
- گوشی‌های موبایل در این دسته قرار می‌گیرند و امکانات متنوعی دارند.
- معمولا این کامپیوترها صفحه کلید ندارند و امکاناتی دارند که معمولا در کامپیوترهای شخصی وجود ندارند، مثل تماس تلفنی و مکان‌یاب.

- سیستم‌های توزیع شده، کامپیوترهایی هستند که به صورت فیزیکی از هم جدا هستند ولی به کمک شبکه به یکدیگر متصل شده‌اند تا امکانات بیشتری را به کابران بدهند.



- افزایش سرعت پردازش (استفاده از پردازنده‌های چند کامپیوتر)
- افزایش توانایی (استفاده از سخت‌افزارهایی که در برخی از کامپیوترها وجود دارد)
- افزایش در دسترس بودن اطلاعات (اطلاعات با سرعت بیشتری به کاربران انتقال می‌یابد)
- افزایش قابلیت اعتماد (اگر یکی از کامپیوترهای سیستم توزیع شده خراب شود، کامپیوترهای دیگر می‌توانند نقش آن را ایفا کنند) را افزایش دهند.

یکی از ساده‌ترین شکل‌های سیستم‌های توزیع شده، مدل Client-Server است.



- ماشین خدمت خاصی مثل قدرت پردازشی، ذخیره‌سازی فایل‌ها یا بازیابی صفحه‌های وب را ارائه می‌دهد.
- کابران برای استفاده از این خدمت، با کمک شبکه به سرور وصل می‌شوند و از آن خدمت استفاده می‌کنند.
- مثال‌هایی از این سرورها، سرورهای HTTP هستند که برای انتشار صفحه‌های وب استفاده می‌شوند.

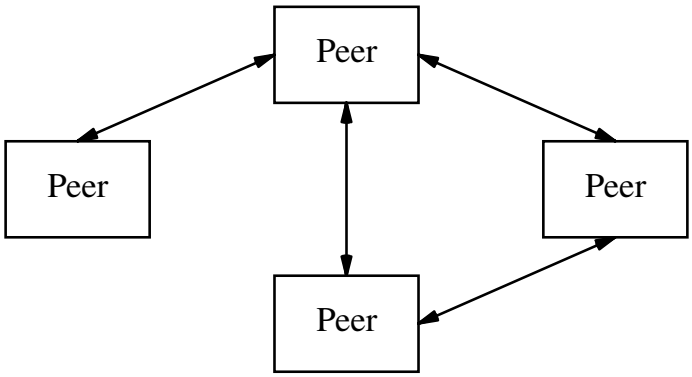
ضعف‌های مدل Client-Server

مدل Client-Server چند ضعف مهم دارد.

- اول اینکه اگر تعداد کاربران زیاد باشد، سرور به دلیل منابع محدود (مثل قدرت پردازشی یا پهنای باند) نمی‌تواند به همه‌ی کاربران پاسخ دهد.
- دوم اینکه اگر سرور دچار مشکل شود، خدمت‌رسانی متوقف می‌شود و درخواست کاربران بدون پاسخ می‌ماند.

مدل نظیر به نظیر (Peer to Peer):

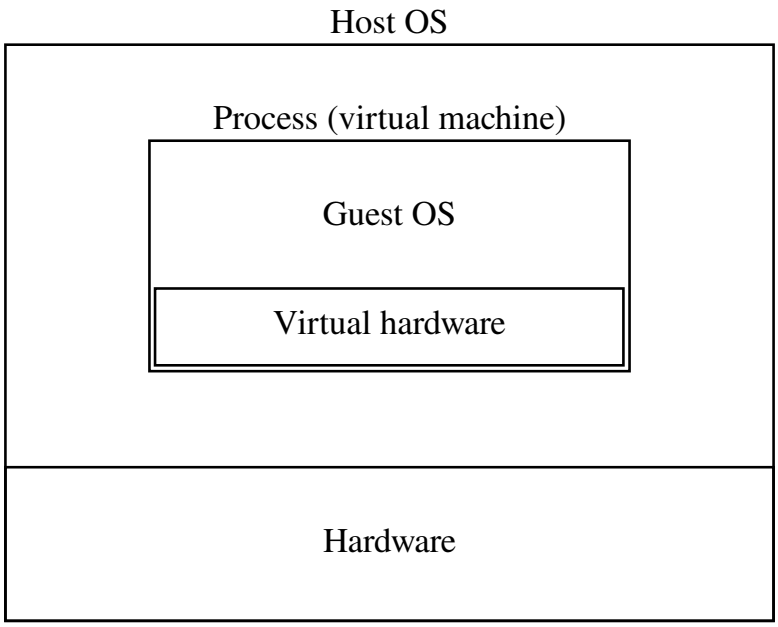
هر ماشین (نظیر یا Peer) هم یک سرور است و هم یک کاربر (هم از سایر کامپیوترها درخواست می‌کند و هم خود به درخواست سایرین پاسخ می‌دهد).



- یک مثال مشهور از این مدل پروتکل BitTorrent است که برای انتقال فایل استفاده می‌شود. هر نظیر می‌خواهد کل فایل را دریافت کند و برای این کار به تعدادی نظیر دیگر وصل می‌شود. هر یک از آنها قسمت‌هایی از فایل را که در اختیار دارد به بقیه می‌دهد و قسمت‌هایی که ندارد را از بقیه درخواست می‌کند.

- به علت اینکه در هر لحظه یک کاربر با کاربران زیادی می‌تواند اتصال برقرار کند، سرعت دریافت اطلاعات می‌تواند در این شبکه بیشتر باشد.
- از طرفی دیگر، چون نسخه‌های متفاوتی از اطلاعات بین نظیرها توزیع شده است، با از کار افتادن یا خارج شدن چند نظیر با هم دسترسی به اطلاعات امکان دارد.

مجازی‌سازی سخت‌افزار (Hardware Virtualization) یعنی اجرای یک سیستم عامل به صورت یک پردازه در یک سیستم عامل دیگر.



- به سیستم عامل بیرونی، سیستم عامل میزبان (Host) به سیستم عامل داخلی، میهمان (Guest) گفته می‌شود و به برنامه‌ای که سخت‌افزار را شبیه‌سازی می‌کند، ماشین مجازی (Virtual Machine) گفته می‌شود.
- در مجازی‌سازی سیستم عامل میهمان به سخت‌افزار واقعی دسترسی ندارد و روی یک سخت‌افزار مجازی اجرا می‌شود.
- مجازی‌سازی به دو شکل اجرا می‌شود: در مجازی‌سازی کامل (Full Virtualization) یک سیستم عامل بدون تغییر اجرا می‌شود و در Para-virtualization سیستم عامل میهمان برای مجازی‌سازی تغییر داده می‌شود.

- در دنیای تجاری، سرورهایی با تعداد پردازنده‌ها و حافظه‌ی زیادی هستند که از آنها فقط برای اجرای تعدادی ماشین مجازی استفاده می‌شود.
- در این سرورها نیازی به یک سیستم عامل عام منظوره نیست و سیستم عامل میزبان فقط برای اجرای ماشین‌های مجازی استفاده می‌شود.
- در این شرایط گاهی به جای استفاده از سیستم عامل عام منظوره از برنامه‌ای به نام Virtual Machine Manager یا Hypervisor استفاده می‌شود.

  مثال: Xen و VMware ESX.

پردازش ابری

پـردازش ابری (Cloud Computing): خدمتـی مثل ذخیره‌سـازی فایل‌هـا، پـردازش داده‌هـا یـا اجرای برنامه‌هـای مختلف از راه شبکه در اختیار کاربران قرار می‌گیرد.

- در واقع، تعدادی کامپیوتر در ابر شبکه‌ی اینترنت و بدون اینکه مکان آنهـا مشخص باشـد در اختیـار کاربران قـرار می‌گیرند.
- برای نمونه، شرکتی را در نظر بگیرید که به مدت چهار روز نیاز به صد کامپیوتر برای پردازش داشته باشد. به جای اینکه این تعداد کامپیوتر خریداری گردند،
- در پردازش ابری به یـک شرکت ارائه دهنده‌ی خدمـات پردازش ابـری مراجعه می‌شـود و کامپیوترهایـی به تـعداد مورد نیاز و در بازه‌ی مورد نظر درخواست می‌شوند و با توجه به تنظیمـات کامپیوترها و زمان استفـاده هزینه‌ای را پرداخت می‌کنند.
- در پردازش ابری از مجازی‌سازی سخت‌افزار استفـاده می‌شود؛ معمـولا سرورهایی بـا توان پردازشـی و حافـظه‌ی زیاد موجود هستند و روی هر یک از این سرورها تعدادی ماشیـن مجازی اجرا می‌شوند و در اختیـار کاربران قـرار می‌گیرند.

دسته‌ی دیگر، سیستم‌های نهفته (Embedded Systems) و بی‌درنگ (Realtime) هستند. سیستم‌های نهفته، کامپیوترهایی هستند که برای کاربردی خاص طراحی می‌شوند، مثل موتور ماشین‌ها، ربات‌های خط تولید، ماشین‌های لباس‌شویی.

- در این سیستم‌ها معمولا محدودیت‌های زمانی برای پاسخ وجود دارد. برای مثال، عمل کیسه‌ی هوای ماشین یا چراغ راهنمایی باید در محدوده‌ی زمانی مشخصی انجام شود.
- در سیستم‌های نهفته، گاهی از سیستم‌های عامل عام‌منظوره مثل لینوکس و گاهی از سیستم‌های عامل خاص‌منظوره استفاده می‌شوند.

در این قسمت ساختار داخلی سیستم‌های عامل را مرور می‌کنیم.

ویژگی‌های مناسب سیستم عامل

سیستم عامل خوب از دید کاربران:

- منعطف باشد (افزودن برنامه‌های جدید یا استفاده از آن در کاربردهایی که تصور نمی‌شد سخت نباشد).
- استفاده از آن آسان باشد.
- قابل اعتماد باشد (دچار اشکال نشود).
- امن باشد (اطلاعات محافظت شوند).
- سریع باشد و ... .

ویژگی‌های مناسب سیستم عامل از دید توسعه دهندگان

مانند هر نرم‌افزار بزرگی، برای توسعه دهندگان سیستم‌های عامل اهمیت دارد

- طراحی، پیاده‌سازی و نگهداری سیستم عامل دشوار نباشد.

دلیل اهمیت ساختار سیستم عامل

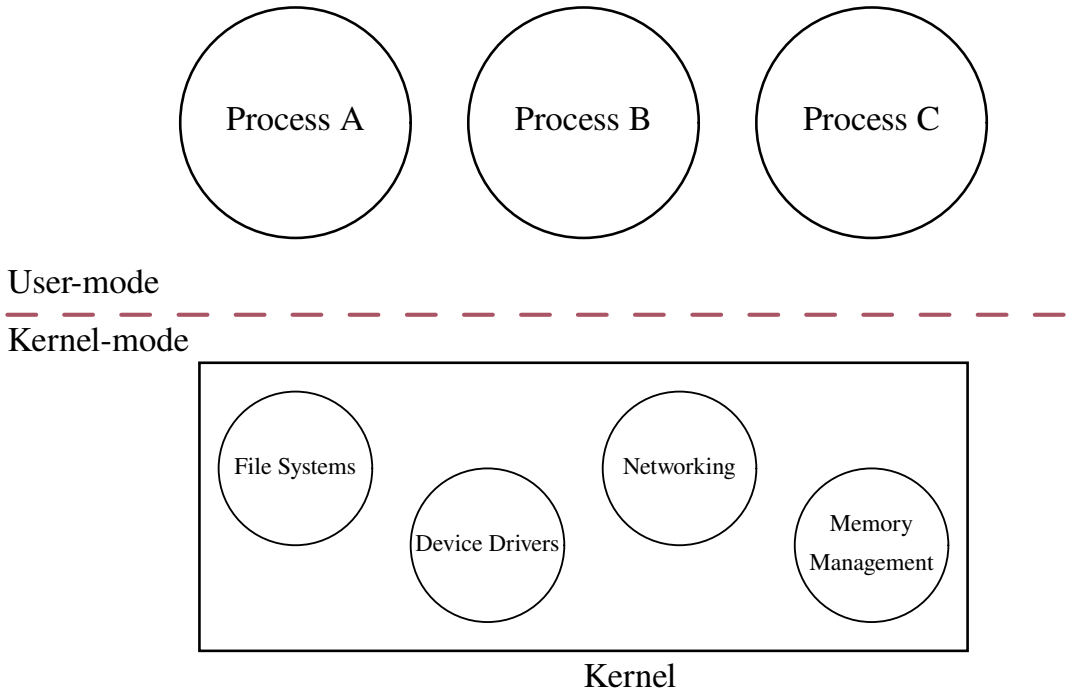
سیستم عامل از قسمت‌هایی زیادی تشکیل شده است

- راه‌اندازهای دستگاه‌های ورودی/خروجی،
- فایل سیستم‌ها،
- الگوریتم‌های زمانبندی پردازنده،
- مدیریت حافظه و
- پروتکل‌های شبکه.

سیستم عامل باید طوری طراحی شود که طراحی و پیاده‌سازی، یافتن اشکال، افزودن ویژگی‌ها به این قسمت‌ها دشوار نباشد.

ساختار یکپارچه

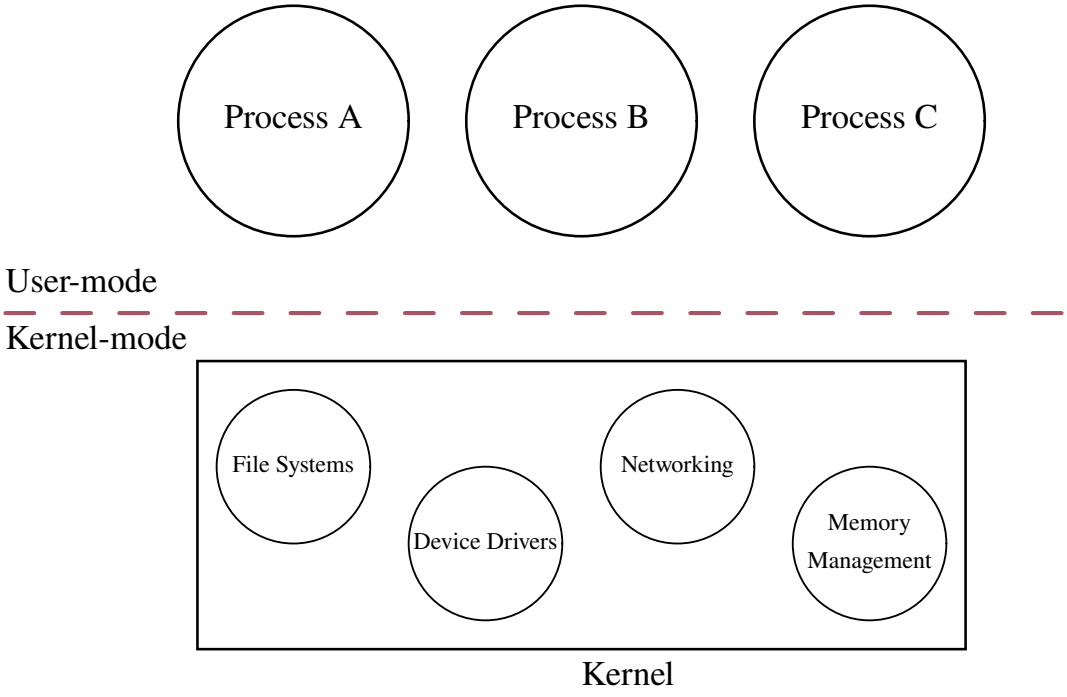
در ساختار یکپارچه (Monolithic) کل سیستم عامل در قالب یک برنامه‌ی یکپارچه پیاده‌سازی می‌شود و کل آن در حالت کرنل (Kernel Mode که در جلسه‌های گذشته بررسی شد) اجرا می‌شود.



- سیستم‌های عامل DOS و Unix نمونه‌هایی از این ساختار هستند.

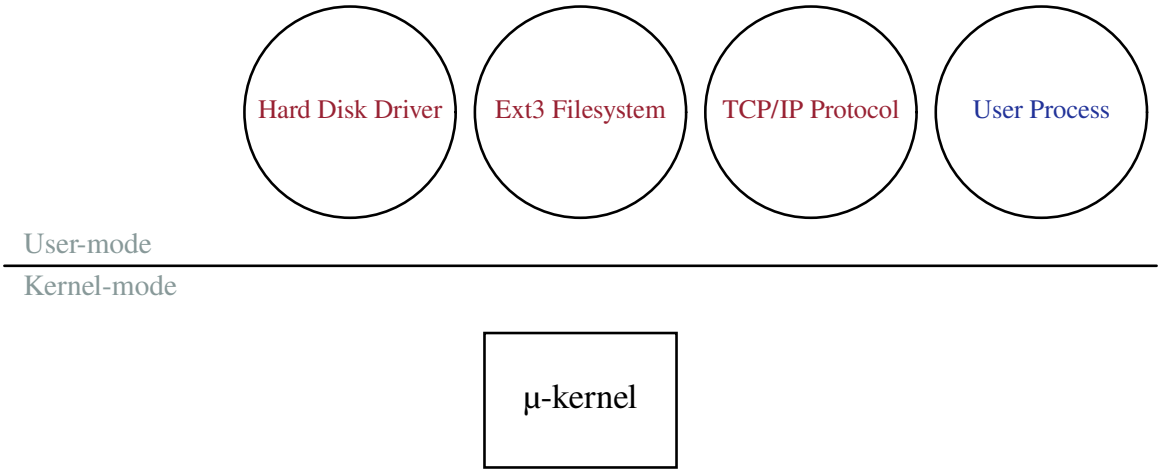
بدی‌های ساختار یکپارچه

- در این ساختار، اگر تغییر جزئی در سیستم عامل ایجاد شود یا ویژگی جدیدی (مثل یک راه‌انداز برای یک سخت‌افزار جدید) به آن اضافه شود لازم است کل سیستم عامل کامپایل شود و کل سیستم عامل جایگزین گردد.
- از طرف دیگر، به خاطر ساختار یکپارچه‌ی این سیستم‌های عامل، حافظه بین قسمت‌های مختلف سیستم عامل مشترک است و اگر قسمتی از سیستم عامل (مثلا راه‌انداز صفحه کلید) مشکلی داشته باشد می‌تواند در حافظه‌ی مربوط به سایر قسمت‌ها بنویسد و کل سیستم عامل را مختل کند (خطاها به سایر قسمت‌های سیستم عامل سرایت می‌کند).
- اما معمولا سربار این سیستم عامل کم است.


Process A
Process B
Process C
User-mode
Kernel-mode
File Systems
Device Drivers
Networking
Memory Management
Kernel

یکی از بزرگ‌ترین مشکلات ساختار یکپارچه، سرایت خطا است.

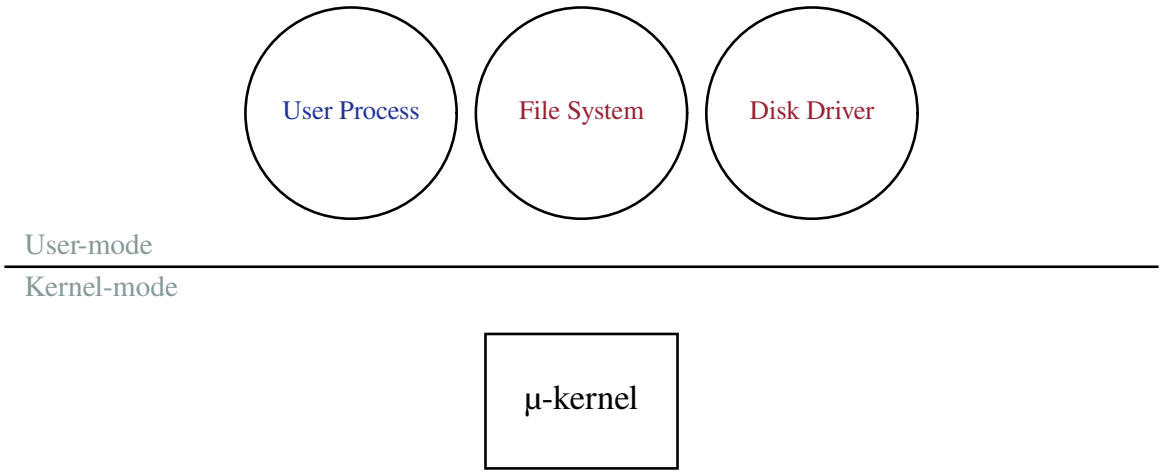
- برای حل این مشکل، در ساختار میکروکرنل (Microkernel یا µ-kernel) هسته‌ی سیستم بسیار کوچک شده است و بیشتر قسمت‌های سیستم عامل به فضای کاربری (User Mode) انتقال می‌یابند و به صورت پردازه‌هایی اجرا می‌شوند.
- به خاطر کوچک شدن هسته‌ی سیستم عامل به آن میکروکرنل (هسته‌ی کوچک) می‌گویند.



- در این ساختار قسمت‌هایی مهمی از سیستم عامل به فضای کاربری انتقال می‌یابند، مثل راه‌اندازها، فایل سیستم‌ها، مدیریت شبکه. این قسمت‌ها لازم است با هم در ارتباط باشند و برای این کار از تبادل پیغام (Message Passing) استفاده می‌کنند.
- قسمت‌هایی مثل مدیریت حافظه، قسمتی از زمانبندی پردازنده و پیاده‌سازی تبادل پیغام به علت ماهیت پایه‌ای آنها نمی‌توانند به فضای کاربری انتقال پیدا کنند و در خود میکروکرنل باقی می‌مانند.
- از میکروکرنل‌های موجود Mach، Minix، L4 و QNX را می‌توان مثال زد.

مثالی از عملکرد ساختار میکروکرنل

برای نمونه، راه‌انداز دیسک سخت در یک پردازه‌ی فضای کاربری قرار می‌گیرد و پیاده‌سازی یک فایل سیستم که در آن دیسک قرار دارد در پردازه‌ی دیگری قرار دارد.

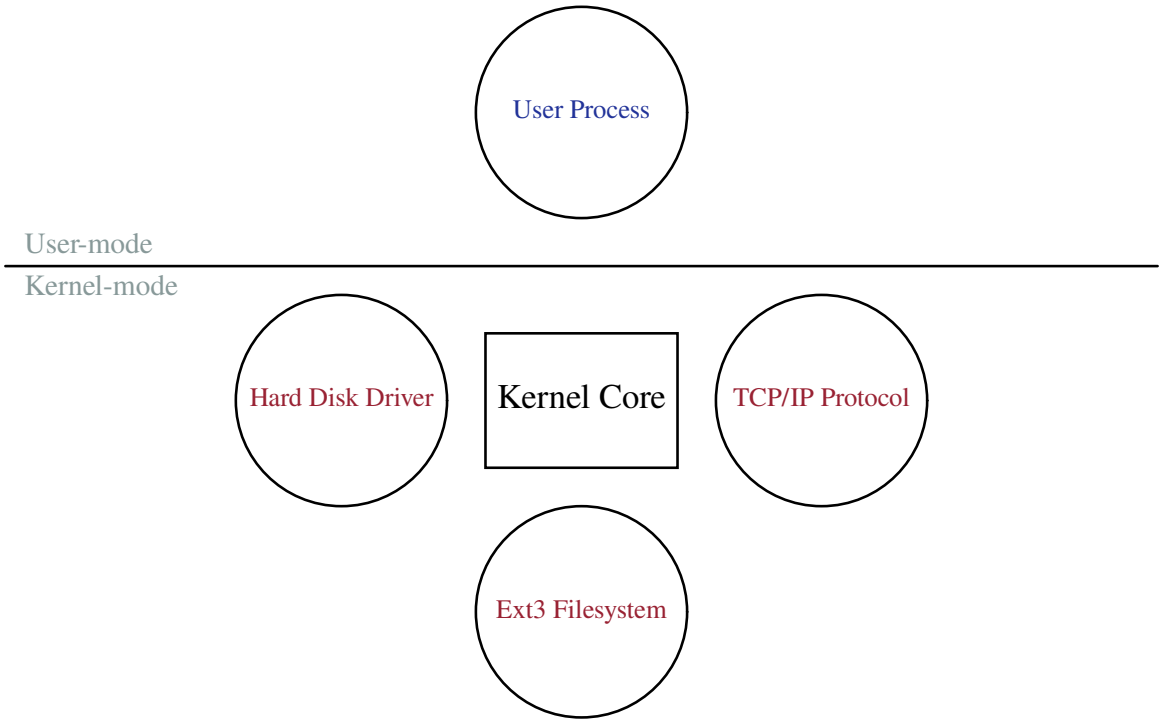


- اگر یک پردازه‌ی کاربر چیزی را از دیسک بخواند، ابتدا پیغامی به پردازه‌ی پیاده‌سازی کننده‌ی فایل سیستم فرستاده می‌شود.
- این پردازه، پیغامی را به راه‌انداز دیسک می‌فرستد تا از قسمت مناسب دیسک اطلاعات خوانده شوند.
- پس از خوانده شدن، محتویات دیسک با پیغامی به پردازه‌ی فایل سیستم فرستاده می‌شود و این پردازه اطلاعات را به پردازه‌ی کاربر انتقال می‌دهد.

- یکی از مزیت‌های مهم ساختار میکروکرنل این است که اگر یکی از پردازه‌ها دچار اشکال شود، سایر قسمت‌های سیستم عامل مصون می‌مانند چون حافظه‌ی هر پردازه مجزا است و پردازه‌ها به حافظه‌ی سایر پردازه‌ها یا هسته دسترسی ندارند.
- همچنین، اگر تغییری در قسمتی از سیستم عامل لازم باشد یا اگر یک ویژگی (مثلا یک راه‌انداز) جدید به آن اضافه گردد کافی است فقط یک پردازه در سیستم عامل اضافه شود یا تغییر کند و لازم به تغییر کل سیستم عامل یا راه‌اندازی دوباره‌ی آن نیست.
- مهم‌ترین بدی میکرو کرنل سربار بالای آن به دلیل تبادل پیغام است (در مثال قبل دلیل این سربار روشن است).

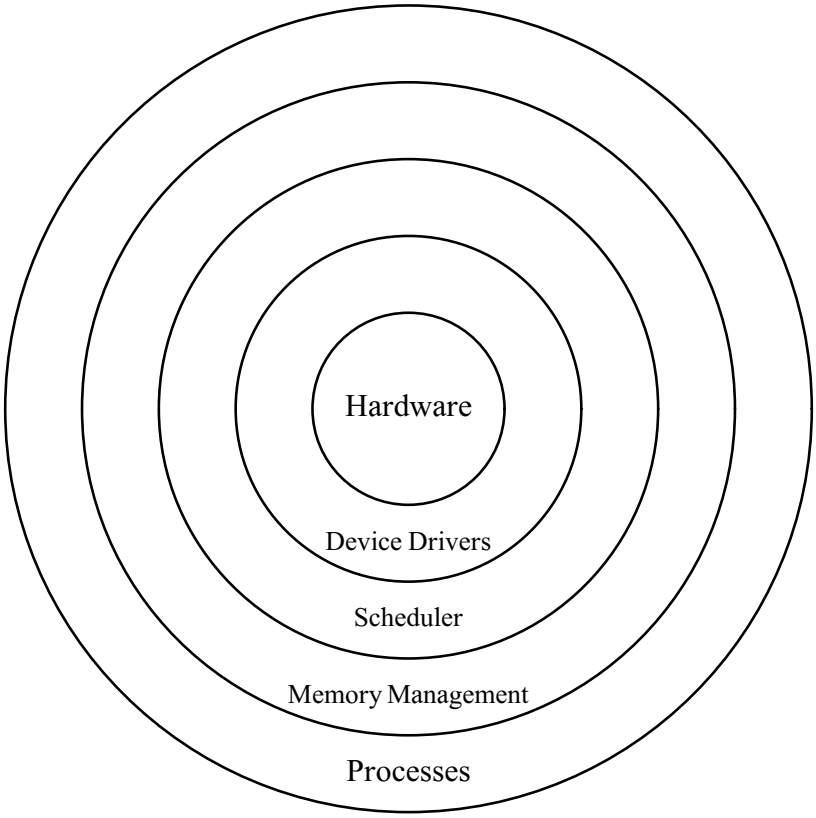
ساختار ماژول‌های قابل بارگذاری هسته

در ساختار ماژول‌های قابل بارگذاری هسته (Loadable Kernel Modules)، هسته‌ی سیستم عامل مشابه ساختار میکروکرنل بسیار کوچک است اما قسمت‌های مختلف سیستم عامل در حالت هسته قرار می‌گیرند و اجرا می‌شوند. به این قسمت‌ها ماژول گفته می‌شود.



- ماژول‌های می‌توانند در زمان اجرا حذف شوند یا اضافه شوند. به این ترتیب، تغییر سیستم عامل یا اضافه کردن قسمت‌های جدید به آن نیازی به تغییر یا راه‌اندازی مجدد سیستم عامل ندارد.
- اما چون ماژول‌ها در فضای هسته اجرا می‌شوند و حافظه‌ی قسمت‌های مختلف هسته با هم مشترک است، اگر یکی از ماژول‌ها دچار مشکل شود، کل سیستم عامل می‌تواند مختل شود.

- در ساختار لایه لایه (Layered): سیستم عامل به صورت تعدادی لایه پیاده‌سازی می‌شود که هر قسمت از قسمت زیرین به صورت مستقیم استفاده می‌کند و به لایه‌ی بعدی خدمت می‌دهد.



- در داخلی‌ترین لایه سخت‌افزار قرار دارد و در بیرونی‌ترین لایه برنامه‌های کاربردی هستند.
- از این ساختار به دلیل طراحی سخت و امکان پذیر نبودن طراحی برخی از قسمت‌ها به صورت لایه لایه در عمل کمتر استفاده می‌شود (اما لازم به اشاره است که طراحی لایه لایه در پروتکل‌های شبکه بسیار موفق است).

ساختار ترکیبی

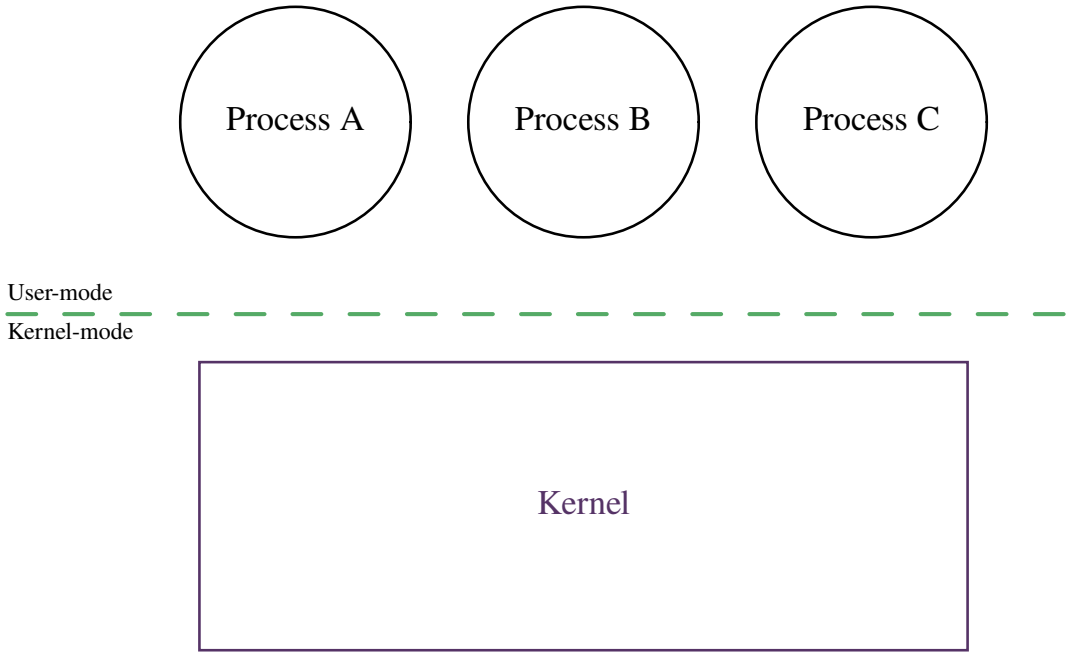
تعداد زیادی از سیستم عامل‌هایی که امروزه استفاده می‌شوند از ترکیبی از ساختارهایی که بررسی کرده‌ایم استفاده می‌کنند.

- برای مثال، سیستم عامل لینوکس (Linux) و سولاریس (Solaris) مشابه ساختار ماژول‌های قابل بارگذاری هسته هستند اما قسمت اصلی آنها بزرگ‌تر است و مشابه سیستم‌های عامل یکپارچه هستند.

سیستم‌های عامل امکاناتی را برای یافتن و رفع اشکال برنامه‌ها و خود سیستم عامل ارائه می‌دهند. با برخی از آنها آشنا می‌شویم.

به یافتن و رفع مشکلات در یک سیستم اصطلاحا Debugging گفته می‌شود.

- فرض کنید پردازه‌ای دچار خطا شده باشد (مثلا به قسمت غیر مجازی از حافظه دسترسی داشته است) و سیستم عامل باید آن را خاتمه دهد.
- چگونه می‌توان مشکل را بررسی کرد؟



- سیستم عامل محتویات حافظه‌ی پردازه و وضعیت رجیسترهای پردازنده در زمان بروز خطا را در یک فایل ذخیره می‌کند.
- به این فایل Coredump می‌گویند (Core به معنای حافظه‌ی اصلی و Dump به معنای ریختن).
- با کمک این فایل و یک برنامه‌ی Debugger می‌توان مشخص کرد خطا در چه خطی از برنامه رخ داده است (دقت کنید که دستوری که اجرا می‌شود در رجیستر IP قرار می‌گیرد) و در هنگام خطا، مقدار متغیرهای برنامه چه بوده است.
- فایل‌های Coredump از دید تجاری اهمیت دارند چرا که اگر برنامه‌ای در حال استفاده‌ی کارفرما دچار اشکال شود امکان یافتن اشکال در آن لحظه نیست.

- اگر سیستم عامل دچار اشکال شود، محتویات حافظه و رجیسترها را در فایلی به نام Crashdump قرار می‌دهند.
- نوشتن Crashdump در دیسک چه خطری دارد؟
- چه راه حلی وجود دارد؟

- مشکلات موجود در برنامه‌ها گاهی مربوط به کارایی پایین آنها است.
- مشکلات کارایی مربوط به متغیرهایی مثل سرعت برنامه، مصرف حافظه، استفاده از حافظه‌ی نهان
- برای این برنامه‌ها چه می‌توان کرد؟

- گلوگاه یک برنامه (Bottleneck) قسمتی از برنامه است که کندتر از سایر قسمت‌ها است و با بهبود کارایی آن قسمت کل برنامه به صورت چشم‌گیری سریع‌تر می‌شود (یا حافظه‌ی کل برنامه به مقدار قابل توجهی کاهش می‌یابد).

```
43.7%   decode_utf8()
15.1%   split_data()
10.3%   reorder_line()
02.8%   strchr()
...
```

- ابزارهایی وجود دارند که مشکلات کارایی برنامه‌ها را شناسایی می‌کنند. به این برنامه‌ها Profiler و به عمل شناسایی گلوگاه‌های برنامه Profiling می‌گویند.
- پس از شناسایی گلوگاه‌های یک برنامه به کمک Profiler، برنامه‌نویسان تلاش می‌کنند که با تغییر برنامه گلوگاه را از بین ببرند.
- برنامه‌های Profiler به صورت تجاری برای افزایش کارایی برنامه‌ها بسیار با اهمیت هستند. برای سیستم عامل لینوکس، Profiler-هایی مثل Gprof، Perf و Valgrind موجود هستند.

استفاده از ثبت رخدادها

عمل Logging یعنی ثبت کردن رخدادهای نرم‌افزاری و به فایلی که این رخدادها را نگه می‌دارد فایل Log گفته می‌شود.

```
1       Reading configuration /etc/xvc.conf
2       Loading data files
3       Locking databases (/var/lock/xvc.db.lock)
4       Registering node via 10.18.5.21
```

- ثبت کردن این رخدادها چگونه به شناسایی خطا کمک می‌کند؟
- در سیستم عامل لینوکس می‌توان با دستور dmesg رخدادهای ثبت شده توسط هسته‌ی سیستم عامل را مشاهده کرد.