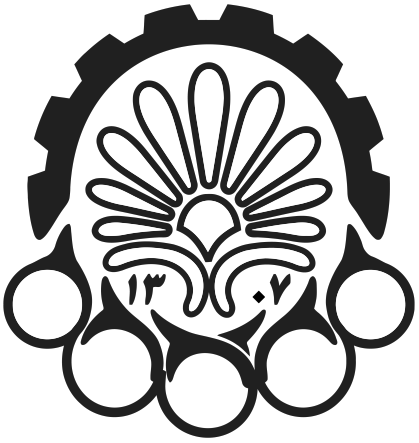
سیستم‌های عامل

دکتر زرندی



دانشگاه صنعتی امیرکبیر

(پلی تکنیک تهران)

دانشکده مهندسی کامپیوتر

رضا آدینه پور ۴۰۲۱۳۱۰۵۵

تمرین سری سوم

۲۶ مهر ۱۴۰۳

سیستم‌های عامل

تمرین سری سوم

رضا آدینه پور ۴۰۲۱۳۱۰۵۵

دانشکده مهندسی کامپیوتر

سوال اول

به سوالات زیر در رابطه با محیط‌های محاسباتی (Computing environment) پاسخ دهید.

۱. مدل‌های Client-server و Peer to peer را تعریف و با یکدیگر مقایسه کنید.

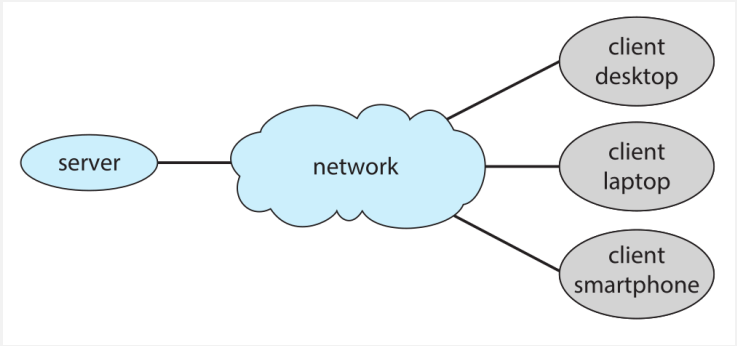
پاسخ

(آ) **معماری Client-server:**

در این مدل، سیستم به دو بخش اصلی تقسیم می‌شود: کلاینت (Client) و سرور (Server). سرور یک سیستم قدرتمند است که خدمات یا منابع خاصی مانند پایگاه‌داده، فایل‌ها، یا اپلیکیشن‌ها را ارائه می‌دهد و کلاینت‌ها دستگاه‌هایی هستند که از این منابع استفاده می‌کنند. ارتباط معمولاً به این شکل است که کلاینت‌ها درخواست‌هایی به سرور می‌فرستند و سرور پاسخ می‌دهد. برای مثال زمانی که ما از مرورگر خود به وب‌سایتی دسترسی پیدا می‌کنیم، مرورگر به عنوان کلاینت و وب‌سرور به عنوان سرور عمل می‌کند.

مطابق با توضیحات صفحه ۴۳ کتاب Silberschatz می‌توان گفت معماری شبکه‌های امروزی معمولاً به این شکل است. به این مدل از سیستم‌های توزیع‌شده، سیستم کلاینت–سرور می‌گویند. همچنین سرورها به دو دسته تقسیم می‌شوند: سرورهای محاسباتی و سرورهای فایل. که چون در صورت سوال توضیحات سرور ها خواسته نشده است، توضیحات آن را نمی‌نویسیم.

شکل زیر که از کتاب Silberschatz آورده شده است، ساختار مدل کلاینت–سرور را نشان می‌دهد.

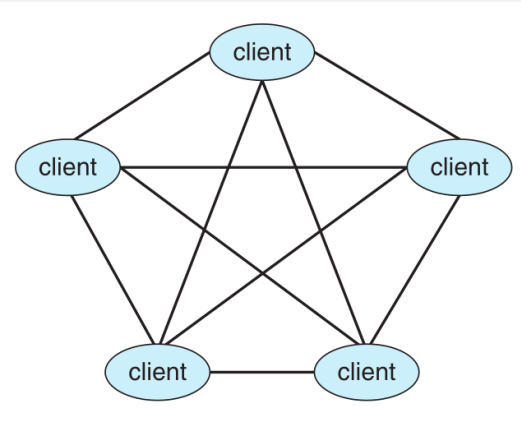


شکل ۱: ساختار سیستم‌هایی با مدل کلاینت–سروری

(ب) **معماری Peer to peer:**

مدل دیگری که برای سیستم‌های توزیع‌شده وجود دارد، مدل P2P است. در این مدل، تفاوتی بین کلاینت‌ها و سرورها وجود ندارد یعنی هر گره برحسب خدمتی که ارائه می‌دهد، می‌تواند هم به عنوان کلاینت و هم به عنوان سرور عمل کند. شکل زیر نمونه‌ای از این معماری است:

پاسخ



شکل ۲: ساختار سیستم‌هایی با مدل P2P

سیستم‌های P2P نسبت به سیستم‌های کلاینت–سرور مزیتی دارند. در سیستم کلاینت–سرور، سرور می‌تواند یک گلوگاه باشد؛ اما در سیستم P2P خدمات توسط چندین گره که در شبکه توزیع شده‌اند ارائه می‌شوند. همچنین Scaleability در مدل P2P بهتر است، چون با افزایش تعداد دستگاه‌ها، شبکه قوی‌تر می‌شود. در حالی که در مدل Client-server، با افزایش تعداد کلاینت‌ها، سرور ممکن است تحت فشار قرار گیرد. از نظر امنیت مدل کلاینت سرور راحت تر مدیریت می‌شود چون یک سرور مرکزی می‌تواند کنترل کامل بر روی داده‌ها و دسترسی داشته باشد، در حالی که در مدل P2P امنیت پیچیده‌تر است چون داده‌ها در میان بسیاری از دستگاه‌ها پخش می‌شوند.

اسکایپ نمونه‌ای از مدل P2P است.

۲. Virtualization و Emulation را تعریف کنید و تفاوت‌های آن‌ها را ذکر کنید.

پاسخ

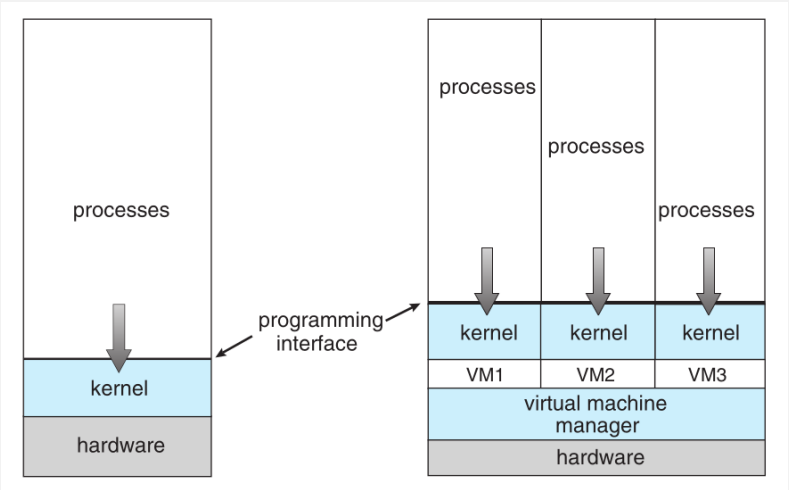
مطابق با تعریف آقای Silberschatz در صفحه ۳۴ کتابشان، می‌توان گفت که: مجازی‌سازی یا Virtualization یک فناوری است که به ما این امکان را می‌دهد تا سخت‌افزار یک کامپیوتر (مانند پردازنده، حافظه، دیسک‌های سخت، کارت‌های شبکه و غیره) را به چندین محیط اجرایی مختلف تجزیه کنیم و به این ترتیب، این حس برای کاربر ایجاد شود که هر محیط به‌صورت مجزا روی یک کامپیوتر اختصاصی خود در حال اجرا است. این محیط‌ها می‌توانند به‌عنوان سیستم‌عامل‌های مختلف (مثل ویندوز و یونیکس) در نظر گرفته شوند که ممکن است هم‌زمان اجرا شوند و با هم تعامل داشته باشند. کاربر یک ماشین مجازی می‌تواند بین این سیستم‌عامل‌های مختلف به همان شکلی که بین فرآیندهای مختلف در یک سیستم‌عامل واحد جابه‌جا می‌شود، جابه‌جا شود.

مجازی‌سازی به سیستم‌عامل‌ها این امکان را می‌دهد که به‌عنوان برنامه‌هایی در درون دیگر سیستم‌عامل‌ها اجرا شوند. در نگاه اول، ممکن است این قابلیت چندان کاربردی به نظر نرسد، اما صنعت مجازی‌سازی بسیار گسترده و در حال رشد است که نشان‌دهنده اهمیت و کاربرد فراوان آن است.

به طور کلی، نرم‌افزارهای مجازی‌سازی یکی از اعضای گروهی از نرم‌افزارها هستند که شبیه‌سازی (Emulation) نیز در آن قرار می‌گیرد. Emulation به معنای شبیه‌سازی سخت‌افزار کامپیوتر در نرم‌افزار است و معمولاً زمانی استفاده می‌شود که نوع پردازنده مبدأ با پردازنده هدف متفاوت باشد. به عنوان مثال، زمانی که شرکت اپل پردازنده‌های IBM Power خود را به پردازنده Intel x86 برای کامپیوترهای دسکتاپ و لپ‌تاپ خود تغییر داد یک قابلیت شبیه‌سازی به نام Rosetta ارائه داد که به برنامه‌هایی که برای پردازنده IBM نوشته شده بودند، اجازه می‌داد روی پردازنده Intel اجرا شوند.

پاسخ

این مفهوم می‌تواند توسعه یابد تا به یک سیستم‌عامل کامل که برای یک پلتفرم خاص نوشته شده، اجازه دهد روی یک پلتفرم دیگر اجرا شود. با این حال، شبیه‌سازی هزینه بالایی دارد، زیرا هر دستورالعمل در سطح ماشین که به‌طور بومی روی سیستم مبدأ اجرا می‌شود، باید به دستورالعمل معادل در سیستم هدف ترجمه شود، که اغلب منجر به چندین دستورالعمل در سیستم هدف می‌شود. اگر پردازنده‌های مبدأ و هدف سطح عملکرد مشابهی داشته باشند، ممکن است کد شبیه‌سازی‌شده بسیار کندتر از کد بومی اجرا شود. شکل زیر نمونه ای از یک کامپیوتر معمولی و یک کامپیوتر مجازی شده را نشان می‌دهد:



شکل ۳: ساختار یک ۳ عدد VM

۳. سه نمونه از دسته سرویس‌های ابری را نام ببرید و به صورت مختصر توضیح دهید.

پاسخ

(آ) **Software as a service (SaaS):**
یک یا چند برنامه (مانند پردازشگرهای کلمه یا spreadsheets) که از طریق اینترنت در دسترس هستند.

(ب) **Platform as a service (PaaS):**
یک پشته نرم‌افزاری آماده برای استفاده از طریق اینترنت (به عنوان مثال، یک سرور پایگاه داده).

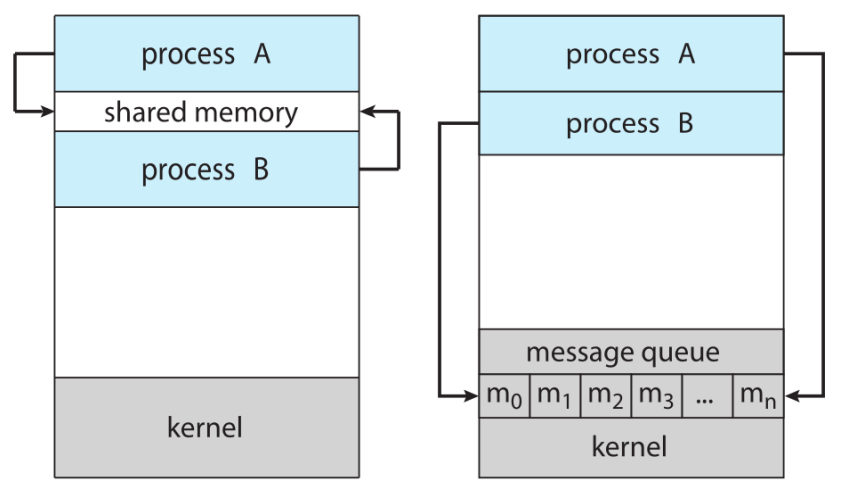
(ج) **Infrastructure as a service (IaaS):**
سرورها یا فضای ذخیره‌سازی که از طریق اینترنت قابل دسترسی هستند (برای مثال، فضای ذخیره‌سازی برای تهیه نسخه‌های پشتیبان از داده‌های تولید).

سوال دوم

دو روش برای ارتباط میان فرآیندها Message passing و Shared memory است. آن‌ها را تعریف و با یکدیگر مقایسه کنید.

پاسخ

در مدل Message passing یک بخش از حافظه که بین فرآیندهای همکار به اشتراک گذاشته می‌شود، ایجاد می‌گردد. سپس فرآیندها می‌توانند با خواندن و نوشتن داده‌ها در این بخش مشترک، اطلاعات را مبادله کنند. در مدل Message passing، ارتباط از طریق پیام‌هایی که بین فرآیندهای همکار رد و بدل می‌شود، انجام می‌گیرد. تفاوت این دو مدل ارتباطی در شکل زیر نشان داده شده است.



شکل ۴: مدل‌های برقراری ارتباط میان فرآیندها

هر دو مدل ذکر شده در سیستم‌عامل‌ها رایج هستند و بسیاری از سیستم‌ها هر دو را پیاده‌سازی می‌کنند. Message passing برای تبادل مقادیر کوچکتر داده‌ها مفید است، زیرا نیازی به جلوگیری از تداخلات نیست. Message passing همچنین در سیستم‌های توزیع‌شده نسبت به حافظه مشترک آسان‌تر پیاده‌سازی می‌شود. Shared memory می‌تواند سریع‌تر از پیام‌رسانی باشد، زیرا سیستم‌های Message passing معمولاً از طریق فراخوانی‌های سیستمی پیاده‌سازی می‌شوند و بنابراین نیازمند مداخله هسته هستند که زمان‌بر است. در سیستم‌های Shared memory فراخوانی سیستمی تنها برای ایجاد بخش‌های حافظه مشترک لازم است. پس از ایجاد حافظه مشترک، تمام دسترسی‌ها مانند دسترسی‌های معمولی به حافظه انجام می‌شود و نیازی به کمک هسته نیست.

سوال سوم

از میان عملیات‌هایی که نیاز به System Call دارند ۳ مثال نام ببرید و توضیح دهید که اگر هر عملیات در لایه User انجام نشود چه مشکل‌هایی را می‌تواند به وجود بیاورد.

پاسخ

۱. **عملیات خواندن و نوشتن فایل‌ها (File Read/Write)**
برای دسترسی به سیستم فایل و خواندن یا نوشتن داده‌ها در فایل‌ها، نیاز به System Call داریم. دسترسی مستقیم کاربران به هارد دیسک یا سیستم فایل از طریق User mode بدون استفاده از System Call بسیار خطرناک است. زیر می‌تواند مشکلات زیر را ایجاد کند:

- عدم کنترل دسترسی: اگر User ها به صورت مستقیم و بدون استفاده از سیستم‌عامل به فایل‌ها دسترسی پیدا کنند، امنیت فایل‌ها به خطر می‌افتد و احتمال خرابی داده‌ها وجود دارد.
- مدیریت ضعیف منابع: بدون واسطه‌ای مانند System Call امکان مدیریت مناسب منابع و جلوگیری از استفاده نادرست یا بیش‌ازحد منابع وجود نخواهد داشت.

۲. **تخصیص و آزادسازی حافظه (Memory Allocation/Deallocation)**
فرآیند تخصیص حافظه به برنامه‌ها توسط System Callهایی مانند malloc یا new انجام می‌شود. سیستم‌عامل کنترل می‌کند که چه بخشی از حافظه به برنامه اختصاص داده شود و چه زمانی حافظه باید آزاد شود. مشکلات این دسته به‌صورت زیر عنوان می‌شود:

- خرابی یا دسترسی غیرمجاز به حافظه: اگر کاربر مستقیماً به حافظه دسترسی پیدا کند، ممکن است به بخش‌هایی از حافظه دسترسی داشته باشد که به برنامه‌های دیگر اختصاص داده شده‌اند، که این می‌تواند منجر به خرابی سیستم یا دسترسی غیرمجاز به اطلاعات شود.
- مدیریت نادرست حافظه: بدون System Call، امکان نشت حافظه (memory leak) و استفاده بی‌رویه از منابع حافظه وجود دارد، چرا که سیستم‌عامل نمی‌تواند حافظه‌ای که دیگر مورد استفاده نیست را آزاد کند.

۳. **اجرای فرآیند جدید (Process Creation)**
عملیات ایجاد یک فرآیند جدید توسط سیستم‌عامل انجام می‌شود که از طریق System Callهایی مانند fork یا exec در سیستم‌عامل‌های یونیکسی انجام می‌شود. این System Callها امکان اجرای برنامه‌های جدید در محیط سیستم‌عامل را فراهم می‌کنند. مشکلات این مثال به صورت زیر معرفی می‌شود:

- تداخل در مدیریت فرآیندها: اگر کاربران بتوانند به‌طور مستقیم فرآیند جدید ایجاد کنند، سیستم‌عامل نمی‌تواند به درستی فرآیندها را زمان‌بندی کند و منابع را به طور عادلانه بین آن‌ها توزیع نماید.
- نقض امنیت: ایجاد فرآیندهای جدید بدون کنترل سیستم‌عامل می‌تواند منجر به اجرای کدهای مخرب یا دسترسی‌های غیرمجاز به منابع سیستم شود، که امنیت کل سیستم را به خطر می‌اندازد.

سوال چهارم

انواع مدل‌های طراحی سیستم‌های عامل را نام ببرید و به صورت مختصر ساختار آن‌ها را نیز توضیح دهید.

پاسخ

بر اساس تقسیم‌بندی آقای می‌توان طراحی سیستم‌های عامل را به گروه‌های زیر تقسیم نمود:

۱. **Monolithic:**

ساده‌ترین ساختار برای سازماندهی یک سیستم‌عامل، نبود هیچ ساختاری است. یعنی تمام عملکردهای هسته را در یک فایل باینری واحد و ایستا که در یک فضای آدرس اجرا می‌شود، قرار دهیم. در این مدل، کل سیستم عامل به عنوان یک واحد پیوسته طراحی شده است. همه عملکردها و سرویس‌ها در یک هسته بزرگ قرار می‌گیرند و این هسته تمام وظایف سیستم عامل را مدیریت می‌کند. تمام سرویس‌ها (مانند مدیریت حافظه، زمان‌بندی فرآیندها، مدیریت فایل و ...) در یک لایه قرار دارند و کرنل مسئول اجرای مستقیم این سرویس‌هاست. تمامی بخش‌های سیستم به یکدیگر متصل هستند و هر بخش می‌تواند به بخش‌های دیگر دسترسی داشته باشد.

- مزایا: سرعت بالا به دلیل نبود واسطه‌های اضافی.
- معایب: به دلیل یکپارچه بودن، خطا در یک بخش می‌تواند کل سیستم را تحت تأثیر قرار دهد و دیباگ کردن سیستم پیچیده است.

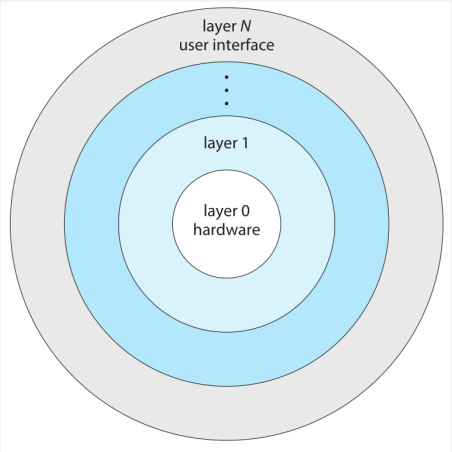
۲. **Layered:**

رویکرد یکپارچه اغلب به‌عنوان یک سیستم به‌هم‌پیوسته شناخته می‌شود، زیرا تغییرات در یک بخش از سیستم می‌تواند تأثیرات گسترده‌ای روی بخش‌های دیگر داشته باشد. در مقابل، می‌توانیم سیستمی طراحی کنیم که به‌طور جداگانه به‌هم‌پیوسته است. چنین سیستمی به اجزای جداگانه و کوچک‌تری تقسیم می‌شود که عملکردهای خاص و محدودی دارند. همه این اجزا با هم هسته را تشکیل می‌دهند. مزیت این رویکرد ماژولار این است که تغییرات در یک جزء تنها بر همان جزء تأثیر می‌گذارد و دیگر اجزا را تحت‌تأثیر قرار نمی‌دهد، که به مجریان سیستم آزادی بیشتری در ایجاد و تغییرات داخلی سیستم می‌دهد. یکی از روش‌های ماژولار کردن سیستم، رویکرد لایه‌ای است، که در آن سیستم‌عامل به تعدادی لایه (سطح) تقسیم می‌شود. لایه پایین (لایه ۰) سخت‌افزار است و بالاترین لایه (لایه N) رابط کاربری است. هر لایه یک عملکرد خاص دارد و هر لایه تنها به لایه‌های پایین‌تر خود دسترسی دارد و از لایه‌های بالاتر مستقل است. لایه‌های بالاتر از طریق لایه‌های پایین‌تر به سخت‌افزار دسترسی پیدا می‌کنند.

- مزایا: طراحی منظم و ساده‌تر برای مدیریت و دیباگ کردن، چرا که هر لایه وظایف محدودی دارد.
- معایب: کندتر بودن نسبت به سیستم‌های تک‌لایه‌ای به دلیل وجود واسطه بین لایه‌ها.

شکل زیر ساختار این مدل است:

پاسخ

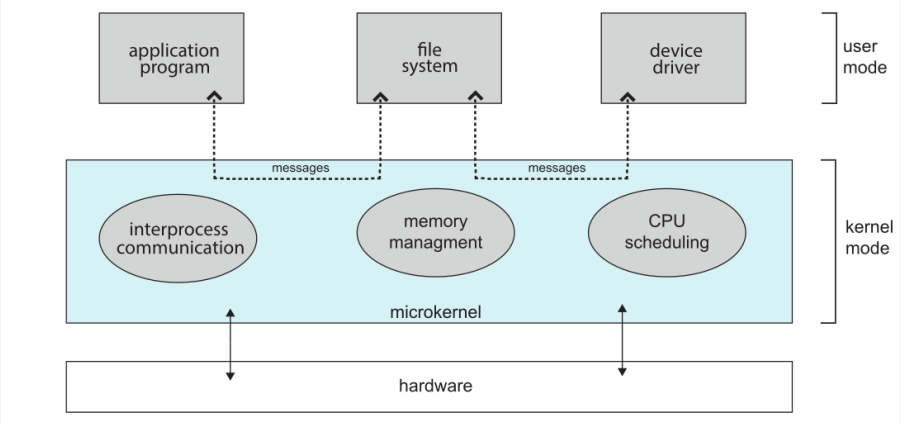


شکل ۵: ساختار لایه‌ای

۳. **Microkernels:**
در مدل ریزهسته‌ای، هسته ماژولار شده است. یعنی تمام اجزای غیرضروری از هسته حذف و به‌عنوان برنامه‌های سطح کاربر در فضاهای آدرس جداگانه پیاده‌سازی شدند. نتیجه آن کوچکتر شدن هسته است. در مورد اینکه کدام خدمات باید در هسته باقی بمانند و کدام باید در فضای کاربر پیاده‌سازی شوند، توافق چندانی وجود ندارد. به‌طور معمول، ریز هسته‌ها مدیریت حداقلی پردازش و حافظه را همراه با یک امکان ارتباطی فراهم می‌کنند

- مزایا: ایمنی و پایداری بالاتر؛ چرا که خطا در یکی از سرویس‌ها بر کل سیستم تأثیر نمی‌گذارد.
- معایب: عملکرد کندتر به دلیل نیاز به پیام‌رسانی بین سرویس‌های مختلف.

شکل زیر ساختار این مدل است:



شکل ۶: ساختار ریزهسته

۴. **Modules:**
یکی از بهترین روش‌های طراحی سیستم‌عامل در حال حاضر، استفاده از ماژول‌های بارگذاری‌شونده هسته (LKMs) است. در این طراحی، هسته شامل مجموعه‌ای از اجزای اصلی است و می‌تواند خدمات اضافی را از طریق ماژول‌ها، در زمان بوت یا حین اجرا، لینک کند.

پاسخ

این نوع طراحی در بسیاری از پیاده‌سازی‌های مدرن UNIX مانند لینوکس، macOS، و سولاریس و همچنین ویندوز رایج است. ایده این طراحی این است که هسته خدمات اصلی را ارائه دهد، در حالی که سایر خدمات به‌طور پویا در حین اجرای هسته پیاده‌سازی می‌شوند. لینوکس به‌طور گسترده از ماژول‌های بارگذاری‌شونده برای پشتیبانی از درایورهای دستگاه‌ها و سیستم‌های فایل استفاده می‌کند. این ماژول‌ها می‌توانند در حین اجرا به هسته اضافه یا حذف شوند و عملکرد هسته پویا و ماژولار را فراهم کنند، در حالی که مزایای عملکردی سیستم‌های یکپارچه را نیز حفظ می‌کنند.

- مزایا: انعطاف‌پذیری بیشتر و قابلیت گسترش سیستم عامل بدون نیاز به بازطراحی کل سیستم.
- معایب: پیچیدگی بیشتر در مدیریت ماژول‌ها.

۵. **Hybrid:**
در عمل، تعداد بسیار کمی از سیستم‌عامل‌ها از یک ساختار تعریف‌شده و واحد استفاده می‌کنند. در عوض، آن‌ها ترکیبی از ساختارهای مختلف را به کار می‌گیرند و به سیستم‌های ترکیبی تبدیل می‌شوند که به مسائل مربوط به عملکرد، امنیت، و کاربرپسندی رسیدگی می‌کنند. به عنوان مثال، لینوکس یک سیستم یکپارچه است، زیرا داشتن سیستم‌عامل در یک فضای آدرس واحد عملکرد بسیار بالایی را فراهم می‌کند. با این حال، لینوکس ماژولار است و امکان افزودن قابلیت‌های جدید به‌طور پویا به هسته را فراهم می‌کند. ویندوز نیز عمدتاً یکپارچه است، اما برخی از ویژگی‌های سیستم‌های ریز هسته‌ای را حفظ کرده است، مانند پشتیبانی از زیرسیستم‌های جداگانه که به‌عنوان فرآیندهای حالت کاربر اجرا می‌شوند.

- مزایا: استفاده از مزایای هر روش :)
- معایب: -

سوال پنجم

به سوالات زیر در رابطه با مدل‌های سیستم‌های عامل پاسخ دهید.

۱. سیستم‌های عامل اولیه از چه مدلی پیروی می‌کردند و دو مورد از معایب این مدل را توضیح دهید.

پاسخ

سیستم‌های عامل اولیه عمدتاً از مدل Monolithic پیروی می‌کردند. در این مدل، تمام بخش‌های سیستم عامل به صورت یکپارچه درون یک هسته بزرگ قرار داشتند.

- مشکل در نگهداری و دیباگ کردن: چون همه بخش‌های سیستم عامل در یک هسته واحد قرار داشتند، خطا در یکی از بخش‌ها می‌توانست کل سیستم را تحت تأثیر قرار دهد. این موضوع باعث می‌شد دیباگ کردن و نگهداری سیستم پیچیده و زمان‌بر باشد.
- عدم انعطاف‌پذیری: در این مدل، اضافه کردن یا حذف یک سرویس یا قابلیت جدید به سیستم عامل بسیار دشوار بود، چرا که تمام بخش‌های سیستم به یکدیگر وابسته بودند. هر تغییری نیازمند بازبینی و بازطراحی بخش‌های دیگر بود.

۲. فواید کاهش سایز kernel (در مدل Microkernel) چیست؟ سه نمونه از عملیات‌هایی که در Kernel نگه داشته می‌شوند را نام ببرید.

پاسخ

- افزایش پایداری و امنیت: با کوچک کردن هسته، بسیاری از سرویس‌ها از هسته خارج می‌شوند و به‌عنوان سرویس‌های کاربری پیاده‌سازی می‌شوند. این باعث می‌شود که اگر یکی از این سرویس‌ها دچار مشکل شود، بر عملکرد هسته تأثیر نگذارد و سیستم به کار خود ادامه دهد.
- قابلیت گسترش و انعطاف‌پذیری: در مدل ریزهسته‌ای، سرویس‌های اضافی مانند مدیریت فایل‌ها یا درایورهای دستگاه‌ها می‌توانند به‌طور جداگانه بارگذاری یا حذف شوند. این امر امکان گسترش سیستم بدون نیاز به تغییر در هسته اصلی را فراهم می‌کند.
- افزایش امنیت: به دلیل کوچک بودن هسته، تعداد عملیات‌هایی که در سطح هسته انجام می‌شوند کاهش می‌یابد، و همین باعث می‌شود که نقاط آسیب‌پذیر کمتری برای حملات وجود داشته باشد.

۳. تفاوت میان مدل لایه ای و مدل ماژولار چیست و چه عاملی باعث برتری مدل ماژولار می‌شود؟

پاسخ

- ساختار: در مدل لایه‌ای، سیستم عامل به لایه‌های مجزا تقسیم می‌شود که هر لایه فقط می‌تواند با لایه‌های مجاور خود ارتباط برقرار کند. در حالی که در مدل ماژولار، سیستم عامل به ماژول‌های جداگانه‌ای تقسیم می‌شود که هر ماژول به صورت مستقل از ماژول‌های دیگر قابل بارگذاری و حذف است و نیازی به وابستگی به لایه‌های مجاور ندارد.
- ارتباط بین بخش‌ها: در مدل لایه‌ای، ارتباط بین لایه‌ها باید به ترتیب و از پایین به بالا یا بالعکس انجام شود. اما در مدل ماژولار، هر ماژول می‌تواند به طور مستقیم با هر بخش دیگری از سیستم ارتباط برقرار کند.

پاسخ

- انعطاف‌پذیری مدل ماژولار بر مدل لایه‌ای برتری دارد. در مدل ماژولار، هر ماژول به صورت مستقل قابل توسعه، بارگذاری یا حذف است، بدون اینکه نیاز باشد کل سیستم عامل را تغییر داد یا از نو طراحی کرد. این امکان باعث می‌شود سیستم عامل در مواجهه با نیازهای جدید یا تغییرات سخت‌افزاری به‌سادگی گسترش یابد.