سوال اول

الف) چه تفاوت‌هایی بین این دو mode وجود دارد؟

در حالت Kernel، کد اجرایی (کدهای هسته سیستم عامل) دسترسی کامل و بدون محدودیت به سخت افزار و منابع سیستم دارد، در حالی که در حالت User کد اجرایی (کدهای کاربر) دسترسی مستقیم به حافظه اصلی یا سخت افزار را ندارد و دسترسی محدود است. به این ترتیب بد افزارها یا کاربرهای بدخیم نمی توانند کد هسته سیستم عامل را تغییر دهند و به سخت افزار دسترسی داشته باشند.

هنگام بوت و بارگذاری سیستم عامل، سیستم در حالت kernel قرار دارد و پس از آن به حالت User تغییر داده می‌شود و برنامه‌های کاربر اجرا می‌شود. در صورت فراخوانی‌های سیستمی یا ایجاد وقفه سیستم مجدد به حالت kernel می‌رود.

ب) سه دستور نام ببرید که سیستم عامل باید روی اجرای آن‌ها کنترل اعمال کند. در کل چه نوع دستوراتی نیازمند کنترل شدن هستند؟

تغییر و پاک کردن حافظه، تغییر آدرس‌های حافظه، دستورالعمل‌های I/O و دستور halt، خاموش کردن دریافت وقفه و ... از جمله دستوراتی هستند که باید تحت نظارت و کنترل سیستم عامل انجام شوند که به آن‌ها دستورات Privileged گفته می‌شود.

ج) این کنترل نرم افزاری انجام می‌شود یا سخت افزاری؟ توضیح دهید، آیا می‌توان بیش از دو mode مختلف دسترسی تعریف کرد (مثلا برای مجازی سازی)؟ چطور؟

این پیاده سازی به صورت سخت افزاری انجام می‌شود. در حالت عادی یک بیت برای تعیین حالت سیستم (Kernel mode, User mode) کافی است. حال در صورتی که CPU قابلیت پشتیبانی از چند حالت را داشته باشد با در نظر گرفتن بیت‌های بیشتر می‌توان حالت‌های بیشتری تعریف نمود. به طور مثال با دو بیت می‌توان 4 حالت ایجاد کرد و یک مقدار را برای حالت user و یک مقدار را برای حالت kernel در نظر بگیریم و دو مقدار دیگر به عنوان حالت‌های میانی در نظر گرفته شود.

سوال دوم

الف) اول برای اینکه ماهان مطمئن بشه که جوابی که میگیره معتبره توضیح بدین که ساختار سیستم عامل و تفاوتش با معماري اون چیه؟

چون سیستم عامل یک برنامه بزرگ و پیچیده است، باید به دقت مهندسی شود تا به خوبی کار کند و راحت تغییر داده شود. یک راه این است که به ماژول‌ها و کامپوننت‌های کوچکتر بشکنیم که هر ماژول یک بخش خوش تعریف از سیستم باشد. به طور کلی میتوان از ساختارهای ساده مثل ساختار MS-DOS، پیچیده تر مثل ساختار UNIX، ساختار لایه‌ای و یا microkernel استفاده کرد. به طور کلی، ساختار روش‌های ارتباط بخش‌های مختلف سیستم عامل به یکدیگر است.

معماری سیستم شامل ارتباط هسته‌ها و پردازنده‌ها و مواردی از این قبیل است که دو حالت تک پردازنده‌ای و چند پردازنده‌ای دارد. سیستم تک پردازنده ای، یک پردازنده و یک هسته دارد. سیستم‌های چند پردازنده‌ای میتواند دو یا چند پردازنده داشته باشد که هرکدام یک CPU تک هسته‌ای دارند. در این حالت، bus و گاهی اوقات clock، حافظه و peripheralها به اشتراک گذاشته میشوند. تعریف سیستم‌های چندپردازنده‌ای با گذر زمان، شامل سیستم‌های چند هسته‌ای می‌شود که در آن‌ها چند هسته روی یک chip قرار می‌گیرند که از سیستم‌ها ی با چند chip که هرکدام یک هسته دارند کارایی بیشتری دارند و ارتباط هسته‌ها نیز سریع‌تر است.

ب) سه ساختاري که بالا گفته شد رو از نظر معیارهاي طراحی، قابلیت اطمینان، قابلیت اشکال زدایی و کارایی کلی با هم مقایسه کنید و دلیل خودتون هم براي هر مورد ذکر کنید.

ساختار monolithic: چون فانکشن‌ها ی زیاد در یک لایه قرار میگیرند به آن یکپارچه گفته میشود. در این ساختار، kernel شامل همه چیز بین اینترفیس فراخوانی سیستمی و سخت افزار است. ساده ترین ساختار است که در آن تمام عملکرد kernel داخل یک فایل باینری استاتیک قرار میگیرد و در یک فضای حافظه اجرا میشود. با وجود سادگی ظاهری، پیاده سازی و گسترش این ساختار دشوار است. با این حال، ارتباطات درون kernel سریع است و overhead کمتری هنگام فراخوانی سیستمی پیش می‌آید و سرعت و کارایی بالایی دارد.

سطح حمله در این ساختار گسترده است که یکی از مشکلات آن است و در نتیجه امنیت پایین تری دارد و چون همه بخش‌ها در دست kernel است قابلیت اطمینان هم کمتر است چون ممکن است از کار افتادن یک قسمت روی بقیه هم اثرگذار باشد. چون همه بخش‌ها یکجاست، تغییر در یک قسمت تاثیر زیادی روی بقیه قسمت‌ها خواهد داشت و اشکال زدایی نسبت به microkernel دشوارتر است چون همه قسمت‌ها به هم وابسته و در یک unit هستند.

ساختار microkernel: در این حالت، کامپوننت‌های غیر ضروری از kernel حذف شده و به عنوان برنامه‌های user پیاده سازی می‌شوند و در آدرس‌های جدا قرار میگیرند. در نتیجه، kernel کوچک تر خواهیم داشت.

کاربرد اصلی این است که بین برنامه client و بقیه سرویس‌هایی که در فضای user در حال اجرا هستند، ارتباط برقرار شود. این ارتباط توسط message passing انجام می‌شود که در این حالت client و سرویس به صورت غیر مستقیم با یکدیگر ارتباط دارند و پیام‌ها را از طریق microkernel منتقل میکنند. از فواید این ساختار این است که گسترش سیستم عامل را آسان تر میکند. تمام سرویس‌های جدید به فضای user اضافه میشوند و به اصلاح و تغییر kernel نیازی نخواهد بود و در نتیجه تغییرات کمتر میشوند چون kernel هم کوچک تر است. انتقال سیستم عامل در این ساختار از یک طراحی سخت افزاری به بقیه راحت تر است. پیاده سازی و نگهداری آسان تر است. این ساختار امنیت و قابلیت اطمینان بیشتری دارد زیرا بیشتر سرویس‌ها به عنوان user اجرا می‌شوند و اگر سرویسی از کار بیفتد، روی بقیه اجزا تاثیری نخواهد داشت. امنیت به دلیل کوچک شدن سطح حمله بالاتر و به دلیل کوچک شدن بخش‌ها اشکال زدایی هم راحت‌تر است.

کارایی این ساختار به علت افزایش overhead نسبت به حالت قبل پایین می‌آید. مثلا وقتی دو سرویس user-level قرار است ارتباط برقرار کنند، پیام‌ها باید بین سرویس‌ها که هر کدام در فضای جدا قرار دارند کپی شود.

همچنین ممکن است سیستم عامل مجبور شود از یک process به دیگری سوییچ کند که پیام‌ها را مبادله کند.

Overheadای که در این مراحل ایجاد میشود بزرگترین مانع گسترش سیستم عامل‌های مبتنی بر این ساختار است.

ترکیب monolithic و modular: میتوان یک سیستم را به بخش‌های کوچک تر تقسیم کرد که هر کدام کاربرد خاصی داشته و در کنار هم kernel را تشکیل دهند. (loosely coupled) یک سیستم به روش‌های مختلفی می‌تواند ماژولار شود که ساختار لایه‌ای یکی از این روش هاست. در این حالت، سیستم عامل به لایه‌های مختلفی شکسته میشود که پایین‌ترین آن‌ها سخت افزار و بالاترین interface است. هر لایه شامل یک سری ساختمان داده و فانکشن است. فایده اصلی این روش سادگی پیاده سازی و اشکال زدایی است. مثلا لایه اول میتواند بدون تاثیری روی بقیه سیستم، اشکال زدایی شود. وقتی این لایه اشکال زدایی شود، سراغ لایه دوم می‌رود و به همین ترتیب تا آخر. در این صورت میتوان مطمئن شد که اگر یک خطا در لایه‌ای پیدا شود، این خطا مربوط به همان لایه است چون لایه‌های پایینی اشکال زدایی شده‌اند. بنابراین طراحی و پیاده سازی سیستم آسان میشود. یکی از سیستم‌های عاملی که از ترکیب این دو استفاده می‌کند، UNIX است که در آن، دو بخش مجزای kernel و system program است

که خود kernel به چند اینترفیس و device driver تقسیم می‌شود که به همین دلیل می‌توان آن را ترکیبی از monolithic و modular دانست. این حالت مزایا و معایب هر دو روش را شامل میشود.

به صورت کلی، بهترین شیوه استفاده از LKM هاست که در آن، kernel سرویس‌های هسته را انجام داده و سایر سرویس‌ها به صورت پویا پیاده‌سازی می‌شوند.

** نمره این قسمت به دانشجویانی که سوال را با توجه به ساختارهای monolithic microkernel modular به درستی پاسخ داده‌اند نیز به صورت کامل تعلق گرفته است.

سوال سوم

الف) مزاياي استفاده از اين واسطها را بيان كنيد.

این واسطها (API) مجموعه‌ای از فانکشن‌ها که برای برنامه‌نویس قابل استفاده‌اند، پارامترهایی که به هر فانکشن داده می‌شود و آن فانکشن برمیگرداند را مشخص میکند. از دلایلی که برنامه نویسان ترجیح میدهند از API استفاده کنند، میتوان به آنها اشاره کرد: این که برنامه روی هر سیستمی که همان API را ساپورت میکند اجرا شود. (portability)، بالا بردن سرعت، اینکه لازم نیست برنامه نویس از تمام جزئیات فراخوانی سیستمی مطلع باشد و می‌تواند کارها را به سیستم عامل بسپارد، یکپارچه شدن و بالا رفتن انعطاف پذیری و کارایی برنامه‌ها و افزایش اسکوپ.

ب) سه شيوه رو با يكديگر مقايسه كنيد و معايب و مزاياى هركدام را نام ببريد.

کامپایلر کد در زبان سطح بالا مثل جاوا را میگیرد و به زبان سطح پایین و قابل فهم برای ماشین تبدیل می‌کند. در این روش ابتدا کد کامپایل شده و سپس وارد مرحله اجرا می‌شود. در این روش کد یک جا کامپایل می‌شود و خط به خط نیست و در نتیجه اشکال زدایی سخت تر است. چون کامپایلر وابسته به سیستم عامل است، فقط روی سیستم عاملی که آن کامپایلر برایش طراحی شده میتواند اجرا شود. این روش چون یک جا کامپایل می‌کند نسبت به مفسر سریع تر است.

روش مفسری: این روش در ساختار شبیه روش کامپایل است، با این تفاوت که مفسر دستورالعمل‌ها را مستقیما از منبع اصلی اجرا کرده و آن‌ها را ترجمه نمی کند. مفسر برنامه را به صورت خط به خط اجرا میکند و در نتیجه نسبت به کامپایلر، اشکال زدایی راحت‌تری دارد. این روش سرعت کمتر و انعطاف پذیری بیشتری دارد. مفسر وابسته به سیستم عامل نیست که استفاده راحت تری خواهد داشت.

روش ماشین مجازی: در این روش، ابتدا کامپایلر کد را به یک فایل میانی تبدیل می‌کند که این فایل وابسته به سیستم عامل است و سپس مفسر این فایل را به یک زبان قابل فهم برای ماشین تبدیل می‌کند که در این حالت دیگر به سیستم عامل وابسته نیست. خود ماشین مجازی نیز به سیستم عامل وابسته نیست. کارایی و سرعت نسبت به دو حالت قبل بیشتر است.

ج) سيستم عامل لينوكس از چند راه براي عبور متغيرها (Parameter Passing) استفاده مى‌كند. آن‌ها را ذكر كرده و توضيح دهيد كه استفاده تركيبى از آن‌ها چه فايده‌هايى دارد؟

میتوان پارامترها را در رجیسترهای مختلف ذخیره کرد. چون تعداد رجیسترها معمولا محدود است، ممکن است از تعداد پارامترها کمتر شود. حالت بعدی این است که پارامترها را در استک

push و pop کنیم و حالت سوم این است که پارامترها را در بلاک‌ها یا خانه‌های حافظه ذخیره کنیم و آدرس آن در حافظه را در رجیسترها بریزیم که این روش تعداد زیادی پارامتر را هم پشتیبانی میکند. در این روش سرعت پایین تر است چون ارتباط مستقیم با حافظه زمان‌برتر از ارتباط با رجیسترها است. استفاده از هر سه روش بنا به تعداد پارامترها بهترین حالت است. اگر تعداد پارامترها زیاد باشد باید از روش سوم که کند تر است استفاده کنیم و در غیر این صورت می‌توانیم پارامترها را در رجیستر ذخیره کنیم.