دانشگاه صنعتی امیرکبیر
(پلی تکنیک تهران)

دانشکده مهندسی کامپیوتر

به نام خدا

پاسخ تمرین سری سوم درس سیستم های عامل

پاییز 1403

استاد درس: دکتر زرندی

سوال اول)

به سوالات زیر در رابطه با محیط های محاسباتی (Computing environment) پاسخ دهید

الف) مدل های Client-server و Peer to peer را تعریف و با یکدیگر مقایسه کنید.

مدل Client-server: دو گره (Node) تحت عنوان Client و Server دارد که سرور نقش سرویس دهنده به کلاینت را دارد (کلاینت: درخواست کننده خدمات یا منابع، سرور: ارائه دهنده خدمات یا منابع)

مدل Peer to peer: در این مدل، تمام گره‌ها (nodes) نقش‌های مشابه ای دارند و می‌توانند هم به‌عنوان ارائه‌دهنده و هم به‌عنوان دریافت‌کننده خدمات یا منابع عمل کنند. هیچ سرور مرکزی وجود ندارد؛ هر گره مستقیماً با سایر گره‌ها تعامل می‌کند.

مقایسه:

ویژگی	Client-server	Peer to peer
ساختار	متمرکز	غیر متمرکز
هزینه اجرا	هزینه بالا به دلیل نیاز به سرور قدرتمند	هزینه کمتر به دلیل عدم نیاز به سرور مرکزی
امنیت	امنیت بیشتر (کنترل توسط سرور)	امنیت کمتر
مقیاس پذیری	محدود به ظرفیت سرور	مقیاس پذیری بالا
قابلیت اطمینان	اگر سرور از کار بیفتند، سیستم مختل می شود	گره های متعدد مانع از خرابی کل سیستم می شوند
سرعت	ممکن است با افزایش کاربران کاهش یابد	بسته به تعداد گره ها متغیر است

کاربرد ها	**سیستم های متمرکز (بانک، وبسایت)**	**اشتراک گذاری فایل و شبکه های غیر متمرکز**

ب) Emulation و Virtualization را تعریف کنید و تفاوت های آن ها را ذکر کنید.

شبیه سازی (Emulation): فرایندی که در آن یک سیستم، سخت‌افزار یا نرم‌افزاری را شبیه‌سازی می‌کند که با سیستم اصلی سازگار نیست. در این روش یک سطح سخت افزار یا نرم افزار به صورت کامل تقلید می شود

مجازی سازی (Virtualization): فرایندی که به چند سیستم عامل اجازه می دهد که مستقیما بر روی سخت افزار موجود اجرا شوند

مقایسه:

ویژگی	Emulation	Virtualization
هدف	**شبیه‌سازی سخت‌افزار یا محیطی ناسازگار**	**اشتراک منابع سخت‌افزاری بین سیستم‌های سازگار**
کاربرد	**اجرای نرم‌افزارهای غیرسازگار یا سخت‌افزارهای قدیمی**	**اجرای سیستم‌عامل‌ها یا محیط‌های مجازی**
عملکرد	**به دلیل تقلید کامل، کندتر از سیستم واقعی است**	**عملکرد نزدیک به سیستم واقعی، با اندکی افت**
سازگاری سخت افزاری	**نیازمند شبیه‌سازی کامل سخت‌افزار**	**نیازمند سازگاری سخت‌افزار با Host**
مثال ها	**QEMU، RetroArch، Wine**	**VMware، VirtualBox، KVM**
سطح دسترسی	**به سخت‌افزار میزبان محدود**	**نیازمند سازگاری سخت‌افزاری**

	نیست	Host با Guest

ج) سه نمونه از دسته سرویس های ابری را نام ببرید و به صورت مختصر توضیح دهید.

سرویس IaaS: ارائه زیرساخت‌های فیزیکی یا مجازی مانند سرورها، فضای ذخیره‌سازی، شبکه‌ها، و ماشین‌های مجازی به‌صورت ابری.

سرویس PaaS: ارائه محیطی برای توسعه، تست، و اجرای برنامه‌ها بدون نیاز به مدیریت مستقیم زیرساخت‌های سخت‌افزاری یا نرم‌افزاری.

سرویس SaaS: ارائه نرم‌افزارهای کاربردی آماده که به‌صورت آنلاین در دسترس هستند و نیاز به نصب یا مدیریت توسط کاربر ندارند.

سوال دوم)

دو روش برای ارتباط میان فرایند ها Message passing و Shared memory است. آن ها را تعریف کنید و با یکدیگر مقایسه کنید.

روش Message Passing: در این روش، فرآیندها با ارسال و دریافت پیام به یکدیگر ارتباط برقرار می‌کنند. پیام‌ها می‌توانند شامل داده‌ها یا سیگنال‌ها باشند.

روش Shared Memory: در این روش، بخش خاصی از حافظه توسط چندین فرآیند به اشتراک گذاشته می‌شود. فرآیندها می‌توانند مستقیماً داده‌ها را در این حافظه بخوانند یا بنویسند.

ویژگی	Message Passing	Shared Memory
روش ارتباط	ارسال و دریافت پیام	دسترسی مستقیم به حافظه اشتراکی
سرعت	کندتر (به دلیل نیاز به تعامل با سیستم‌عامل)	سریع‌تر (بدون واسطه مستقیم)

پیچیدگی مدیریت	**ساده‌تر؛ سیستم‌عامل ارتباط را مدیریت می‌کند**	**پیچیده‌تر؛ نیازمند همگام‌سازی دقیق**
قابلیت استفاده در شبکه	**مناسب برای سیستم‌های توزیع‌شده**	**محدود به سیستم‌های محلی**
انعطاف‌پذیری	**مستقل از معماری حافظه**	**نیازمند اشتراک‌گذاری حافظه**
همگام‌سازی (Race Condition)	**نیازی به همگام‌سازی مستقیم بین فرآیندها نیست**	**نیازمند ابزارهای همگام‌سازی (Semaphore، Mutex)**
کاربرد ها	**مناسب برای سیستم‌های غیرمتمرکز**	**مناسب برای سیستم‌های متمرکز**

سوال سوم)

از میان عملیات هایی که نیاز به System Call دارند ۳ مثال نام ببرید و توضیح دهید که اگر هر عملیات در لایه User انجام میشد و نیاز به System Call نداشت چه مشکل هایی می توانست به وجود بیاورد.

الف) ایجاد یا باز کردن فایل: برای دسترسی به فایل‌ها، سیستم‌عامل باید اجازه دسترسی به دیسک را صادر کند، وضعیت فایل‌ها را بررسی کند (وجود داشتن یا نداشتن)، و مکان‌های ذخیره‌سازی را مدیریت کند.

مشکل در لایه User:

- **دسترسی مستقیم به دیسک می‌تواند امنیت سیستم را به خطر بیندازد (به‌عنوان‌مثال، یک برنامه مخرب ممکن است به داده‌های حساس دسترسی پیدا کند).**

- ممکن است چندین فرآیند هم‌زمان به فایل‌ها دسترسی پیدا کنند و بدون مدیریت سیستم‌عامل، مشکلاتی مانند **Race Condition** به وجود آید.
- کاربر ممکن است داده‌ها را در بخش اشتباه دیسک ذخیره کند یا ساختار فایل را تخریب کند.

ب) ارتباط با شبکه: ارتباط با شبکه نیازمند مدیریت دقیق منابع شبکه، سوکت‌ها، و اطمینان از استفاده ایمن و کنترل‌شده از منابع شبکه است.

مشکل در لایه **User**:

- یک برنامه می‌تواند منابع شبکه را بدون محدودیت مصرف کند و باعث اختلال در سرویس‌دهی (**Denial of Service**) شود.
- اطلاعات حساس ممکن است بدون رمزگذاری مناسب ارسال شوند.
- برنامه‌های کاربر ممکن است به شبکه دسترسی ناامن داشته باشند و خطر حملات سایبری را افزایش دهند.

ج) مدیریت فرایند ها: سیستم‌عامل وظیفه مدیریت فرآیندها را دارد (مانند تخصیص منابع، زمان‌بندی، و خاتمه فرآیندها). فرآیندها نمی‌توانند مستقیماً فرآیند دیگری را ایجاد یا حذف کنند.

مشکل در لایه **User**:

- یک برنامه ممکن است تعداد زیادی فرآیند ایجاد کند و باعث کمبود منابع سیستم (مانند حافظه و پردازنده) شود.
- فرآیندهای کاربر می‌توانند فرآیندهای دیگر را به‌طور نادرست خاتمه دهند یا مختل کنند، که منجر به از دست دادن داده‌ها یا بی‌ثباتی سیستم می‌شود.
- نظارت سیستم‌عامل بر حیات فرآیندها از بین می‌رود، که ممکن است منجر به فرآیندهای معلق یا بی‌هدف شود.

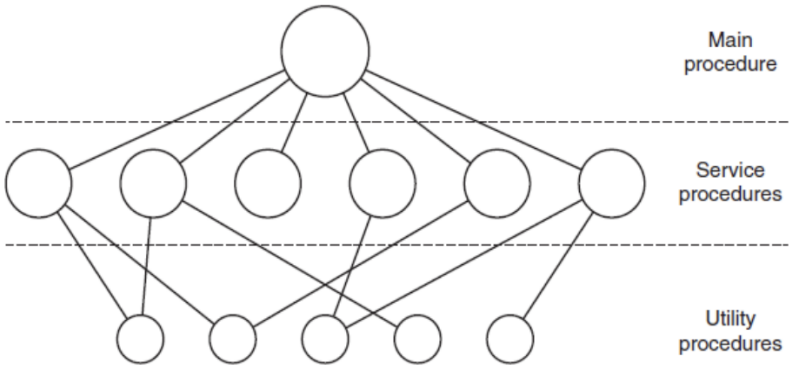
* مثال های دیگر نیز مورد قبول هستند *

سوال چهارم)

انواع مدل های طراحی سیستم های عامل را نام ببرید و به صورت مختصر ساختار آن ها را توضیح دهید.

OS structure: 1. Simple structure (Monolithic)

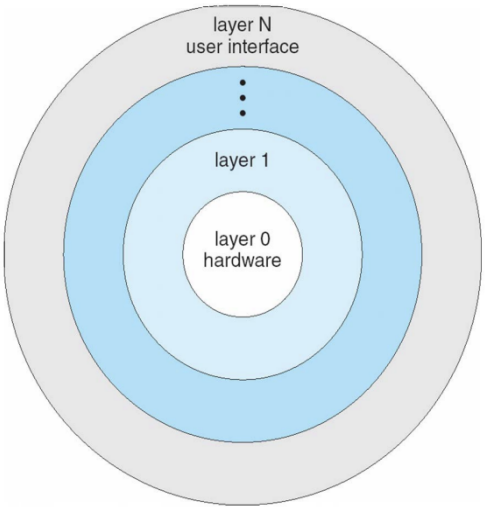
- The most common organization
- OS is a single large program in kernel mode
- Problems
  - Crash in called procedures?
  - Unwieldy & difficult to understand



A simple structuring model for a monolithic system

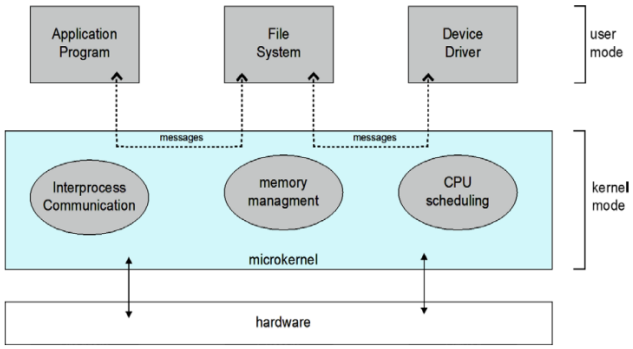
OS structure: 2. Layered approach

- Layered approach
  - abstractions
  - adv.
    - Simplicity
      - ✓ Construction
      - ✓ Debugging
    - Functions and operations of low layers
  - dis. adv.
    - Layer definition problem
      - ✓ MMU, backing store, scheduler (?)
    - Less efficient



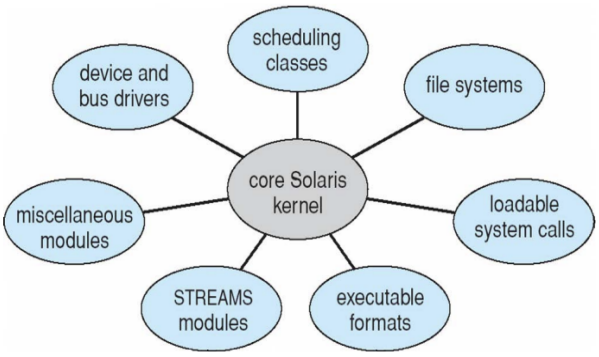
OS structure: 3. Microkernel

- Microkernel
  - Moves as much from the kernel into "*user*" space
  - Communication takes place between user modules using message passing
- Examples
  - Mach (CMU)
  - Mac OS X kernel (Darwin)
- Benefits:
  - Easier to extend a microkernel
  - Easier to port the operating system to new architectures
  - More reliable (less code is running in kernel mode)
  - More secure
- Detriments:
  - Performance overhead of user space to kernel space communication
  - Windows NT 4 (microkernel) slow!
    - vs. Windows XP (monolithic) fast!

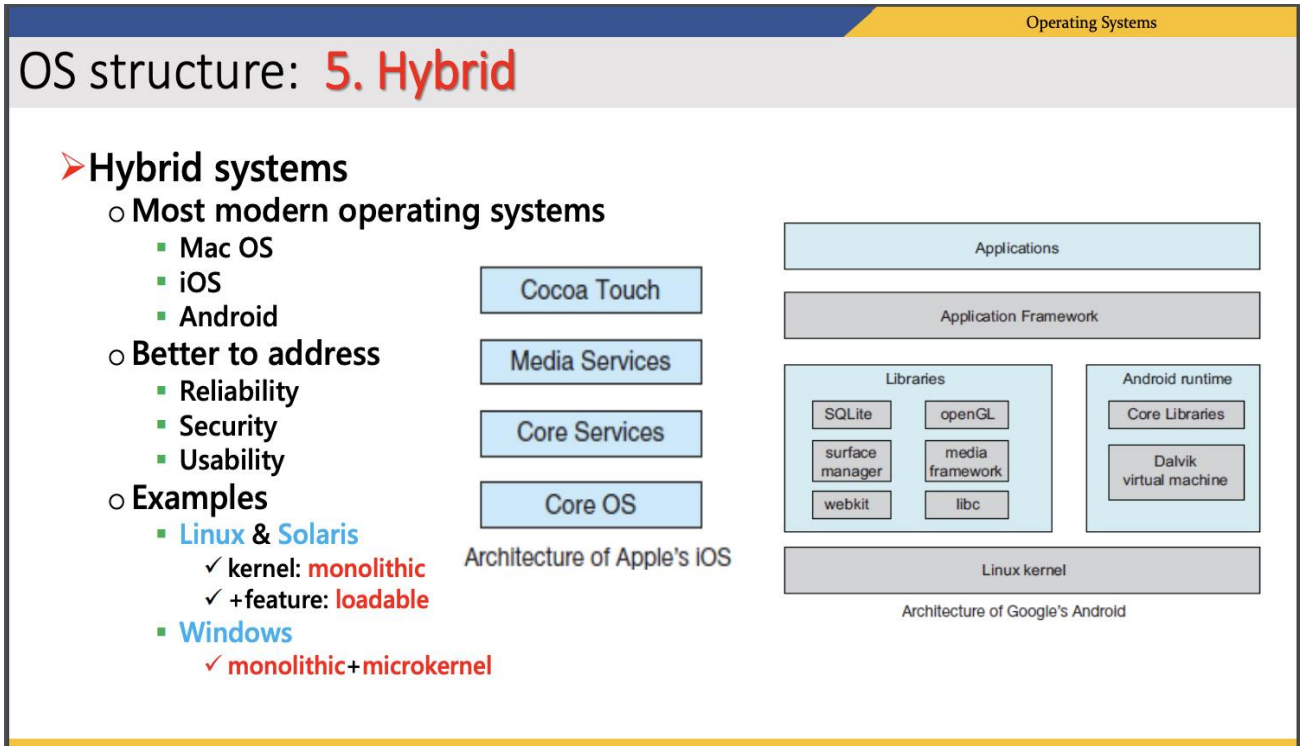


OS structure: 4. Modules

- Modules
  - Most modern operating systems implement kernel modules
    - Uses object-oriented approach
    - Each core component is separate
    - Each talks to the others over known interfaces
    - Each is loadable as needed within the kernel
    - Faster than microkernel
      - ✓ No need of message passing
    - Better than layered
      - ✓ Direct module communications
  - Overall, similar to layers but with more flexible
    - Linux, Solaris, etc



Solaris loadable modules



سوال پنجم)

به سوالات زیر در رابطه با مدل های سیستم های عامل پاسخ دهید.

الف) سیستم های عامل اولیه از چه مدلی پیروی میکردند و دو مورد از معایب این مدل را توضیح دهید.

سیستم عامل های اولیه از روش Monolithic استفاده می کردند که در آن تمام اجزای سیستم‌عامل (مانند مدیریت فایل، مدیریت حافظه، مدیریت دستگاه‌ها و ...) در یک فضای آدرس (Kernel Space) قرار دارند و به‌عنوان یک ماژول بزرگ عمل می‌کنند.

معایب:

- **پیچیدگی بالا: سیستم‌عامل‌های Monolithic شامل تعداد زیادی ماژول و کد هستند که همگی در فضای هسته اجرا می‌شوند. این پیچیدگی باعث می‌شود که فهم و مدیریت کد سخت‌تر شود.**
- **آسیب پذیری نسبت به خطا ها: از آنجا که تمام اجزا در یک فضای آدرس مشترک اجرا می‌شوند، اگر یک ماژول دچار خرابی شود (مانند اشکال در درایور یا مدیریت حافظه)، می‌تواند کل سیستم را مختل کند.**

- **کاهش امنیت: چون همه ماژول‌ها به فضای آدرس هسته دسترسی کامل دارند، یک اشکال امنیتی در هر ماژول می‌تواند به سوءاستفاده و دسترسی غیرمجاز به کل سیستم منجر شود.**
- **مقیاس پذیری پایین: افزودن ماژول‌های جدید یا سفارشی‌سازی سیستم‌عامل نیاز به دسترسی و تغییر در کل هسته دارد.**
- **دشواری در توسعه و نگهداری: هر تغییری در یک بخش از کد ممکن است به تغییرات گسترده در بخش‌های دیگر نیاز داشته باشد.**

ب) فواید کاهش سایز kernel (در مدل Microkernel) چیست؟ و سه نمونه از عملیات هایی در Kernel نگه داشته می شوند را نام ببرید.

فواید:

- **افزایش امنیت: کاهش تعداد اجزای موجود در هسته به معنای کاهش نقاط آسیب‌پذیری است. بسیاری از خدمات سیستم‌عامل (مانند مدیریت فایل و درایورها) به فضای کاربر منتقل می‌شوند، که در صورت وقوع خطا یا حمله، تأثیری بر هسته و سیستم اصلی نخواهد داشت.**
- **پایداری بیشتر: در صورت خرابی یا خطا در یکی از اجزای سیستم (مانند درایور)، این خرابی به هسته سرایت نمی‌کند و سیستم به‌طور کامل از کار نمی‌افتد.**
- **کاهش پیچیدگی: هسته کوچک‌تر به معنای کد کمتر و ساده‌تر است، که موجب کاهش احتمال بروز خطاهای برنامه‌نویسی می‌شود همچنین اشکال‌زدایی و نگهداری سیستم‌عامل آسان‌تر می‌شود.**
- **افزایش انعطاف پذیری: خدمات و ماژول‌ها می‌توانند به‌صورت مستقل توسعه داده شوند و به‌راحتی به سیستم اضافه یا از آن حذف شوند همچنین ارتقاء یا تغییر بخش‌های سیستم‌عامل بدون نیاز به بازسازی کامل هسته امکان‌پذیر است.**

عملیات هایی که در Kernel نگهداری می شوند: مدیریت فرایند ها (Process Management)، مدیریت حافظه و ارتباط میان فرایند ها (Inter process communication)

ج) تفاوت میان مدل لایه ای و مدل ماژولار چیست و چه عاملی باعث برتری مدر ماژولار می شود؟

در مدل لایه ای سیستم‌عامل به چندین لایه مرتب تقسیم می‌شود، به طوری که هر لایه تنها با لایه بالایی و پایینی خود تعامل دارد اما در مدل ماژولار سیستم‌عامل به مجموعه‌ای از ماژول‌های مستقل تقسیم می‌شود که هر کدام یک وظیفه خاص را انجام می‌دهند که این ماژول ها می‌توانند به‌صورت

پویا بارگذاری یا حذف شوند همچنین ارتباط ماژول‌ها به شکل مستقیم یا از طریق یک رابط مرکزی (مانند هسته) انجام می‌شود.

عوامل برتری ماژولار:

- **انعطاف پذیری: ماژول‌ها مستقل از یکدیگر هستند و می‌توان آن‌ها را بدون تأثیر بر سایر بخش‌ها تغییر داد، اضافه یا حذف کرد.**
- **بهبود کارایی: مدل ماژولار امکان تغییر یا ارتقاء یک بخش از سیستم‌عامل بدون تأثیر بر سایر بخش‌ها را فراهم می‌کند.**
- **کاهش پیچیدگی: تفکیک وظایف به ماژول‌های مستقل باعث می‌شود مدیریت کد ساده‌تر و توسعه آسان‌تر شود.**
- **بهبود پایداری و امنیت: اگر یک ماژول خراب شود، کل سیستم دچار مشکل نمی‌شود. خرابی محدود به همان ماژول باقی می‌ماند.**
- **قابلیت بارگذاری پویا: ماژول‌ها می‌توانند به‌صورت پویا در حین اجرای سیستم‌عامل بارگذاری یا حذف شوند (مانند درایورهای دستگاه).**