به نام خدا

دانشگاه صنعتی امیرکبیر
(پلی تکنیک تهران)

سیستم‌های عامل (بهار ۱۴۰۱)

پاسخنامه تمرین اول

استاد درس:

دکتر جوادی

1. نمودار *Process State Transition* را در نظر بگیرید.

الف) آیا می‌توان یک انتقال مستقیم از موقعیت *waiting* برای *I/O* به موقعیت *terminated* داشت. چرا؟

ب) مشخص کنید که در هر یک از موارد زیر، از چه وضعیتی به چه وضعیتی انتقال انجام می‌شود؟ (موارد از یکدیگر مجزا هستند و به هم ارتباطی ندارند)

- پردازه‌ی p (که در حال اجراست)، دستوری را برای خواندن از دیسک اجرا می‌کند.
- مدت زمانی که *CPU* به پردازه‌ی p اختصاص یافته‌است، تمام می‌شود.
- اجرای پردازه‌ی p به پایان می‌رسد.
- خواندن از دیسک برای پردازه p تمام می‌شود..

الف) خیر. با توجه به دیاگرام صفحه‌ی ۱۲ اسلاید پردازه، ابتدا پردازه باید وارد ready queue شود و سپس وارد running شود و در نهایت وارد terminated شود

ب)

1) از running به waiting
2) از running به ready
3) از running به terminated
4) از waiting به ready

2. در مورد *Multiprogramming* به سوالات زیر پاسخ دهید:

الف) تفاوت آن با *Multitasking* را بیان کنید.

ب) اگر یک سیستم تک پردازنده و تک هسته‌ای داشته‌باشیم که از *Multiprogramming* پشتیبانی کند، در هر لحظه چند پردازه می‌تواند در وضعیت *running* باشد؟

ج) فرض کنید دو پردازه A و B داریم که فرآیند کاری آن‌ها به صورت زیر است:

A: 20ns Memory - 10ns CPU - 5ns I/O

B: 30ns Memory - 15ns CPU - 8ns I/O

مقدار *CPU Utilization* را در صورتی که یک سیستم تک پردازنده و تک هسته‌ایی داشته‌باشیم، که از *Multiprogramming* پشتیبانی می‌کند، حساب کنید.

توجه: فرض کنید دستگاه‌های *I/O* و دسترسی به حافظه به صورت موازی عملیات‌های پردازه‌ها را انجام می‌دهند.

الف) در Multiprogramming اساس کار بر مبنای context -switch است اما Multitasking بر مبنای time sharing است.

تمرکز اصلی Multiprogramming استفاده ی حداکثری از cpu است(اینکه cpu هرگز بیکار نباشد) اما تمرکز اصلی Multitasking کم کردن زمان پاسخگویی cpu است.
در Multiprogramming بیشترین زمان ممکن صرف اجرای یک پردازه می شود اما در Multitasking کمترین زمان ممکن صرف اجرای یک پردازه می شود.
Multiprogramming با وجود حجم کم حافظه هم امکان پذیر است ولی Multitasking نیازمند حجم زیادی از حافظه است.
ب) یکی. زیرا فقط یک هسته داریم و یک cpu. پس در هر لحظه فقط به یک پردازه می تواند اختصاص یابد.

ج)

$$\frac{15 + 10}{53} = 47\ \%$$

3. همانطور که در پروژه‌ی درس دیدید، برای کار کردن با سیستم عاملی به غیر از سیستم عامل اصلی سیستم، می‌توانیم از ماشین مجازی مثل نرم‌افزار *vmware* استفاده کنیم. ولی غیر از این روش، با تغییر در نخستین برنامه‌ای که در هنگام روشن شدن سیستم اجرا می‌شود، که یک سفت‌افزار (*firmware*) است، می‌توانیم این امکان را فراهم کنیم.

الف) اسم این برنامه چیست؟ کارکرد اصلی آن را توضیح دهید.

برنامه bootstrap.

با روشن شدن سیستم، اولین برنامه ای که اجرا می شود برنامه bootstrap است. یک برنامه کاملا سخت افزاری می باشد که در ROM یا EPROM ذخیره شده است و تمامی ویژگی های سیستم را مقداردهی اولیه می کند. همچنین kernel را بارگذاری می کند و شروع به اجرا می کند.

برنامه bootstrap اولین بخش (sector) از دیسک را داخل رم اجرا می کند. در اولین بار روشن کردن سیستم، این بخش همان برنامه سیستم عامل ماست و از بار بعدی، سیستم با مراجعه به رم، سیستم عامل نصب شده را اجرا می کند.

ب) چگونه با استفاده از آن می‌توانیم چند سیستم عامل داشته باشیم و به خواسته سوال برسیم؟

زمانی که چند سیستم عامل میخواهیم داشته باشیم، در واقع سیستم به چند partition (هرکدام برای یک سیستم عامل) تقسیم می شود. برنامه bootstrap با روشن شدن سیستم، متوجه می شود که چند سیستم عامل روی سیستم نصب شده است و از کاربر می خواهد که سیستم عامل مورد نظر را انتخاب کند.

سپس مانند توضیح داده شده در بخش الف عمل می کند، یعنی برنامه سیستم عامل مورد نظر را بر روی رم می آورد و سپس اجرا می کند.(برای هر سیستم عامل یک bootloader داریم.)

4. به هنگام برنامه‌نویسی، برنامه‌های ساده نیز ممکن است از سیستم عامل استفاده زیادی کنند. اغلب، سیستم‌ها هزاران فراخوان سیستمی (*System call*) را در ثانیه اجرا می‌کنند. با این حال، اکثر برنامه‌نویسان هرگز این سطح از جزئیات را نمی‌بینند. به طور معمول، توسعه‌دهندگان برنامه، بر اساس رابط برنامه نویسی (*API*)، برنامه‌ها را طراحی می‌کنند. با این حال توابع مربوط به رابط‌های برنامه‌نویسی (*API*) خود از فراخوان‌های سیستمی (*System Calls*) استفاده می‌کنند.
   a) در ابتدا توضیح دهید که چرا برنامه‌نویسان به جای استفاده مستقیم از فراخوان‌های سیستمی (*System Calls*) حاضر هستند از این API ها استفاده کنند؟
   b) یکی از عوامل مهم در رسیدگی به فراخوان‌های سیستمی (*System Calls*)، محیط زمان اجرا (*Runtime environment*) است. محیط زمان اجرا یا به طور اختصار *RTE* مجموعه کامل از نرم‌افزار مورد نیاز برای اجرای برنامه‌های کاربردی نوشته شده در یک زبان برنامه‌نویسی خاص، از جمله کامپایلرها یا مفسرهای آن و همچنین نرم‌افزارهای دیگر مانند کتابخانه‌ها و لودرها (*Loaders*) است. تحقیق کنید و توضیح دهید، وجود *RTE* چگونه باعث می‌شود تا استفاده از فراخوان‌های سیستمی راحت‌تر بشود؟

a)

زیرا این api های برای سادگی ارتباط با os طراحی شده اند و استفاده از آن ها روند برنامه نویسی را نسبت به اینکه مستقیما با system call ها کار کنند بسیار راحت تر می کند. همچنین مهم ترین دلیل استفاده از آن ها این است که اگر یک برنامه با تکیه بر این api ها نوشته شود و کامپایل و اجرا شود و تغییراتی در system call ها ایجاد شود، این برنامه همچنان می تواند اجرا شود و نیازی به تغییر نخواهد داشت. (portability)

b)

به عنوان مثال یکی از معروف ترین RTE ها JVM می باشد، فرض کنیم میخواهیم برنامه ای بنویسیم که بر روی یک فایل عملیات خواندن و نوشتن را انجام می دهد. با استفاده از JVM فقط کافی است یکبار این برنامه را با استفاده از API هایی که تعبیه شده اند بنویسیم و پس از کامپایل می توانیم آن را هر جایی که JVM داشته باشد اجرا کنیم، که اینکار پیچیدگی خاصی ندارد و با نوشتن چند خط کد امکان پذیر است.

ولی اگر این RTE وجود نداشت، باید برای هر سیستم عامل با توجه به system call های مخصوص به خودش، یک نسخه از برنامه را می نوشتیم و در برنامه مستقیما با OS درگیر می شدیم که باعث افزایش چشمگیر پیچیدگی برنامه و همچنین زمان توسعه آن می شد.

5. در آینده، در درس سیستم عامل با مفهوم *Security* و اهمیت آن در سیستم‌های کامپیوتری آشنا می‌شوید. یکی از انواع حمله‌هایی که به یک سیستم کامپیوتری می‌تواند اعمال شود، *DMA Attack* است. با مفهوم *DMA* در درس آشنا شدید. حال ابتدا مفهوم *DMA* و کاربرد آن را شرح دهید و سپس نحوه انجام این حمله را به طور مختصر شرح دهید.

مفهوم DMA و کاربرد آن:

همانطور که میدانید CPU یا پردازنده سیستم در فرآیند فراخوانی اطلاعات از حافظه RAM دخیل میباشد و برای انجام پردازش روی داده ها بایستی اطلاعات را از حافظه RAM فرخوانی کند همین امر موجب میشود که زمان زیادی برای انتقال داده ها از حافظه RAM به سایر دستگاه ها در کامپیوتر شود. اینجاست که تکنولوژی DMA پا به میان میگذارد تا انجام این فرآیند با سرعت بیشتری صورت پذیرد.

DMA یا Direct Memory Access روشی برای انتقال داده ها از حافظه RAM به اجزاء دیگر کامپیوتر بدون پردازش آن توسط CPU میباشد. از آنجا که داده های ورودی و خروجی از کامپیوتر توسط CPU پردازش میشوند، اما برخی از داده ها نیاز به پردازش ندارند و یا میتوانند توسط دستگاه دیگری در کامپیوتر مورد پردازش قرار گیرند. یکی از مزیت های اصلی تکنولوژی DMA در همین است که از بار پردازشی CPU میکاهد و راهی بهینه برای انتقال داده ها از حافظه RAM به سایر دستگاه های موجود در کامپیوتر ایجاد میکند.

نحوه انجام حمله DMA attack:

همانطور که گفته شد، از آن جایی که در DMA دسترسی مستقیم به حافظه فراهم است، مهاجمین می توانند با استفاده از DMA Attack به RAM دسترسی داشته باشند و اطلاعات دلخواه را بنویسند یا بخوانند.

موفق باشید

تیم تدریس‌یاری درس سیستم‌های عامل