به نام ایزد یکتا

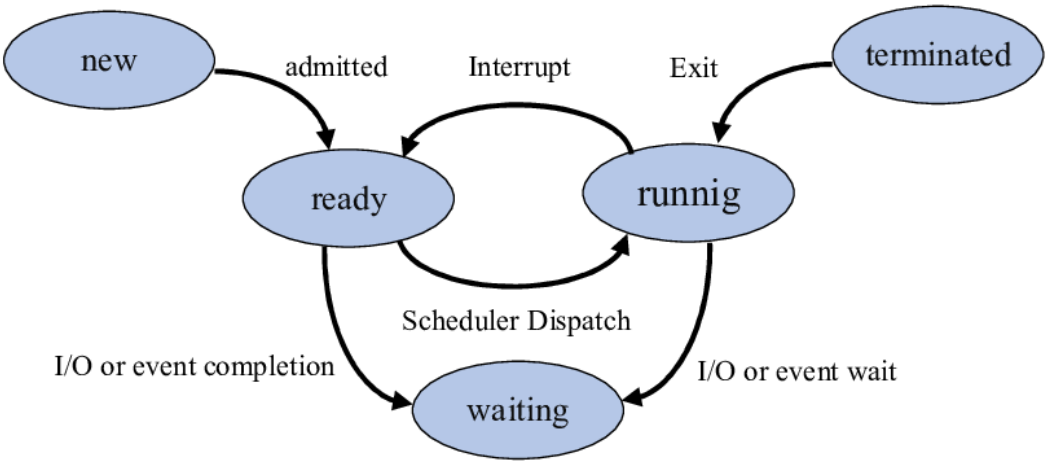
تمرین اول درس سیستم عامل

استاد: دکتر جوادی

تهیه کننده: بردیا اردکانیان

۹۸۳۱۰۷۲

سوال اول)



الف) خیر، یک پردازه از حالت waiting تنها می‌تواند به حالت ready تغییر وضعیت دهد. در واقع اگر پردازه به هر دلیلی وارد وضعیت waiting شود (می‌تواند به علت درخواستی از I/O یا event باشد) منتظر می‌ماند تا درخواست I/O یا event به انجام برسد و در نهایت از صف waiting queue خارج شده و به ready queue وارد می‌شود. در نهایت با تمام شدن پردازه‌اش وارد حالت terminated می‌شود.

اگر فرض کنیم پردازه والدی داریم که فرزندش در حالت waiting و منتظر I/O می‌باشد تنها در صورتی این شرایط امکان پذیر است که پردازه والد متوقف شود (به عنوان مثال با دستور kill) و پردازه فرزند از حالت waiting به حالت terminated خواهد رفت. (در بعضی سیستم عامل‌ها که پردازه فرزند بعد از خاتمه یافتن پردازه والد می‌تواند به زیستن ادامه دهد این شرایط رخ نمی‌دهد)

ب)

1. waiting. پردازه قبل از این در وضعیت running بوده است و با توجه به اینکه پردازه منتظر است تا یک I/O رخ دهد، به وضعیت waiting می‌رود.
2. ready. پردازه قبل از این در وضعیت running بوده است ولی با توجه به پارادیم timesharing مدت زمان زیادی را به این پردازه اختصاص داده و بنابر این پردازه به وضعیت ready تغییر وضعیت می‌دهد.
3. terminated. پردازه قبل از این در وضعیت running بوده است و با تمام شدن آن به وضعیت terminated می‌رود.
4. ready. پردازه قبل از این در وضعیت waiting بوده است و با تمام شدن خواندن از دیسک به وضعیت ready می‌رود و وارد صف ready queue می‌شود.

سوال دوم)

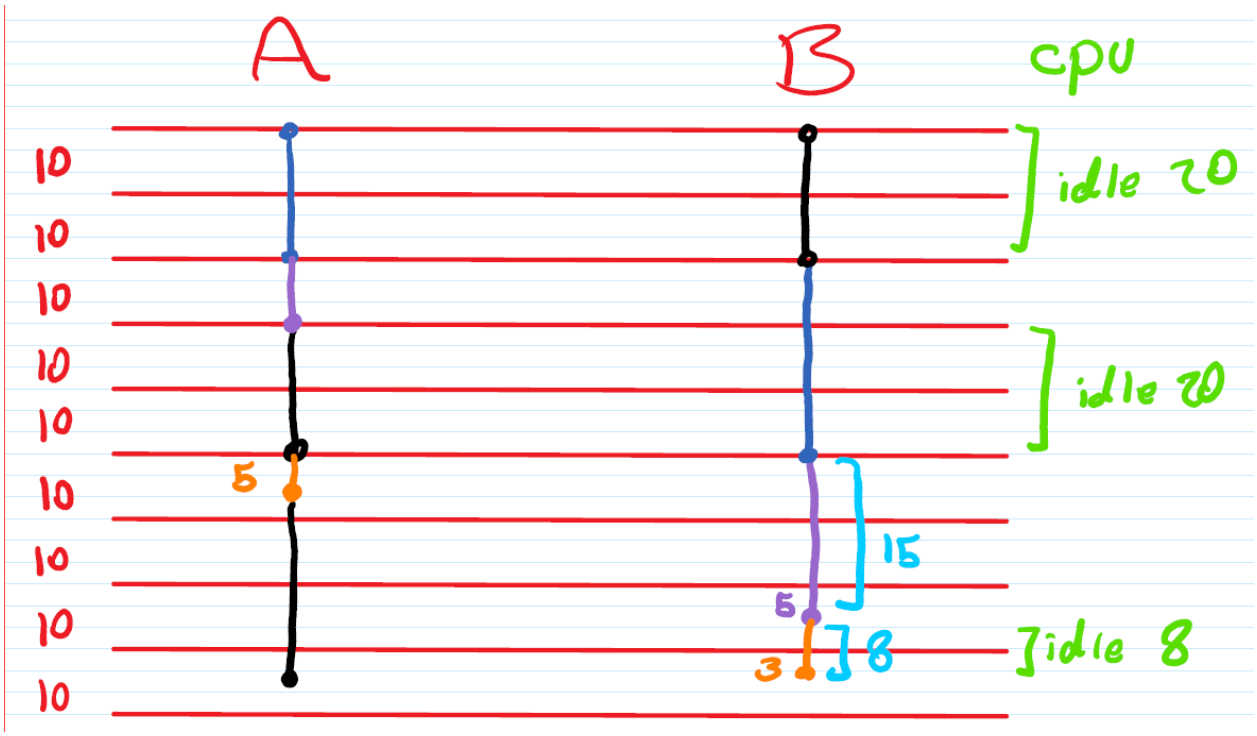
الف) در حالی که یک سیستم عامل multiprogramming اجازه می‌دهد تا بیش از یک برنامه به طور همزمان با استفاده از یک CPU اجرا شود، یک سیستم عامل multitasking اجازه می‌دهد چندین پردازه یا تسک همزمان با استفاده از چندین CPU اجرا شود.

ب) در یک سیستم multiprogramming، معمولاً یک صف از فرآیندهای مسدود (blocked) شده در انتظار I/O وجود دارد، و یک صف از فرآیندهایی که در انتظار اجرا هستند. در واقع فقط 0 یا 1 پردازش در حال اجرا (وضعیت running) وجود دارد.

2. C)

A: 20 ns Mem. - 10 ns CPU - 5 ns I/O

B: 30 ns Mem. - 15 ns CPU - 8 ns I/O



$T_{Total} = 83\,ns$

$Idle\_Total = 48ns$

$\Rightarrow \frac{48}{83} \times 100 = 57.83$

CPU Utilization is 100 - 57.83 which is equal to 42.17%

سوال سوم)

الف) برنامه bootstrap.

bootstrap برنامه‌ای است که سیستم عامل (OS) را در هنگام روشن شدن مقدار دهی (initialize) می‌کند.

- اولین برنامه‌ای که اجرا می‌شود؛ این برنامه بخش اولیه از 28 میلیون خط کد سیستم عامل را در حافظه اصلی لود می‌کند
- تمام ابعاد سیتم عامل را داخل حافظه لود می‌کند
- تمام جنبه های سیستم را راه اندازی می کند
- تمامی کار های اولیه مثل انتخاب منطقه، چند پارتیشنی و ... را انجام میده
- هسته سیستم عامل را بارگیری می کند و اجرا را شروع می کند
- با روشن شدن یا reboot مجدد بارگذاری می‌شود
- معمولاً در ROM یا EPROM ذخیره می شود که به عنوان firmware شناخته می شود

هر زمانی که کامپیوتر را روشن می‌کنیم، برنامه bootstrap از داخل ROM در حافظه اصلی (RAM) اجرا می‌شود. این بخش که اولین sector از disk را اجرا می‌کند مقادیر اولیه را بازگذاری (initialize) می‌کند و سیستم عامل را لود می‌کند. بعد از لود شدن سیستم عامل، کنترل به دست سیستم عامل می‌افتد.

ب)

هنگامی که دو سیستم عامل بر روی سیستم کامپیوتری نصب می‌شود، به آن dualboot می‌گویند. در واقع می‌توان چندین سیستم عامل را روی چنین سیستمی نصب کرد. اما چگونه سیستم می داند که کدام سیستم عامل باید بوت شود؟ یک بوت لودر که چندین سیستم فایل و چندین سیستم عامل را درک می‌کند، می‌تواند فضای بوت را اشغال کند. پس از بارگیری، می‌تواند یکی از سیستم عامل های موجود روی دیسک را بوت کند. دیسک می‌تواند چندین پارتیشن داشته باشد که هر کدام شامل یک نوع سیستم عامل متفاوت است. هنگامی که یک سیستم کامپیوتری روشن می‌شود، یک برنامه مدیریت بوت منویی را نمایش می دهد که به کاربر اجازه می‌دهد سیستم عامل مورد استفاده را انتخاب کند.

بوت لودرها می توانند در چندین مکان وجود داشته باشند.

یک MBR (Master Boot Record) وجود دارد، اما در کنار آن چندین پارتیشن هم حضور دارند. هر پارتیشن می‌تواند یک بوت لودر در Volume Boot Record (VBR) خود داشته باشد - اولین بخش از یک پارتیشن قابل بوت. شما

می‌توانید یک بوت لودر روی MBR و/یا درون VBR داشته باشید، بنابراین MBR به بوت لودر ثانویه در VBR زنجیر می‌شود. اینگونه است که ابتدا Grub و سپس سیستم عامل را می‌بینید. Grub روی MBR است.

بنابراین وقتی چندید سیستم عامل (بر فرض 2 عدد) نصب کرده باشیم، Bootstrap متوجه می‌شود چند سیستم عامل وجود دارد. در واقع Bootstrap متوجه می‌شود که اطلاعات چندین سیستم عامل را می‌تواند بخواند.

سوال چهارم)

الف)

برنامه نویسان اغلب به فراخوانی سیستمی دسترسی مستقیم ندارند اما می‌توانند از طریق رابط‌های برنامه نویسی (API) به آنها دسترسی داشته باشند. توابعی که در API گنجانده شده‌اند، در واقع فراخوانی‌های سیستم را فراخوانی می‌کنند. با استفاده از API می توان مزایای خاصی به دست آورد:

- ***Portability***: تا زمانی که یک سیستم از API پشتیبانی می کند، هر برنامه ای که از آن API استفاده می‌کند می تواند کامپایل و اجرا شود
- ***Ease of Use***: استفاده از API می تواند بسیار ساده تر از استفاده از فراخوانی سیستمی باشد
- ***Security/Protection***: نوعی امنیت/محافظت را برای کرنل فراهم می کند زیرا شما با هسته تعامل ندارید
- ***Efficiency***: API ها مقداری کارایی را افزایش می‌دهند
- ***door/access***: همچنین یک درب/دسترسی برای سرویس‌های خارج از سیستم که در داخل با سیستم تعامل دارند، فراهم می‌کند.

فراخوانی‌های سیستمی با هسته ارتباط برقرار می کنند و درخواست می‌دهند و از آن پاسخ می‌گیرند. به یاد داشته باشید که پایین ترین سطح سیستم عامل (OS)، هسته سیستم عامل است. مسئول تفسیر درخواستی است که توسط کامپیوتر قابل درک باشد. به این ترتیب فراخوانی سیستم رابط هسته است.

این فرآیندها در سطح پایینی رخ می‌دهند.

در حالی که API در سطح بالاتری قرار دارد، با فراخوانی سیستمی تعامل دارد که به نوبه خود با هسته تعامل دارد. API یک system-call interface است در حالی که فراخوانی سیستمی یک رابط هسته با سیستم عامل است.

API در سطح بالا تعامل دارد در حالی که فراخوانی سیستمی در سطح پایین تعامل دارد.

بنابراین برای کارایی و Ease of use، به API نیاز دارید.

ب)

محیط های زمان اجرا (به اختصار RTE) به عنوان سیستم عامل های کوچک عمل می کنند و تمام عملکردهای لازم برای اجرای یک برنامه را فراهم می کنند. این شامل رابط‌های بخش‌های فیزیکی سخت‌افزار، تعاملات کاربر و اجزای نرم‌افزاری است.

یک محیط زمان اجرا برنامه‌ها را بارگیری می‌کند و آنها را روی یک پلتفرم اجرا می‌کند. تمام منابع لازم برای اجرای مستقل از سیستم عامل در این پلتفرم موجود است. به همین دلیل است که، برای مثال، پخش ویدیوهای فلش تنها با محیط زمان اجرا مناسب امکان پذیر است - در این مورد Adobe Flash Player. در این محیط، فیلم های فلش را می‌توان با همان کیفیت و با عملکردهای یکسان بدون توجه به اینکه از کدام مرورگر یا سیستم عامل استفاده می شود پخش کرد.

یک محیط زمان اجرا چگونه کار می کند؟

برنامه‌ای که در حال حاضر در حال اجرا است از طریق یک سیستم زمان اجرا با محیط زمان اجرا تعامل دارد. محیط زمان اجرا به نوبه خود به عنوان یک رابط بین برنامه و سیستم عامل عمل می‌کند. به محض اجرای برنامه، دستورالعمل ها را به پردازنده و رم کامپیوتر ارسال می‌کند و به منابع سیستم دسترسی پیدا می‌کند. بنابراین محیط زمان اجرا شامل سخت افزار، حافظه، متغیرهای محیط و تعامل با کاربر و اجزای نرم افزار می‌شود.

یک محیط زمان اجرا عملکردهای اساسی مختلفی را برای حافظه، شبکه‌ها و سخت افزار فراهم می‌کند. این توابع به جای برنامه کاربردی توسط محیط زمان اجرا انجام می‌شود و مستقل از سیستم عامل کار می‌کند. آنها شامل خواندن و نوشتن فایل‌ها، مدیریت دستگاه های ورودی و خروجی، جستجو و مرتب سازی فایل ها، و انتقال داده ها از طریق شبکه می‌باشد.

یکی از مزایای اصلی محیط‌های زمان اجرا این است که برنامه ها به تمام عملکردهای مورد نیاز خود دسترسی دارند و بنابراین مستقل از سیستم عامل ها کار می‌کنند. علاوه بر این، برنامه ها صرف نظر از اینکه روی ویندوز، macOS یا لینوکس اجرا می‌شوند، رابط کاربری یکسانی دارند. توسعه دهندگان همچنین از محیط های زمان اجرا برای آزمایش برنامه ها در اجرای خود استفاده می کنند. در صورت بروز خطا، RTE دلیل سقوط را گزارش می‌کند. فریم ورک‌ها نیز به محیط‌های زمان اجرا مرتبط هستند. این ساختارهای برنامه توسعه نرم افزار را ساده می‌کند و می تواند شامل محیط‌های زمان اجرا باشد که برنامه ها در آن اجرا می‌شوند.

همانطور که در بالا ذکر شد، محیط‌های زمان اجرا عملکرد cross-platform را برای برنامه‌ها فعال می‌کنند. این فرآیند توسعه را ساده می‌کند، زیرا برنامه نیازی به تطبیق با سیستم عامل‌های مختلف ندارد. اگر یک برنامه از عملکردهای یک محیط زمان اجرا برای اجرای خود استفاده کند، افرادی که از سیستم عامل های مختلف استفاده می‌کنند می‌توانند از عملکردهای مشابه و یک رابط کاربری تقریباً یکسان بهره‌مند شوند.

مزیت دیگر حفظ منابع است: برنامه های مشابه می‌توانند از محیط های زمان اجرا یکسان استفاده کنند و اجزای مشترک را به اشتراک بگذارند.

سوال پنجم)

Direct Memory Access (DMA) یکی از ویژگی‌های سیستم‌های کامپیوتری است که به برخی از subsystemهای سخت‌افزاری اجازه می‌دهد تا به حافظه اصلی سیستم (RAM) مستقل از واحد پردازش مرکزی (CPU) دسترسی داشته باشند.

بدون DMA، زمانی که CPU از ورودی/خروجی (I/O) استفاده می‌کند، معمولاً در تمام مدت عملیات خواندن یا نوشتن به طور کامل اشغال می‌شود و بنابراین برای انجام کارهای دیگر در دسترس نیست. با DMA، CPU ابتدا انتقال را آغاز می‌کند، سپس در حین انجام انتقال، عملیات دیگری را انجام می دهد و در نهایت هنگام انجام عملیات، یک وقفه از کنترلر DMA (DMAC) دریافت می کند. این ویژگی در هر زمانی مفید است که CPU نتواند با سرعت انتقال داده‌ها هماهنگی داشته باشد، یا زمانی که CPU نیاز به انجام کار دارد در حالی که منتظر انتقال داده I/O نسبتا کند است. بسیاری از سیستم های سخت افزاری از DMA از جمله کنترل کننده های درایو دیسک، کارت های گرافیک، کارت های شبکه و کارت های صدا استفاده می‌کنند. DMA همچنین برای انتقال داده های درون تراشه‌ای در پردازنده های چند هسته‌ای استفاده می‌شود. رایانه‌هایی که کانال‌های DMA دارند می‌توانند داده‌ها را به و از دستگاه‌هایی با CPU overhead بسیار کمتر از رایانه‌های بدون کانال DMA منتقل کنند. به طور مشابه، یک عنصر پردازشی در داخل یک پردازنده چند هسته‌ای می تواند داده‌ها را بدون اشغال زمان پردازش به و از حافظه محلی خود منتقل کند و به محاسبات و انتقال داده‌ها اجازه دهد تا به صورت موازی پیش بروند.

DMA همچنین می تواند برای کپی کردن یا انتقال داده ها در حافظه "memory to memory" استفاده شود. DMA می‌تواند عملیات گران‌قیمت حافظه مانند کپی‌های بزرگ یا عملیات جمع‌آوری پراکنده را از CPU به یک موتور اختصاصی DMA تخلیه کند. یک مثال پیاده سازی، فناوری شتاب I/O است. DMA در معماری های محاسباتی شبکه روی تراشه و درون حافظه مورد توجه است.

حمله DMA نوعی حمله side channel در امنیت رایانه است که در آن مهاجم می‌تواند با بهره‌برداری از وجود پورت‌های سریع که دسترسی مستقیم به حافظه (DMA) را امکان‌پذیر می‌سازد، به رایانه یا دستگاه دیگری نفوذ کند.

DMA در تعدادی از اتصالات گنجانده شده است، زیرا به یک دستگاه متصل (مانند دوربین فیلمبرداری، کارت شبکه، دستگاه ذخیره سازی یا سایر لوازم جانبی مفید یا کارت PC داخلی) اجازه می‌دهد تا داده‌ها را با حداکثر سرعت ممکن بین خود و رایانه انتقال دهد. دسترسی مستقیم سخت افزاری برای خواندن یا نوشتن مستقیم در حافظه اصلی بدون هیچ گونه نظارت یا تعامل سیستم عامل. استفاده‌های قانونی از چنین دستگاه‌هایی منجر به پذیرش گسترده لوازم جانبی و اتصالات DMA شده است، اما مهاجم می‌تواند به همان اندازه از همان تسهیلات برای ایجاد لوازم جانبی استفاده کند که با استفاده از همان پورت متصل می‌شود و سپس به طور بالقوه می‌تواند به بخشی یا تمام فضای آدرس حافظه فیزیکی رایانه دسترسی مستقیم داشته باشد، مکانیزم‌های امنیتی سیستم‌عامل و هر lock screen را دور بزند، تمام کارهایی که رایانه انجام می‌دهد را بخواند، داده‌ها یا کلیدهای رمزنگاری را بدزدد، نرم‌افزارهای جاسوسی نصب و اجرای کند و یا سیستم را طوری تغییر دهید که درهای پشتی یا بدافزارهای دیگر مجاز باشد.

جلوگیری از اتصالات فیزیکی به چنین پورت‌هایی از حملات DMA جلوگیری می کند. در بسیاری از رایانه‌ها، اتصالات پیاده‌سازی DMA نیز می‌توانند در صورت عدم استفاده در BIOS یا UEFI غیرفعال شوند، که بسته به دستگاه می‌تواند پتانسیل این نوع سوءاستفاده را از بین ببرد یا کاهش دهد.