باسمه تعالی

تکلیف ۱ درس سیستم عامل- دکتر جوادی

پاییز ۱۴۰۱

سید امیرمهدی میرشریفی - ۹۸۳۱۱۰۵

۱- به سوالات زیر در مورد دستگاه های ورودی و خروجی[۱] و نحوه‌ی انتقال اطلاعات از آنها به پردازنده و برعکس پاسخ دهید.

الف) وظیفه‌ی کنترولر[۲] و درایور[۳] دستگاه‌ها چیست؟ تعامل این دو قسمت با یکدیگر و با دستگاه مربوط به خودشان از آغاز تا پایان یک عملیات *I/O* به چه صورت است؟

ب) می‌دانیم یک روش انتقال داده بین دستگاه‌های ورودی و خروجی و پردازنده، مبتنی بر وقفه[۴] است. عیب (ضعف) این روش چیست و چگونه در سیستم های کامپیوتری امروزی رفع شده؟

پاسخ:

الف)

هر دیوایس کنترولری مسئولیت کنترل ، استفاده و بهره وری از یک دستگاه جانبی را دارد.
در داخل هر دیوایس کنترولری یک بافر وجود دارد که واسطی جهت انتقال اطلاعات بین دستگاه جانبی و کامپیوتر است.
بر خلاف دیوایس کنترولر که از جنس سخت افزار است ، درایور یک برنامه است که تمام سیستم عامل ها از آن جهت شناسایی کنترولر و ارتباط آن با خود استفاده میکنند.
برای انتقال اطلاعات بین دستگاه جانبی و کامپیوتر ، همانطور که در بالا اشاره شد ، چه برای خواندن و چه برای نوشتن ابتدا اطلاعات به بافری که داخل کنترولر قرار دارد و ظرفیت محدودی دارد منتقل میشود و سپس یک اینتراپت به پردازنده داده میشود و سپس با نظارت پردازنده یا به وسیله خود پردازنده اطلاعات بین حافظه رم و دستگاه جانبی رد و بدل میشود.

ب)

پس از ارسال وقفه به پردازنده، پردازنده حالت فعلی خود رو در رجیستر هایش ذخیره میکند و سپس به سراغ جدول مدیریت وقفه میرود تا پردازش های مناسب با آن و برگرداندن جواب مناسب وقفه را انجام دهد.
اما ما نیاز هایی مهم تر درباره وقفه داریم که در ادامه آمده است و کامپیوتر های مدرن این نیاز ها را برای ما حل کرده اند:

۱) توانایی به تعویق انداختن مدیریت وقفه حین پردازش بحرانی
۲) روش کارآمد جهت ارسال پاسخ پردازنده به وقفه به دستگاه عامل به وجود آمدن وقفه
۳) توانایی اولویت بندی وقفه ها

در سخت افزار کامپیوتر های مدرن این خدمات توسط پردازنده ارائه میشود.

۲ - در مورد فراخوانی‌های سسیستمی[۵] و *API* به موارد زیر پاسخ دهید:

الف) توضیح دهید چگونه استفاده از *API* به جای فراخوانی مستقیم فراخوانی‌های سیستمی به برنامه‌نویسان کمک می‌کند؟

ب) می‌دانیم هنگام استفاده از *API* مختلف سیستم عامل در برنامه‌هایمان، در حقیقت بعضی فراخوانی‌های سیستمی استفاده می‌شوند. توضیح دهید این تبدیل فراخوانی‌های API به صدا زدن فراخوانی‌های سیستمی چگونه شکل می‌گیرد؟

پ) روش های کلی انتقال پارامترها به سیستم عامل هنگام استفاده از فراخوانی‌های سیستمی را توضیح دهید.

پاسخ:

الف)

برنامه نویس یک برنامه با استفاده از یک API می تواند انتظار داشته باشد که برنامه او روی هر سیستمی که از همان API پشتیبانی می کند، کامپایل و اجرا شود (اگرچه، در واقعیت، تفاوت های معماری اغلب این کار را دشوارتر از آنچه به نظر می رسد، می کند).

همچنین استفاده مستقیم از فراخوانی سیستمی کمی سخت تر و با جزئیات بیشتر است. بنابر این با استفاده از API ها کار برای برنامه نویس ساده تر میشود.

ب)

Run time environment اینترفیسی برای فراخوانی های سیستمی توسعه داده است که با توابعی که در API ها هستند لینک هستند. این اینترفیس ها زمان فراخوانی توابع API ها فراخوانی های سیستمی مورد نظر را صدا میزنند، بدین صورت که جدولی حاوی ایندکس فراخوانی های سیستمی را نگه میدارند و با دریافت شماره ایندکس فراخوانی سیستمی را برای سیستم عامل و هسته آن میفرستد و پاسخ آن را برمیگرداند.

پ)

برای ارسال پارامتر برای سیستم عامل هنگام صدا زدن فراخوانی های سیستمی از ۳ روش استفاده میشود.

۱) رجیستر ها: در این روش شماره فراخوانی سیستمی را در رجیستر EAX و سه پارامتر دیگر را در EDX... EBX وارد میشود.
۲) اشاره به قسمتی از حافظه: در این روش آدرس حافظ در رجیستر قرار میگیرد و سیستم عامل با دستور العمل مشخص پارامتر ها را از رجیستر میخواند.
۳) استک: در این روش هم پارامتر ها در استک پوش میکنیم و سپس سیستم عامل آن ها را پاپ میکند.

۳- در مورد فراخوانی‌های سیستمی زیر در سیستم عامل لینوکس ۶۴ بیتی با معماری *x86* تحقیق کنید و بنویسید که روش ارسال پارامتر های هر کدام به چه صورت است.

- sys_write
- sys_close
- sys_getpid
- sys_sysinfo

sys_write : برای فراخوانی این سیستم کال ابتدا شماره آن را در یک رجیستر ، سپس آدرس فایلی که میخواهیم در آن بنویسیم در رجیستر di ، بافری که میخواهیم از آن بخوانیم را در رجیستر si و تعداد بایت هایی که میخواهیم از بافر بخوانیم و در فایل مقصد بنویسیم را در رجیسترdx مینویسیم. بنابراین شیوه پاس دادن متغیر ها بر اساس **اشاره به حافظه** است.

sys_close :برای فراخوانی این سیستم کال ابتدا شماره آن را در یک رجیستر، سپس آدرس فایلی که میخواهیم آن را ببندیم را در رجیستر di مینویسیم. بنابراین شیوه پاس دادن متغیر بر اساس **رجیستر** است.

sys_getpid: برای فراخوانی این سییستم کال تنها نیاز است که شماره سیستم کال را در رجیستر بنویسیم و هیچ پارامتر دیگری نیاز به ارسال نیست.

sys_sysinfo: این سیستم کال که جهت گرفتن آمار کلی سیستم هست به یک پارامتر که ساختار اطلاعاتی که مورد نیاز هست را مشخص میکند، نیاز دارد. با مشخص کردن ساختار و قرار دادن آدرس آن در یک رجیستر پارامتر را برای فراخوانی ارسال میکنیم. بنابراین روش ارسال پارامتر **اشاره به حافظه** است.

۴- چگونه می‌توان سیستمی طراحی کرد که اجازه‌ی انتخاب یک سیستم عامل از چند سیستم عامل را هنگام بوت شدن به کاربر بدهد؟ برنامه‌ی bootstrap برای این منظور چه کاری باید انجام بدهد؟

پاسخ:

برای این کار باید از روشی به نام **Multi booting** (یا dual booting برای نصب دو سیستم عامل) استفاده کرد که در ادامه توضیح آن آورده شده است.

Multi booting عمل نصب چند سیستم عامل روی یک دستگاه است و شما را قادر میسازد که یک سیستم عامل را جهت بوت شدن انتخاب کنید.

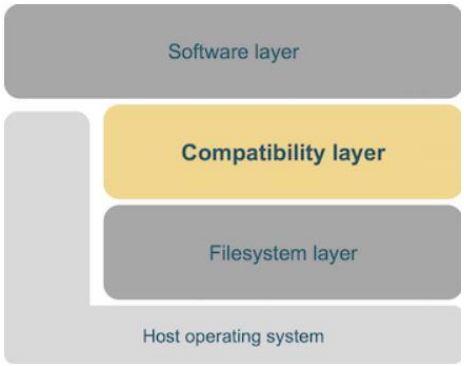
Bootstrap هنگام بوت کردن سیستم عامل کدی را از حافظه را میخواند که مخصوص بوت کردن سیستم عامل است به نام bootloader. هرسیستم عامل boot loader خاص خود را دارد.بنابراین برنامه بوت استرپ ،بوت لُدر سیستم عاملی که کاربر انتخاب کرده است را بوت میکند.

۵- یک برنامه‌ی کمکی می‌خواد به گونه‌ای این امکان را فراهم کند که یک برنامه که برای ویندوز کامپایل شده را در لینوکس اجرا کند. اگر معماری پردازنده‌ی دو سیستم عامل یکسان باشد، توضیح دهید این برنامه چه کارهایی باید انجام دهد.

compatibility layer یک اینترفیس هست که این اجازه را به کاربر میدهد که فایلی که برای یک سیستم عامل دیگر کامپایل شده است را روی سیستم عامل دستگاه میزبان اجرا کند.

روش کار compatibility layer این است که فراخوانی های سیستمی برنامه میهمان را به فراخوانی های سیستمی میزبان ترجمه کند.

برای این کار نرم افزار هایی مانند wine وجود دارند که میشود از آنها بهره برد.



۶- وقتی یک پردازه با صدا زدن fork پردازه‌ی جدیدی می‌سازد، کدام از موارد زیر بین پردازه‌ی والد و فرزند مشترک خواهد بود؟ (هر مورد را کوتاه توضیح دهید)

- استک
- هیپ
- قسمت های حافظه‌ی مشترک[۶]

وقتی یک پردازه به وسیله فُرک یک پردازه فرزند میسازد (درصورت درخواست فرزند) یک کپی از تمام اطلاعات پدر اعم از استک هیپ و حافظه و دیتا برای فرزند گرفته میشود و در حافظه فرزند به صورت متناظر منتقل میشود. بعد از آن دیگر هیچ ارتباطی بین استک و هیپ و حافظه آنها نیست و هرکدام راه خود را میروند.

گاهی دوتا پردازه میخواهند با یکدیگر در ارتباط باشند و اطلاعاتی را بین خود رد و بدل کنند که یکی از آن روش ها حافظه مشترک است که به وسیله دو پردازنده مشخص میشود و بعد از آن آن دو پردازه میتوانند روی آن بخش از حافظه بخوانند و بنویسند.