AUT-CE-OS HW1 – Keivan Ipchi Hagh - 9831073

1. سه مورد مزایا و معایب جداسازی سطح User و Kernel را نام ببرید.

برخی سیستم عامل ها این دو سطح را جداسازی نکرده اند. این جداسازی مزایا و معایب خاصی دارد، مزایای این روش:

- مسائل امنیتی؛ به طوری که اگر یک بد افزار در سطح User ایزوله میشود و این جداسازی مانع آسیب رسیدن به سخت افزار و بخش های حساس سیستم عامل شده و بقیه سیستم را تحت تاثیر قرار ندهد. به طور کلی kernel اولین لایه از چهار لایه امنیتی است.
- سطح دسترسی؛ به طوری که kernel همیشه سطح دسترسی بالاتری نسبت به بقیه برنامه ها دارد (is privileged) و این باعث میشود نرم افزار های دیگر قوانین را نشکنند و همیشه تحت کنترل سیستم مادر باشند.
- آسیب کمتر؛ crash کردن سیستم باعث خسارت های زیادی میشود. برای جلوگیری از این اتفاق این جداسازی باعث میشود user space که عملا یک interface است crash کند و بقیه اجزای سیستم تحت شعاع قرار نگیرند. لازم به ذکر است در صورت crash کردن kernel خسارت زیادی وارد میشود. (از دست رفتن دیتا و..)

از معایب این جداسازی میتوان گفت:

- مصرف منابع؛ به طوری که بدون این جداسازی هیچ مشکلی پیش نمی آید و این کار صرفا یک overhead بر روی سیستم میگذارد.
- پیچیدگی و حساسیت؛ kernel های بزرگ تر پیچیدگی بیشتری دارند و یک باگ در یک درایو باعث crash کردن کل سیستم میشود. حساسیت سازگاری dependency ها بالاست و درصورت ناسازگاری کل سیستم از کار می افتد.
- توسعه پذیری؛ توسعه دادن کرنل آسان نیست زیرا اکثر کتابخانه های معروف قابل استفاده نیستند و اکثر کد باید از اول زده شود. همچنین به طور مرتب دستگاه باید reboot شود تا تست صورت بگیرد و عیب یابی کردن سخت است.

2. درباره multitasking و multiprogramming به صورت جداگانه توضیح داده و تفاوت های آن ها را بیان کنید.

معنی multiprogramming یا parallel processing آن است که برای مثال چندین کاربر بتوانند یک (یا چند) برنامه را بطور همزمان استفاده کنند (توسط یک CPU). به طوری که چندین process برای هر کاربر داریم. اگر process یک کاربر به هر دلیل متوقف شود سیستم عامل CPU را به process کاربر بعدی در صف اختصاص میدهد تا با این کار گپ خالی را پر کند. (اصطلاحا maximize the CPU utilization)

معنی multitasking آن است که یک کاربر بتواند چندین برنامه (process) را به طور همزمان استفاده کند. (برای مثال گوش کردن به موزیک هنگام تایپ تکلیف OS در ورد). به طور کلی multitasking زمانیست که چندین برنامه مختلف از منابع یکسانی استفاده کنند (در اینجا CPU به طور اختصار)

(تفاوت ها:)

به طور کلی multiprogramming (که بر اساس context switching کار میکند) برای اجرای چندین process به طور همزمان است در صورتی که multitasking برای اجرای چندین task یا job. این یعنی process میتواند از چندین task ساخته شده باشد پس multitasking تقریبا زیر مجموعه ای از multiprogramming است.

همچنین در multiprogramming ما یک CPU داریم که چندین process را انجام میدهد اما در multitasking چندین CPU داریم که چندین task را انجام میدهند.

3. چگونگی بالا آمدن و بار گذاری سیستم عامل در لحظه شروع به کار سیستم کامپیوتری را با جزئیات توضیح دهید.

در مرحله اول برنامه Bootstrap که در ROM ذخیره شده (firmware) لود شده و در مرحله بعدی به CPU دستور میدهد که دستورات لازم مربوط به خواندن سیستم عامل و بقیه سیستم های مورد نیاز برای کارکرد را از BIOS که در ROM ذخیره شده خوانده و سپس به دنبال سیستم عامل گشته و آنرا در RAM بارگذاری کند. (اگر چندین سیستم عامل وجود داشت امکان انتخاب وجود دارد و سپس سیستم عامل انتخاب شده لود میشود) سپس کنترل از Bootstrap به سیستم عامل منتقل شده و ادامه کار بر دوش سیستم عامل خواهد بود.

4. دلیل اینکه برنامه های مختص به هر سیستم عامل هستند و براحتی بر روی سیستم عامل های دیگر اجرا نمیشوند، چیست؟

سختی این امر به دلیل آن است که ویندوز و لینوکس از API های متفاوتی استفاده میکنند زیرا kernel های متفاوتی دارند (و در نتیجه کتابخانه های مختلفی). به طور دقیق تر system call های سیستم عامل ها متفاوت است و این باعث میشود نسخه binary برنامه ها به طور مشترک بر روی هر دو سیستم عامل کار نکند.

به طور دقیق تر، هر برنامه ی باینری عملا یک لیست از دستوراتی است که برنامه باید انجام بدهد؛ نحوه ساخت این دستورات بستگی به processor سیستمی دارد که برنامه در آن کامپایل شده است. اکثر سیستم های جدید از processor های مشابهی (شیوه طراحی processor) استفاده میکنند اما همه این دستورات به طور native مختص processor نیستند و میتوانند برای ارتباط برقرار کردن با I/O، گرفتن ورودی از کاربر، نمایش چیزی یا... باشند که برای

اینکار به سیستم عامل system call میفرستد. حال سیستم عامل های متفاوت API های متفاوت و در نتیجه system call های متفاوت دارند که باعث میشود این برنامه ها OS compatible نباشند.

5. درباره نحوه عملکرد DMA توضیح دهید. نحوه همکاری DMA و پردازنده به چه صورت است؟ پردازنده به چه صورت از به پایان رسیدن کار DMA مطلع میشود؟

بدون وجود DMA یا همان Direct Access Memory در کنار CPU، CPU مجبور است برای هر دستور I/O منتظر بماند تا خروجی گرفته شود و در نتیجه CPU به طور کامل اشغال میشوند (عملا سیستم freeze میشود) و دیگر قابلیت multiprogramming نخواهد داشت.

در نتیجه هنگام ارتباط برقرار کردن با I/O، CPU فعالیت را به DMA منتقل کرده و در حین این عمل میتواند ادامه فعالیت خود را بکند. هنگامی که عمل I/O تمام شود DMA با فرستادن یک interrupt به CPU آن را مطلع کرده و دسترسی (کنترل) به CPU بازگردانده میشود. این ویژگی زمانی خوب زیرا I/O سرعت پایینی دارد و CPU دیگر منتظر پایان رسیدن عملیات های کند نمی ماند.

برای مثال زمانی که در برنامه نیاز به ورودی کاربر دارد، کنترل از CPU به DMA منتقل شده و CPU به ادامه کار خود میپردازد. زمانی که ورودی از کاربر گرفته شد interrupt مربوطه به CPU داده میشود و کنترل دوباره به CPU بازگردانده میشود. این روش باعث حداکثر شدن کارایی CPU میشود (یا به عبارتی maximum CPU utilization)