تمرین تئوری سری اول سیستم عامل

دکتر انتظاری

رامین بوذرپور

402541206

سوال ۱:

اگه بخوایم سیستم‌عامل گوشی‌های هوشمند رو بسازیم، بهترین ساختار Microkernel هست. یعنی فقط کارهای اصلی مثل مدیریت حافظه و زمان‌بندی توی هسته انجام میشه و بقیه کارها مثل درایورها و پروتکل‌های شبکه میرن توی فضای کاربر. اینطوری:

- **امنیت و پایداری سیستم بیشتر میشه چون اگه یه بخش خراب بشه، کل سیستم نمی‌ریزه**
- **توسعه و نگهداری راحت‌تر میشه چون تغییرات توی بخش‌های مختلف سیستم تأثیر کمتری روی هم دارن**
- **مصرف انرژی کمتر میشه چون بخش‌های مختلف سیستم منابع رو بهینه‌تر استفاده می‌کنن**

البته یه نکته هم هست؛ این ساختار ممکنه کمی عملکرد رو پایین بیاره چون ارتباط بین بخش‌های مختلف از طریق پیام‌گذاری انجام میشه که ممکنه زمان‌بر باشه

سوال ۲:

- **Multiprogramming یعنی چند تا برنامه رو می‌ذاری توی حافظه و CPU بینشون جابه‌جا میشه. اینطوری از زمان‌های بیکاری CPU استفاده میشه. ولی فقط یه برنامه در هر لحظه اجرا میشه و بقیه منتظر می‌مونن**
- **Multitasking یعنی CPU می‌تونه چند تا برنامه رو همزمان اجرا کنه. هر برنامه یه مدت زمان مشخص اجرا میشه و بعد میره سراغ برنامه بعدی. اینطوری کاربر احساس می‌کنه که همه برنامه‌ها همزمان در حال اجرا هستن**

در واقع، Multitaskingیه نوع پیشرفته‌تر از Multiprogrammingهست که منابع رو بهینه‌تر استفاده می‌کنه

سوال ۳:

سیستم‌عامل‌ها برای اینکه امنیت و پایداری بیشتری داشته باشن، از دو حالت User Mode و Kernel Mode استفاده می‌کنن:

- **User Mode : برنامه‌های کاربردی توی این حالت اجرا می‌شن و دسترسی محدودی به منابع سیستم دارن. اینطوری اگه برنامه‌ای خراب بشه، سیستم آسیب نمی‌بینه**
- **Kernel Mode : هسته سیستم‌عامل توی این حالت اجرا میشه و دسترسی کامل به منابع سیستم داره. این حالت برای کارهای حساس مثل مدیریت حافظه و پردازش درخواست‌های ورودی/خروجی استفاده میشه**

این تفکیک باعث میشه که سیستم‌عامل بتونه منابع رو به صورت مؤثر و ایمن مدیریت کنه

سوال ۴ :

Race Condition زمانی پیش میاد که دو یا چند نخ به طور همزمان به یه منبع مشترک دسترسی پیدا کنن و ترتیب اجرای اون‌ها تأثیرگذار باشه. این می‌تونه باعث رفتارهای غیرمنتظره یا خطا بشه

برای جلوگیری از Race Condition می‌تونیم از روش‌های زیر استفاده کنیم:

- **Mutexes : قفل‌هایی که فقط یه نخ می‌تونه در هر لحظه به منبع مشترک دسترسی داشته باشه**
- **Semaphores : شمارنده‌هایی که تعداد نخ‌های مجاز برای دسترسی به منبع رو کنترل می‌کنن**
- **Monitors : ساختارهایی که دسترسی به منابع مشترک رو هماهنگ می‌کنن و از بروز مشکلات جلوگیری می‌کنن**

استفاده از این ابزارها باعث میشه که سیستم پایدار و قابل اعتماد باقی بمونه

سوال ۵ :

Copy-on-Write یه تکنیک مدیریت حافظه است که وقتی می‌خوای یه کپی از داده‌ها بسازی، تا زمانی که نیاز به تغییر نباشه، از کپی کردن جلوگیری می‌کنه. یعنی تا زمانی که داده‌ها تغییر نکنن، از حافظه مشترک استفاده میشه و فقط وقتی که یکی از فرآیندها بخواد داده‌ای رو تغییر بده، کپی انجام میشه

مزایای این تکنیک عبارتند از:

- **صرفه‌جویی در حافظه چون داده‌ها فقط زمانی کپی می‌شن که تغییر کنن**
- **افزایش کارایی چون عملیات کپی فقط در صورت لزوم انجام میشه**
- **کاهش زمان ایجاد فرآیندها چون نیازی به کپی کامل داده‌ها نیست**

این تکنیک به ویژه در زمان استفاده از سیستم‌عامل‌هایی مثل Linuxکه از ()fork برای ایجاد فرآیند جدید استفاده می‌کنن، کاربرد زیادی داره

سوال ۶:

مدل‌های مختلف نخ‌ها به نحوه مدیریت نخ‌ها در سیستم‌عامل اشاره دارن:

- **Many-to-One : چندین نخ کاربر به یه نخ هسته‌ای نگاشت می‌شن. این مدل محدودیت‌هایی داره چون اگه یکی از نخ‌ها مسدود بشه، همه نخ‌ها مسدود می‌شن و نمی‌تونن از پردازنده‌های چند هسته‌ای استفاده کنن**
- **One-to-One : هر نخ کاربر به یه نخ هسته‌ای اختصاص داره. این مدل امکان استفاده از پردازنده‌های چند هسته‌ای رو فراهم می‌کنه و کارایی بالاتری داره، اما ایجاد نخ‌های زیاد می‌تونه منابع زیادی مصرف کنه**

- **Many-to-Many : چندین نخ کاربر به چندین نخ هسته‌ای نگاشت می‌شن. این مدل تعادلی بین کارایی و مصرف منابع ایجاد می‌کنه و امکان استفاده بهینه از پردازنده‌ها رو فراهم می‌کنه**

انتخاب مدل مناسب بستگی به نیازهای خاص سیستم و برنامه‌های در حال اجرا داره

سوال ۷ :

Implicit Threading به تکنیک‌هایی اطلاق میشه که در اون‌ها ایجاد و مدیریت نخ‌ها به صورت خودکار توسط سیستم انجام میشه و برنامه‌نویس نیازی به مدیریت دستی نخ‌ها نداره. این تکنیک‌ها شامل موارد زیر می‌شن:

- **Thread Pools :در این روش، تعدادی نخ از پیش ایجاد می‌شن و در صورت نیاز برای انجام وظایف جدید استفاده می‌شن. این باعث میشه که از ایجاد و نابودسازی مکرر نخ‌ها جلوگیری بشه و کارایی سیستم افزایش پیدا کنه**
- **OpenMP : یه API برای زبان‌های برنامه‌نویسیC ، ++Cو Fortran است که به برنامه‌نویسان اجازه میده که به راحتی برنامه‌های موازی بنویسن**
- **Grand Central Dispatch (GCD) :یه تکنولوژی در سیستم‌عامل‌های macOSو iOS است که مدیریت نخ‌ها رو به صورت خودکار انجام میده**

استفاده از این تکنیک‌ها باعث میشه که برنامه‌نویسان تمرکز بیشتری روی منطق برنامه داشته باشن و نیاز به مدیریت پیچیده نخ‌ها نداشته باشن

سوال ۸:

برای مدیریت تعداد زیادی اتصال همزمان، چندین رویکرد وجود داره:

- **یک فرآیند به ازای هر اتصال :برای هر اتصال یه فرآیند جدید ایجاد میشه. این رویکرد ساده است، اما مصرف منابع بالایی داره و مقیاس‌پذیری کمی داره**

- یک نخ به ازای هر اتصال : برای هر اتصال یه نخ جدید ایجاد میشه. این رویکرد مصرف منابع کمتری نسبت به روش قبلی داره، اما هنوز هم ممکنه با مشکلاتی مثل مسدود شدن نخ‌ها مواجه بشه
- استفاده از Thread Pool : تعدادی نخ از پیش ایجاد می‌شن و برای انجام وظایف مختلف استفاده می‌شن. این باعث میشه که از ایجاد و نابودسازی مکرر نخ‌ها جلوگیری بشه و کارایی سیستم افزایش پیدا کنه
- رویکرد رویداد-محور با I/O غیرمسدود : از یه یا چند نخ برای مدیریت تمامی اتصالات استفاده میشه و عملیات I/O به صورت غیرمسدود انجام میشه. این رویکرد مقیاس‌پذیری بالایی داره و برای سیستم‌هایی با تعداد زیاد اتصال مناسب است

انتخاب رویکرد مناسب بستگی به نیازهای خاص سیستم و منابع موجود داره

سوال ۹:

در مدل Many-to-Many، چندین نخ کاربر به چندین نخ هسته‌ای نگاشت می‌شن. این مدل مزایای زیادی داره، اما ممکنه با مشکلاتی هم مواجه بشه:

- مسدود شدن نخ‌ها : اگه یکی از نخ‌های هسته‌ای به دلیل عملیات I/O مسدود بشه، ممکنه نخ‌های کاربر مرتبط با اون هم مسدود بشن. برای جلوگیری از این مشکل، از Scheduler Activationsاستفاده میشه. در این روش، هسته سیستم‌عامل به کتابخانه نخ کاربر اطلاع میده که نخ هسته‌ای مسدود شده، تا کتابخانه نخ بتونه نخ‌های کاربر رو به نخ‌های هسته‌ای دیگه نگاشت کنه و از مسدود شدن جلوگیری بشه
- مدیریت پیچیده : مدیریت نخ‌ها در این مدل ممکنه پیچیده باشه، چون نیاز به هماهنگی بین هسته و کتابخانه نخ کاربر داره. برای ساده‌تر کردن این مدیریت، از تکنیک‌هایی مثل Thread Poolsو OpenMP استفاده میشه