بسمه تعالی

تمرین شماره ۱

سیستم‌های عامل

محسن کربلائی امینی، ۹۸۲۴۲۱۲۸

مهر ۱۴۰۲

سوال ۱:

الف)

اهداف کاربر:

۱. راحتی در استفاده
۲. قابلیت اتکا
۳. امنیت
٤. سرعت

اهداف سیستم:

۱. اجرای برنامه‌های کاربر و راحت‌کردن حل مشکلات کاربر
۲. راحتی در استفاده، طراحی(design)، پیاده‌سازی، و نگه داری
۳. استفاده از سخت‌افزار به صورت بهینه

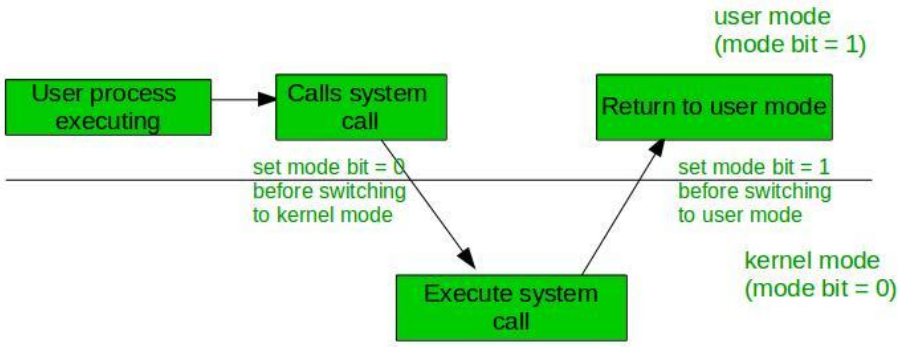
ب)

سیاست: چه چیزی نیاز هست تا انجام شود.(استراتژی انجام کار و تخصیص منابع)
مکانیزم: چگونگی انجام آن کار.(نحوه پیاده‌سازی استراتژی)

سوال ۲:

فضای عملیاتی دو حالته، برای ایجاد یک لایه محافظتی و پایداری سیستم طراحی شده‌است که فضای کاربر و فضای سیستم را از یکدیگر جدا میکند:

- فضای کاربر: هنگامی که برنامه در این فضا قرار دارد، دسترسی وی برای منابع سیستم محدود می باشد. کاربر می‌تواند با بهره‌مندی از sys call ها درخواست منابع مختلف سیستمی را تقاضا کند.
- فضای سیستم(کرنل): در این فضا محدودیتی وجود ندارد و دسترسی منابع به صورت کامل باز می‌باشد.



سوال ۳:

سیستم کال‌ها مکانیزمی برای برنامه‌های فضای کاربر هست تا از سیستم‌عامل منابع و یا سرویس‌هایی را درخواست کنند. برای مثال درخواست‌هایی برای دستکاری در فایل‌سیستم، تخصیص منابع حافظه یا پردازش، دسترسی به منابع ارتباطی سیستم و ...

به این ترتیب مدیریت دسترسی و تخصیص منابع توسط سیستم‌عامل امکان‌پذیر و عملی می‌شود.

سوال ٤:

۱. **Polling:**

در قالب یک فرایند چک کردن(نمونه برداری) در بازه‌های زمانی مختلف/منظم انجام می‌شود تا انجام شدن/نشدن یک اتفاق را بررسی کند و در صورت گرفتن نتیجه مناسب یک عملیاتی را انجام دهد.

برای مثال بررسی مقدار یک متغییر در یک حلقه بی‌نهایت و ادامه در صورت داشتن یک مقدار خاص.

۲. **Interrupt:**

در این فرایند به جای نمونه برداری، در هنگامی که اتفاق مورد نظر رخ می‌دهد، خود برنامه یک درخواست وقفه به سیستم ارسال می کند و سیستم از آن رخداد مطلع می‌شود و در صورت صلاحدید، تحت یک روتین عملیات مربوط به آن وقفه را انجام می‌دهد.

سوال ٥:

۱. **رابط کاربری:** رابطی که از کاربر فرامین مختلف را دریافت کند. این رابط می‌تواند انواع مختلفی همچون خط فرمان، گرافیکی و یا Batch دارند.
۲. **اجرای برنامه:** اجرای برنامه‌های دلخواه کاربر و نمایش نتایج آن(چه نتایج موفق و چه ناموفق)
۳. **عملیات ورودی/خروجی:** در اجرای برنامه‌های مختلف ممکن است برنامه به ورودی/خروجی‌های مختلف همچون نوشتن در یک فایل در دیسک و .. نیاز داشته باشد.

سوال ٦:

مزایا:

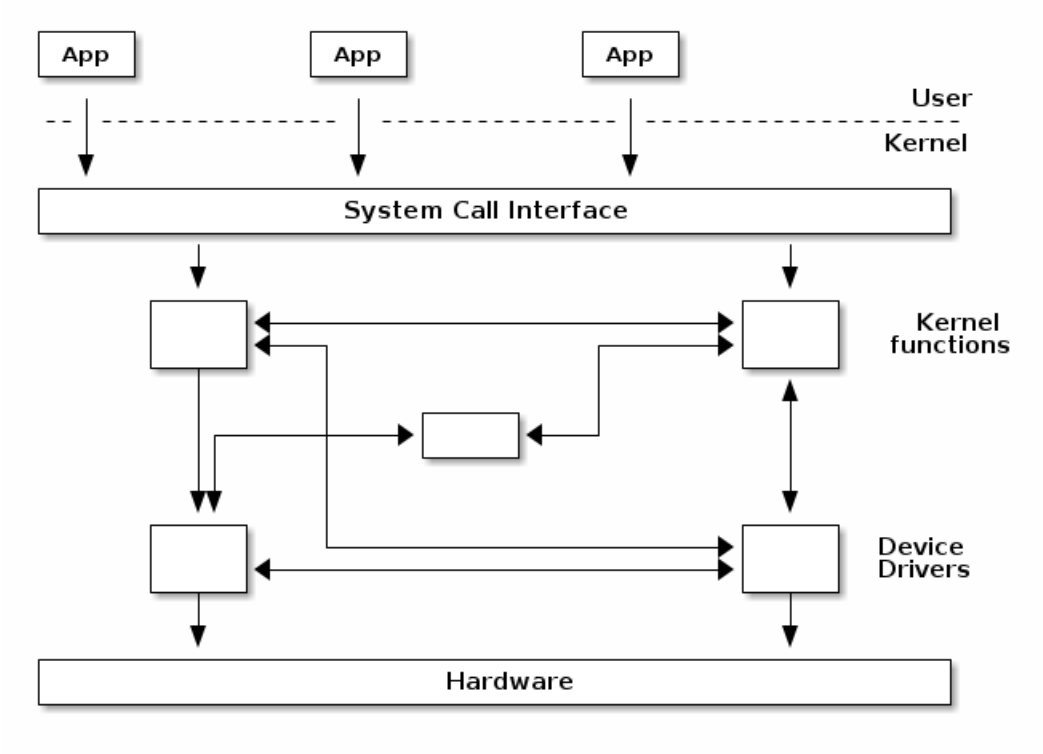
۱. کرنل راحت‌تر گسترش و توسعه داده می‌شود.
۲. سیستم‌عامل راحت‌تر برای معماری‌های جدید آماده و ارائه می‌شود.
۳. قابلیت اتکای بیشتر به دلیل کد کمتر در حال اجرا در کرنل
٤. امنیت بیشتر

در این نوع طراحی کرنل، سعی می‌شود هر بخشی از کرنل که قابلیت جدا شدن را دارد، از کرنل جدا و به فضای کاربر منتقل کنیم. به این صورت کرنل کوچک‌تر شده و برای مصارف سبک‌ترهم آمادگی بیشتری خواهد داشت.

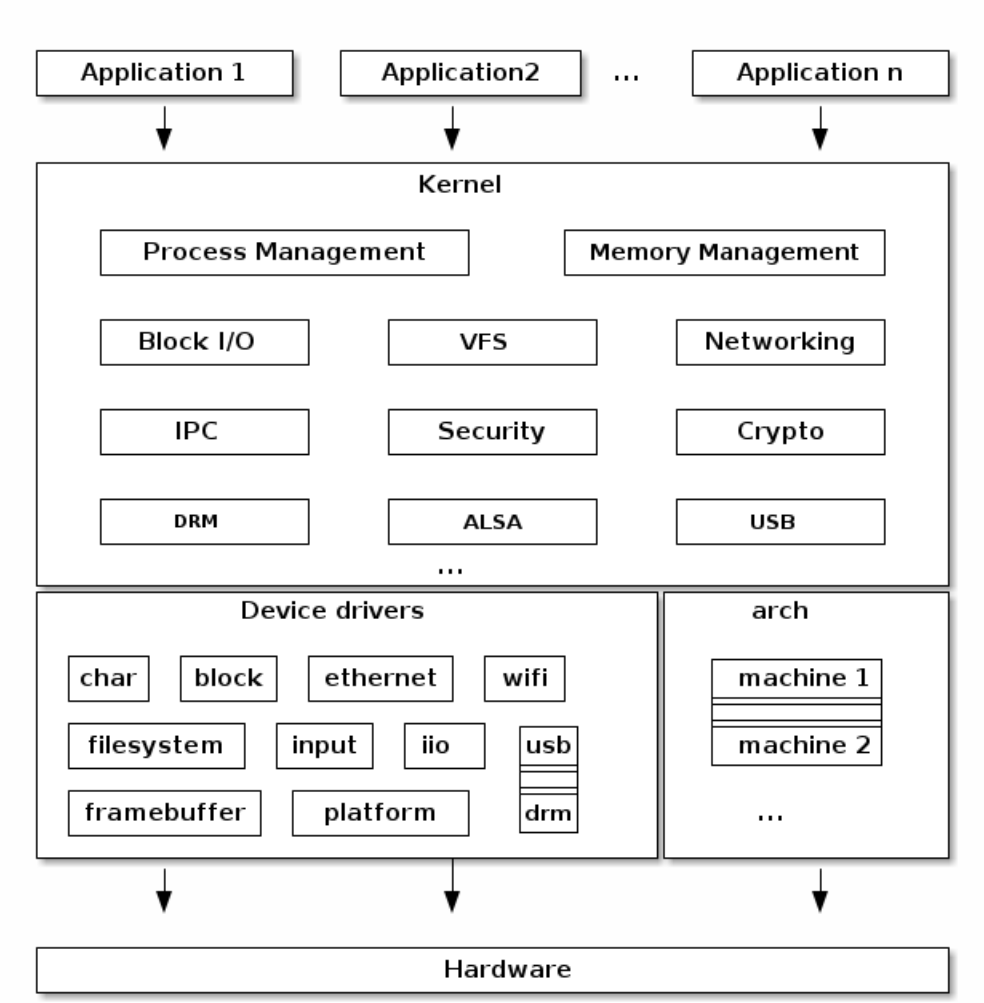
سوال ۷:

معماری کرنل لینوکس:

کرنل لینوکس اصطلاحا monolithic می‌باشد. به این معنی که بین تمامی سرویس‌ها و زیرسیستم‌های کرنل محافظت دسترسی وجود ندارد و این اجزا می‌توانند آزادانه با یکدیگر ارتباط برقرار کنند. البته بیشتر کرنل‌های monolithic داشتن یک جدایی logical میان زیرسیستم‌ها را enforce می‌کنند. این موضوع به خصوص در مورد هسته کرنل و درایورها، با ایجاد API های محدودشده انجام می‌شود.



معماری کلی کرنل لینوکس را می‌توان در چنین هیئتی تصور کرد:



توضیح برخی بخش‌های کرنل:

- **Arch**

در این بخش، کدهای وابسته به معماری قرار دارند. لینوکس در ابتدا برای کامپیوترهای بر پایه معماری ۳۲-بیت x۸۶ توسعه داده شد اما در حال حاضر بر روی بسیاری از معماری‌ها و ماشین‌های مختلف از جمله arm اجرا می‌شود.
کد لینوکس در این قسمت برای مدیریت interrupt ها، BUS ها و exception ها و ... که مربوط به سخت‌افزارهای گوناگون می‌باشد توسعه داده می‌شود.

- **Device drivers**

لینوکس درایورهای ارتباطی برای انواع مختلف سخت‌افزارها را پشتیبانی می‌کند. از جمله:
TTY, serial, SCSI, fileystem, ethernet, USB, framebuffer, input, sound, etc.

- **Process management**
  - standard Unix process management APIs such as fork(), exec(), wait()
  - standard POSIX threads

منبع:

https://linux-kernel-labs.github.io/refs/heads/master/lectures/intro.html#overview-of-the-linux-kernel