به نام خدا

آزمایشگاه سیستم عامل

آشنایی، اجرا و اشکال‌زدایی هسته سیستم عامل xv6

(بخش اول: آشنایی با xv6)

مقدمه

سیستم عامل xv6 یک سیستم عامل آموزشی است که در سال ۲۰۰۶ توسط محققان دانشگاه MIT به وجود آمده است. این سیستم عامل به زبان C و با استفاده از هسته Unix Version 6 نوشته شده و برروی معماری Intel x86 قابل اجرا می‌باشد. سیستم عامل xv6 علی‌رغم سادگی و حجم کم، نکات اساسی و مهم در طراحی سیستم عامل را دارا است و برای مقاصد آموزشی بسیار مفید می‌باشد. تا پیش از این، در درس سیستم عامل دانشگاه تهران از هسته سیستم عامل لینوکس استفاده می‌شد که پیچیدگی‌های زیادی دارد. در ترم پیشرو، دانشجویان آزمایشگاه سیستم عامل بایستی پروژه‌های مربوطه را برروی سیستم عامل xv6 اجرا و پیاده‌سازی نمایند. در این پروژه، ضمن آشنایی به معماری و برخی نکات پیاده‌سازی سیستم عامل، آن را اجرا و اشکال‌زدایی خواهیم کرد و همچنین برنامه‌ای در سطح کاربر خواهیم نوشت که برروی این سیستم عامل قابل اجرا باشد.

آشنایی با سیستم‌عامل xv6

کدهای مربوط به سیستم عامل xv6 از لینک زیر قابل دسترسی است:

https://github.com/mit-pdos/xv6-public

همچنین مستندات این سیستم عامل و فایلی شامل کدهای آن در صفحه درس بارگزاری شده است. برای این پروژه، نیاز است که فصل‌های ۰ و ۱ از مستندات فوق را مطالعه کرده و به سوالات زیر پاسخ دهید. پاسخ این سوالات را در قالب یک گزارش آپلود خواهید کرد.

- سوال ۱ : معماری سیستم عامل xv6 چیست؟ چه دلایلی در دفاع از نظر خود دارید؟
- سوال ۲ : یک پردازه[1] در سیستم‌عامل xv6 از چه بخش‌هایی تشکیل شده است؟ این سیستم‌عامل به طور کلی چگونه cpu را به پردازه‌های مختلف اختصاص می‌دهد.
- سوال ۳ : فراخوانی‌های سیستمی fork و exec چه عملی انجام می‌دهند؟ از نظر طراحی، ادغام نکردن این دو چه مزیتی دارد؟
- سوال ۴: مفهوم file descriptor در سیستم عامل‌های مبتنی بر Unix چیست؟ عملکرد pipe در سیستم‌عامل xv6 چگونه است و به طور معمول برای چه هدفی استفاده می‌شود؟

اجرا و اشکال‌زدایی

در این بخش به اجرای سیستم عامل xv6 خواهیم پرداخت. علی‌رغم این که این سیستم عامل قابل boot شدن مستقیم بر روی سیستم است، به دلیل آسیب‌پذیری بالا و رعایت مسائل ایمنی، از این کار اجتناب می‌کنیم و سیستم عامل را به کمک امولاتور qemu روی سیستم عامل لینوکس اجرا می‌کنیم. برای این منظور لازم است که کدهای مربوط به سیستم عامل را از لینک ارائه شده clone و یا دانلود کنیم. در ادامه با اجرای دستور make در دایرکتوری دانلود، سیستم عامل کامپایل می‌شود. در نهایت با اجرای دستور make qemu سیستم عامل برروی ماشین‌مجازی[2] اجرا میشود (توجه شود که فرض شده qemu از قبل برروی سیستم شما نصب بوده است. در غیر این صورت ابتدا آن را نصب نمایید).

[1] Process

[2] Emulator

اضافه کردن یک متن به Boot Message

در این بخش، شما باید نام اعضای گروه را پس از بوت شدن سیستم‌عامل روی ماشین مجازی qemu، در انتهای پیام‌های نمایش داده شده در کنسول نشان دهید.
تصویر این اطلاعات را در گزارش خود قرار دهید.

اضافه کردن چند قابلیت به کنسول xv6

- پس از اجرای سیستم عامل برروی qemu، مشاهده می‌کنیم که درصورت استفاده از کلید Tab معادل ASCII این کاراکتر در کنسول نوشته می‌شوند. همان‌طور که از تجربه‌ی استفاده از ترمینال لینوکس می‌دانید،استفاده از این کلید باعث می‌شود تا دستور در صورت امکان کامل شود.
  در این قسمت از پروژه، باید یک ویژگی مشابه به این ویژگی به کنسول xv6 اضافه کنید. به این‌صورت که شما باید همواره ۱۰ دستور آخری که کاربر وارد کرده را نگه دارید و در صورت فشردن کلید Tab دستور را با بهترین گزینه از میان آن‌ها تکمیل کند. در صورتی که در بین این ۱۰ دستور هیچ‌کدام به عنوان تکمیل‌شده‌ی این دستور قابل قبول نبود هیچ‌کاری انجام داده نشود و اگر چند گزینه وجود داشت آن دستوری که زودتر استفاده شده بود جایگزین شود.

- همچنین در صورتی که کاربر دستور Ctrl + r را وارد کرد باید تمامی ترمینال پاک شده و بتوان یک دستور جدید در ترمینال وارد کرد. این دستور همانند دستور clear در ترمینال لینوکس می‌باشد.

اجرا و پیاده‌سازی یک برنامه‌ی سطح کاربر

در این قسمت شما باید یک برنامه‌ی سطح کاربر و به زبان C بنویسید به برنامه‌های سطح کاربر سیستم‌عامل اضافه کنید. اسم این برنامه‌ی سطح کاربر touch می‌باشد. دو حالت کلی برای اجرای این برنامه وجود دارد.
زمانی که تنها یک ورودی اولیه به برنامه‌ی سطح کاربر داده شود، برنامه یک فایل جدید با آن اسم ایجاد می‌کند.

```
$ touch file.txt
```

زمانی که پرچم w- در این دستور فعال باشد برنامه باید منتظر ورودی گرفتن از کاربر باشد و سپس یک با فایل آن محتوا ایجاد کند. در صورتی که این فایل موجود باشد باید محتوا بر روی فایل قبلی بازنویسی[3] شود.

```
$ touch -w file.txt
Hello World!
```

از فراخوانی‌های سیستمی open، read، write و close استفاده کنید که برای باز کردن، خواندن، نوشتن و بستن فایل‌ها استفاده می‌شود.

[3] overwrite

نکات مهم

- برای تحویل پروژه ابتدا یک مخزن خصوصی در سایت GitLab ایجاد نموده و سپس پروژه خود را در آن Push کنید. سپس اکانت UT_OS_TA را با دسترسی Maintainer به مخزن خود اضافه کنید. کافی است در محل بارگذاری در سایت درس، آدرس مخزن، شناسه آخرین Commit و گزارش پروژه را بارگذاری نمایید.
- در نهایت کدهای خود را در کنار گزارش با فرمت pdf در یک فایل zip آپلود نمایید.
- به تمامی سوالاتی که در صورت پروژه از شما پرسیده شده‌است پاسخ دهید و آن‌ها را در گزارش‌کار خود بیاورید.
- همه‌ی اعضای گروه باید به پروژه‌ی آپلود شده توسط گروه خود مسلط باشند و لزوماً نمره‌ی افراد یک گروه با یکدیگر برابر نیست.
- در صورت مشاهده‌ی هرگونه مشابهت بین کدها یا گزارش دو گروه، نمره‌ی ۰ به هر دو گروه تعلق می‌گیرد.
- تمامی سؤالات را در **کوتاه‌ترین اندازه ممکن** پاسخ دهید.