پروژه اول آزمایشگاه سیستم عامل

اهداف آزمایش

- آشنایی با ابزارهایی که تو دنیا و آخرت به دردمون می‌خوره
- آشنایی با کامپایل، بوت و اشکال زدایی از لینوکس
- آشنایی با تکنولوژی‌های مجازی‌سازی[۱]
- آشنایی با کد منبع لینوکس

مقدمه

هدف از این پروژه آشنایی با ابزارهاییه که توی بقیه‌ی پروژه‌ها به دردتون می‌خوره و اگه منصفانه و با این نگاه، نگاه کنیم که قراره چیزی یاد بگیریم می‌بینیم که خیلی راحت از چیزایی که الان یاد می‌گیریم می‌تونیم برای کارهای دیگمون استفاده کنیم. هدف دیگه ما توی طراحی این پروژه این بوده شما چیزایی که قراره یاد بگیرید توی عمل باهاش مواجه بشید و تا جای ممکن سعی بر این بوده که از چیزایی که حافظه شما و نه توانایی‌هاتون رو استفاده می‌کنه پرهیز کنیم. تا درصد خیلی خوبی این پروژه این طوریه اما برای این که مطمئن بشیم همه کارا درست انجام شده مجبوریم موقع تحویل حافظه شما رو هم کند و کاو کنیم از همین جا عذر خواهی من رو بپذیرید. اگه کسی راه بهتری داره پیشنهاد بده اگه خوب باشه چرا اون طوری عمل نکنیم.

[۱] Virtualization

چکیده

این پروژه شامل این بخش‌هاست:

- آشنایی با ابزارها
  - مجازی‌سازی
  - نصب
  - اشکال‌زدایی
  - بازی کردن با GDB
- آشنایی با سایت lxr.free-electrons.com
- بررسی کد منبع
  - بوت شدن سیستم عامل
  - فراخوان سیستمی
- آشنایی با حالت‌های[۲] مختلف سیستم

در ابتدا با VirtualBox و Qemu به عنوان تکنولوژی‌های مجازی‌سازی کار و از آن‌ها برای نصب و اشکال‌زدایی از هسته استفاده می‌کنیم. در ادامه با GDB آشنا می‌شویم که می‌توانیم با آن در کد منبع سیستم‌عامل پیمایش کنیم و یک تمرین هیجان انگیز خواهیم داشت. وقتی که الان پیمایش توی کد رو یاد گرفتیم حیف نیست که یه سر به کرنل نزنیم ببینیم چی توشه؟ در مرحله بعد از شما می‌خوایم که کدهای لینوکس رو بررسی کنید و جواب چند سوال رو بدید. در پایان نیز با حالت‌های مختلف سیستم آشنا می‌شویم.

[۲] modes

گام اول: آشنایی با مجازی‌سازی، نصب و اشکال‌زدایی

آترین دانشجوی سال سوم برق این ترم درس سیستم عامل رو برداشته و می‌خواد پروژه‌های آزمایشگاه رو انجام بده، برای همین خوشحال و خندون کرنل لینوکس رو دانلود و یه کم کدش رو دست کاری کرد، بعد کامپایل، بعد ریست، بعد پوفففففففففف ..... لپ‌تاپ آترین به خاطر این‌که درایورهاش تنظیم نشده بودن ترکید.[۳]

دفعه بعد آترین از آرینا که ترم پیش اون هم این درس رو برداشته بود کمک گرفت و آرینا بهش گفت برای این که سیستم عامل رو نصب و اشکال‌زدایی کنن از نرم‌افزارهای مجازی‌سازی مثل VirtualBox و Qemu استفاده می‌کنند. استفاده از VirtualBox این امکان رو به ما می‌ده که کامپیوترمون نترکه و چند تا سیستم عامل رو هم زمان بیاریم بالا و باهاشون کار کنیم. Qemu هم این امکان رو به ما می‌ده که فقط هسته رو بدون چیزای اضافه بیاریم بالا. در مرحله بعد آترین لازم داشت که به کرنل وصل بشه و اون رو خط به خط اجرا کنه تا ببینه کجاهاش کار می‌کنه و کجاهاش کار نمی‌کنه. آرینا به آترین استفاده کردن از GDB رو پیشنهاد کرد. اما گفت که کارای ترمای پیش رو انجام نده که خیلی چیزی ازشون یاد نمی‌گیره. برای همین یه آموزش بهش داد که اگه اون رو انجام بده کامل یاد بگیره که چه طوری باید با GDB کار بکنه. توضیح اون رو توی ادامه این بخش میاریم. اما به جز اون آرینا بهش گفت از VirtualBox استفاده کنه و برای بالا آوردن کرنل هم از Qemu به شما هم پیشنهاد می‌دیم همین کار رو انجام بدید. آرینا یه سری ویدئو هم پیدا کرده بود که کمکتون می‌کنه راحت‌تر پروژه رو انجام بدید اون‌ها رو هم برای شما آپلود کردم. در کل هم آزادید از هر سیستم مجازی‌سازی و اشکال‌زدایی که دوست دارید استفاده کنید، اما اون سیستم باید نیازمندی‌های زیر رو برآورده کنه.[۴]

- باید بتونیم کرنل رو بالا بیاریم
- یه کرنل جدید نصب کنیم
- باید بتونیم کرنل رو تا یه خط خاص ببریم جلو
- باید بتونه خروجی‌های سیستم رو نمایش بده
- باید بتونید ثابت کنید که واقعا این کارها رو انجام دادید

---

[۳] این اتفاق احتمالش کمه روی کامپیوتراتون بیفته ولی بعید نیست، شاید ترم بعد یه داستان واقعی از شما تعریف کردم ☺

[۴] می‌تونیم بدون این که Qemu رو نصب کنیم به VirtualBox وصل بشیم و کرنل رو اشکال‌زدایی کنیم. خودم براش تلاش کردم اما موفق نشدم، شما شادی بتونید. اگه این کار رو کردید نمره اضافه می‌گیرید.

ما از شما موقع تحویل می‌خوایم که یه کرنل دیگه رو بوت کنید، با اشکال‌زا به اون وصل بشید، اون رو تا یک خط خاص جلو ببرید و خروجی‌های سیستم رو به ما نشون بدید. برای این مرحله در گزارش کار توضیح دهید که چه دستوراتی را وارد کردید و معنای این آرگومان‌های این دستورات چیست. برای مثال وقتی از دستور make menuconfig استفاده می‌کنید باید توضیح دهید که آرگومان menuconfig باعث چه کاری انجام می‌دهد.

توجه داشته باشید که وقتی با qemu سیستم را بوت می‌کنید وجود kernel panic یعنی یک جایی از کار ایراد دارد.

گام دوم: آشنایی با سایت lxr.free-electrons.com

آرینا ترم پیش یه سایت جالب پیدا کرده بود که بد نیست شما رو هم باهاش آشنا کنم. این سایت حالت یه مرجع رو داره که می‌شه توی کدهای لینوکس و نسخه‌های مختلفش دنبال یه چیزی گشت. برای مثال وقتی به آدرس سایت مراجعه می‌کنید یه مستطیل هست که می‌تونید توی اون کلیدواژتون رو بنویسید.

بعد اون براتون آدرس هر جایی که اون کلیدواژه اومده رو نشون می‌ده. اگه روی لینکی که براتون آورده کلیک کنید به جایی می‌ریم که اون کلیدواژه اومده و می‌تونید با کلیک روی بخش‌هایی که می‌خواید، توی فایل‌های هسته لینوکس جابجا بشید. این طوری موقع کار کردن با کدهای لینوکس سرگیجه نمی‌گیرید.

یه تابع یا یه Structure رو توی لینوکس انتخاب کنید، جایی که تعریف شده رو پیدا کنید و توضیحی درباره این که اون تابع یا Structure برای چی استفاده می‌شه توی گزارشتون بنویسید. معمولا این طوریه که اگه بچه‌ها با هم تعامل نداشته باشن یه پروژه‌ای که قراره ۵ ساعت طول بکشه ۵ روز طول می‌کشید. برای همین ما ازتون می‌خوایم چیزی که می‌خواید انتخاب کنید یکتا باشه یعنی گروه دیگه‌ای اون رو انتخاب نکرده باشه. این طوری مجبورید برید ببینید گروه‌های دیگه چی کار کردن :)

راستی ترمای پیش هم همین سوال رو داده بودیم. احتمالا بعضی‌هاتون برید سراغ این که از اون‌ها بپرسید ولی این کار کلا نیم ساعت بیش‌تر طول نمی‌کشه. پیدا کردن کسی که ترم پیش این کار رو کرده باشه و یادش باشه که چی کرده فکر کنم بیش‌تر طول بکشه.

گام سوم: بررسی کد منبع

مرحله نخست: بوت شدن سیستم عامل

دکمه‌ی پاور کامپیوترتون رو فشار می‌دید و بعد از چند لحظه صفحه ورود لینوکس ظاهر می‌شه. آیا تا الان از خودتون پرسیدید توی این فاصله چه اتفاقی می‌افته؟

صفر. پنج مرحله کار انجام می‌شه تا به صفحه ورود برسیم. اون‌ها چی هستند توضیح مختصری درباره هر کدوم بدید. (بعضی از سایت‌ها و مراجع شش مرحله برای بوت شدن در نظر می‌گیرند. اگر توضیح آن‌ها را هم بیاورید قبول است.)

اولین مسئله قابل توجه برای شروع به کار در سطح هسته سیستم عامل، مرور نحوه بوت شدن هسته است. با توجه به سوالات زیر می‌خواهیم این بخش از سیستم عامل را مورد بررسی قرار دهیم.

۱. فایل header.S از آدرس /arch/x86/boot را باز کنید. چرا این فایل به زبان اسمبلی است و آن را به زبان C ننوشته‌اند؟ دو دلیل برای آن بیاورید. با ذکر خلاصه‌ای از فرایند طی شده در این فایل (بدون ورود به جزئیات کدهای اسمبلی) با فرض بدون خطا بودن فرایند بگویید که در انتهای این فایل چه اتفاقی رخ می‌دهد؟ دلیل این کار چیست؟ چرا این فایل هنوزم در فایل‌های کرنل وجود دارد؟
۲. تابع main() از فایل main.c در آدرس /arch/x86/boot را به طور مختصر شرح دهید. در انتهای این تابع چه اتفاقی می‌افتد؟
۳. در ادامه‌ی راه تابع start_kernel فراخوانی می‌شود. آیا این تابع برای معماری‌های مختلف متفاوت است؟ توسط چه کدهایی فراخوانی می‌شود و مقدار بازگشتی آن به چه کسی برمی‌گردد؟ مسئولیت این تابع چیست و با چه دسترسی‌ای اجرا می‌شود؟
۴. در ادامه فراخوانی console_init() انجام می‌شود. می‌توانستیم بعد از بوت شدن کامل سیستم مقداردهی کنسول را انجام دهیم پس چرا در مراحل ابتدایی بوت شدن عملیات مقدار دهی کنسول را انجام می‌دهیم؟ اگر این کار را انجام ندهیم با چه مشکلی مواجه می‌شویم.

مرحله دوم: فراخوان سیستمی[۵]

آخرین مسئله مورد بررسی نحوه‌ی عملکرد فراخوان‌های سیستمی هستند. برای بررسی این موضوع در معماری x86 به فایل entry_32.S در آدرس /arch/x86/kernel مراجعه کنید. به طور مختصر شرح دهید که اجرای تابع مربوطه به فراخوانی سیستمی و دریافت نتایج چگونه انجام می‌شود.

(راهنمایی: برای شروع فرآیند می‌توانید به خط ENTRY(system_call) از همین فایل یا فصل پنجم کتاب Love مراجعه کنید.)

مرحله سوم: فایل desctiptors

می‌توانیم هر تعداد فراخوان سیستمی که لازم داشته باشیم به سیستم اضافه کنیم. در پروژه‌های آینده با این فرآیند آشنا خواهید شد. اما این کار نیازمند این است که کرنل را کامپایل کنیم که محدودیت نسبتا بزرگیست. این جاییست که file desctiptors به ما کمک می‌کنند.

به منبع [1] مراجعه کنید و با توجه به آن به سوالات زیر پاسخ دهید.

۱. File desccriptors را تعریف کنید.
۲. چگونه توسط آن می‌توان دستور pipe کردن در ترمینال را پیاده‌سازی کرد؟
۳. مثال دیگری از استفاده کردن از file descriptors را توضیح دهید.
۴. هنگامی که از دستور open برای باز کردن یک فایل استفاده می‌کنیم چه اتفاقی می‌افتد؟

[۵] System call

گام چهارم: حالت‌های[۶] مختلف سیستم

آنوشا[۷] طی اکتشافاتش در درس سیستم عامل به این نتیجه رسید که سیستم عامل حالت‌های مختلفی داره User mode و Kernel mode. بعضی چیزها Kernel mode هستن و بعضی دیگه User mode. آنوشا یه شرح کامل نوشت و توش درباره این حالت‌ها توضیح داد و این که چه چیزهایی توی Kernel mode باشه بهتره چه چیزهایی توی User mode. یه تحلیل هم نوشت که چه تفاوت‌های کلیدی بین این دو حالت هست و داد به ما. اما متاسفانه این برگه رو گم کردیم و ایشون هم رفتند. الان هم به شدت به چیزی شبیه به اون احتیاج داریم. برای همین از شما می‌خوایم که یه چیزی مثل اون برای ما درست کنید[۸] و توش جواب سوالاتی که گفتیم بدید.

یادش به خیر آنوشا حواسش نبود که حالت‌های دیگه‌ای به نام Real Mode و Protected Mode داریم. ما از بچه‌ها خواستیم که برای ما اطلاعاتی جور کنند که بتونیم به رخ بکشیم اما اون اطلاعات به همراه اطلاعات بالا گم شده. حدس ما اینه که دست‌های کثیفی توی کاره. توی اون برگه‌ها تفاوت‌های کلیدی این دو حالت، این که چه دسترسی‌هایی در هر حالت داریم و هدف اصلی از اضافه کردن Protected Mode به سیستم‌ها نوشته شده بود اما چه فایده که گم شدند. برای همین از شما می‌خوایم به ما کمک کنید و چیزی شبیه به اون برای ما آماده کنید. ممنون.

لطفا درباره درایورها هم تحقیق کنید و توضیح دهید اون‌ها در چه mode ای قراد دارند و اجرا می‌شوند.

---

[۶] modes

[۷] یکی که ترم قبل از آرینا درس رو داشت.

[۸] آرینا و بچه‌های ترمای پیش درست کرده بودن ولی چیز به درد بخوری نبود.

نکات آخر

- باید جواب تمام سوالات و مراحلی رو که طی کردید توی گزارشتون بنویسید.
- به این علت که مطمئن بشیم پروژه رو خودتون انجام دادید و گزارش رو خودتون نوشتید هنگام تحویل از شما سوالاتی خواهیم پرسید و مجبوریم سخت‌گیری کنیم.
- اگر هنگام تحویل جواب سوالات را بلد نباشید از نمره شما کسر خواهد شد اما ملاک برای نمره دهی به شما پاسخ‌هاییست که در گزارش نوشته‌اید.
- به علت ذات پروژه‌های آزمایشگاه ممکن است به مشکلات پیش‌بینی نشده‌ای بر بخورید. برای همین توصیه می‌کنیم زود شروع کنید.
- ملاک نمره دهی مقدار کار هر یک از اعضای گروه است و لزوما نمره یکسانی به آن‌ها تعلق نخواهد گرفت.
- گزارش شما نباید بیش‌تر از ۵ صفحه A4 شود.

منابع

1. https://www.bottomupcs.com/file_descriptors.xhtml