علی اطهری

پرهام سلطانی

پروژه سیستم عامل

نصب و راه اندازی

ابتدا با توجه به راهنمایی داخل متن پروژه موارد مورد نیاز رو نصب کردیم و یک بار با دستور make qemu سیستم عامل رو اجرا کردیم.

Qemu یک تقلید کننده (Emulator) است که برای اجرا فرایند های کامپیوتری استفاده میشه. یک سری از ویژگی های Qemu که باعث میشود گزینه مناسبی برای شبیه سازی و تقلید سیستم عمال باشد عبارت است از :

- تقلید سیستم: توانایی تولید یک مدل مجازی کامل از یک ماشین که شامل cpu و حافظه و ... است.
- تقلید حالت کاربر: قابلیت اجرا یک برنامه که برای مجموعه دستورات مختلف کامپایل شده است.
- پشتیبانی Hypervisor : عملکرد به عنوان یک مدریت کننده ماشین های مجازی.
- پشتیبانی معماری : قابلیت تقلید معماری های مختلف مانند x86, ARM, PowerPC, RISC-V .
- نگهداری حالت : قابلیت ذخیره و بازیابی حالت یک ماشین مجازی.

ایجاد سیستم کال ساده

برای ایجاد یک سیستم کال ساده ابتدا تابع مورد نظر رو در فایل های زیر تعریف کردیم (مانند بقیه سیستم کال های تعریف شده):

syscall.h

syscall.c

بعد از اون خود عملکرد تابع رو توی فایل زیر نوشتیم :

sysproc.c

در این مرحله سیستم کال فقط یک عدد ثابت رو برمیگردوند و کار خاصی نمیکرد. برای تست صدا زدن سیستم کال فایلی به نام mytest.c رو ایجاد کریدم و داخلش یه کد ساده نوشتیم که سیستم کال رو صدا میزد. برای اجرا این فایل اون رو داخل Makefile در بخش های مربوطه اضافه کردیم. با انجام این کار هنگام بالا اومدن سیستم عامل فایل mytest کامپایل میشود و میتوان آن را اجرا کرد. با این کاری وقتی make qemu رو اجرا میکنیم و سیستم عامل بالا میاد میتونیم با دسترو mytest فایل که نوشته بودیم رو اجرا کنیم.

Linker.id یک اسکریپت است که موقع build شدن سیستم عامل xv6 به کار میرود. این اسکریپت مستقیما به qemu مربوط نیست ولی برای کامپایل شدن xv6 ضروریه. این اسکریپت سمبل های مختلف رو تعریف میکنه و ساختار حافظه را مشخص میکند. مانند آدرس قسمت هایی که کد و دیتا باید لود شود و ... . این اسکریپت مشخص میکنند که قسمت های مختلف کرنل کجای حافظه باید باشند.

در این قسمت متوجه شدیم برای اجرای سیستم کال باید آن را در دو قسمت دیگه نیز تعریف کنیم. تغییرات لازم رو در فایل های زیر دادیم:

user.h

usys.s

با اضافه کردن تعریف سیستم کال در این دو فایل سیستم کال به درستی اجرا شد.

در مرحله بعد سیستم کال رو تغییر دادیم تا جدل فرایند ها رو به صورت ساده پرینت کند. اینجا موتجه شدیم که برای دستری به جدول فرایند باید تابع رو داخل فایل proc.c بنویسیم. و تعریفش رو در فایل defs.h وارد کنیم.

در نتیجه تابع جدید در فایل proc.c نوشتیم و آن را از داخل سیستم کال خود صد زدیم.

آرگومان ها و تغییر منطق سیستم کال

ابتدا یک آرگومان ساده اضافه کردیم و عملکرد رو بررسی کردیم . نیاز بود که در بعضی فایل هایی که سیستم کال رو تعریف کرده بودیم تغییر ایجاد کنیم.

بعد از ارسال یک آرگومان ساده تلاش کردیم که پوینتر به یک struct رو پاس بدیم. ابتدا ساختاری به نام process_info_t رو ایجاد کردیم و در فایل process_info_t.h گذاشتیم وتعریف سیستم کال رو عوض کردیم.

نمیتوان به صورت مستقیم پوینتر رو از ورودی تابع دریافت کرد. زیرا سیستم کال های xv6 طوری طراحی شده اند تا با تعداد ورودی های ثابت که پونتر و یا عدد هستند کار کنند. به جای پاس دادن مستقیم ساختار، باید از argptr استفاده کرد.

ما ابتدا یک پونتر رو پاس دادیم و داخل سیستم کال به متغییر آن را عوض کردیم و درست کار کرد و بیرون سیستم کال میتوان آن تغییرات رو خواند.

برای state یک فرایند هم یک عدد دریافت میکنیم که متانظر مقدار های زیر است:

UNUSED, EMBRYO, SLEEPING, RUNNABLE, RUNNING, ZOMBIE

با توجه به ورودی های گرفته شده روی جدول فرایند حرکت میکنیم و فرایندی که با ورودی ها مطابقت داشت رو انتخاب میکنیم و اطلاعات آن را داخر پونتری که به ساختار process_info_t دریافت کرده بود میریزیم.

برای تست عملکرد سیستم کال فایل mytest رو تغییر دادیم و بعد از اجرا make qemu دستور mytest رو اجرا میکنیم.

آدرس گیت هاب

https://github.com/Ali-Athary/OS_Project/tree/syscall-ps

توجه کنید آخرین تغییرات روی برنچ syscall-ps قرار دارد.