سیستم عامل

پروژه دوم آزمایشگاه : فراخوانی سیستم

تاریخ تحویل : ۲۰ آبان

اهداف این پروژه

- **اشنایی با روش های مختلف دسترسی به سرویس های هسته**
- **پیاده سازی یک فراخوانی سیستم ساده**
- **اشنایی با ابزار های ردگیری**

مقدمه

برنامه‌ها به دلایل متفاوتی نمی‌توانند تمام نیازهایشان را در سطح کاربر برطرف کنند و به برخی سرویس‌هایی که فقط توسط هسته ارائه می‌شوند نیاز پیدا می‌کنند. متداول ترین روش برای دسترسی به سرویس‌های هسته، فراخوانی سیستمی است.

برای هماهنگی بیشتر استفاده از کرنل ورژن 3.16 که در لینک زیر قابل دسترسی است توصیه می‌شود.

https://www.kernel.org/pub/linux/kernel/v3.x/linux-3.16.tar.xz

گام اول :اشنایی با روش‌های مختلف دسترسی به سرویس‌های هسته

این گام را با بیان چند سوال برای یادآوری اهمیت مساله‌ی فراخوانی سیستمی، آغاز میکنیم.

اصولا یک برنامه در سطح کاربر چه کارهایی را نمی‌تواند انجام دهد و نیازمند چه سرویس‌هایی از هسته می‌شود؟

فراخوانی‌های سیستم با توجه به کارکرد هایشان چگونه دسته‌بندی می‌شوند؟

اصولا وجود داشتن دو سطح کاربر و هسته به چه علت بود؟

پاسخی مختصر برای هر یک سوالات زیر در گزارش خود بیاورید.

پارامترها در پردازنده‌های x86 چطور ارسال می‌شوند و حداکثر چند پارامتر قابل ارسال است؟

با توجه به محدود بودن حداکثر تعداد پارامتر قابل ارسال، برای ارسال تعداد بیشتر پارامتر باید چه کرد؟

تابع copy_from_user چه استفاده‌ای در رابطه با فراخوانی‌های سیستم دارد؟ چرا دو فضای آدرس برای کرنل و کاربر در نظر گرفته شده است؟

و در پایان آیا روش های دیگری غیر از فراخوانی سیستمی برای دسترسی به اطلاعات و سرویس‌های هسته وجود دارد؟ توضیح مختصری در پاسخ به این سوال در گزارش خود بیاورید.

فراخوانی سیستمی دو بخش دارد: یک بخش که بستگی به معماری سخت افزار سیستم دارد و بخش دیگر آن یک handler function به زبان C است. هر بخش آن در کجا قرار دارند؟ با مثال توضیح دهید.

برای مثال فایل اسمبلی entry_64.S در معماری ۸۶x سیستم‌های ۶۴ بیتی کنونی کارهای زیادی انجام می‌دهد که یکی از آن ها آماده سازی شرایط برای اجرای یک فراخوانی سیستمی است. منظور از آماده‌سازی این است که روی رجیسترها تغییراتی ایجاد می‌کند و کنترل پردازه کاربر را سلب می‌کند و به پس از تعیین فراخوانی سیستم مرتبط به شروع کد مربوط به آن پرش می‌کند. این تغییرات رو رجیسترها و نحوه تعیین فراخوانی سیستم مرتبط را توضیح دهید.
دستور SYSCALL چیست و دستوری که قبلا از آن استفاده می‌شد چه بود؟

گام دوم: پیاده سازی یک فراخوانی سیستم ساده

پس از آن‌که با نحوه انتقال کنترل به هسته آشنا شدید، اکنون می‌توانید یک Handler Function برای فراخوانی سیستم بنویسید. و با نحوه رد کردن پارامترها به فراخوانی سیستم و مقدار بازگشتی آن نیز آشنا شوید.
برای انجام این کار مستندات خوبی در اینترنت (خصوصا برای کرنل ورژن 3.16) و دیگر منابع موجود است. مطالعه فصل های مرتبط با فراخوانی سیستم در کتاب های قرار داده شده بر روی سایت درس توصیه می‌شود.
یک فراخوانی سیستمی hello world بنویسید.
فراخوانی hello_there بنویسید که با گرفتن یک نام به عنوان پارامتر به آن نام سلام کند!
فراخوانی hello_user بنویسید که به نام یوزری که این فراخوانی را انجام داده سلام کند.

و در پایان آیا اضافه کردن فراخوانی سیستمی برای هر سرویس جدید در سیستم عامل کار صحیحی است؟ چرا؟

گام سوم: آشنایی با ابزارهای ردگیری (Tracing)

در این بخش چند ابزار متفاوت که در سطوح متفاوتی عمل می‌کنند اما کارکرد همه‌ی آن ها نوعی ردگیری است آشنا می‌شوید و از آن ها استفاده می‌کنید.
در کرنل یک فراخوانی سیستمی وجود دارد که با آن پردازه‌ها را ردگیری می‌کنند، نام این فراخوان سیستمی چیست؟ پارامترهای آن را توضیح دهید و بگویید چگونه می‌توان با آن ابزار strace را پیاده‌سازی نمود؟
با اینکه فراخوانی‌های سیستمی روشی برای دسترسی بدون واسطه به سرویس‌های هسته اند، اما اغلب فراخوانی‌های سیستمی توسط کتابخانه‌ها انجام می‌شود و برنامه‌نویس بدون درگیری جزییات سطح پایین کدش را با استفاده از توابعی که کتابخانه‌ها در اختیارش قرار می‌دهند می‌نویسد. چرا فراخوانی‌های سیستمی، مستقیما در برنامه فراخوانده نمی‌شوند؟
برخی فراخوانی‌های سیستمی تقریباً بدون هیچ تغییری توسط توابع کتابخانه‌ای (wrapperها) پیاده‌سازی شده و در برنامه فراخوانی می شوند. اما برخی دیگر در پیاده‌سازی توابع کتابخانه‌ای استاندارد یا حتی توابع کتابخانه‌ای دیگر به کار گرفته می‌شوند. به عنوان مثال پیاده‌سازی تابع کتابخانه‌ای استاندارد malloc مستلزم استفاده از فراخوانی‌های سیستمی است. دو فراخوانی سیستمی که توسط این تابع، فراخوانی می‌شوند را نام

برده و کارکرد آن‌ها را بگویید. (امتیازی: یک مورد استفاده از هر کدام را به طور کامل توضیح داده و ردگیری نمایید. یعنی پارامترهای ورودی malloc که منجر به اجرای آن‌ها می‌شود را وارد نمایید و حین ردگیری، علت پیمایش مسیر اجرایی مربوطه را توضیح دهید. برای هر فراخوانی سیستم تنها یک مسیر اجرایی را به طور کامل توضیح دهید کافی است.)

به منظور ردگیری کتابخانه‌های استفاده شده‌ی یک برنامه می‌توان از ابزار ltrace بهره برد. دقت شود که ltrace فقط می‌تواند کتابخانه‌های مشترک که به صورت داینامیک لینک شده‌اند را تشخیص بدهد و کتابخانه‌ای که استاتیک لینک شده با کد اصلی برنامه ادغام شده و قابلیت تشخیص آن کتابخانه توسط ltrace وجود ندارد.

برای این‌کار شما باید برنامه‌ی دلخواه را بیلد کنید و پنج کتابخانه پرکاربردتر منجر به فراخوانی سیستمی را با ltrace بیابید. سپس توابع کتابخانه‌ای مربوط به آن فراخوانی ها را خودتان با glibc ای که خودتان دانلود کردید و بیلد کردید لینک کنید. دلیل این‌کار این است که بیلد کردن خود برنامه پیچیده خواهد بود

دقت شود که glibc فقط باید در سیستم شما بیلد شود و نباید نصب بشود.

می‌توانید glibc را از آدرس زیر دریافت کنید:

https://ftp.gnu.org/gnu/glibc/

شما به بتوانید به کمک یک دیباگر مانند gdb خط به خط آن را اجرا کنید و دقیقا مشاهده کنید که در کدام خط از اجرای توابع کتابخانه‌ای از فراخوانی سیستمی استفاده شده است.

نکات آخر

- در نظر داشته باشید که باید پاسخ سوالات را برای معماری ۶۴ بیتی بنویسید.
- باید جواب تمام سوالات خواسته شده و مراحل طی شده را در گزارش بنویسید.
- هنگام تحویل به جواب سوال ها و گزارشتان مسلط باشید.
- ملاک نمره دهی مقدار کار هر یک از اعضای گروه است و لزوما نمره یکسانی به همه اعضا تعلق نمیگیرد.
- توصیه می‌شود که پروژه را زود شروع کنید و سوال‌هایتان راهم زودتر بپرسید.