دانشگاه صنعتی اصفهان

دانشکده برق و کامپیوتر

آزمایشگاه سیستم عامل

دستور کار جلسه هشتم

آبان ۱۴۰۱

فهرست مطالب

۱ فراخوانی سیستم

فراخوانی سیستم[1] روشی است که در آن برنامه در حال اجرا از کرنل سیستم عاملی که روی آن اجرا می شود، درخواست سرویس می دهد. در واقع فراخوانی سیستم راه تعاملی برنامه ها با سیستم عامل می باشد. فراخوانی‌های سیستمی از طریق API خدماتشان را به برنامه های کاربر ارائه می دهند و از این طریق برنامه ای که در سطح کاربر اجرا می شود می‌تواند از سیستم عامل سرویس‌های سطح پایینتری که مربوط به سیستم عامل است استفاده نماید.

۲ هدف آزمایش

در این آزمایش قصد داریم با شیوه اضافه کردن یک فراخوانی سیستمی در کرنل لینوکس آشنا شویم. در این مسیر علاوه بر اضافه کردن یک فراخوانی سیستم با کامپایل کردن کرنل و initramfs آشنا می‌شویم. لطفا مراحل را با دقت و گام به گام انجام دهید، تا به نتیجه ای که این دستور کار برای آن طراحی گردیده است برسید.

۳ آماده سازی کرنل

برای اجرای این آزمایش از کرنل نسخه ۵٫۱۵٫۷۷ استفاده شده است. فایل کرنل در فایل فشرده در اختیارتان قرار گرفته است. فرآیند کامپایل کردن کرنل طولانی است و ممکن است ساعت‌ها طول بکشد، اما راهکارهایی برای سریع تر کردن این فرآیند وجود دارد. یکی از این راهکارها غیر فعال کردن تنظیمات کرنل است. برای تنظیم کرنل از دستورات مختلفی استفاده می‌شود، یکی از این دستورات make tinyconfig است(برای اطلاعات بیشتر به اینجا مراجعه کنید). این دستور کمترین تنظیمات ممکن را برای کرنل فعال می‌کند. خروجی دستور make tinyconfig یک فایل config. است که در پوشه کرنل اضافه می‌شود(فایل config. را با نرم افزار vim باز کنید و محتویات آن را مشاهده کنید). کرنلی که با تنظیمات tinyconfig کامپایل می‌شود برای آزمایشگاه سیستم عامل کفایت نمی‌کند، به همراه دستور کار فایل تنظیمات کرنل با نام kconfig در اختیارتان قرار می‌گیرد. این فایل را با نام config. در داخل پوشه کرنل

[1] system call

کپی کنید(از مسئول آزمایشگاه درباره تنظیمات اضافی که در kconfig اضافه بر تنظیمات tinyconfig فعال شده است سوال کنید). برای راحتی این فایل در فایل فشرده‌ی کرنلی که در اختیارتان قرار گرفته است، جانمایی شده و در صورتی که از کرنل ارائه شده استفاده کنید، نیازی به کپی وجود ندارد.

٤ یک فراخوانی سیستم ساده

به مسیر کرنلی که از حالت فشرده خارج کردیم می‌رویم، پوشه‌ی hello را می‌سازیم و وارد آن می‌شویم:

```
cd /usr/src/linux-4.19.257
mkdir hello
cd hello
```

فایلی با نام hello.c بسازید و کد زیر را وارد نمایید.

```
#include <linux/kernel.h>
#include <linux/syscalls.h>

SYSCALL_DEFINE1(hello, int, number)
{
	printk("Hello From Kernel\n");
	return number*number;
}

SYSCALL_DEFINE2(hello2, int, number, int, number2)
{
	printk("Hello From Kernel2\n");
	return number*number2;
}
```

نکته: ماکرو SYSCALL_DEFINEn برای معرفی یک فراخوانی سیستم استفاده می‌شود و مقدار n تعداد پارامترها را مشخص می‌کند. ورودی اول در این ماکرو نام فراخوانی سیستم است. پس از آن به ازای هر پارامتر نوع و سپس نام پارامتر می‌آید.

نکته۲: فراخوانی سیستم فقط مقدارهایی از نوع long را باز می‌گرداند(return می‌کند).

یک فایل Makefile در کنار hello.c با محتویات زیر ایجاد کنید.

```
obj-y := hello.o
```

فایل Makefile کرنل را باز می‌کنیم و به دنبال خط زیر می‌گردیم:

```
core-y          += kernel/ certs/ mm/ fs/ ipc/ security/ crypto/ block/
```

و در انتهای خط پیدا شده پوشه hello را به صورت زیر اضافه می‌کنیم:

```
core-y          += kernel/ certs/ mm/ fs/ ipc/ security/ crypto/ block/ hello/
```

برای معرفی فراخوانی سیستم باید مانند هر تابع دیگر، تعریف آن را در یک فایل header اضافه کنیم. برای اینکار باید فایل زیر را ویرایش کنیم.

```
cd include/linux/syscalls.h
```

قبل از #endif که در آخر فایل قرار دارد، تعریف تابع را به صورت زیر را وارد نمایید.

```
asmlinkage long sys_hello(int);
asmlinkage long sys_hello2(int, int);
```

نکته: درصورتی که به عنوان پارامتر اول به ماکرو SYSCALL_DEFINEn مقدار x داده شود، نام فراخوانی سیستم برابر با sys_x خواهد بود.

پس از تغییر Makefile باید فایل syscall_64.tbl را تغییر دهیم(توجه شود که اگر از نسخه ۳۲ بیتی استفاده می کنید فایل syscall_32.tbl را می بایستی ویرایش نمایید).

```
cd arch/x86/entry/syscalls/
nano syscall_64.tbl
```

جدول system call از ۴ ستون تشکیل شده است که هر ستون با tab از مقدار قبلی جدا شده است. فرمت این جدول به شکل زیر است:

```
<number> <abi> <name> <entry point>
```

- number: فراخوانی‌های سیستم با یک شماره واحد شناخته می شوند. به هنگام استفاده از یک فراخوانی سیستم شماره آن به کرنل اطلاعات داده می‌شود.

- ABI: مخفف Application Binary Interface است. برای اینکه مشخص شود که فراخوانی سیستم مختص معماری ۶۴ بیتی(x64) یا معماری ۳۲ بیتی(x32) یا هر دو(common) است.
- name: نام system call
- entry point: نام تابعی که قرار است system call با آن صدا زده شود که حتما با پیشوند sys_ شروع می شود.

به آخرین ردیف رفته و مقادیر زیر را وارد جدول نمایید. در اینجا آخرین فراخوانی سیستم شماره ۵۴۷ است، یک شماره به آن اضافه می کنیم و فراخوانی سیستم با شماره ۵۴۸ را اضافه می‌کنیم.

```
548   common   hello          sys_hello
549   common   hello2         sys_hello2
```

در نهایت با دستور make کرنل را کامپایل می‌کنیم. پس از تکمیل فرآیند کامپایل، فایلی با نام bzImage در مسیر arch/x86/boot/bzImage ایجاد می‌شود که کرنل کامپایل شده است.

نکته: در این آزمایش قصد داریم تا کرنل را با سرعت کامپایل کنیم و به همین دلیل برای استفاده از آن نیز از روشی ابتکاری استفاده می‌کنیم(که در ادامه با آن آشنا می‌شویم). در صورتی که بخواهیم کرنل سیستم را برای استفاده واقعی کامپایل کنیم مراحل زیر طی می‌شود:

- با استفاده از دستور make oldconfig تنظیمات کرنل جاری را برای کرنل جدید فعال می‌کنیم.
- با استفاده از دستور make کرنل کامپایل می‌شود.
- با استفاده از دستور make modules_install کرنل ماژول‌ها را کامپایل می‌کنیم.
- با استفاده از دستور make install کرنل را روی سیستم نصب می‌کنیم.
- با استفاده از دستور update-initramfs -c -k 5.15.77 فایل initramfs را بروزرسانی می‌کنیم(با این فایل در ادامه آشنا می‌شویم).
- با استفاده از دستور update-grub کرنل جدید را به عنوان یکی از کرنل‌هایی که سیستم می‌تواند با استفاده از آن اجرا شود را به boot menu اضافه می‌کنیم.

۵ یک برنامه ساده برای استفاده از فراخوانی سیستم

تا اینجا توانستیم یک کرنل را کامپایل کنیم. برای اینکه ببینیم آیا تابعی که به عنوان فراخوانی سیستم ایجاد کردیم به درستی فراخوانی می شود با استفاده از قطعه کد زیر برنامه ساده‌ای می‌نویسیم و کامپایل می‌کنیم.

```
#include <stdio.h>
#include <linux/kernel.h>
#include <sys/syscall.h>
#include <unistd.h>
int main()
{
        long ret = syscall(548, 4);
        printf("System call sys_hello returned %ld\n", ret);
        return 0;
}
```

نکته: در صورتی که کرنل را برای استفاده واقعی کامپایل کرده باشید می‌توانید کد بالا را با gcc کامپایل کنید و از آن استفاده کنید، در غیر این صورت(استفاده از kconfig) باید قطعه کد بالا را با استفاده از مراحلی که در بخش بعد توضیح داده شده است کامپایل و برای اجرا آماده کنید.

۶ آشنایی با Initramfs

به هنگام اجرای کرنل فایلی با نام initramfs به عنوان پارامتر ورودی به کرنل داده می‌شود که کرنل پس از اجرای خود آن فایل را از حالت فشرده خارج می‌کند. داخل فایل initramfs مجموعه‌ای از فایل‌ها وجود دارد که کرنل آن‌ها را در حافظه موقت بارگذاری می‌کند(معمولا initramfs شامل تمام پوشه‌هایی که در مسیر ریشه لینوکس وجود دارد مانند proc/، dev/، run/ و ... است.)؛ از این فایل‌ها برای بارگذاری کامل فایل‌های مرتبط با سیستم عامل که روی دیسک نصب شده‌اند کمک گرفته می‌شود(در مراحل اولیه بارگذاری سیستم عامل ممکن است بسیاری از قابلیت‌ها، از جمله دسترسی به دیسک به صورت کامل وجود نداشته باشد، initramfs کمک می‌کند تا ماژول‌های مورد نیاز بارگذاری شود و دسترسی به پارتیشنی که فایل‌های سیستم عامل روی آن نصب شده است برقرار شود). Initramfs در ساده ترین حالت حداقل یک فایل باینتری یا اسکریپت به نام init دارد. فایل init اولین فرآیندی است که توسط

کرنل(PID 1) اجرا می‌شود، بنابراین والدی ندارد و نباید هیچ وقت خاتمه پیدا کند. دقت داشته باشید که فایل init، یک برنامه کاربردی است، بنابراین می‌توان از آن برای کارهای دیگری غیر از هدف اصلی آن استفاده کرد. در این آزمایش قصد داریم تا از init برای اجرای یک فراخونی سیستم استفاده کنیم. برای ایجاد initramfs مورد نظرمان، مراحل زیر باید طی شود:

- در ابتدا قطعه کدی مانند زیر را آماده می‌کنیم تا به عنوان برنامه init مورد استفاده قرار بگیرد. دقت کنید که این برنامه باید در یک loop بی‌نهایت باشد تا خاتمه پیدا نکند. زیرا این فرآیند اصلی سیستم عامل است و خاتمه یافتن آن سیستم عامل را از کار می‌اندازد.

```
#include <stdio.h>
#include <unistd.h>

int main(void) {
	while (1){
		printf("Hello world\n");
		fflush(stdout);
		sleep(2);
	}
	return 0;
}
```

این برنامه را با دستور زیر کامپایل می‌کنیم:

```
gcc -static init.c -o init
```

برای کامپایل برنامه init از پرچم static استفاده می‌کنیم، زیرا برنامه init اولین برنامه‌ای است که اجرا می‌شود و نباید هیچ پیش‌نیازی داشته باشد.

برای ساختن فایل initramfs از دستور زیر استفاده می‌کنیم:

```
echo init | cpio -ov --format=newc > initramfs.img
```

در نهایت با استفاده از دستور زیر کرنل و initramfs را در ماشین مجازی اجرا می‌کنیم.

```
qemu-system-x86_64 -kernel linux-5.15.77/arch/x86/boot/bzImage -initrd initramfs.img
```