bhnum Fix broken Linux programming chapter 8 link 2c039ec · 4 months ago

292 lines (204 loc) · 14.7 KB

آزمایش ۲ - آشنایی با فراخوانی های سیستمی

۲.۱ مقدمه

در این جلسه از آزمایشگاه با برخی از مهم‌ترین فراخوانی‌های سیستمی در سیستم‌عامل لینوکس آشنا خواهیم شد و به کمک آنها چند برنامه خواهیم نوشت. همچنین روش اضافه کردن فراخوانی‌های سیستمی به هسته ی لینوکس را خواهیم آموخت.

۲.۱.۱ اهداف

انتظار می‌رود در پایان این جلسه دانشجویان مطالب زیر را فرا گرفته باشند:

- آشنایی با مفهوم فراخوانی سیستمی.
- نحوه ی اجرای فراخوانی‌های سیستمی.
- فراخوانی‌های سیستمی مهم و پرکاربرد در سیستم‌عامل لینوکس.
- نحوه ی ایجاد فراخوانی‌های سیستمی جدید.

۲.۱.۲ پیش‌نیازها

انتظار می‌رود که دانشجویان با موارد زیر از پیش آشنا باشند:

- برنامه‌نویسی به زبان c/c++

۲.۲ فراخوانی سیستمی چیست؟

فراخوانی‌های سیستمی یا system call ها توابعی هستند که در هسته ی سیستم‌عامل پیاده‌سازی شده اند. هنگامی که یک برنامه یک فراخوانی سیستمی انجام می‌دهد، کنترل اجرا از آن برنامه به هسته منتقل می‌شود تا عملیات درخواست شده صورت پذیرد. فراخوانی‌های سیستمی برای کارهای مختلفی مانند دسترسی به منابع، تخصیص آنها، خاموش کردن یا راه‌اندازی مجدد سیستم‌عامل و ... مورد استفاده قرار می گیرند. برخی از این فراخوانی های سیستمی تنها در پروسه‌هایی قابل استفاده اند که توسط super-user اجرا شده باشند.

هر فراخوانی سیستمی با یک شماره ی ثابت شناخته می‌شود. این شماره، پیش از کامپایل شدن هسته باید مشخص گردد. به همین دلیل در سیستم‌عامل لینوکس افزودن فراخوانی‌های سیستمی تنها با کامپایل و نصب مجدد هسته امکان پذیر است.

برای اطلاعات بیشتر در مورد فراخوانی‌های سیستمی در لینوکس به اینجا مراجعه کنید.

۲.۳ شرح آزمایش

۲.۳.۱ مشاهده ی فراخوانی‌های سیستمی تعریف شده

1. وارد سیستم‌عامل مجازی ایجاد شده در جلسه ی قبل شوید.

2. سیستم‌عامل لینوکس در حال حاضر شامل بیش از ۳۰۰ فراخوانی سیستمی است. فایل زیر را به کمک یک ویرایشگر باز کنید؛ در این فایل می توانید لیست فراخوانی‌های سیستمی به همراه شماره ی آنها را بیابید.

```
/usr/include/i386-linux-gnu/asm/unistd_32.h
```

۲.۳.۲ اجرای یک فراخوانی سیستمی

1. در پوشه ی خانه ی خود فایلی با نام testsyscall.cpp ایجاد کنید.

2. کد زیر با استفاده از فراخوانی سیستمی mkdir یک پوشه ی جدید ایجاد می کند. آن را در فایلی که در مرحله ی قبل ایجاد کرده‌اید وارد کنید:

برنامه ی نمونه برای ایجاد یک پوشه:

```
#include <stdio.h>
#include <unistd.h>
#include <sys/syscall.h>
int main () {
    long result;
    result = syscall(__NR_mkdir , "testdir", 0777);
```

```
    printf("The result is %ld.\n", result);
    return 0;
}
```

3. کد را کامپایل کنید و سپس اجرا نمایید.
4. نتیجه ی اجرای آن را شرح دهید.
5. در مثال بالا، نقش `__NR_mkdir` چیست؟
6. در مورد نحوه ی استفاده از دستور syscall و ورودی‌ها و خروجی‌های آن توضیح دهید.

۲.۳.۳ اجرای ساده‌تر فراخوانی سیستمی

برای کاربرد ساده‌تر فراخوانی‌های سیستم بدون نیاز به شماره ی آن‌ها، می توان از توابعی استفاده کرد که از پیش به عنوان wrapper برای آن ها نوشته شده‌اند. برای مثال برای فراخوانی سیستمی بخش قبلی می توان از تابع ()mkdir که در `sys/stat.h` قرار دارد استفاده کرد. به دلیل خوانایی بالاتر و سادگی کاربرد، معمولا ترجیح بر استفاده از این توابع به جای استفاده ی مستقیم از دستور syscall است.

1. کد بخش قبل را به کمک تابع ()mkdir بازنویسی کنید و در فایل testsyscall2.cpp ذخیره نمایید.

۲.۳.۴ آشنایی با چند فراخوانی سیستمی پرکاربرد

در هر کدام از فعالیت‌های این بخش، یک فراخوانی سیستمی معرفی می شود؛ به کمک این فراخوانی سیستمی برنامه‌های خواسته‌شده را بنویسید. برای دریافت راهنمایی در مورد هر کدام از این فراخوانی‌های سیستمی می توانید از دستور `man 2 [syscall_name]` استفاده کنید.

- برای دیدن امکان دسترسی به فایل‌ها، فراخوانی سیستمی access مورد استفاده قرار می‌گیرد. برنامه‌ای بنویسید که به عنوان آرگومان ورودی یک آدرس را دریافت کند و ببیند که آیا اولاً آن آدرس وجود دارد یا خیر و ثانیاً آیا دسترسی به آن برای پروسه ی اجرا شده امکان پذیر است؟
- به کمک فراخوانی‌های سیستمی open, write, close برنامه‌ای بنویسید که یک فایل با اسم oslab2.txt ایجاد کرده و نامتان را در آن فایل بنویسد.
- به کمک فراخوانی سیستمی sysinfo برنامه‌ای بنویسید که میزان حافظه RAM کل و همچنین حافظه ی خالی را در خروجی چاپ کند.
- به کمک فراخوانی سیستمی getrusage برنامه‌ای بنویسید که تعداد context swith های خود (داوطلبانه یا غیر داوطلبانه) را چاپ کند.

۲.۳.۵ اضافه کردن یک فراخوانی سیستمی به سیستم‌عامل

همان طور که در ابتدا بیان شد، برای اضافه کردن فراخوانی های سیستمی به هسته ی لینوکس نیازمند آن هستیم که هسته را مجدداً کامپایل و نصب کنیم. برای اضافه کردن یک فراخوانی سیستمی سه گام اصلی باید انجام شود:

1. اضافه کردن تابع جدید،
2. به‌روزرسانی فایل‌های سرآیند،
3. به‌روزرسانی جدول فراخوانی‌های سیستمی.

در اینجا قصد داریم که یک فراخوانی سیستمی ساده را به سیستم‌عامل اضافه کنیم. مراحل دقیق این کار بسته به این که چه نسخه ای از هسته را انتخاب می‌کنید و قصد دارید تا آن را برای اجرا روی چه معماری‌های کامپیوتری ای کامپایل کنید، تفاوت می‌کند. در این جا دستورالعملی را معرفی می‌کنیم که تا حد خوبی حالات مختلف را پوشش می‌دهد. اگر با دنبال کردن آن نتوانستید هسته را با موفقیت کامپایل کنید، بایستی منابع مناسب خود را در اینترنت پیدا کنید.

1. مطمئن شوید که با دسترسی root به سیستم‌عامل وارد شده اید.
2. وارد پوشه ی کد منبع هسته ی سیستم‌عامل که در جلسه ی قبل ایجاد کردیم شوید.
3. دستور make oldconfig را اجرا کنید. این دستور هسته ی جدید را مطابق با ویژگی های هسته ی فعلی که بر روی سیستم نصب شده است تنظیم می کند.
4. با دستور make هسته را کامپایل کنید و مطمئن شوید که این عملیات به درستی صورت می‌گیرد.
5. به کمک دستور make modules_install هسته ی جدید را نصب نمایید.
6. سیستم‌عامل را مجدداً راه‌اندازی کنید. دقت کنید که در منوی بوت، هسته ی جدید را انتخاب کنید.
7. یک پوشه ی خالی با نام hello در شاخه ی اصلی کد منبع هسته ایجاد کنید.
8. در این پوشه یک فایل hello.c شامل کد فراخوانی سیستمی با محتوای زیر ایجاد کنید:

```c
#include <linux/kernel.h>
asmlinkage long sys_hello(void) {
    printk("Hello World\n");
    return 0;
}
```

9. یک فایل با نام Makefile در همین شاخه با محتوای زیر ایجاد کنید:

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

10. فایل Makefile موجود در ریشه ی کد منبع را باز کنید. در حوالی خط ۸۰۰ این فایل متنی مشابه زیر وجود دارد که به انتهای آن /hello را اضافه نمایید.

```
Core-y += kernrl/ mm/ fs/ ipc/ security/ crypto/ block/
```

11. حال جداول مربوط به فراخوانی سیستمی را به روزرسانی می کنیم. فایل زیر را باز کرده و خط بلوک بعدی را در انتهای آن اضافه نمایید. آدرس فایل: (به جای X، عدد 32 یا 64 قرار می‌گیرد بسته به آنکه سیستم‌عامل شما 32 بیتی است یا 64 بیتی.)

```
./arch/x86/syscalls/syscall\_X.tbl
```

توجه کنید که بسته به نسخه ی هسته ی شما، ممکن است آدرس مورد نظر به شکل زیر باشد.
آدرس فایل:

```
./arch/x86/entry/syscalls/syscall\_X.tbl
```

خطی که باید اضافه کنید: (توجه کنید که عدد 357 لزوماً ثابت نیست و این عدد برابر با شماره ی آخرین خط از فایل شما است. صرفاً کافی است تا به آخرین عددی که وجود دارد یکی اضافه کنید و سپس آن را به جای عدد 357 قرار دهید.)

```
357 i386 hello sys_hello
```

12. در فایل زیر خطی به صورت بلوک بعدی اضافه کنید:
آدرس فایل:

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

خطی که باید اضافه کنید:

```
asmlinkage long sys_hello(void);
```

13. هسته را مجدداً کامپایل و نصب کنید و سیستم را دوباره راه‌اندازی نمایید.

کامپایل ورژن ۵ کرنل لینوکس

در صورتی که از ورژن‌های جدیدتر هسته ی لینوکس استفاده می‌کنید کمی پروسه ی نوشتن فراخوانی سیستمی متفاوت می‌تواند باشد.

در ابتدا محتوای فایل hello.c را به صورت زیر عوض کنید:

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

SYSCALL_DEFINE0(hello)
{
    printk("Hello World\n");
    return 0;
}
```

همچنین به آدرس زیر رفته:

```
./arch/x86/include/generated/uapi/asm
```

و فایل unistd_32.h را برای ویرایش باز کنید (اگر کرنل را برای اجرا در حالت ۶۴ بیتی کامپایل می‌کنید، ممکن است نیاز باشد فایل unistd_64.h تغییر داده شود). در دو سه خط انتهایی، مقدار __NR_syscalls را در نظر گرفته و فرض کنید برابر x باشد. در این صورت، پیش از خط

```
#ifdef __KERNEL__
```

، خط زیر را اضافه کنید:

```
#define __NR_hello x
```

همچنین در خط

```
#define __NR_syscalls x
```

مقدار x را یکی زیاد کنید:

```
#define __NR_syscalls x+1
```

حال می‌توانید کرنل لینوکس را کامپایل کنید و از syscall جدید خود استفاده کنید!

حالا دو برنامه با کارکرد زیر بنویسید:

- برنامه‌ای بنویسید که از فراخوانی سیستمی hello استفاده کند. برای مشاهده ی خروجی چاپ شده آن از دستور dmesg استفاده کنید.
- یک فراخوانی سیستمی با نام adder بنویسید که دو عدد را با یکدیگر جمع کند.

لینک‌هایی برای مطالعه ی بیشتر

در این لینک توضیحات خوبی در مورد روند کار ارائه شده. همچنین به syntax مورد نیاز برای فراخوانی های سیستمی دارای آرگومان نیز پرداخته شده. به عنوان مرجعی دیگر برای راهنمایی گام به گام از این لینک می‌توانید استفاده کنید.

پیشنهاد می‌شود که به این لینک زیر برای توضیحاتی فنی، به‌روز و قابل اعتماد در خصوص اضافه کردن system call مراجعه کنید.

برای نسخه‌های قدیمی هسته نیز این لینک می‌تواند مرجع مناسبی باشد.