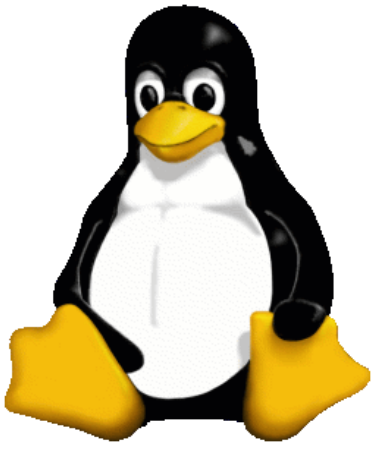
تهیه کرده

دانشگاه صنعتی شریف
دانشکده مهندسی کامپیوتر

دستورکار آزمایشگاه سیستم‌های عامل

جلسه دوم: آشنایی با فراخوانی‌های سیستمی



پاییز ۱۳۹۲

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 لینوکس را خواهیم آموخت.

اهداف

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

- آشنایی با مفهوم فراخوانی سیستمی.
- نحوه‌ی اجرای فراخوانی‌های سیستمی.
- فراخوانی‌های سیستمی مهم و پرکاربرد در سیستم عامل لینوکس.
- نحوه ایجاد فراخوانی‌های سیستمی جدید.

پیش‌نیازها

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

- برنامه‌نویسی به زبان C/C++

فراخوانی سیستمی چیست؟

فراخوانی سیستمی یا System Call تابعی است که در هسته‌ی سیستم‌عامل پیاده‌سازی شده است و 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 اعمال مختلفی مانند دسترسی به منابع، تخصیص آن‌ها، خاموش کردن یا راه‌اندازی مجدد سیستم‌عامل و ... مورد استفاده قرار می‌گیرند. برخی از این فراخوانی‌های سیستمی تنها در پروسه‌هایی قابل استفاده هستند که توسط super-user اجرا شده باشند.

هر فراخوانی سیستمی با یک شماره ثابت شناخته می‌شود که 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 امکان‌پذیر است.

برای اطلاعات بیشتر در مورد فراخوانی‌های سیستمی در لینوکس به [۱] مراجعه کنید.

شرح آزمایش

الف) مشاهده فراخوانی‌های سیستمی تعریف شده

۱. وارد سیستم‌عامل مجازی نصب شده در جلسه قبل شوید.
۲. سیستم‌عامل لینوکس در حال حاضر شامل بیش از ۳۰۰ فراخوانی سیستمی است. فایل زیر را به کمک یک ویرایشگر باز کنید؛ در این فایل می‌توانید لیست فراخوانی‌های سیستمی به همراه شماره‌ی آنها را بیابید:

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

ب) اجرای یک فراخوانی سیستمی

۱. در پوشه‌ی خانه‌ی خود یک فایل testsyscall.cpp ایجاد کنید.
۲. کد زیر با استفاده از فراخوانی سیستمی mkdir یک پوشه جدید ایجاد می‌کند. آن را در فایلی که در مرحله قبل ایجاد کرده‌اید وارد کنید:

```
#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;
}
```

۳. کد را کامپایل کنید و سپس اجرا نمایید.
۴. نتیجه‌ی اجرای آن را شرح دهید.

✓ فعالیت‌ها

- در مثال بالا، نقش `__NR_mkdir` چیست؟
- در مورد نحوه استفاده از دستور `syscall` و ورودی‌ها و خروجی‌های آن توضیح دهید.

پ) اجرای ساده‌تر فراخوانی‌های سیستمی

برای کاربرد ساده‌تر فراخوانی‌های سیستمی بدون نیاز به شماره آن‌ها، می‌توان از توابعی استفاده کرد که از پیش به عنوان Wrapper برای آن‌ها نوشته شده‌اند. برای مثال برای فراخوانی سیستمی بخش قبلی می‌توان از

تابع `mkdir()` که در `<sys/stat.h>` قرار دارد استفاده کرد. به دلیل خوانایی بالاتر سادگی کاربرد، معمولاً ترجیح بر استفاده از این توابع به جای استفاده مستقیم از دستور `syscall` است.

✓ فعالیت‌ها

- کد بخش قبلی را به کمک تابع `mkdir()` بازنویسی کنید و در فایل testsyscall2.cpp ذخیره نمایید.

ت) آشنایی با چند فراخوانی سیستمی پرکاربرد

در هرکدام از فعالیت‌های این بخش، یک فراخوانی سیستمی معرفی می‌شود؛ به کمک این فراخوانی سیستمی برنامه‌های خواسته شده را بنویسید. برای دریافت راهنمایی در مورد هرکدام از این فراخوانی‌های سیستمی می‌توانید از دستور `man 2 [syscall_name]` استفاده کنید.

✓ فعالیت‌ها

- 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 پروسه‌ی اجرا شده امکان‌پذیر است؟
- به کمک فراخوانی‌های سیستمی open، close و write 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 حافظه‌ی RAM کل و همچنین حافظه‌ی خالی را در خروجی چاپ کند.
- به کمک فراخوانی سیستمی getrusage برنامه‌ای بنویسید که میزان حافظه مصرفی خود را چاپ کند.

ث) اضافه کردن یک فراخوانی سیستمی به سیستم‌عامل

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

۱. اضافه کردن تابع جدید،
۲. به روزرسانی فایل‌های سرآیند،
۳. به روزرسانی جدول فراخوانی‌های سیستمی.

در اینجا قصد داریم که یک فراخوانی سیستمی ساده را به سیستم‌عامل اضافه کنیم.

۰. مطمئن شوید که با دسترسی root به سیستم‌عامل وارد شده‌اید.
۱. وارد پوشه‌ی کد منبع هسته سیستم‌عامل ـ که در جلسه قبل ایجاد کردیم ـ شوید.
۲. دستور `make oldconfig` را اجرا کنید. این دستور هسته‌ی جدید را مطابق با ویژگی‌های هسته‌ی فعلی که بر روی سیستم نصب شده است تنظیم می‌کند.
۳. با دستور `make` هسته را کامپایل کنید و مطمئن شوید که این عملیات به درستی صورت می‌گیرد.
۴. به کمک دستور `make modules_install install` هسته‌ی جدید را نصب کنید.
۵. سیستم‌عامل را مجدداً راه‌اندازی کنید. دقت کنید که در منوی بوت، هسته جدید را انتخاب کنید.

حال نوبت به نوشتن فراخوانی سیستمی می‌رسد.

۶. یک پوشه خالی با نام hello در شاخه‌ی اصلی کد منبع هسته ایجاد کنید.
۷. در این پوشه یک فایل hello.c شامل کد فراخوانی سیستمی با محتوای زیر ایجاد کنید:

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

asmlinkage long sys_hello(void) {
    printk("Hello World\n");
    return 0;
}
```

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

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

۹. فایل Makefile موجود در ریشه‌ی کد منبع را باز کنید. در حوالی خط ۷۰۰ این فایل خط زیر را پیدا کنید و به انتهای آن /hello را اضافه کنید:
۱۰. حال باید جداول مربوط به فراخوانی‌های سیستمی را به روزرسانی کنیم.
فایل ./arch/x86/kernel/syscall_32.tbl را باز کنید و خط زیر را در انتهای آن اضافه کنید:

```
.long sys_hello
```

۱۱. سپس فایل ./arch/x86/include/asm/unistd_32.h را باز کرده و خط زیر را در آن بیابید:

```
#define NR_syscalls 346
```

این خط تعداد فراخوانی‌های سیستمی تعریف شده را مشخص می‌کند. مقدار این عدد را یک واحد افزایش دهید.

۱۲. در همین فایل، یک خط به صورت زیر اضافه کنید:

```
#define __NR_hello 350
```

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

✓ فعالیت

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

منابع

[1] http://www.advancedlinuxprogramming.com/alp-folder/alp-ch08-linux-system-calls.pdf

[2] http://seshagiriprabhu.wordpress.com/2012/11/15/adding-a-simple-system-call-to-the-linux-3-2-0-kernel-from-scratch/