به نام خدا

دانشگاه شهید بهشتی

رشته‌ی مهندسی کامپیوتر

درس آزمایشگاه سیستم عامل
استاد : دکتر آهوز

پروژه‌ی امتیازی :

افزودن system call جدید به هسته‌ی لینوکس

اعضای گروه :

متین ارجمند **امیرحسین جعفرزاده**
محدثه صفاری **علی نصرالله پور**

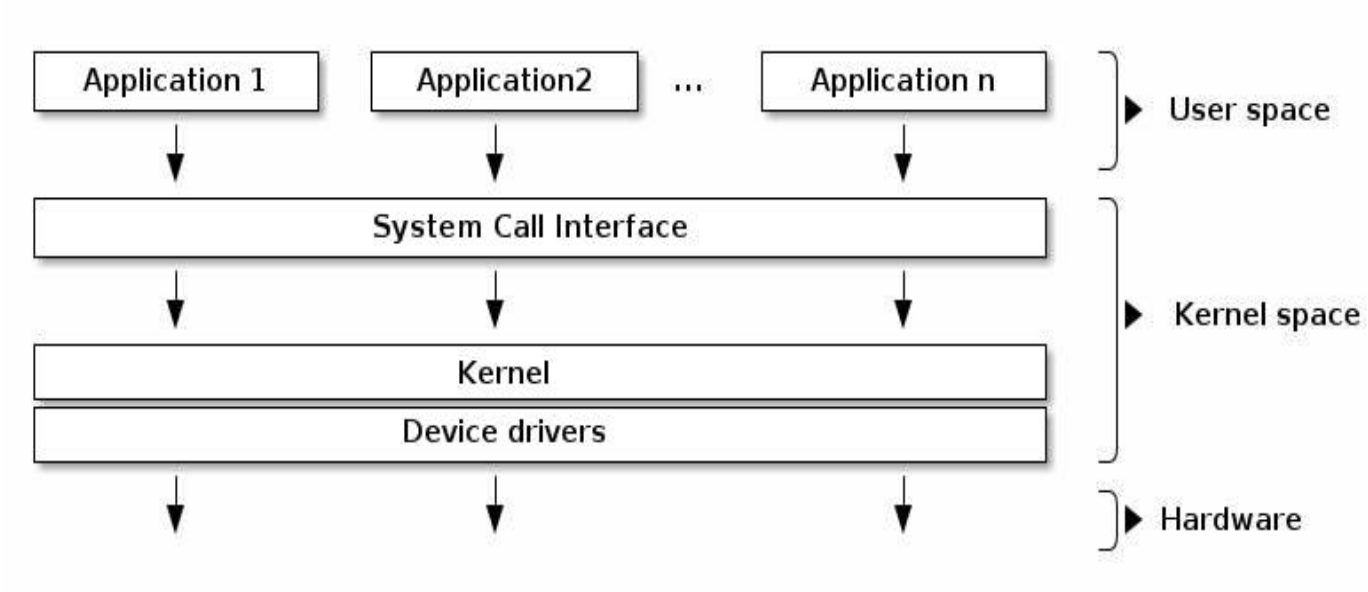
فهرست :

مقدمه :

سیستم عامل لینوکس از دو فضای اصلی تشکیل شده است : User Space فضای کاربر و Kernel Space فضای هسته.
برنامه‌های عادی که کاربر اجرا می‌کند در فضای کاربر قرار دارند و به دلایل امنیتی و حفاظتی، اجازه دسترسی مستقیم به منابع سخت‌افزاری یا حافظه مدیریت‌شده توسط سیستم‌عامل را ندارند.

توضیحات‌تئوری :

System Call یا فراخوان سیستمی، رابطی (Interface) میان فضای کاربر و فضای هسته است. زمانی که یک برنامه نیاز به خدماتی از هسته دارد (مانند خواندن فایل، ایجاد پردازش جدید، یا در این پروژه "انجام یک عملیات ریاضی و ثبت در لاگ سیستم")، از طریق System Call درخواستی به کرنل ارسال می‌کند. پردازنده با دریافت این درخواست، حالت خود را به Kernel Mode تغییر داده، تابع مورد نظر را اجرا کرده و نتیجه را به برنامه باز می‌گرداند.



شرح پیاده سازی(نحوه‌ی اضافه کردن به کرنل):

ابتدا سورس کد کرنل را دانلود کردیم. بهترین کار برای پروژه این بود که یک نسخه تمیز از سایت kernel.org بگیریم (زیرا سورس کد خود ابونتو گاهی پچ‌های خاص دارد که کار را پیچیده میکند).

نسخه‌ی 5.4.81 به ابونتو 18.04 نزدیک و بسیار پایدار است.

با این دستورات آن را دانلود و extract کردیم :

```
wget https://cdn.kernel.org/pub/linux/kernel/v5.x/linux-5.4.81.tar.xz
tar -xvf linux-5.4.81.tar.xz
```

سپس مراحل زیر روی سورس کد کرنل نسخه 5.4.81 انجام شد :

الف) ایجاد دایرکتوری و سورس کد

یک پوشه به نام my_syscall در مسیر اصلی سورس کرنل ایجاد کردیم سپس درون این پوشه فایلی به نام add_syscall.c با کد زیر نوشتیم:

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

// system call with 2 args
SYSCALL_DEFINE2(my_add_syscall, int, a, int, b)
{
    int result = a + b;
    // print in kernel log
    printk(KERN_INFO "[System Call] Adding %d + %d = %d\n", a, b, result);
    return result;
}
```

از ماکروی SYSCALL_DEFINE2 استفاده شد زیرا تابع ما دو ورودی دارد. تابع printk وظیفه نوشتن در لاگ سیستم (dmesg) را بر عهده دارد.

ب) ایجاد Makefile اختصاصی

در همان پوشه my_syscall، فایلی به نام Makefile ایجاد شد تا به کامپایلر بگوید که فایل آبجکت را بسازد:

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

ج) ویرایش Makefile اصلی

برای اینکه کرنل هنگام کامپایل، پوشه جدید ما را هم در نظر بگیرد، فایل Makefile اصلی (در ریشه سورس) را ویرایش کردیم و نام پوشه/hello را به انتهای خط core-y اضافه گردید:

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

د) ثبت در جدول فراخوان‌های سیستمی(System Call Table)

فایل مربوط به جدول سیستم‌کال‌های معماری ۶۴ بیتی
(arch/x86/entry/syscalls/syscall_64.tbl) را ویرایش کردیم و خط زیر را در انتهای بخش common اضافه کردیم:

```
436 common  my_add_syscall   __x64_sys_my_add_syscall
```

عدد 436 به عنوان شناسه منحصربه فرد (ID) این سیستم‌کال تعیین شد.

ه) معرفی در فایل هدر

برای اینکه تابع در سطح کرنل شناخته شود، پروتوتایپ آن به فایل include/linux/syscalls.h اضافه شد :

```
asmlinkage long sys_my_add_syscall(int a, int b);
```

کامپایل و نصب کرنل:

پس از اعمال تغییرات، مراحل زیر طی شد:

1. پیکربندی : اجرای make menuconfig
   (جهت بهینه‌سازی فضا، گزینه‌های Debug Info و گواهی‌های امنیتی غیرضروری غیرفعال شدند).
2. کامپایل : اجرای دستور make
3. نصب : اجرای sudo make modules_install و sudo make install
4. راه‌اندازی : بروزرسانی گراب و ریستارت سیستم با کرنل جدید.

تست و نتیجه‌ی اجرا:

برای بررسی صحت عملکرد، یک برنامه به زبان C در فضای کاربر (User Space) نوشته شد که سیستم‌کال شماره 436 را فراخوانی می‌کند.

کد تست برنامه (test_syscall.c) :

```c
#include <stdio.h>
#include <linux/kernel.h>
#include <sys/syscall.h>
#include <unistd.h>

#define __NR_my_add_syscall 436

int main()
{
    int a = 10;
    int b = 20;

    scanf("%d %d", &a, &b);

    printf("Calling kernel function to add %d and %d...\n", a, b);

    // system call
    long res = syscall(__NR_my_add_syscall, a, b);

    printf("System Call returned: %ld\n", res);

    return 0;
}
```

خروجی برنامه :

پس از کامپایل و اجرای برنامه تست، خروجی زیر در ترمینال نمایش داده شد که نشان‌دهنده دریافت صحیح نتیجه جمع (۳۰) از کرنل است:

```
matin@ubuntu:~/Desktop$ gcc test_syscall.c -o test
matin@ubuntu:~/Desktop$ ./test
10 20
Calling kernel function to add 10 and 20...
System Call returned: 30
```

بررسی لاگ کرنل (dmesg) :

با اجرای دستور dmesg، پیغام ثبت شده توسط تابع printk در کرنل مشاهده شد که انجام موفقیت‌آمیز عملیات در سطح کرنل را تایید می‌کند:

```
matin@ubuntu:~/Desktop$ dmesg | tail -n 1
[  648.129220] [System Call] Adding 10 + 20 = 30
```

نتیجه گیری :

در این پروژه، با موفقیت یک System Call سفارشی به کرنل لینوکس اضافه شد. این تابع توانست داده‌ها را از فضای کاربر دریافت کند، عملیات ریاضی را در فضای هسته انجام دهد، نتیجه را در لاگ سیستم ثبت کند و خروجی را به درستی به برنامه کاربر بازگرداند.