دانشگاه صنعتی شریف
دانشکده‌ی مهندسی کامپیوتر

درس آز سیستم‌عامل

استاد حمید بیگی

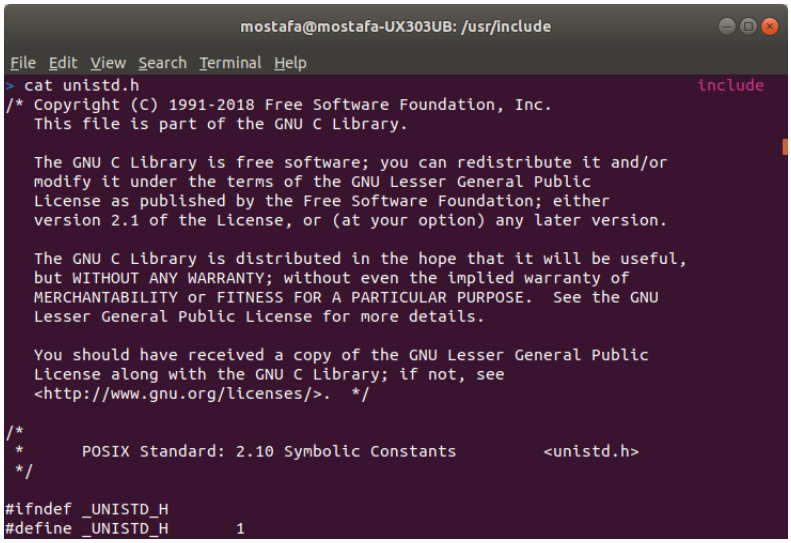
گزارش آزمایش دوم

علی‌رضا دیزجی، امیرحسن فتحی، مصطفی قدیمی

سؤال ۱. مشاهده فراخوانی‌های سیستمی تعریف شده

در این قسمت با توجه به این‌که ساختار سخت‌افزار ما با سوال فرق دارد، فایل asm را به کمک دستور find پیدا کردیم:

```
> cd /usr/include/
> find . -name asm
./x86_64-linux-gnu/asm
> cd ./x86_64-linux-gnu/asm/
> cat unistd.h
```



سؤال ۲. اجرای یک فراخوانی سیستمی

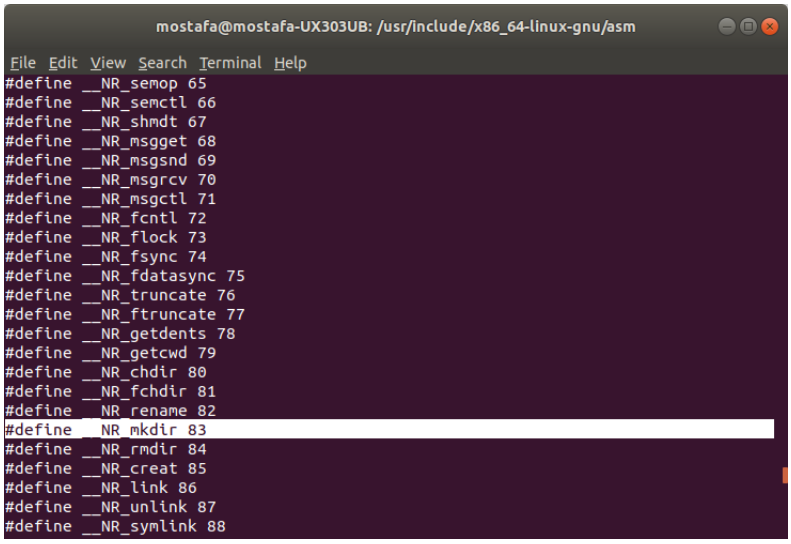
- دستورات زیر را درون terminal اجرا می‌کنیم.

```
> mkdir 2 && cd 2
> sudo nano testsyscall.cpp
```

- کد داده شده در سوال را در این قسمت کپی کرده و سپس با استفاده از دستور ctrl + x آن را ذخیره می‌کنیم.
- به کمک دستورات زیر کد را کامپایل کرده و اجرا می‌کنیم.

```
> sudo gcc -o testsyscall testsyscall.cpp
> ./testsyscall
```

- نتیجه‌ی اجرای آن ساخته شدن یک پوشه به نام testdir در مسیر [1] فعلی است و در آخر پیام The result is 0. را چاپ می‌کند.
- همان‌طور که در توضیحات فراخوانی سیستمی آمده است، هر فراخوانی سیستم با یک شماره‌ی ثابت شناخته می‌شود. نقش `__NR_mkdir` (که به‌طور سراسری [2] تعریف شده است) در این‌جا این است تا عدد مربوط به این فراخوانی جایگزین آن شود.



- تابع `syscall()` یک تابع کوچک است که رابط زبان assembly آن دارای شماره و آرگومان‌های مشخص است. تابع پر کاربردی است اگر از آن بدون wrapperها استفاده کنیم، رجیسترهای پردازنده قبل از فراخوانی سیستم ذخیره می‌کند و بعدا آن‌ها را بازگردانی می‌کند. خروجی آن در صورتی که با موفقیت انجام گیرد، «۰» و در غیر این صورت «۱ـ» خواهد بود.

سؤال ۳. اجرای ساده‌تر فراخوانی‌های سیستمی

```
#include <stdio.h>
#include <unistd.h>
#include <sys/stat.h>
int main() {
    long result;
    result = mkdir("testdir", 0777);
    printf("The result is %ld.\n", result);
    return 0;
}
```

سؤال ۴. آشنایی با چند فراخوان سیستمی پرکاربرد

- فراخوان سیستمی access

---

[1] directory
[2] global

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

int main(int argc, char *argv[])
{
    int result;
    char *path = argv[1];
    result = access(path, F_OK);
    if (result == 0)
    {
        printf("%s exists\n", path);
    }
    else
    {
        printf("%s doesn't exist.", path);
    }
    return 0;
}
```

- فراخوان‌های سیستمی open, close, write

```
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <unistd.h>
#include <string.h>

int main() {
    int open_result;
    int write_result;
    open_result = open("oslab2.txt", O_CREAT | O_WRONLY, 0777);
    write_result = write(open_result, "Mostafa Ghadimi\n", strlen("Mostafa Ghadimi\n"));
    close(open_result);
    return 0;
}
```

- فراخوان سیستمی sysinfo

```
#include <linux/kernel.h>
#include <stdio.h>
#include <sys/sysinfo.h>

int main()
{
    const double megabyte = 1024 * 1024;
    struct sysinfo si;
    sysinfo(&si);
    printf("total RAM: 1f.%5 MB\n", si.totalram / megabyte);
    printf("free RAM: 1f.%5 MB\n", si.freeram / megabyte);
    return 0;
}
```

- فراخوان سیستمی getrusage

```
#include <sys/time.h>
#include <sys/resource.h>
#include <stdio.h>

int main() {
    struct rusage ru;
    getrusage(RUSAGE_SELF, &ru);
    printf("maximum resident set size: %ld\n", ru.ru_maxrss);
    printf("integral shared memory size: %ld\n", ru.ru_ixrss);
    printf("integral unshared stack size: %ld\n", ru.ru_isrss);
}
```