



## 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
```

```
mostafa@mostafa-UX303UB: /usr/include  
File Edit View Search Terminal Help  
> cat unistd.h  
/* Copyright (C) 1991-2018 Free Software Foundation, Inc.  
   This file is part of the GNU C Library.  
  
   The GNU C Library is free software; you can redistribute it and/or  
   modify it under the terms of the GNU Lesser General Public  
   License as published by the Free Software Foundation; either  
   version 2.1 of the License, or (at your option) any later version.  
  
   The GNU C Library is distributed in the hope that it will be useful,  
   but WITHOUT ANY WARRANTY; without even the implied warranty of  
   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU  
   Lesser General Public License for more details.  
  
   You should have received a copy of the GNU Lesser General Public  
   License along with the GNU C Library; if not, see  
   <http://www.gnu.org/licenses/>. */  
  
/*  
 *      POSIX Standard: 2.10 Symbolic Constants      <unistd.h>  
 */  
#ifndef _UNISTD_H  
#define _UNISTD_H      1
```

## 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<sup>۱</sup> 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<sup>۲</sup> تعریف شده است) در این‌جا این است تا عدد مربوط به این فراخوانی جایگزین آن شود.

```

mostafa@mostafa-UX303UB: /usr/include/x86_64-linux-gnu/asm
File Edit View Search Terminal Help
#define __NR_semop 65
#define __NR_semctl 66
#define __NR_shmctl 67
#define __NR_msgget 68
#define __NR_msgsnd 69
#define __NR_msgrcv 70
#define __NR_msgctl 71
#define __NR_fcntl 72
#define __NR_flock 73
#define __NR_fsync 74
#define __NR_fdatasync 75
#define __NR_truncate 76
#define __NR_ftruncate 77
#define __NR_getdents 78
#define __NR_getcwd 79
#define __NR_chdir 80
#define __NR_fchdir 81
#define __NR_rename 82
#define __NR_mkdir 83
#define __NR_rmdir 84
#define __NR_creat 85
#define __NR_link 86
#define __NR_unlink 87
#define __NR_symlink 88

```

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

directory<sup>۱</sup>  
global<sup>۲</sup>

```

#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;
}

```

• sysinfo فراخوان سیستمی

```

#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;
}

```

```
#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);
}
```