



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^۱ 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^۲ 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 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;
}
```

^۱directory
^۲global