دستور کار جلسه سوم

فراخوانی سیستمی: System Call

دستورات دریافت شده توسط پوسته به هسته ارسال می شوند. این دستورات در قالب فراخوانی های سیستمی به هسته ارسال شده و در آنجا به فراخوانی های هسته تبدیل می شوند. در نهایت این فراخوانی های هسته اند که عملیات مشخص شده را به انجام می رسانند.

در ادامه هشت دسته از مهمترین فراخوانی های سیستم در استاندارد POSIX بررسی می شوند.

این دسته ها عبارتند از فراخوانی های سیستمی برای :

۱. مدیریت فایل ها
۲. مدیریت فایل سیستم و دایرکتوری ها
۳. مدیریت سطح دسترسی و حفاظت از فایل ها
۴. مدیریت زمان
۵. مدیریت فرآیندها
۶. مدیریت سیگنال ها (signal)
۷. مدیریت سوکت ها (socket)
۸. مدیریت حافظه مشترک (shared memory)

فراخوانی های مطرح شده در هر بخش برای حل مسائل بعدی نیز به کار گرفته می شوند.

در این جلسه فراخوانی های سیستمی برای مدیریت فایل ها، مدیریت فایل سیستم و مدیریت سطح دسترسی بررسی می شوند.

تمامی توابع و فراخوانی های مطرح شده در این دستور کار از دو آدرس زیر آورده شده اند، برای مطالعه جزییات به آنها به صفحات راهنما و یا آدرس های آورده شده مراجعه کنید :

http://www.minix3.org/manpages
http://linux.die.net/man

فراخوانی های سیستمی برای مدیریت فایل ها

-به منظور ایجاد و یا بازکردن فایل از پیش ساخته شده از توابع زیر استفاده میشود..در اینجا منظور از flag، این است که این فایل به چه هدفی ایجاد میشود..تنها برای خواندن یا خواندن و نوشتن و ....

منظور از mode در واقع مجوز دسترسی به این فایل است که بصورت عددی محاسبه میشود و برای این فایل منظور میشود.

```
int open(const char *pathname, int flags);
int open(const char *pathname, int flags, mode_t
mode);
```

flags:

O_RDONLY	open for reading only
O_WRONLY	open for writing only
O_RDWR	open for reading and writing
O_NONBLOCK	do not block on open
O_APPEND	append on each write
O_CREAT	create file if it does not exist
O_TRUNC	truncate size to 0
O_EXCL	error if create and file exists

modes:

S_ISUID	04000	set user ID on execution
S_ISGID	02000	set group ID on execution
S_ISVTX	01000	`sticky bit'
S_IRWXU	00700	read, write, execute by owner
S_IRUSR	00400	read by owner
S_IWUSR	00200	write by owner
S_IXUSR	00100	execute (search on directory) by owner
S_IRWXG	00070	read, write, execute by group
S_IRGRP	00040	read by group
S_IWGRP	00020	write by group

S_IXGRP	00010	execute (search on directory) by group
S_IRWXO	00007	read, write, execute by others
S_IROTH	00004	read by others
S_IWOTH	00002	write by others
S_IXOTH	00001	execute (search on directory) by others

از تابع زیر برای بستن فایل در انتهای کار استفاده میشود.

```
int close(int fd);
```

close a file descriptor

تابع read برای خواندن به اندازه count از فایل مشخص شده با اشاره گر fd و ریختن آن در buf است.

```
ssize_t read(int fd, void *buf, size_t count);
```

Read data from a file into a buffer

تابع write برای نوشتن به اندازه count که در buf قراردارد داخل فایل مشخص شده با fd

```
ssize_t write(int fd, const void *buf, size_t count);
```

Write data from a buffer into a file

سطح دسترسی amode را برای فایل name بررسی میکند.

```
s = access (name, amode)
```

Check a file's accessibility

فراخوانی های سیستمی برای مدیریت فایل سیستم و دایرکتوری ها

از تابع زیر برای ایجاد دایرکتوری در مسیر path و با سطح دسترسی mode استفاده میشود.

```
int mkdir(const char *pathname, mode_t mode);
```

creates a directory in path

برای حذف دایرکتوری خالی path از تابع زیر استفاده میشود.

```
int rmdir(const char *pathname);
```

Remove an empty directory

C Library <stdio.h>

Printf

تابع printf به منظور چاپ خروجی استفاده میشود..به همین ترتیب تابع fprintf برای چاپ در فایل و تابع sprintf برای چاپ در رشته مورد استفاده قرار میگیرد.

```
int printf(const char *format, ...);
int fprintf(FILE *stream, const char *format, ...);
int sprintf(char *str, const char *format, ...);
int scanf (const char *format, ...);
int fscanf (FILE *stream, const char *format, ...);
int sscanf (const char *str, const char *format,
...);
```

تابع Scanf برای دریافت ورودی استفاده میشود و تابع fscanf برای دریافت ورودی از فایل و تابع sscanf جهت دریافت ورودی از رشته استفاده میشود.

C Library <string.h>

تابع strcmp جهت مقایسه دو رشته استفاده میشود..اگر دو رشته با هم برابر بودند خروجی تابع صفر و در غیر اینصورت برابر یک میباشد.

```
int strcmp(const char *s1, const char *s2);
int strncmp(const char *s1, const char *s2, size_t
n);
```

The strcmp() function compares the two strings s1 and s2. It returns an integer less than, equal to, or greater than zero if s1 is found, respectively, to be less

than, to match, or be greater than s2.
The strncmp() function is similar, except it only compares the first (at most) n bytes of s1 and s2.

Strlen

تابع strlen طول رشته را محاسبه کرده و برمیگرداند.

```
size_t strlen(const char *s);
```

The strlen() function calculates the length of the string *s*, excluding the terminating null byte (‘\0’).

مثال ها

مثال های آورده شده در زیر را نوشته و اجرا کنید :

open:

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <fcntl.h>

int main()
{
char buffer[256];
sprintf(buffer,"%s","this is string in open");
int openFile=open("open.txt",O_CREAT|O_RDWR,00777);
return 0;
}
```

write:

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

int main()
{
char buffer[256];
sprintf(buffer,"%s","this is string in write");
int openFile=open("open.txt",O_CREAT | O_TRUNC
| O_RDWR,00777);
write(openFile,buffer,strlen(buffer));
return 0;
}
```

read:

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

int main()
{
char readBuffer[256];
int readFile=open("open.txt",O_RDONLY,00777);
read(readFile,readBuffer,255);
fprintf(stdout,"%s\n",readBuffer);
return 0;
}
```

mkdir:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/types.h>
#include <sys/stat.h>

int main()
{
char dirName[256];
sprintf(dirName,"%s","/home/dirTest");
mkdir(dirName,00755);
return 0;
}
```