Linux programming
Lab 6 | 19-02-2020

Name: k.prudhvi
Reg:17mis1086

System Calls For File Operations:

1.System calls for file open,read,write

Code:

open():

```
#include<stdio.h>
#include<fcntl.h>
#include<errno.h>
extern int errno;
int main()
{
        int fd = open("file1.txt", O_RDONLY);
        printf("fd = %d\n", fd);
        if (fd ==-1)
        {
                printf("Error Number % d\n", errno);
                perror("Program");
        }
        return 0;
}
```

Output

```
linux@prudhvi:~$ gcc ex1.c
linux@prudhvi:~$ ./a.out
fd=3
```

Read & Write:

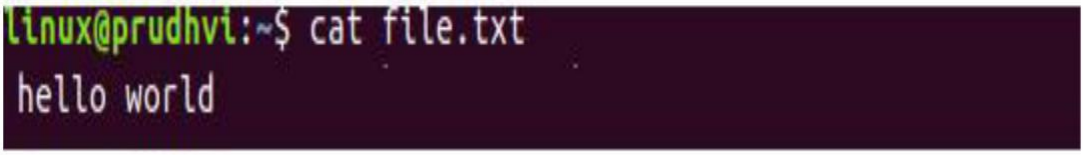
Code:

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

int main()
{
        int fd;
        fd=open("/home/prudhvi/lab6/file1.txt",O_RDWR);
        char word[50];
        read(fd,word,sizeof(word));
        const char *buf="Okay Read from file1 and written to file2";
        ssize_t nr;
        nr=write(fd,buf,strlen(buf));

}
```

Output:



2. Manage EINTR while accessing file using system calls.

Code:

```
#include<stdio.h>
#include<fcntl.h>s
#include<errno.h>
#include<stdlib.h>
#include<string.h>
int main()
{

int fd = open("file1.txt", O_RDONLY );
   int sz;
   sz = write(fd, "I am Inevitable\n", strlen("I am Inevitable"));
   if (sz == -1 && errno != EINTR)
```

```
        {
                perror("Read");
                exit(EXIT_FAILURE);
        }
    return 0;
}
```

Output:

```
linux@prudhvi:~$ gcc ex1.c
linux@prudhvi:~$ ./a.out
ex1.c: In function 'main':
ex1.c:13:10: warning: implicit declaration of function 'write'; did you mean 'fwrite'
? [-Wimplicit-function-declaration]
     sz = write(fd, "I am Inevitable\n", strlen("I am Inevitable"));
          ^~~~~
          fwrite
linux@prudhvi:~$ Read:Bad file Descriptor
```

3. Do Non-Block read and write using system calls.

Code:

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

int main()
{
        int fd,ret;
        fd=open("/home/prudhvi/17mis1086/file1.txt",O_RDWR);
        ssize_t nr;
        char buf[BUFSIZ];
        start:
        nr=read(fd,buf,BUFSIZ);
        while(BUFSIZ!=0 && (ret = read(fd,buf,BUFSIZ))!=0)
        {
                if(nr==-1)
                {
                        if(errno == EINTR)


                        {
                                goto start;
                        }
                        if(errno == EAGAIN)
                        {
                                continue;
                        }
                        else
                        {
                                perror("Read");
```

```
                    break;
                }
            }
        }

}
```

Output:

```
linux@prudhvi:~$ gcc ex2.c
linux@prudhvi:~$ ./a.out
linux@prudhvi:~$ Read:Bad file Descriptor
```

File Permissions:

4. Disable Write permissions to user for all the files in specific folder.

Code:

So let's create a directory called "acldemo" and create two files namely file1.txt,file2.txt in it.

Changing permissions to r-x i.e, disabling write permissions.

```
linux@prudhvi:~$ cd lab6/acldemo
file1.txt     file2.txt
```

```
eg1@AB1614SCSE73:~/Desktop$ setfacl -m u:eg1:r prudhvii.c
eg1@AB1614SCSE73:~/Desktop$ # setfacl -dm "entry"
eg1@AB1614SCSE73:~/Desktop$ setfacl -x g:staff prudhvii.c
eg1@AB1614SCSE73:~/Desktop$ getfacl prudhvii.c
# file: prudhvii.c
# owner: eg1
# group: eg1
user::rw-
user:eg1:r--
group::r--
mask::r--
other::r--

# file: prudhvii.c
# owner: eg1
# group: eg1
user::rw-
user:eg1:r--
group::r--
mask::r--
other::r--
```

5. **Set write permission for only one user on a file**

```
eg1@AB1614SCSE73:~/Desktop$ setfacl -m u:eg1:r prudhvii.c
eg1@AB1614SCSE73:~/Desktop$ # setfacl -dm "entry"
eg1@AB1614SCSE73:~/Desktop$ setfacl -x g:staff prudhvii.c
eg1@AB1614SCSE73:~/Desktop$ getfacl prudhvii.c
# file: prudhvii.c
# owner: eg1
# group: eg1
user::rw-
user:eg1:r--
group::r--
mask::r--
other::r--
```