File System Calls

1. Create system Call

```
#include <sys/types.h>

#include <sys/stat.h>

#include <fcntl.h>

#include <unistd.h>

#include<stdio.h>

#include<errno.h>


void main(){

int fd;


//Step-1 Create a file with some permission

fd=creat("first.txt", 0666);

// fd= integer number other than [stdin=0, stdout=1, stderr=2, are reserved]. fd=-1 in case of error

//Octal 0 [User=R+W=4+2=6]

//Octal 0 [Group=R+W=4+2=6]

//Octal 0 [Other=R+W=4+2=6]

 if(fd==-1){

printf("Error=%d\n",fd);

perror("What happened:");//Whatever happens perror returns that

}

else{

printf("Success Creating File FD=%d\n",fd);

perror("What happened:");//Whatever happens perror returns that

}


int retval=close(fd); //0 in case of success and -1 in case of an error
//You can simply do close(fd);

printf("retval=%d\n", retval);


}//main




/* Output
gcc file1.c -o file1
./file1
Success Creating File FD=3
What happened:: Success
retval=0
*/
```

Example: 1 Advanced

```
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <unistd.h>
#include<stdio.h>
#include<errno.h>
#include<stdlib.h>  //System
void main(){
int fd;

//Step-1 Create a file with some permission
```

```
//fd=creat("first.txt", 0666); //octal

//System Will Reset it.


fd=creat("second.txt",S_IRUSR|S_IWUSR|S_IXUSR|S_IXGRP|S_IXOTH);//Execute

//System Will Reset it again


// fd= integer number other than [stdin=0, stdout=1, stderr=2, are reserved]. fd=-1 in case of error


/*

S_IRUSR, S_IWUSR, S_IXUSR  - Owner: read, write, execute.

S_IRGRP, S_IWGRP, S_IXGRP - Group: read, write, execute.

S_IROTH, S_IWOTH, S_IXOTH- Others: read, write, execute.

*/


if(fd==-1){

printf("Error=%d\n",fd);

perror("What happened:");//Whatever happens perror returns that

}


else{

printf("Success Creating File FD=%d\n",fd);

perror("What happened:");//Whatever happens perror returns that

system("ls -l second.txt"); //To execute linux commands here

system("chmod 777 second.txt"); //To execute linux commands here

system("ls -l second.txt"); //To execute linux commands here

}


int retval=close(fd); //0 in case of success and -1 in case of an error

//You can simply do close(fd);
```

```
printf("retval=%d\n", retval);



}//main

/* Output
root@ubuntu:~/Desktop/shell_scripts# gcc file11.c -o file11
root@ubuntu:~/Desktop/shell_scripts# ./file11
Success Creating File FD=3
What happened:: Success
---x--x--x 1 root root 0 Aug 28 13:22 second.txt
-rwxrwxrwx 1 root root 0 Aug 28 13:22 second.txt
retval=0
*/
```

Example-2   Open System Call to open a file

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

/////System Calls For Open/////
void main(){
int fdr, fdw, fdrw;

//Step-2 OPen a File
fdr=open("first.txt", O_RDONLY);
// fd= integer number other than [stdin=0, stdout=1, stderr=2, are reserved]. fd=-1 in case of error

///Read only Mode
if(fdr>0){
printf("The file is open for read only\n");
}
close(fdr); //close returns 0 in case of success and -1 in case of an error

///Write Only Mode
fdw=open("first.txt", O_WRONLY);
if(fdw>0){
printf("The file is open for write only\n");
}
close(fdw); //close returns 0 in case of success and -1 in case of an error

/////Read Write////
fdrw=open("first.txt", O_RDWR);
// fd= integer number other than [stdin=0, stdout=1, stderr=2, are reserved]. fd=-1 in case of error

if(fdr>0){
printf("The file is open for read only\n");
}

close(fdw); //close returns 0 in case of success and -1 in case of an error
perror("What Happened:");
}//main
```

```
/* Output

 gcc file2.c -o file2

 ./file2

The file is open for read only

The file is open for write only

The file is open for read only

What Happened:: Success

*/
```

Example: 3 Read System Call

```
#include <sys/types.h>

#include <sys/stat.h>

#include <fcntl.h>

#include <unistd.h>

#include<stdio.h>

#include<errno.h>

#define MAXBYTES 20


void main(){

int fd;

char buffer_1[MAXBYTES];

int num_bytes;


//Step-1. Open file system call


fd=open("first.txt",O_RDONLY);


if(fd>0){
```

```
//No error while opening file descriptor
//Step-2 Read  file
num_bytes=read(fd,buffer_1,MAXBYTES);
if(num_bytes==0){
printf("Empty File : %d\n", num_bytes);
}//if
else{
printf("File has: %d bytes \n", num_bytes);
}//else
}//if
close(fd);//Must close fd at the end
}//main

/* Output
root@ubuntu:~/Desktop/shell_scripts# gcc file3.c -o file3
root@ubuntu:~/Desktop/shell_scripts# ./file3
Empty File : 0
*/
```

Example: 4 Write System Call

```
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <unistd.h>
#include<stdio.h>
#include<errno.h>
#define MAXBYTES 20
```

```
void main(){

int fd;
char buffer_1[MAXBYTES];
int num_bytes;

//Fill up the array with characters
//These will be written in the file
//Step-1. Open file system call
fd=open("first.txt",O_WRONLY);
if(fd>0){
//No error while opening file descriptor
num_bytes= write(fd,buffer_1,MAXBYTES);
// write function returns the number of bytes written and the value -1 in case of an error.
if(num_bytes>0){
printf("Success\n");
perror("What Happened");
}//if

else{
printf("Error\n");
perror("What Happened");
}//else
}//if
close(fd);//Must close fd at the end
}//main


/* Output
```

```
root@ubuntu:~/Desktop/shell_scripts# gcc file4.c -o file4

root@ubuntu:~/Desktop/shell_scripts# ./file4

Success

What Happened: Success
```

/// Now if you again execute or run the read system call program you will see 20 bytes have been written

```
root@ubuntu:~/Desktop/shell_scripts# ./file3

File has: 20 bytes

root@ubuntu:~/Desktop/shell_scripts#
```

*/

//If you open the file just by gedit you may see some garbage values as well

//To remove this I have used the system function to change the permission first in next code

//Also how to use screen - use FD=1 and FD=2 is shown

Example4 Advanced:

```
#include <sys/types.h>

#include <sys/stat.h>

#include <fcntl.h>

#include <unistd.h>

#include<stdio.h>

#include<errno.h>

#include<stdlib.h>


#define MAXBYTES 20
```

```
void main(){


int fd;

char buffer_1[MAXBYTES]; //1 byte

int num_bytes;


//Fill up the array with characters

//These will be written in the file

int i=0;

for(i=0;i<MAXBYTES;i++){

buffer_1[i]=65+i; //Writing A, B ........ A starts with ASCII 65

//You can write anything

}



//Step-1. Open file system call

//Slight Modification

system("chmod 777 first.txt");

fd=open("first.txt",O_WRONLY);


if(fd>0){

//No error while opening file descriptor


num_bytes= write(fd,buffer_1,sizeof(MAXBYTES));


//FD=1 or 2

write(1,buffer_1,sizeof(MAXBYTES));///Writing to monitor

write(2,buffer_1,sizeof(MAXBYTES));///Writing to stderr or monitor
// write function returns the number of bytes written and the value -1 in case of an error.

if(num_bytes>0){
printf("Success\n");
perror("What Happened");
}//if

else{
printf("Error\n");
perror("What Happened");
}//else
}//if
close(fd);//Must close fd at the end

}//main

/* Output
root@ubuntu:~/Desktop/shell_scripts# gcc file4.c -o file4
root@ubuntu:~/Desktop/shell_scripts# ./file4
ABCDABCDSuccess
What Happened: Success

/// Now if you again execute or run the read system call program you will see 20 bytes have been
written
root@ubuntu:~/Desktop/shell_scripts# ./file3
```

```
File has: 20 bytes
root@ubuntu:~/Desktop/shell_scripts#
//If you open the file by gedit you shall find no garbage values
*/
```

Example: Dup System Call

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

#define MAXBYTES 20

void main(){

int fd, fd_dup;
char buffer_read[MAXBYTES]; //1 byte
int num_bytes;


//Step-1. Open file system call
//Slight Modification
system("chmod 777 first.txt");
fd=open("first.txt",O_RDONLY);
if(fd>0){
//No error while opening file descriptor
//Duplicate file descriptor -Sharing

fd_dup=dup(fd);

num_bytes=read(fd_dup,buffer_read,MAXBYTES);

if(num_bytes>0){
printf("%d bytes read from buffer is %s\n",num_bytes,buffer_read);
perror("What Happened");
}//if

else{
printf("Error\n");
perror("What Happened");
}//else
}//if
close(fd);//Must close fd at the end
close(fd_dup);//Must close fd at the end
}//main
/* Output
root@ubuntu:~/Desktop/shell_scripts# gcc file14.c -o file14
root@ubuntu:~/Desktop/shell_scripts# ./file14
4 bytes read from buffer is ABCD
What Happened: Success
```

```
*/
```