```
// C program to illustrate open system call
#include<stdio.h>
#include<fcntl.h>
int main()
{
    int fd = open("sample.c", O_RDONLY);
     printf("\nfd = %d\n", fd);
      if (fd ==-1)
    printf("\n fail to open\n");
     return 0;
}
```

output1(if file exist):

fd=3

output2 (if file doesnot exist):

fd=-1

fail to open

```
// C program to illustrate close system Call
#include<stdio.h>
#include <fcntl.h>
#include<stdlib.h>
int main()
{
   int fd = open("open.c", O_RDONLY);
   if (fd < 0)
   {
      printf("\n no such file\n");
      //exit(1);
      printf("opened the fd = % d\n", fd);
   }
   // Using close system Call
   if (close(fd) < 0)
    {
       printf("error in closing the file\n");
       exit(1);
   }
   printf("closed the file fd=%d\n",fd);
}
```

output1:

closed the file fd = 3

output2:

no such file

opened the fd=-1

error in closing the file

```
// C program to illustrate close system Call
#include<stdio.h>
#include<fcntl.h>
int main()
{
    // assume that kits.txt is already created
    int fd1 = open("kits.txt", O_RDONLY);
    printf("fd1 = % d\n", fd1);
    close(fd1);
       // assume that cse.txt is already created
    int fd2 = open("cse.txt", O_RDONLY);
      printf("fd2 = % d\n", fd2);
    exit(0);
}
```

output1:

fd1 = 3

fd2 = 3

```
// C program to illustrate read system Call
#include<stdio.h>
#include <fcntl.h>
#include<stdlib.h>
int main()
{
int fd, sz;
char c[100];
 fd = open("cse.txt", O_RDONLY);
if (fd < 0)
{
printf("error in opening file");
exit(1);
}
 sz = read(fd, c, 10);
printf("called read(% d, c, 10). returned that  %d bytes were read.\n", fd, sz);
c[sz] = '\0';
printf("Those bytes are as follows: %s\n", c);
}
```

```
// C program to illustrate read system Call
#include <unistd.h>
#include <fcntl.h>
#include <sys/types.h>
#include <stdio.h>
int main()
{
 int file=0,sz,i;
char buffer[19];
 if((file=open("testfile.c",O_RDONLY)) <= -1)
  printf("file opening error");
  printf("\n file descriptor = % d\n", file);
  sz=read(file,buffer,19);
for(i=0;i<sz;i++)
printf("%c",buffer[i]);
printf("\n");
return 0;
}
```

output1:

file opening error

file descriptor = -1

output2:

create a file named testfile.c with the contents CSE Department KITS Singapur

file descriptor = 3

CSE Department KITS

```
// C program to illustrate write system Call
#include<stdio.h>
#include <fcntl.h>
#include<string.h>
#include<stdlib.h>
int main()
{
int sz;
int fd = open("foo.txt", O_WRONLY | O_CREAT);
if (fd < 0)
{
   printf("file opening error");
   exit(1);
}
sz = write(fd, "hello CSE A Section\n", strlen("hello CSE A Section\n"));
printf( "%d",sz);
close(fd);
}
```

output1:

20

check the file foo.txt

output2:

file opening error

```
// C program to illustrate lseek system Call
#include<stdio.h>
#include<unistd.h>
#include<fcntl.h>
#include<sys/types.h>
#include<sys/stat.h>
int main()
{
int n,f;
char buff[100];
f=open("sample.txt",O_RDWR);
read(f,buff,10);
write(1,buff,10);
printf("\n");
lseek(f,7,SEEK_CUR);//skips 7 characters from the current position
read(f,buff,8);
write(1,buff,8);
printf("\n");
}
```

create a file named sample.txt with the contents "Computer Science and Engineering"

Output:

```
"Computer
 and Eng
```

```
// C program to illustrate lseek system Call
#include<unistd.h>
#include<fcntl.h>
#include<sys/types.h>
#include<sys/stat.h>
#include<stdio.h>
int main()
{
int n,f,f1;
char buff[100];
f=open("seeking.txt",O_RDWR);
f1=lseek(f,10,SEEK_SET);
printf("Pointer is at %dth position\n",f1);
read(f,buff,15);
write(1,buff,15);
printf("\n");
}
```

Output:

Pointer is at 10th Position

Science and Eng

```
/* C program to illustrate opendir system call  */
#include <stdio.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <stdlib.h>
#include <dirent.h>
int main()
{
    DIR *d;
   d = opendir("/home/kits/CSE2021");
  if (d == NULL)
{
    printf("Couldn't open \n");
    exit(1);
  }
else
printf("\ndirectory opened successfully\n");
 closedir(d);
  return 0;
}
```

output1:

Couldn't open

output2: if /home/kits/CSE2021 directory exists

directory opened successfully

```
// Unix system calls opendir, readdir
#include <stdio.h>
#include <dirent.h>
int main()
{
        struct dirent *de;
        DIR *dr = opendir(".");
        if (dr == NULL)
        {
                printf("Could not open current directory" );
                return 0;
        }
        while ((de = readdir(dr)) != NULL)
            printf("%s\n", de->d_name);
        closedir(dr);
        return 1;
}
```

output:

list the contents of the current directory

```
/* C program to illustrate readdir system call  */
#include <stdio.h>
#include <sys/types.h>
#include <dirent.h>
#include <errno.h>
#include <string.h>
int main(int c, char *v[]) {
   DIR *myDirectory;
   struct dirent *myFile;
   if (c == 2) {
      myDirectory = opendir(v[1]);
      if (myDirectory) {
         puts("OK the directory is opened, let's see its files:");
         while ((myFile = readdir(myDirectory)))
            printf("%s\n", myFile->d_name);
         /*            ** closedir            */
         if (closedir(myDirectory) == 0)
            puts("The directory is now closed.");
         else
            puts("The directory can not be closed.");
      } else if (errno == ENOENT)
         puts("This directory does not exist.");
      else if (errno == ENOTDIR)
         puts("This file is not a directory.");
      else if (errno == EACCES)
         puts("You do not have the right to open this folder.");
      else
```

```
            puts("That's a new error, check the manual.");
    } else
        puts("Sorry we need exactly 2 arguments.");
    return (0);
}
```

Output1:

./a.out .

OK the directory is opened, let's see its files:

list the contents of the current directory

The directory is now closed.

Output2:

create a directory test and a file named example in that test directory

./a.out test

OK the directory is opened, let's see its files:

.

..

example

The directory is now closed.

Output3:

create a subdirectory ctest in that test directory

./a.out test

OK the directory is opened, let's see its files:

.

ctest

..

example

The directory is now closed.

Output4:

./a.out

Sorry we need exactly 2 arguments.

```
//C program to illustrate stat system call
/* C program to find file permission, size, creation and last modification date of a given file. */
#include <stdio.h>
#include <unistd.h>
#include <sys/stat.h>
#include <time.h>
void printFileProperties(struct stat stats);
int main()
{
   char path[100];
   struct stat stats;
   printf("Enter source file path: ");
   scanf("%s", path);
   // stat() returns 0 on successful operation,
   // otherwise returns -1 if unable to get file properties.
   if (stat(path, &stats) == 0)
   {
      printFileProperties(stats);
   }
   else
   {
      printf("Unable to get file properties.\n");
      printf("Please check whether '%s' file exists.\n", path);
   }
   return 0;
}
```

```
/**
 * Function to print file properties.
 */
void printFileProperties(struct stat stats)
{
    struct tm dt;

    // File permissions
    printf("\nFile access: ");
    if (stats.st_mode & R_OK)
        printf("read ");
    if (stats.st_mode & W_OK)
        printf("write ");
    if (stats.st_mode & X_OK)
        printf("execute");

    // File size
    printf("\nFile size: %ld", stats.st_size);

    // Get file creation time in seconds and
    // convert seconds to date and time format
    dt = *(gmtime(&stats.st_ctime));
    printf("\nCreated on: %d-%d-%d %d:%d:%d", dt.tm_mday, dt.tm_mon, dt.tm_year + 1900,
                                    dt.tm_hour, dt.tm_min, dt.tm_sec);
```

```
    // File modification time
    dt = *(gmtime(&stats.st_mtime));
    printf("\nModified on: %d-%d-%d %d:%d:%d", dt.tm_mday, dt.tm_mon, dt.tm_year + 1900,
                                        dt.tm_hour, dt.tm_min, dt.tm_sec);
}
```

output1:

```
Enter source file path: /home/kits/test/example
File access: read
File size: 11
created on: 30-06-2021 11:40:16
modified on: 30-06-2021 11:40:16
```

output2:

```
Unable to get file attributes
Please check whether /home/kits/test/nofile file exists.
```