**Index:**

1. **Program to illustrate stat System call.**
2. **Program to illustrate Directory Management System calls.**
3. **Implementation of cat command(Exercise)**

**Note:** Outputs: Left side

**Stat System Call**

The stat function returns information about the specified file.

**int stat(const char \*file\_name, struct stat \*buf);**

This function retrieves the status of the file pointed to by file\_name and fills in buf.

It returns a stat structure, which contains the following fields:

struct stat {

dev\_t st\_dev; /\* device \*/

ino\_t st\_ino; /\* inode \*/

mode\_t st\_mode; /\* protection \*/

nlink\_t st\_nlink; /\* number of hard links \*/

uid\_t st\_uid; /\* user ID of owner \*/

gid\_t st\_gid; /\* group ID of owner \*/

dev\_t st\_rdev; /\* device type (if inode device) \*/

off\_t st\_size; /\* total size, in bytes \*/

unsigned long st\_blksize; /\* blocksize for filesystem I/O \*/

unsigned long st\_blocks; /\* number of blocks allocated \*/

time\_t st\_atime; /\* time of last access \*/

time\_t st\_mtime; /\* time of last modification \*/

time\_t st\_ctime; /\* time of last change \*/

};

**Experiment # 2: Display Attributes Of A File Using Stat() Systemcall**

**AIM:** To write a program in C to Display attributes of a given file using stat system call**.**

**PROGRAM:**

#include<stdlib.h>

#include<stdio.h>

#include<sys/stat.h>

int main(int argc,char \*argv[])

{

if(argc!=2)

{

printf("wrong no: of arguments");

exit(1);

}

else

{

struct stat str;

stat(argv[1],&str);

printf("\nmode is %o\n",str.st\_mode);

printf("device type is %lu\n",str.st\_dev);

printf("mode no: is %lu\n",str.st\_ino);

printf("no: of inks is %lu\n",str.st\_nlink);

printf("user id is %lu\n",str.st\_uid);

printf("group id is %lu\n",str.st\_gid);

printf("device name is %lu\n",str.st\_rdev);

printf("block size is %lu\n",str.st\_blksize);

printf("no: of blocks is %lu\n",str.st\_blocks);

}

return(0);

}

**Viva Questions:**

**1.What is about stat() system call**

The **stat() system call** returns information about the specified file.

**2. Syntax of stat() system call**

int stat(const char \*file\_name, struct stat \*buf);

##### **opendir,readdir , closedir System Calls**

**DIR \*opendir(const char \****name***);**

The **opendir**() system call opens a directory stream corresponding to the directory *name*, and returns a pointer to the directory stream. The stream is positioned at the first entry in the directory.

The **opendir**() system call return a pointer to the directory stream. On error, NULL is returned

**structdirent \*readdir(DIR \****dirp***);**

The **readdir**() function returns a pointer to a *dirent* structure representing the next directory entry in the directory stream pointed to by *dirp*. It returns NULL on reaching the end of the directory stream or -1 if an error occurred.

The *dirent* structure is defined as follows:

structdirent {

ino\_t d\_ino; /\* Inode number \*/

off\_t d\_off; /\* Not an offset; see below \*/

unsigned short d\_reclen; /\* Length of this record \*/

unsigned char d\_type; /\* Type of file; not supported by all filesystem types \*/

char d\_name[256]; /\* Null-terminated filename \*/

};

The fields of the *dirent* structure are as follows:

*d\_ino* This is the inode number of the file.

*d\_off* The value returned at the current position in the director stream.

*d\_reclen*This is the size (in bytes) of the returned record.

*d\_type* This field contains a value indicating the file type

glibc defines the following macro constantfor the value returned in *d\_type*:

**DT\_BLK** This is a block device.

**DT\_CHR** This is a character device.

**DT\_DIR** This is a directory.

**DT\_FIFO** This is a named pipe (FIFO).

**DT\_LNK** This is a symbolic link.

**DT\_REG** This is a regular file.

**DT\_SOCK** This is a UNIX domain socket.

**DT\_UNKNOWN** The file type could not be determined.

*d\_name* This field contains the null terminated filename.

On success, **readdir**() returns a pointer to a *dirent*structure.If the end of the directory stream is reached, NULL is returned. -1 If an error occurs.

**intclosedir(DIR \****dirp***);**

The **closedir**() function closes the directory stream associated with *dirp*. A successful call to **closedir**() also closes the underlying file descriptor associated with *dirp*. The directory stream descriptor *dirp* is not available after this call.

The **closedir**() function returns 0 on success. On error, -1 is returned

##### **Experiment # 3: use of opendir,readdir , closedir System Calls .**

##### **Aim:**Towrite a program in C to Illustrate the use of opendir,readdir , closedir System Calls .

**Program**

#include<stdio.h>

#include<stdlib.h>

#include<dirent.h>

structdirent \*dptr;

int main()

{

char buff[100];

DIR \*dirp;

printf("\n\n ENTER DIRECTORY NAME");

scanf("%s", buff);

if((dirp=opendir(buff))==NULL)

{

printf("The given directory does not exist");

exit(1);

}

while(dptr=readdir(dirp))

{

printf("%s\n",dptr->d\_name);

}

closedir(dirp);

}