**3.1** Write the program to show file statistics using the stat system call.

Take the filename / directory name from user including path.

**Objectives:**

1. To learn about File system Internals.

**Theory:**

Name:

stat, fstat, lstat - get file status

Syntax:

#include <[sys/types.h](https://linux.die.net/include/sys/types.h)>

#include <[sys/stat.h](https://linux.die.net/include/sys/stat.h)>

#include <[unistd.h](https://linux.die.net/include/unistd.h)>

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

int fstat(int *fd*, struct stat \**buf*);

int lstat(const char \**path*, struct stat \**buf*);

Description:

These functions return information about a file. No permissions are required on the file itself, but-in the case of stat() and lstat() - execute (search) permission is required on all of the directories in *path* that lead to the file.

stat() stats the file pointed to by *path* and fills in *buf*.

lstat() is identical to stat(), except that if *path* is a symbolic link, then the link itself is stat-ed, not the file that it refers to.

fstat() is identical to stat(), except that the file to be stat-ed is specified by the file descriptor *fd*.

All of these system calls return a *stat* structure, which contains the following fields:

struct stat {

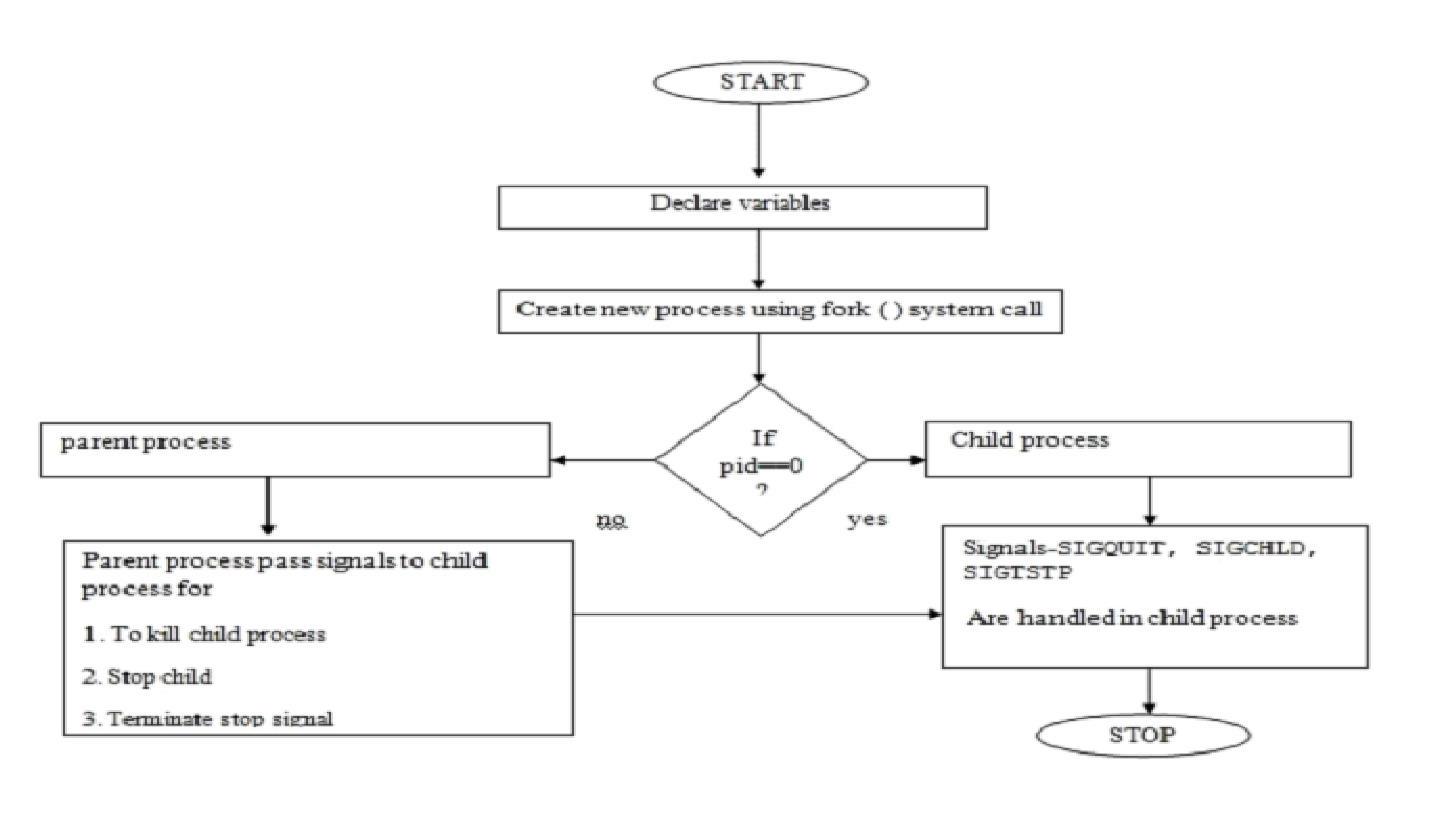
|  |  |  |  |
| --- | --- | --- | --- |
| dev\_t | st\_dev; | /\* ID of device containing file \*/ | |
| ino\_t | st\_ino; | /\* inode number \*/ | |
| mode\_t | st\_mode; | | /\* protection \*/ |
| nlink\_t | st\_nlink; |  | /\* number of hard links \*/ |
| uid\_t | st\_uid; | /\* user ID of owner \*/ | |
| gid\_t | st\_gid; | /\* group ID of owner \*/ | |
| dev\_t | st\_rdev; | /\* device ID (if special file) \*/ | |
| off\_t | st\_size; /\* total size, in bytes \*/ | | |
| blksize\_t st\_blksize; /\* blocksize for file system I/O \*/ | | | |
| blkcnt\_t st\_blocks; | | | /\* number of 512B 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 status change \*/ |

};

**Data Dictionary:**

|  |  |  |  |
| --- | --- | --- | --- |
| Sr Number | Variable/Function | Datatype | Use |
|  |  |  |  |
| 1 | fileStat | struct stat | Store information about files. |
|  |  |  |  |

**Flowchart:**



**Program:**

#include<stdio.h>

#include<unistd.h>

#include<sys/types.h>

#include<sys/stat.h>

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

{

struct stat fileStat;

stat("/home/it/Sarita/UOS/Demo.txt",&fileStat)

if(stat("/home/it/Sarita/UOS/Demo.txt",&fileStat) < 0)

{

printf("Failed\n");

return 1;

}

printf("---------------------------\n");

printf("File Size: \t\t%ld bytes\n",(long)fileStat.st\_size);

printf("Number of Links: \t%ld\n",(long)fileStat.st\_nlink);

printf("File inode: \t\t%ld\n",(long)fileStat.st\_ino);

printf("File Permissions: \t");

printf( (S\_ISDIR(fileStat.st\_mode)) ? "d" : "-");

printf( (fileStat.st\_mode & S\_IRUSR) ? "r" : "-");

printf( (fileStat.st\_mode & S\_IWUSR) ? "w" : "-");

printf( (fileStat.st\_mode & S\_IXUSR) ? "x" : "-");

printf( (fileStat.st\_mode & S\_IRGRP) ? "r" : "-");

printf( (fileStat.st\_mode & S\_IWGRP) ? "w" : "-");

printf( (fileStat.st\_mode & S\_IXGRP) ? "x" : "-");

printf( (fileStat.st\_mode & S\_IROTH) ? "r" : "-");

printf( (fileStat.st\_mode & S\_IWOTH) ? "w" : "-"); printf( (fileStat.st\_mode & S\_IXOTH) ? "x" : "-"); printf("\n\n");

printf("The file %s a symbolic link\n", (S\_ISLNK(fileStat.st\_mode)) ? "is" : "is not"); return 0;

}

**Output:**

sarita@sarita-HP-Laptop-15g-dr0xxx:~/Sarita/UOS$ ./a.out

---------------------------

File Size: 6299 bytes

Number of Links: 1

File inode: 15736649

File Permissions: -rw-rw-r--

The file is not a symbolic link

**Conclusion:**

Stats of file like file size,links, permissions, inode number and type of link can be retrieved using stat() and stored in a structure.

**References:**

https://www.lix.polytechnique.fr/~liberti/public/computing/prog/c/C/FUNCTIONS/stat.html