Practical – 5
Reading Directory Information

Opendir() - open a directory

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

Syntax : DIR *opendir(const char *name);

The opendir() function 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.

Readdir() - read a directory

Syntax :
#include <dirent.h>

struct dirent *readdir(DIR *dir);

readdir() function returns a pointer to a dirent structure, or NULL if an error occurs or end-of-file is reached.

Struct Dirent

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 if an error occurred.

The *dirent* structure is defined as follows:

```
struct dirent {
    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; */
    char           d_name[256]; /* Null-terminated filename */
};
```

Access permission bits for directory maintenance

S_ISUID	04000	Set user ID on execution
S_ISGID	020#0	Set group ID on execution if # is 7,5,3 or 1
S_ISVTX	01000	Save text image after execution
S_IRWXU	00700	Read, write, execute by owner
S_IRUSR	00400	Read by owner
S_IWUSR	00200	Write by owner

S_IXUSR	00100	Execute (search if a directory) by owner
S_IRWXG	00070	Read, write, execute by group
S_IRGRP	00040	Read by group
S_IWGRP	00020	Write by group
S_IXGRP	00010	Execute by group
S_IRWXO	00007	Read, write, execute (search) by other
S_IROTH	00004	Read by other
S_IWOTH	00002	Write by other
S_IXOTH	00001	Execute by other

st_mode member of stat combines the file type with its permission in a space of 16 bits

Bits	File attribute
1 - 4	Type
5 – 7	SUID, SGID and sticky bit permission
8 – 10	Owner permission
11 – 13	Group permission
14 – 16	Other permission

Obtaining File Information: stat(), lstat(), and fstat()

Monitor obtains its file information by calling “stat()”, which works as follows:

System Call : int stat(const char* name, struct stat* buf)
int lstat(const char* name, struct stat* buf)

“stat()” fills the buffer *buf* with information about the file *name*.

The “stat” structure is defined in “/usr/include/sys/stat.h”.

“lstat()” returns information about a symbolic link itself rather than the file that it references.

Stat Struct

This structure is defined in sys/stat.h header file as follows:

```
struct stat {
        mode_t          st_mode;
        ino_t           st_ino;
```

```
        dev_t                   st_dev;
        dev_t                   st_rdev;
        nlink_t                 st_nlink;
        uid_t                   st_uid;
        gid_t                   st_gid;
        off_t                   st_size;
        struct timespec         st_atim;
        struct timespec         st_mtim;
        struct timespec         st_ctim;
        blksize_t               st_blksize;
        blkcnt_t                st_blocks;
};
```

The "stat" structure contains the following members:

NAME	MEANING
st_dev	the device number
st_ino	the inode number
st_mode	the permission flags
st_nlink	the hard-link count
st_uid	the user ID
st_gid	the group ID
st_size	the file size
st_atime	the last access time
st_mtime	the last modification time
st_ctime	the last status-change time

Extracting the information

- To extract the information S_IFMT mask is used
- & operation with st_mode and S_IFMT mask gives the file type
- &~ operation with st_mode and S_IFMT mask gives the file permission
- Example

```
struct stat statbuf;
mode_t file_type, file _perm;
file_type = statbuf.st_mode & S_IFMT;   //1 – 4 bit
file_perm = statbuf.st_mode  & ~S_IFMT;   //5 – 16 bit
```

- For the file type Unix provides number of macros. The following macros are defined to check the file type using the st_mode field :

S_ISREG	Regular file
S_ISDIR	Directory file
S_ISBLK	Block special file
S_ISCHR	Character special file
S_ISLNK	Symbolic link file
S_ISFIFO	FIFO file
S_ISSOCK	Socket file

- example

```
if(S_ISDIR(statbut.st_mode))
        printf("Directory file");
```

Program – 1

```
//This program shows use of chdir() and getwcd() system calls

#include<stdio.h>
#include <unistd.h>

int main(int argc, char *argv[])
{
        char olddir[80];
        char newdir[80];

        if(getcwd(olddir,80)==-1)
                exit(1);
        else
                printf("pwd:%s\n",olddir);

        if(chdir(argv[1])==-1)
                exit(2);
        else
                printf("cd:%s\n",argv[1]);

        getcwd(newdir,80);
        printf("pwd:%s\n",newdir);
}
```

Program – 2

```
//This program reads directory and list files in it

#include<stdio.h>
#include<dirent.h>

int main()
{
        struct dirent *de;

        DIR *dr = opendir(" . ");
        if(dr == NULL)
        {
                printf("Could not open");
                return 0;
        }

        while((de == readdir(dr)) != NULL)
```

```
        {
                printf("%s\n", de->d_name);
        }

        closedir(dr);
        return 0;
}
```

Program – 3

```
//This program performs various file and directory operations
#include<stdio.h>
#include<dirent.h>

int main()
{
        char r,nm[30],lnm[30];
        int ch,mod;

        printf("MENU\n");
        printf("1:Create directory\n2:Remove directory\n3:Hard link\n");
        printf("4:Soft Link\n5:unlink\n6:Rename file\n7:Exit");
        printf("\nEnter ur choice:");
        scanf("%d",&ch);

        switch(ch)
        {
                case 1:
                        printf("Enter directory name:");  scanf("%s",nm);
                        printf("Enter octal permission:"); scanf("%d",&mod);
                        r=mkdir(nm,mod);
                        break;
                case 2:
                        printf("Enter name of directory to be removed:");
                        scanf("%s",nm);
                        rmdir(nm);
                        break;
                case 3:
                        printf("Enter the existing filename:"); scanf("%s",nm);
                        printf("Enter the link filename:");    scanf("%s",lnm);
                        link(nm,lnm);
                        break;
                case 4:
                        printf("Enter the existing filename:"); scanf("%s",nm);
                        printf("Enter the link filename:");    scanf("%s",lnm);
                        symlink(nm,lnm);
                        break;
                case 5:
                        printf("Enter filename to be unlinked:");
                        scanf("%s",nm);
                        unlink(nm);
                        break;
```

```
                case 6:
                        printf("Enter oldfilename:");
                        scanf("%s",nm);
                        printf("Enter new filename:");
                        scanf("%s",lnm);
                        rename(nm,lnm);
                        break;
                default:exit(0);
        }
        printf("Completed...\n");
        exit(0);
}
```

Program – 4

```
//This program displays various information of file using stat structure

#include<stdio.h>
#include<sys/stat.h>

int main(int argc, char *argv[])
{
        struct stat statbuf;

        if((lstat(argv[1],&statbuf))==-1)
                exit(1);

        printf("File:%s\n",argv[1]);
        printf("Inode number:%d\n",statbuf.st_ino);
        printf("Number of links:%d\n",statbuf.st_nlink);
        printf("User ID:%d\n",statbuf.st_uid);
        printf("Permissiion:%o\n",statbuf.st_mode);
        printf("Size:%d\n",statbuf.st_size);
}
```