st_blsize ; // block size for I/O
st_blocks ; // number of 512B blocks allocated.
st_atime ; // time of last access
st_mtime ; // time of last modification
st_ctime ; // time of last status change

}

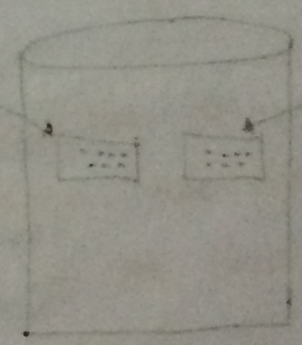
Some Macros for the st_mode field :-

S_ISREG (m) → is it a regular file ?
S_ISDIR (m) → directory ?
S_ISCHR (m) → character device ?
S_ISBLK (m) → block device ?
S_ISFIFO (m) → FIFO (named pipe) ?
S_ISLNK (m) → Symbolic link ?
S_ISSOCK (m) → socket ?

Return Value :-

on success = 0
on error = -1

IF (S_IS REG (u.st_mode))



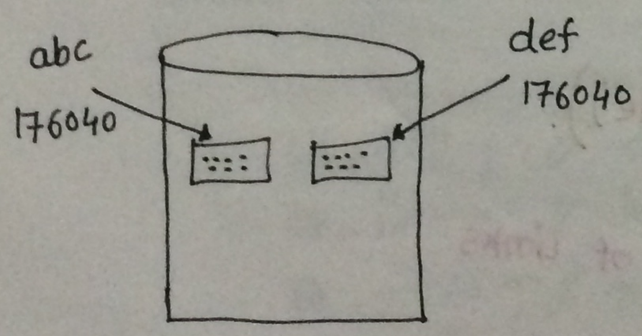
Two types of Links

Hardlink Softlink

→ Using ln command create a link between two files.

Hard-link	Soft-link
1) ln abc def	1) ln -s one two
2) ls -li (inode number are same).	2) ls -li (inode number are different).
3) abc file is also regular and def is also regular.	3) two file is a link file and one file is a regular file.
4) no special file is created.	4) special file is created (link file).
5) number of hard links increased.	5) link count is not increased.
6) Conclusion Hardlink is putting a alice name of existing file.	6) –
7) remove original file than other file is not delet, all the data is in safe mode.	7) remove the original file than link is called as a dangling link.

① Hard link :-



Vi abc
abcd
abcd
abcd
abcd

Then ln abc def

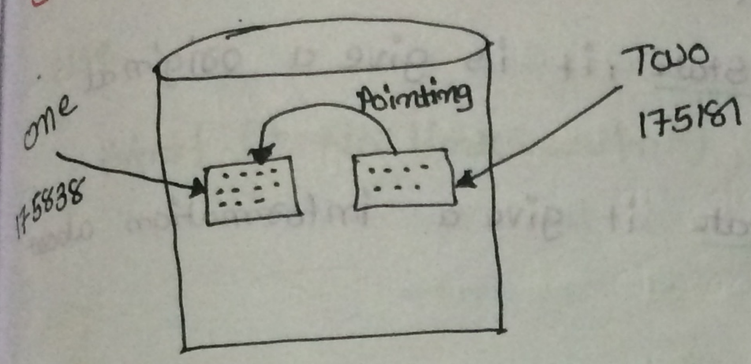
→ abc content is copy in def.

→ we change the abc data then automatically change the def data.

ln command :-

- link is more than 1 that time file manager can't delet the other file.
- link count is zero than filemanager give the another file.
- other file is existing then compiler generate the error.

(2) Soft link :-



ln -s one two

Two file is created as a different colour.

- Softlink file size is number of character of original filename.
- If you modified one file than two file automatically modified.
- delet the symbolic file than you collect the data from the original file.
- delet the original file than link is called the "dangling link."
- softlink is created when all the permission is enabled.

Program :- WAP to take two filename from the user and check is there any link between two files. If link is there display which type of the link.

- ./a.out abc def → Hard link
- ./a.out one two → Soft link
- ./a.out abc one → no link

	State (inode)	lstate (inode)
abc	176040	176040
def	176040	176040
one	176117	176117
two	176117	175181

→ two is a soft link.

→ two is pass to the state, it is give a original file information.

→ two is pass the lstate it give a information about "two file".

```
#include <stdio.h>
# include <sys/types.h>
# include <sys/stat.h>
# include <unistd.h>
main (int argc, char ** argv)
{
        Struct stat v, v1 ;
        if (argc != 3)
        {
                Printf ("Usage : ./a.out fname fname\n");
                return ;
        }
        if (stat (argv[1], &v) <0)
        {
                Perror ("stat");
                return ;
        }
        if (stat(argv[2], &v1) <0)
        {
                Perror ("stat");
                return ;
        }
```

```
if (v.st_ino == v1.st_ino)
{
    lstat (argv[1], &v);
    lstat (argv[2], &v);
    if (v.st_ino == v1.st_ino)
        Printf (" Hard link---\n");
    else
        Printf (" Soft link----\n");
}
else
{
  Printf (" No link----\n");
}
}
```

Opendir() & readdir()

→ A directory is stored in the file system in a similian way to a regular file two things distingwish a directory from a regular file.

- A directory is marked with a diffrent file type in its inode entry.
- A directory is a file with a special organization. Essentially it is a table consisting of filenames and inode numbers.

$ man opendir

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

DIR * opendir (const char * name);

DIR * fdopendir (int fd);
```

Success → directory pointer.

Failure → 0 (Null).

$ man readdir

```
#include <dirent.h>
struct dirent * readdir ( DIR * dirp);
int readdir_r ( DIR *dirp, struct dirent *entry, struct
                dirent **result);
```

```
struct_dirent
{
    d_ino ; // inode number
    d_OFF ; // offset to the next dirent.
    d_reclen ; // length of this record.
    d_type ; // types of the file.
    d_name[256] ; // file name.
};
```

Program my_ls command :-

```
#include <stdio.h>
# include <sys/types.h>
# include <dirent.h>
main (int argc, char **argv)
{
    Struct dirent *p ;
    DIR *dp ;
    if (argc != 2)
    {
        Printf (" Usage : ./a.out dirname\n");
        return ;
    }
    dp = opendir (argv[1]);
    if (dp == 0)
    {
        Perror (" opendir");
    }   return ;
```

```
while ( p = readdir (dp))
    if ( p -> d_name [0] != '.' )
        printf (" %s ", p -> d_name);

// printf (" Thanks ----\n");
}
```

Create vi .abc file then ls command .abc file not display.
ls -a The .abc is display.

1) ./my_ls
2) ./my_ls path
3) ./my_ls -l path
4) ./my_ls -l
5) ./my_ls -li path
6) ./my_ls -ls
7) ./my_ls -ls path

Assignment :-

1) write a program two inputs from the command line one is the path and other is filename.
   * write a program to find out How many time the files is present from that path onward.

logic :-

```
main(------) :
{
    ret = search( ___ , ___ );
}
int search (char *path, char *name)
{
    dp = opdir (        );
    while ( p = readdir (dp)) .
    {
        if ( p -> d_name is directory)
            search ( ___ , name );
```

```
        Else if ( p -> d_name is regular):
            C++ ;
    }
    return C ;
}
```

② design a small prototype of make Utility :-

[Note :- In the linux file manager is not maintaining any file Creation time.

- It's maintaining only st_atime
  st_mtime
  st_ctime].

$ man time :-

```
#include <time.h>

time_t  time (time_t *t);
```

⇒ get time in seconds.

Success → time in second.

Failure → errno.

⇒ time() returns the time as the number of seconds since the Epoch, 1970-01-01 00:00:00 +0000 (UTC).

⇒ If t is non-NULL, the return value is also stored in memory pointed by t.

Program :-

```
#include <stdio.h>
#include <time.h>
main()
{
    time_t  t1 ;
    while(1)
    {
```