FILE RELATED SYSTEM CALLS – 2

AIM: To demonstrate link() system call.

DESCRIPTION : The link system call in Unix-like operating systems is used to create a new link (or hard link) to an existing file. A hard link is essentially a directory entry that associates a name with an inode (data structure on a filesystem that stores information about afile or directory). Multiple hard links can point to the same inode, and all hard links to a file are essentially equivalent. If you modify the content of a file through one hard link, the changes are reflected in all other hard links as well.

PROGRAM:

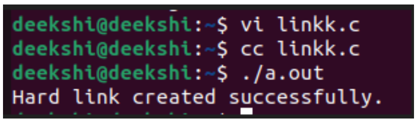
```
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>

int main() {
    const char *existingFileName = "existing_file.txt";
    const char *newLinkName = "hard_link.txt";

    // Create a file (if it doesn't exist)
    FILE *file = fopen(existingFileName, "w");
    if (file == NULL) {
        perror("Error creating file");
        exit(EXIT_FAILURE);
    }
    fprintf(file, "This is the content of the file.\n");
    fclose(file);

    // Use the link system call to create a hard link
    if (link(existingFileName, newLinkName) == -1) {
        perror("Error creating hard link");
        exit(EXIT_FAILURE);
    }
    printf("Hard link created successfully.\n");
    return 0;
}
```

OUTPUT:



```
deekshi@deekshi:~$ vi linkk.c
deekshi@deekshi:~$ cc linkk.c
deekshi@deekshi:~$ ./a.out
Hard link created successfully.
```

AIM: To demonstrate unlink() system call.

DESCRIPTION :

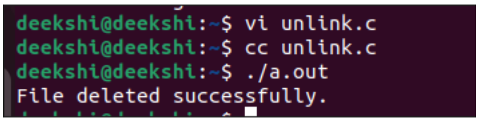
The unlink system call in Unix-like operating systems is used to delete a hard link to a file. Ittakes a single argument, which is the pathname of the file to be deleted. If the file has multiple hard links, deleting one link does not affect the others until the last link is removed.When the last link is deleted, the file's data blocks and inode are deallocated, freeing up the associated disk space.

PROGRAM:

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

int main() {
        const char *fileName = "file_to_delete.txt";
        FILE *file = fopen(fileName, "w");
        if (file == NULL) {
                perror("Error creating file");
                exit(EXIT_FAILURE);
        }
        fprintf(file, "This file will be deleted.\n");
        fclose(file);
        // Use the unlink system call to delete the file
        if (unlink(fileName) == -1){
                perror("Error deleting file");
                exit(EXIT_FAILURE);
      }
        printf("File deleted successfully.\n");
        return 0;
}
```

OUTPUT:



```
deekshi@deekshi:~$ vi unlink.c
deekshi@deekshi:~$ cc unlink.c
deekshi@deekshi:~$ ./a.out
File deleted successfully.
```

AIM: To demonstrate stat() system call.

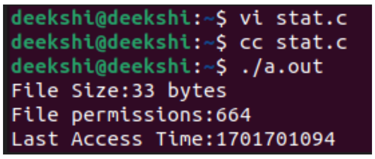
DESCRIPTION:

The stat system call in Unix-like operating systems is used to retrieve information about a file.It provides details such as file size, ownership, permissions, and timestamps. The C library function corresponding to the stat system call is stat or fstat (for a file descriptor).
SYNTAX:
  int stat(const char *pathname, struct stat *statbuf);

PROGRAM:

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/stat.h>
#include <unistd.h>
int main(){
        const char *filename="example.txt";
        struct stat fileStat;
        FILE  *file=fopen(filename,"w");
        if(file==NULL){
                perror("Error creating file");
                exit(EXIT_FAILURE);
        }
        fprintf(file,"This is the content of the file.\n");
        fclose(file);
        if(stat(filename,&fileStat)==-1){
                perror("Error getting file information");
                return -1;
        }
        printf("File Size:%ld bytes\n",fileStat.st_size);
        printf("File permissions:%o\n",fileStat.st_mode & 0777);
        printf("Last Access Time:%ld\n",fileStat.st_atime);
        return 0;
}
```

OUTPUT:



```
deekshi@deekshi:~$ vi stat.c
deekshi@deekshi:~$ cc stat.c
deekshi@deekshi:~$ ./a.out
File Size:33 bytes
File permissions:664
Last Access Time:1701701094
```

AIM: To demonstrate mount() system call.

DESCRIPTION:

The mount() system call is used in Unix-like operating systems to attach the filesystem found on some device (such as a hard disk partition or a CD-ROM) to the file tree. It provides a way to make the content of the filesystem accessible to the system at a specified mount point in the directory tree.

The mount system call is quite complex, and its usage often involves interacting with various data structures and providing specific options. Additionally, in modern Unix-like systems, the mount command is typically used instead of the mount system call directly.

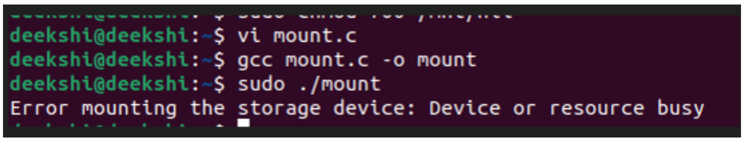
SYNTAX:

int mount(const char *source, const char *target, const char *filesystemtype, unsigned long mountflags, const void *data);

PROGRAM:

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/mount.h>
int main() {
    const char *source = "/dev/sda";     // The storage device you want to connect
    const char *target = "/mnt/hii";  // Where you want to attach it in your file system
    const char *filesystemtype = "ext4";    // The type of the storage device
    unsigned long mountflags = 0;           // Flags, for example, MS_RDONLY for read-only
    const void *data = NULL;                // Optional data specific to the filesystem
    if (mount(source, target, filesystemtype, mountflags, data) == -1) {
        perror("Error mounting the storage device");
        return 1;
    }
    printf("Storage device mounted successfully at /mnt/usb_drive.\n");
    return 0;
}
```

OUTPUT:



```
deekshi@deekshi:~$ vi mount.c
deekshi@deekshi:~$ gcc mount.c -o mount
deekshi@deekshi:~$ sudo ./mount
Error mounting the storage device: Device or resource busy
```

AIM: To demonstrate unmount() system call.

DESCRIPTION:

The umount() system call in Unix-like operating systems is used to unmount a previously mounted filesystem from the directory hierarchy. The corresponding C library function is umount.

SYNTAX:- int umount(const char *target);

PROGRAM:

```
#include <sys/umount.h>
#include <stdio.h>
#include <stdlib.h>
 int main() {
   const char *target = "/mnt/mydrive";
   if (umount(target) == -1) {
       perror("Error unmounting filesystem");
       exit(EXIT_FAILURE);
   }
   printf("Filesystem unmounted successfully.\n");
   return 0;
}
```

CHAITANYA BHARATHI
INSTITUTE OF TECHNOLOGY
1979

AIM: To demonstrate chmod() system call.

DESCRIPTION:

The chmod() and fchmod() system calls change a file’s mode bits. (the mode consists of the file permission bits plus the set-user-ID, set-group-ID, and sticky bits)

Header files:

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

Syntax:

int chmod(const char *path, mode_t mode);

int fchmod(int fildes, mode_t mode);

Return Value:

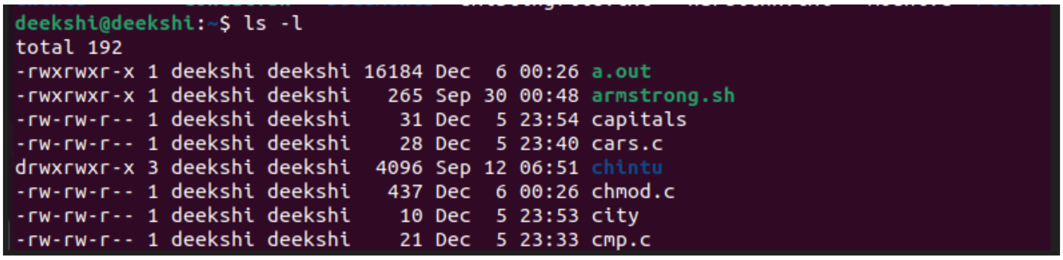
On success, zero is returned. On error, -1 is returned, and errno is set appropriately.

PROGRAM:

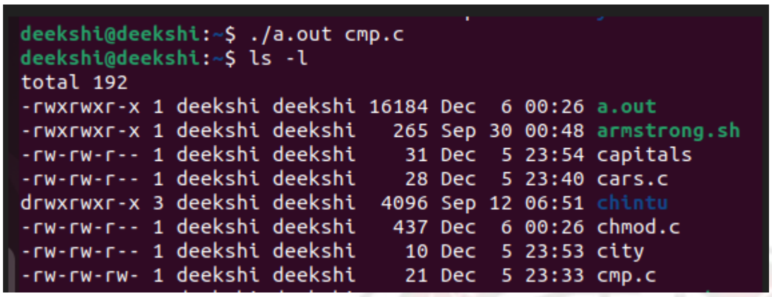
```
#include <stdio.h>
#include <stdlib.h>
#include <sys/stat.h>
int main(int argc, char *argv[])
{
const char *filename;
struct stat fs;
int r;
filename = argv[1];
r = stat(filename,&fs);
if( r==-1)
{
 fprintf(stderr,"Error reading '%s'\n",filename);
 exit(1);
}
r = chmod( filename, fs.st_mode | S_IWGRP+S_IWOTH );
if( r!=0)
{
 fprintf(stderr,"Unable to reset permissions on '%s'\n",filename);
 exit(1);
}
stat(filename,&fs);
return(0);
}
```

CHAITANYA BHARATHI INSTITUTE OF TECHNOLOGY 1979

OUTPUT:



```
deekshi@deekshi:~$ ls -l
total 192
-rwxrwxr-x 1 deekshi deekshi 16184 Dec  6 00:26 a.out
-rwxrwxr-x 1 deekshi deekshi   265 Sep 30 00:48 armstrong.sh
-rw-rw-r-- 1 deekshi deekshi    31 Dec  5 23:54 capitals
-rw-rw-r-- 1 deekshi deekshi    28 Dec  5 23:40 cars.c
drwxrwxr-x 3 deekshi deekshi  4096 Sep 12 06:51 chintu
-rw-rw-r-- 1 deekshi deekshi   437 Dec  6 00:26 chmod.c
-rw-rw-r-- 1 deekshi deekshi    10 Dec  5 23:53 city
-rw-rw-r-- 1 deekshi deekshi    21 Dec  5 23:33 cmp.c
```



```
deekshi@deekshi:~$ ./a.out cmp.c
deekshi@deekshi:~$ ls -l
total 192
-rwxrwxr-x 1 deekshi deekshi 16184 Dec  6 00:26 a.out
-rwxrwxr-x 1 deekshi deekshi   265 Sep 30 00:48 armstrong.sh
-rw-rw-r-- 1 deekshi deekshi    31 Dec  5 23:54 capitals
-rw-rw-r-- 1 deekshi deekshi    28 Dec  5 23:40 cars.c
drwxrwxr-x 3 deekshi deekshi  4096 Sep 12 06:51 chintu
-rw-rw-r-- 1 deekshi deekshi   437 Dec  6 00:26 chmod.c
-rw-rw-r-- 1 deekshi deekshi    10 Dec  5 23:53 city
-rw-rw-rw- 1 deekshi deekshi    21 Dec  5 23:33 cmp.c
```

AIM: To demonstrate chown() system call.

DESCRIPTION:

The chown() function sets the owner ID and group ID of the file that pathname specifies.

Header files:

#include <sys/types.h>

#include <unistd.h>

Syntax:

int chown(const char *pathname, uid_t owner, gid_t group);

int fchown(int fildes, uid_t owner, gid_t group);

int lchown(const char *pathname, uid_t owner, gid_t group);

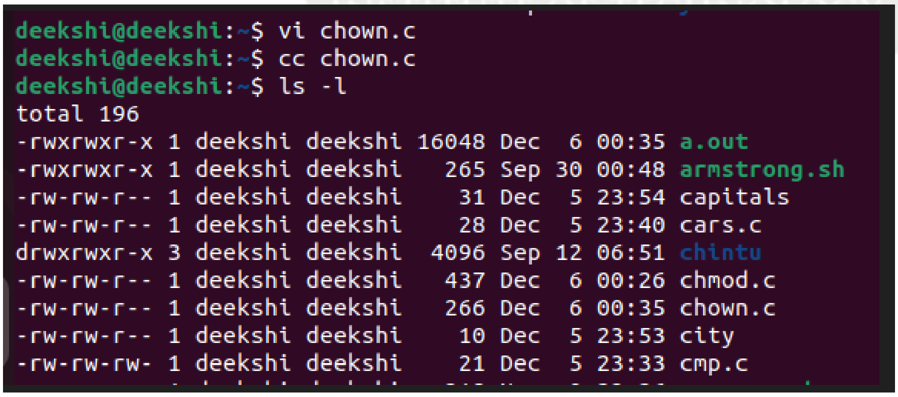
Return Value:

If successful, chown(), fchown(), and lchown() return zero. On failure, they return -1, make no changes to the owner or group of the file, and set errno

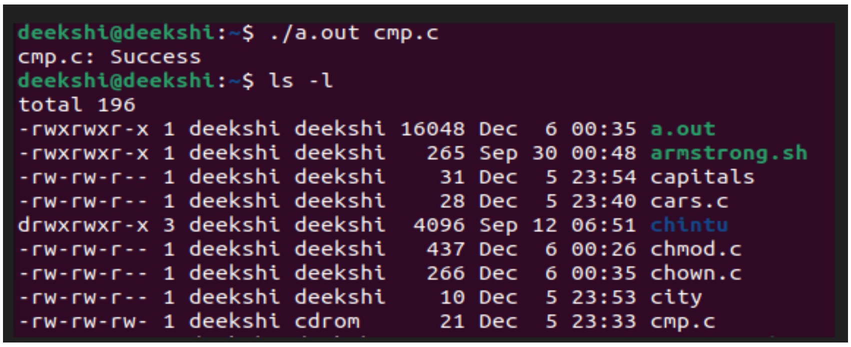
PROGRAM:

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <unistd.h>
int main( int argc, char** argv )
 {
int i;
int ecode = 0;
for( i = 1; i < argc; i++ )
{
 if( chown( argv[i], 1000, 24 ) == 0 )
{
 perror( argv[i] );
 ecode++;
 }
}
exit( ecode );
}
```

OUTPUT:



```
deekshi@deekshi:~$ vi chown.c
deekshi@deekshi:~$ cc chown.c
deekshi@deekshi:~$ ls -l
total 196
-rwxrwxr-x 1 deekshi deekshi 16048 Dec  6 00:35 a.out
-rwxrwxr-x 1 deekshi deekshi   265 Sep 30 00:48 armstrong.sh
-rw-rw-r-- 1 deekshi deekshi    31 Dec  5 23:54 capitals
-rw-rw-r-- 1 deekshi deekshi    28 Dec  5 23:40 cars.c
drwxrwxr-x 3 deekshi deekshi  4096 Sep 12 06:51 chintu
-rw-rw-r-- 1 deekshi deekshi   437 Dec  6 00:26 chmod.c
-rw-rw-r-- 1 deekshi deekshi   266 Dec  6 00:35 chown.c
-rw-rw-r-- 1 deekshi deekshi    10 Dec  5 23:53 city
-rw-rw-rw- 1 deekshi deekshi    21 Dec  5 23:33 cmp.c
```



```
deekshi@deekshi:~$ ./a.out cmp.c
cmp.c: Success
deekshi@deekshi:~$ ls -l
total 196
-rwxrwxr-x 1 deekshi deekshi 16048 Dec  6 00:35 a.out
-rwxrwxr-x 1 deekshi deekshi   265 Sep 30 00:48 armstrong.sh
-rw-rw-r-- 1 deekshi deekshi    31 Dec  5 23:54 capitals
-rw-rw-r-- 1 deekshi deekshi    28 Dec  5 23:40 cars.c
drwxrwxr-x 3 deekshi deekshi  4096 Sep 12 06:51 chintu
-rw-rw-r-- 1 deekshi deekshi   437 Dec  6 00:26 chmod.c
-rw-rw-r-- 1 deekshi deekshi   266 Dec  6 00:35 chown.c
-rw-rw-r-- 1 deekshi deekshi    10 Dec  5 23:53 city
-rw-rw-rw- 1 deekshi cdrom      21 Dec  5 23:33 cmp.c
```

CONCLUSION:

By executing the above program, we have successfully demonstrated the File Related System Calls –link(),unlink(),stat(),mount(),unmount(), chmod(), chown().

CHAITANYA BHARATHI
INSTITUTE OF TECHNOLOGY
స్వయం తేజస్విన్ భవ
1979