**Lab 4 - File System**

**Lab Exercises**

1. Write a program to find the inode number of an existing file in a directory. Take the input as a filename and print the inode number of the file.

#include <sys/types.h>

#include <sys/stat.h>

#include <unistd.h>

#include <stdio.h>

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

{

struct stat sb;

int ret;

if (argc < 2)

{

fprintf (stderr, "usage: %s <file>\n", argv[0]);

return 1;

}

ret = stat (argv[1], &sb);

if (ret)

{

perror ("stat");

return 1;

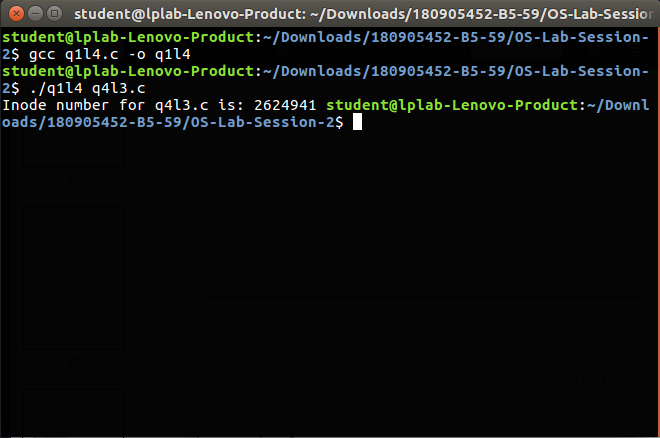
}

printf ("Inode number for %s is: %ld ", argv[1], sb.st\_ino);

return 0;

}

Output



1. Write a program to print out the complete stat structure of a file.

#include <sys/types.h>

#include <sys/stat.h>

#include <unistd.h>

#include <stdio.h>

#include <time.h>

#include <stdlib.h>

#include<dirent.h>

#include<string.h>

char\* formatdate(char\* str, time\_t val)

{

strftime(str, 36, "%d.%m.%Y %H:%M:%S", localtime(&val));

return str;

}

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

{

struct stat sb;

if(argc < 2)

{

printf("Insufficient arguments!\n");

return 1;

}

int ret;

char date[36];

ret = stat(argv[1], &sb);

if(ret)

{

perror("stat");

return 1;

}

printf("ID of device - %d\n", sb.st\_dev);

printf("INO Number is - %llu\n", sb.st\_ino);

printf("File mode - %hu\n", sb.st\_mode);

printf("Number of hard links - %d\n", sb.st\_nlink);

printf("User ID - %d\n", sb.st\_uid);

printf("Group owner - %d\n", sb.st\_gid);

printf("File size - %lld\n", sb.st\_size);

printf("Blocksize - %d\n", sb.st\_blksize);

printf("Number of Blocks - %lld\n", sb.st\_blocks);

printf("Last access time - %s\n", formatdate(date,sb.st\_atime));

printf("Last modification time - %s\n", formatdate(date,sb.st\_mtime));

printf("Last change time - %s\n", formatdate(date,sb.st\_ctime));

DIR \* dp;

struct dirent \* entry;

struct stat statbuf;

if((dp = opendir(".")) == NULL)

{

printf("Cannot open directory \n");

return 0;

}

chdir(".");

while((entry = readdir(dp)) != NULL)

{

lstat(entry->d\_name,&statbuf);

if(!S\_ISDIR(statbuf.st\_mode))

{

if(strcmp(entry->d\_name,argv[1])==0)

{

printf("Permissions\t");

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

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

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

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

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

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

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

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

printf( (statbuf.st\_mode & S\_IWOTH) ? "w" : "-");

printf( (statbuf.st\_mode & S\_IXOTH) ? "x" : "-");

printf("\n\n");

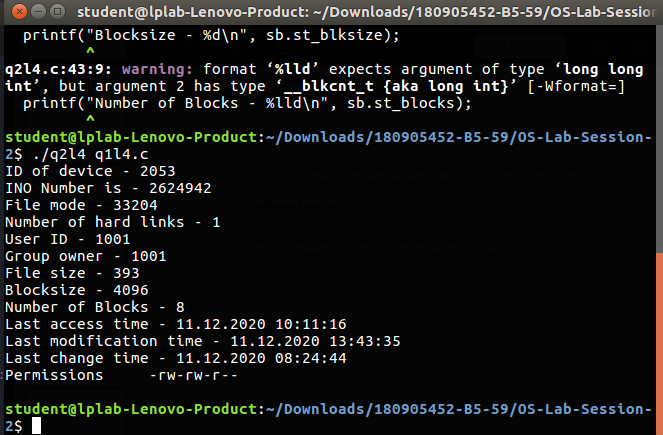
}

}

}

}

Output



1. Write a program to create a new hard link to an existing file and unlink the same. Accept the old path as input and print the newpath.

#include <sys/types.h>

#include <sys/stat.h>

#include <unistd.h>

#include <stdio.h>

#include <inttypes.h>

#include <stdlib.h>

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

{

if(argc<2)

{

printf("Insufficient arguments\n");

return;

}

char new\_path[100]="q1l3.c";

struct stat start;

int ret1 = stat(argv[1],&start);

printf("Number of hard links:%ld\n", start.st\_nlink);

system("ls");

printf("Linking..\n");

int ret2 = link(argv[1],new\_path);

struct stat intermediate;

int ret3 = stat(argv[1],&intermediate);

printf("Number of hard links:%ld\n", intermediate.st\_nlink);

printf("New path:%s\n",new\_path);

system("ls");

int ret4 = unlink(argv[1]);

struct stat ending;

int ret5 = stat(new\_path,&ending);

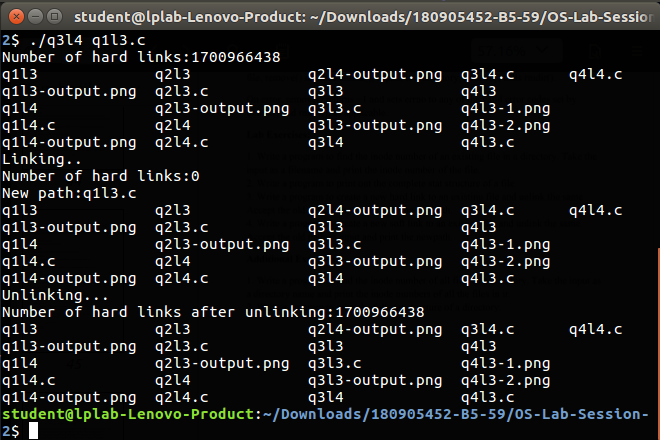
printf("Unlinking...\n");

printf("Number of hard links after unlinking:%ld\n", ending.st\_nlink);

system("ls");

}

Output



4.) Write a program to create a new soft link to an existing file and unlink the same. Accept the old path as input and print the newpath.

#include<sys/types.h>

#include<sys/stat.h>

#include<unistd.h>

#include<stdio.h>

#include <inttypes.h>

#include<stdlib.h>

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

{

if(argc<2)

{

printf("Insufficient arguments\n");

return;

}

char new\_path[100]="q2l3.c";

struct stat start;

int ret1 = stat(argv[1],&start);

system("ls");

printf("Linking..\n");

int ret2 = symlink(argv[1],new\_path);

struct stat intermediate;

int ret3 = stat(argv[1],&intermediate);

printf("New path:%s\n",new\_path);

system("ls");

int ret4 = unlink(argv[1]);

struct stat ending;

int ret5 = stat(new\_path,&ending);

printf("Unlinking...\n");

system("ls");

}

Output

