SEC : A **FILE SYSTEM** **BATCH : A3**

REG: 190905513 **LAB 04** **NAME: MOHAMMAD DANISH EQBAL**

LAB EXERCISES:

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

Program:

```
#include <sys/types.h>
#include <sys/stat.h>
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>

void help() { printf("USAGE: inode
     [name_of_file]\n");
}

int main(int argc, char * argv[]) {
     if ( argc != 2 ) { help();
     exit(EXIT_FAILURE);
     }
     struct stat curr_stat;
     int r_val = lstat( argv[1], &curr_stat );
     if ( r_val < 0 ) { perror("lstat");
     exit(EXIT_FAILURE);
     }
     printf("Inode number: %ld\n", curr_stat.st_ino);
}
```

Output:

```
@lplab-ThinkCentre-M71e: ~/Documents/190905513/OS_LAB/LAB4
student@lplab-ThinkCentre-M71e:~/Documents/190905513/OS_LAB/LAB4$ gcc q1.c -o q1
student@lplab-ThinkCentre-M71e:~/Documents/190905513/OS_LAB/LAB4$ ls
190905513_OSLab4.odt  q1  q1.c
student@lplab-ThinkCentre-M71e:~/Documents/190905513/OS_LAB/LAB4$ ./q1 190905513_OSLab4.odt
Inode number: 263537
student@lplab-ThinkCentre-M71e:~/Documents/190905513/OS_LAB/LAB4$
```

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

Program:

```
#include <sys/types.h>
#include <sys/stat.h>
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <pwd.h>

#include <grp.h>

void help() { printf("USAGE: stat
     [name_of_file]\n");
}

void pp(int r_val, char * str)
     { if ( r_val == 0 )
     { printf("-");
     } else { printf("%s",
          str);
     }
}
void print_filetype(mode_t m)
     { if ( S_ISREG(m) )
     { printf("Regular file");
     } else if ( S_ISDIR(m) )
          { printf("Directory");
     } else if ( S_ISCHR(m) )
          { printf("Character device");
     } else if ( S_ISBLK(m) )
          { printf("Block device");
     } else if ( S_ISFIFO(m) )
          { printf("Named pipe");
     } else if ( S_ISLNK(m) )
          { printf("Symbolic link");
     } else if ( S_ISSOCK(m) )
          { printf("Socket");
     }
}

int main(int argc, char * argv[]) {
     if ( argc != 2 ) { help();
     exit(EXIT_FAILURE);
     }
     struct stat curr_stat;
     struct passwd * file_usr = NULL;
     struct group * file_grp = NULL;
     int r_val = lstat( argv[1], &curr_stat );
```

```
     if ( r_val < 0 )
          { perror("lstat");
          exit(EXIT_FAILURE);
     }
     mode_t m = curr_stat.st_mode;
     printf("Inode number: %ld\n", curr_stat.st_ino);
     printf("Device id: %ld\n", curr_stat.st_dev);
     printf("Mode: %d\n", m); printf("File
     permissions: ");
     pp( m & S_IRUSR, "r" );
     pp( m & S_IWUSR, "w" );
     pp( m & S_IXUSR, "x" );
     pp( m & S_IRGRP, "r" );
     pp( m & S_IWGRP, "w" );
     pp( m & S_IXGRP, "x" );
     pp( m & S_IROTH, "r" );
     pp( m & S_IWOTH, "w" );
     pp( m & S_IXOTH, "x" );
     printf("\n");
     printf("File type: ");
     print_filetype(m);
     printf("\n");
     file_usr = getpwuid(curr_stat.st_uid);
     if ( file_usr == NULL )
          { perror("getpwuid");
          exit(EXIT_FAILURE);
     }
     file_grp = getgrgid(curr_stat.st_gid);
     if ( file_grp == NULL )
     { perror("getgrgid");
     exit(EXIT_FAILURE);
     }
     printf("UID: %d(%s)\n", curr_stat.st_uid, file_usr->pw_name);
     printf("GID: %d(%s)\n", curr_stat.st_gid, file_grp->gr_name);
     printf("Size: %ld bytes\n", curr_stat.st_size);
     printf("Block size: %ld\n", curr_stat.st_blksize);
     printf("No of blocks used: %ld\n", curr_stat.st_blocks);
     printf("Last access time: %s", ctime(&curr_stat.st_atime));
     printf("Last modified time: %s", ctime(&curr_stat.st_mtime));
     printf("Last status change time: %s",
ctime(&curr_stat.st_ctime));
free(file_usr); free(file_grp);
}
```

Output:

```
@lplab-ThinkCentre-M71e: ~/Documents/190905513/OS_LAB/LAB4
student@lplab-ThinkCentre-M71e:~/Documents/190905513/OS_LAB/LAB4$ gcc q2.c -o q2
student@lplab-ThinkCentre-M71e:~/Documents/190905513/OS_LAB/LAB4$ ls
190905513_OSLab4.odt  q1  q1.c  q2  q2.c
student@lplab-ThinkCentre-M71e:~/Documents/190905513/OS_LAB/LAB4$ ./q2 190905513_OSLab4.odt
Inode number: 263537
Device id: 2055
Mode: 33204
File permissions: rw-rw-r--
File type: Regular file
UID: 1001(student)
GID: 1001(student)
Size: 141752 bytes
Block size: 4096
No of blocks used: 280
Last access time: Fri Oct 29 11:38:23 2021
Last modified time: Fri Oct 29 11:38:23 2021
Last status change time: Fri Oct 29 11:38:23 2021
```

3. 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.

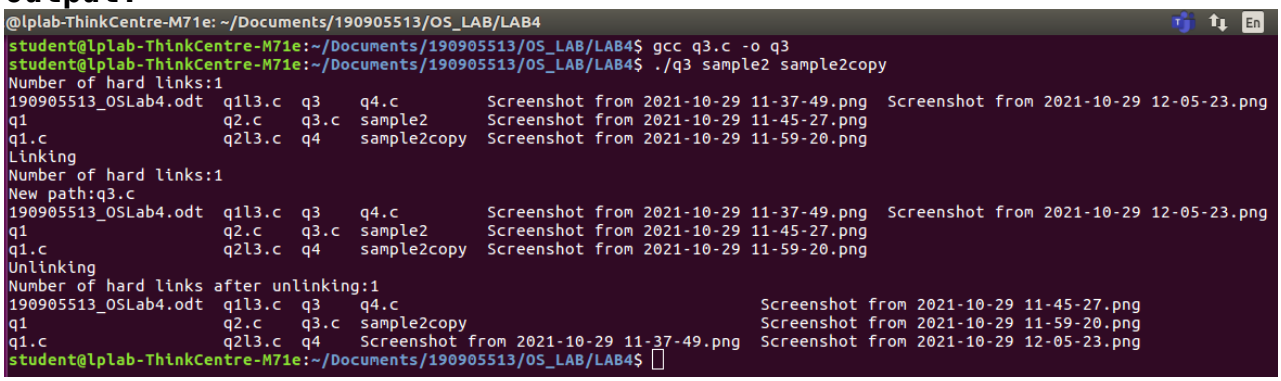
Program:

```
#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]="q3.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.

Program:

```
#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]="q4.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: