

EXPERIMENT 3a:

PROGRAM

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

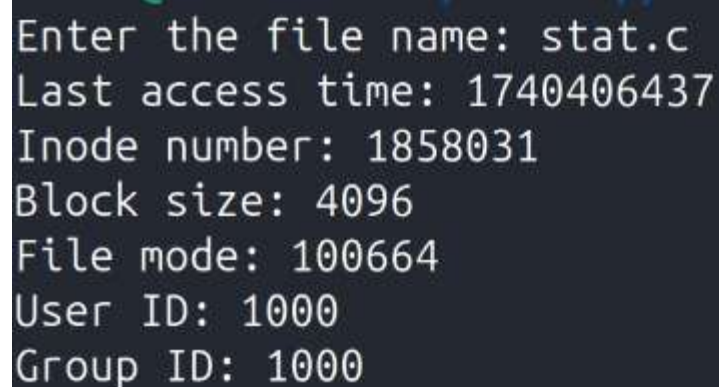
void main()
{
    char file[50];
    struct stat *nfile;
    nfile = (struct stat*)malloc(sizeof(struct stat));

    printf("Enter the file name: ");
    scanf("%s", file);

    stat(file, nfile);

    printf("Last access time: %ld\n", nfile->st_atime);
    printf("Inode number: %lu\n", nfile->st_ino);
    printf("Block size: %ld\n", nfile->st_blksize);
    printf("File mode: %o\n", nfile->st_mode);
    printf("User ID: %u\n", nfile->st_uid);
    printf("Group ID: %u\n", nfile->st_gid);
    free(nfile);
}
```

OUTPUT

A screenshot of a terminal window with a dark background and light green text. It shows the output of the program for the file 'stat.c'. The output lines are: 'Enter the file name: stat.c', 'Last access time: 1740406437', 'Inode number: 1858031', 'Block size: 4096', 'File mode: 100664', 'User ID: 1000', and 'Group ID: 1000'.

```
Enter the file name: stat.c
Last access time: 1740406437
Inode number: 1858031
Block size: 4096
File mode: 100664
User ID: 1000
Group ID: 1000
```

EXPERIMENT 3b:

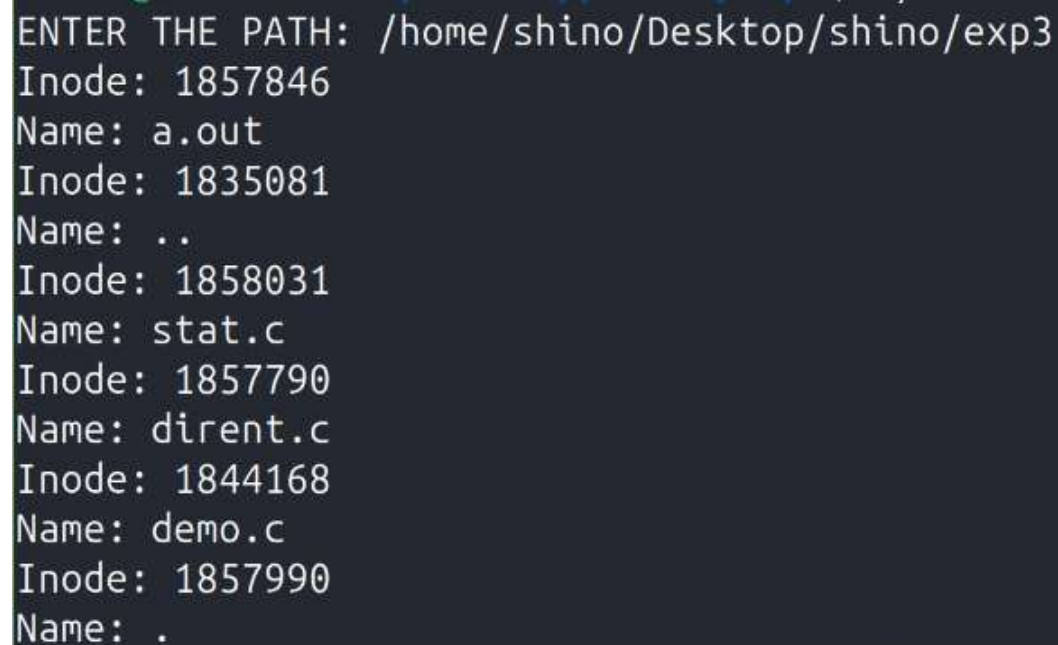
PROGRAM

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

void main() {
    char path[50];
    DIR *dir;
    struct dirent *directory;

    printf("ENTER THE PATH: ");
    scanf("%s", path);
    dir = opendir(path);
    while ((directory = readdir(dir)) != NULL)
    {
        printf("Inode: %lu\n", directory->d_ino);
        printf("Name: %s\n", directory->d_name);
    }
    closedir(dir);
}
```

OUTPUT



```
ENTER THE PATH: /home/shino/Desktop/shino/exp3
Inode: 1857846
Name: a.out
Inode: 1835081
Name: ..
Inode: 1858031
Name: stat.c
Inode: 1857790
Name: dirent.c
Inode: 1844168
Name: demo.c
Inode: 1857990
Name: .
```

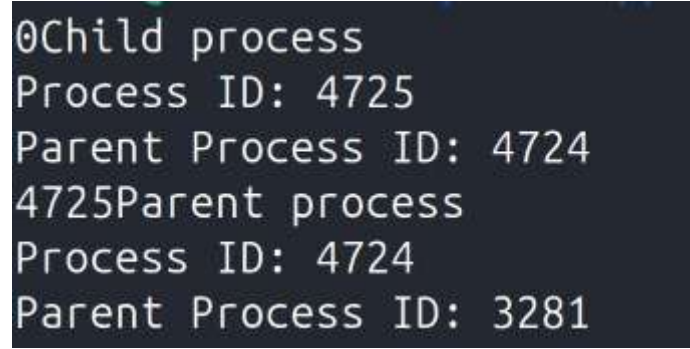
EXPERIMENT 3c:

PROGRAM

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

void main() {
    pid_t p;
    p = fork();
    printf("%d", p);
    if (p < 0) {
        printf("ERROR");
        exit(1);
    }
    else if (p > 0) {
        wait(NULL);
        printf("Parent process \n");
        printf("Process ID: %d\n", getpid());
        printf("Parent Process ID: %d\n", getppid());
    }
    else {
        printf("Child process \n");
        printf("Process ID: %d\n", getpid());
        printf("Parent Process ID: %d\n", getppid());
    }
}
```

OUTPUT



```
0Child process
Process ID: 4725
Parent Process ID: 4724
4725Parent process
Process ID: 4724
Parent Process ID: 3281
```