Experiment No. 3

Title: File and Process handling using system calls

Batch: A1 **Roll No: 16010422013** **Experiment No: 3**

Aim: Implementation of basic commands in Linux and write a program to show file and process handling using system call in Linux.

Resources needed: Ubuntu 15.04 GNU.

Theory:
Pre lab/Prior concepts:

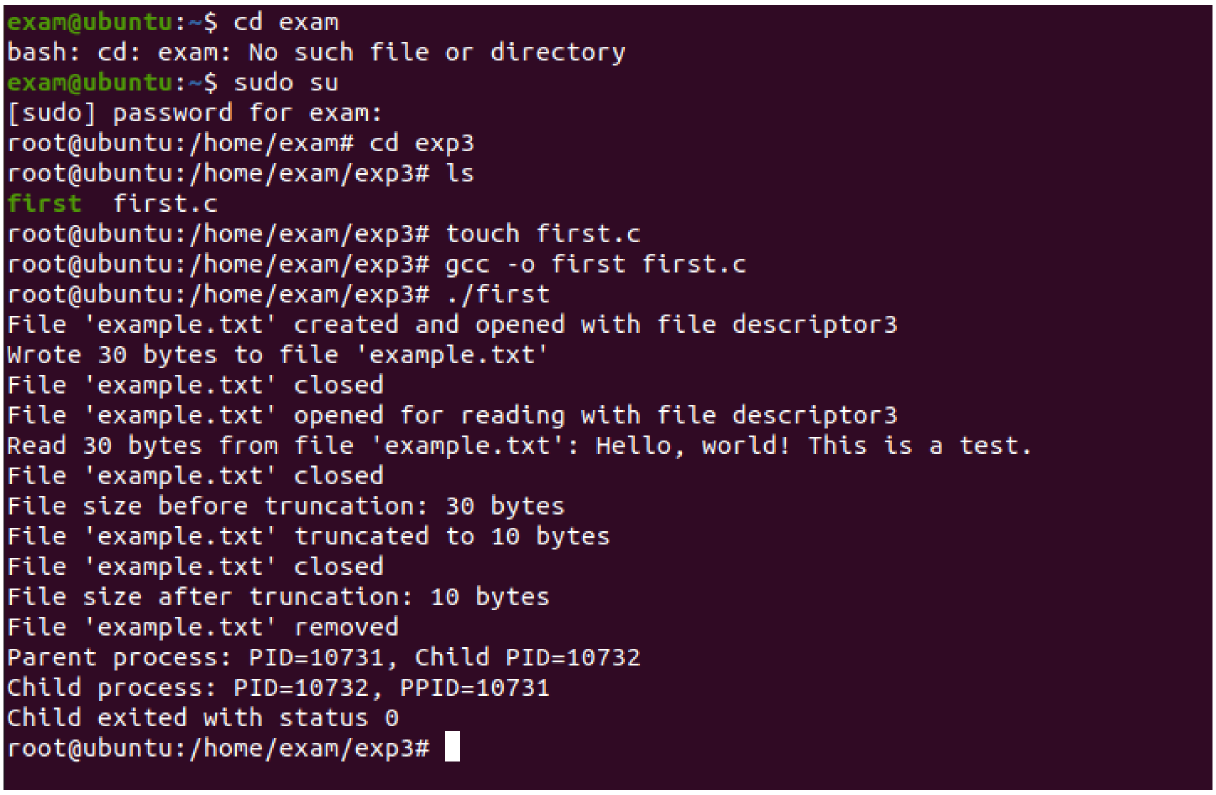
Study the commands given.

Activity:

1. Write a program to show file management and process management using system call.

Results: Perform the activity task and attach the snapshots here

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <fcntl.h>
#include <sys/stat.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <errno.h>
#include <string.h>

int main() {
    const char *filename = "example.txt";
    int fd;
    ssize_t bytes_written;
    char buffer[100];
    struct stat file_info;

    fd = open(filename, O_CREAT | O_WRONLY | O_TRUNC, S_IRUSR | S_IWUSR);

    if (fd == -1) {
        perror("open");
        exit(EXIT_FAILURE);
    }

    printf("File '%s' created and opened with file descriptor%d\n", filename, fd);
    const char *text = "Hello, world! This is a test.\n";
    bytes_written = write(fd, text, strlen(text));

    if (bytes_written == -1) {
        perror("write");
        close(fd);
    exit(EXIT_FAILURE);
}
printf("Wrote %zd bytes to file '%s'\n", bytes_written, filename);

if (close(fd) == -1) {
    perror("close");
    exit(EXIT_FAILURE);
}

printf("File '%s' closed\n", filename);
fd = open(filename, O_RDONLY);

if (fd == -1) {
    perror("open");
    exit(EXIT_FAILURE);
    }
printf("File '%s' opened for reading with file descriptor%d\n", filename, fd);
ssize_t bytes_read = read(fd, buffer, sizeof(buffer) - 1);

if (bytes_read == -1) {
    perror("read");
    close(fd);
    exit(EXIT_FAILURE);
}

buffer[bytes_read] = '\0';
printf("Read %zd bytes from file '%s': %s", bytes_read, filename, buffer);

if (close(fd) == -1) {
    perror("close");
    exit(EXIT_FAILURE);
}

printf("File '%s' closed\n", filename);

if (stat(filename, &file_info) == -1) {
    perror("stat");
    exit(EXIT_FAILURE);
}

printf("File size before truncation: %ld bytes\n", file_info.st_size);
fd = open(filename, O_WRONLY);

if (fd == -1) {
    perror("open");
    exit(EXIT_FAILURE);
}

if (ftruncate(fd, 10) == -1) {
    perror("ftruncate");
    close(fd);
    exit(EXIT_FAILURE);
    }

    printf("File '%s' truncated to 10 bytes\n", filename);

    if (close(fd) == -1) {
        perror("close");
        exit(EXIT_FAILURE);
    }

    printf("File '%s' closed\n", filename);

    if (stat(filename, &file_info) == -1) {
        perror("stat");
        exit(EXIT_FAILURE);
    }
    printf("File size after truncation: %ld bytes\n", file_info.st_size);

    if (unlink(filename) == -1) {
        perror("unlink");
        exit(EXIT_FAILURE);
    }
    printf("File '%s' removed\n", filename);
    pid_t pid = fork();

    if (pid == -1) {
        perror("fork");
        exit(EXIT_FAILURE);
    }

    if (pid == 0) {
        printf("Child process: PID=%d, PPID=%d\n", getpid(), getppid());
        _exit(EXIT_SUCCESS);
    }

    else {
        printf("Parent process: PID=%d, Child PID=%d\n", getpid(), pid);
        int status;
        if (wait(&status) == -1) {
            perror("wait");
            exit(EXIT_FAILURE);
        }
        if (WIFEXITED(status)) {
            printf("Child exited with status %d\n", WEXITSTATUS(status));
        } else {
            printf("Child did not terminate normally\n");
        }
    }

    return 0;
}
```



```
exam@ubuntu:~$ cd exam
bash: cd: exam: No such file or directory
exam@ubuntu:~$ sudo su
[sudo] password for exam:
root@ubuntu:/home/exam# cd exp3
root@ubuntu:/home/exam/exp3# ls
first  first.c
root@ubuntu:/home/exam/exp3# touch first.c
root@ubuntu:/home/exam/exp3# gcc -o first first.c
root@ubuntu:/home/exam/exp3# ./first
File 'example.txt' created and opened with file descriptor3
Wrote 30 bytes to file 'example.txt'
File 'example.txt' closed
File 'example.txt' opened for reading with file descriptor3
Read 30 bytes from file 'example.txt': Hello, world! This is a test.
File 'example.txt' closed
File size before truncation: 30 bytes
File 'example.txt' truncated to 10 bytes
File 'example.txt' closed
File size after truncation: 10 bytes
File 'example.txt' removed
Parent process: PID=10731, Child PID=10732
Child process: PID=10732, PPID=10731
Child exited with status 0
root@ubuntu:/home/exam/exp3#
```

Outcomes: CO1: Understand basic structure of modern operating system

Conclusion: Understanding the results how system calls are used for file and process handling.

Grade: AA/AB/BB/BC/CC/CD/DD

Signature of faculty in-charge with date

References:

Books/ Journals/ Websites:

1. Ri$_n$c$_d$hard Blum and Christine Bresnahan, “Linux Command Line & Shell Scripting”, II Edition edition, Wiley, 2012.