Experiment Number: 4

Problem Statement: **Design menu driven application demonstrating use of different system calls.**

NAME: Aadesh Chawla ROLLNO: 12

CLASS: TY-IT-A BATCH: B1

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Code:**

#include <iostream>

#include <unistd.h>

#include <sys/types.h>

#include <sys/stat.h>

#include <fcntl.h>

#include <sys/wait.h>

#include <string.h>

using namespace std;

// Function prototypes

void processSystemCalls()

{

pid\_t child\_id;

int status;

child\_id = fork();

if (child\_id < 0)

{

perror("fork");

return;

}

else if (child\_id == 0)

{

printf("Child process: PID = %d\n", getpid());

exit(0);

}

else

{

printf("Parent process: PID = %d, Child PID = %d\n", getpid(), childPid);

wait(&status);

}

}

void fileSystemCalls()

{

int fd;

char buffer[100];

ssize\_t bytes\_Read, bytes\_Written;

fd = open("sample.txt", O\_CREAT | O\_RDWR, S\_IRUSR | S\_IWUSR);

if (fd < 0)

{

perror("FILE OPEN");

return;

}

bytes\_Written = write(fd, "This is a sample file", strlen("This is a sample file"));

if (bytes\_Written < 0)

{

perror("FILE WRITE");

}

lseek(fd, 0, SEEK\_SET);

bytes\_Read = read(fd, buffer, sizeof(buffer));

if (bytes\_Read < 0)

{

perror("FILE READ");

}

else

{

buffer[bytes\_Read] = '\0';

printf("READ FROM THE FILE: %s\n", buffer);

}

close(fd);

unlink("sample.txt");

}

void communicationSystemCalls()

{

int fd[2];

char buffer[100];

ssize\_t bytes\_Read, bytes\_Written;

if (pipe(fd) < 0)

{

perror("PIPE CREATION");

return;

}

pid\_t child\_id = fork();

if (child\_id < 0)

{

perror("fork");

return;

}

else if (child\_id == 0)

{

close(fd[0]);

char message[] = "Child Process Executing.\n";

bytes\_Written = write(fd[1], message, strlen(message));

close(fd[1]);

if (bytes\_Written < 0)

{

perror("write");

}

}

else

{

close(fd[1]);

bytes\_Read = read(fd[0], buffer, sizeof(buffer));

close(fd[0]);

if (bytes\_Read < 0)

{

perror("read");

}

else

{

buffer[bytesRead] = '\0';

printf("Msg from child process: %s\n", buffer);

}

wait(NULL);

}

}

void informationSystemCalls()

{

struct stat file\_Stat;

if (stat("test.txt", &file\_Stat) < 0)

{

perror("stat");

return;

}

printf("Details of teh file:\n");

printf("Size: %lld bytes\n", (long long)fileStat.st\_size);

printf("UID: %d\n", fileStat.st\_uid);

printf("GID: %d\n", fileStat.st\_gid);

printf("File inode: %ld\n", (long)fileStat.st\_ino);

printf("Permissions: %o\n", fileStat.st\_mode);

printf("Number of Links: %ld\n", (long)fileStat.st\_nlink);

}

int main()

{

int choice;

while (true)

{

cout << "\nChoose the appropriate:\n";

cout << "1. Process related system calls\n";

cout << "2. File related system calls\n";

cout << "3. Communication system calls\n";

cout << "4. Information related system calls\n";

cout << "5. Exit\n";

cout << "Enter your choice: ";

cin >> choice;

switch (choice)

{

case 1:

processSystemCalls();

break;

case 2:

fileSystemCalls();

break;

case 3:

communicationSystemCalls();

break;

case 4:

informationSystemCalls();

break;

case 5:

cout << "Exiting..." << endl;

exit(0);

default:

cout << "Invalid choice. Please try again." << endl;

}

}

return 0;

}

**Output:**

aadesh@aadesh-VirtualBox:~/Desktop/icecream$ ./a.out

Choose the appropriate:

1. Process related system calls

2. File related system calls

3. Communication system calls

4. Information related system calls

5. Exit

Enter your choice: 1

Parent process: PID = 54445, Child PID = 54446

Child process: PID = 54446

Choose the appropriate:

1. Process related system calls

2. File related system calls

3. Communication system calls

4. Information related system calls

5. Exit

Enter your choice: 2

READ FROM THE FILE: This is a sample file

Choose the appropriate:

1. Process related system calls

2. File related system calls

3. Communication system calls

4. Information related system calls

5. Exit

Enter your choice: 3

Choose the appropriate:

1. Process related system calls

2. File related system calls

3. Communication system calls

4. Information related system calls

5. Exit

Enter your choice: Msg from child process: Child Process Executing.

4

Details of teh file:

Size: 76 bytes

UID: 1000

GID: 1000

File inode: 4195488

Permissions: 100664

Number of Links: 1