Experiment Number: 04

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

**1) process related system call:fork,exit, wait,**

**2) file realted system call: open,read,write,close,link,unlink,stat**

**3) communication system call:pipe,fifo,**

**4)information related system call**

NAME: **ARMAN NANDESHWAR** ROLLNO: **05**

CLASS: **TY-IT-A** BATCH: **B1**

DATE OF PERFORMANCE: **13/08/023**

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

PROGARM:

#include <stdlib.h>

#include <unistd.h>

#include <sys/types.h>

#include <sys/wait.h>

#include <fcntl.h>

#include <string.h>

#include <sys/stat.h>

int main() {

while (1) {

system("clear");

printf("1. Process Related System Calls\n");

printf("2. File Related System Calls\n");

printf("3. Communication System Calls\n");

printf("4. Information Related System Calls\n");

printf("5. Exit\n");

int c;

printf("Enter your c: ");

scanf("%d", &c);

switch (c) {

case 1: // Process Related System Calls

printf("1. fork\n");

printf("2. exit\n");

printf("3. wait\n");

int sub\_c;

printf("Enter sub-c: ");

scanf("%d", &sub\_c);

switch (sub\_c) {

case 1:

printf("fork system call \n");

pid\_t child\_pid = fork();

if (child\_pid == 0) {

printf("This is the child process.\n");

} else if (child\_pid > 0) {

printf("This is the parent process. Child Process ID: %d\n", child\_pid);

} else {

printf("Fork failed.\n");

}

break;

case 2:

printf("exit SysCall\n");

exit(0);

break;

case 3:

printf("wait SysCall\n");

wait(NULL);

printf("Child process Completed.\n");

break;

default:

printf("Invalid sub-c\n");

break;

}

break;

case 2: // File Related System Calls

printf("1. open\n");

printf("2. read\n");

printf("3. write\n");

printf("4. close\n");

printf("5. link\n");

printf("6. unlink\n");

printf("7. stat\n");

printf("Enter sub-c: ");

scanf("%d", &sub\_c);

switch (sub\_c) {

case 1:

printf("open system cal\n");

int fd = open("sample.txt", O\_CREAT | O\_WRONLY, 0644);

if (fd != -1) {

printf("File opened for writing.\n");

close(fd);

} else {

printf("Error opening file.\n");

}

break;

case 2:

printf("read system call\n");

fd = open("sample.txt", O\_RDONLY);

if (fd != -1) {

char content[100];

ssize\_t bytesRead = read(fd, content, sizeof(content));

if (bytesRead > 0) {

content[bytesRead] = '\0';

printf("Content read from file: %s\n", content);

} else {

printf("Error reading file.\n");

}

close(fd);

} else {

printf("Error opening file.\n");

}

break;

case 3:

printf("write system call\n");

fd = open("sample.txt", O\_WRONLY | O\_CREAT | O\_TRUNC, 0644);

if (fd != -1) {

write(fd, "Hello, world!", 13);

printf("Content written to file.\n");

close(fd);

} else {

printf("Error opening file.\n");

}

break;

case 4:

printf("close system call\n");

int fd = open("sample.txt", O\_CREAT | O\_WRONLY, 0644);

close(fd);

printf("File closed.\n");

break;

case 5:

printf("link system call\n");

link("sample.txt", "sample\_link.txt");

printf("Hard link created.\n");

break;

case 6:

printf("unlink system call\n");

unlink("sample\_link.txt");

printf("Hard link removed.\n");

break;

case 7:

printf("stat system call\n");

struct stat st;

if (stat("sample.txt", &st) == 0) {

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

printf("File Permissions: %o\n", st.st\_mode & 0777);

} else {

printf("Error getting file information.\n");

}

break;

default:

printf("Invalid sub-c\n");

break;

}

break;

case 3: // Communication System Calls

printf("1. pipe\n");

printf("2. fifo\n");

printf("Enter sub-c: ");

scanf("%d", &sub\_c);

switch (sub\_c) {

case 1:

printf("pipe system call\n");

int pipe\_fd[2];

if (pipe(pipe\_fd) == 0) {

printf("Pipe created.\n");

close(pipe\_fd[0]);

close(pipe\_fd[1]);

} else {

printf("Error creating pipe.\n");

}

break;

case 2:

printf("fifo system call\n");

mkfifo("myfifo", 0666);

printf("FIFO created.\n");

break;

default:

printf("Invalid sub-c\n");

break;

}

break;

case 4: // Information Related System Calls

printf("information related system call \n");

printf("Current User: %s\n", getenv("USER"));

system("date");

break;

case 5:

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

exit(0);

default:

printf("Invalid c\n");

break;

}

printf("Press Enter to continue...");

getchar(); // Consume the newline character left in the input buffer

getchar(); // Wait for user to press Enter

}

return 0;

}

**Output:**

1. Process Related System Calls

2. File Related System Calls

3. Communication System Calls

4. Information Related System Calls

5. Exit

Enter your c: 1

1. fork

2. exit

3. wait

Enter sub-c: 1

**fork** system call...

This is the parent process. Child Process ID: 6912

Press Enter to continue...This is the child process.

Press Enter to continue...

Enter sub-choice: 2

**exit** SysCall..

Enter sub-choice: 3

**wait** SysCall...

Child process Completed.

Press Enter to continue....

Enter your choice: 2

1. open

2. read

3. write

4. close

5. link

6. unlink

7. stat

Enter sub-choice: 5

Executing link system call...

Hard link created.

Press Enter to continue...

Enter your choice: 3

1. pipe

2. fifo

Enter sub-choice: 1

**pipe** system call...

Pipe created.

Enter your choice: 3

1. pipe

2. fifo

Enter sub-choice: 2

**fifo** system call…

FIFO created.

Press Enter to continue...

Enter your choice: 4

**information** related system call...

Current User: vboxuser

Sat Aug 19 11:14:30 IST 2023

Press Enter to continue...