4ITRC2 Operating System Lab

Lab Assignment 4

Aim: To study and learn about various system calls

To perform: Comprehensive study of different categories of Linux system calls, categorized as

1. Process Management System calls
   fork(), exec(), wait(), exit().
2. File Management System calls
   open(), read(), write(), close().
3. Device Management System calls
   read(), write(), ioctl(), select().
4. Network Management System calls
   socket(), connect(), send(), recv().
5. System Information Management
   System calls getpid(), getuid(),
   gethostname(), sysinfo().

To Submit: Write up for the exhaustive study of the above mentioned system call categories with their examples.

1. Process Management System Calls

Process management system calls are essential for creating, controlling, and terminating processes in a Linux operating system.

- fork(): Used to create a new process. It returns two values: 0 to the child process and the process ID to the parent. Example

```
#include <stdio.h>

#include <unistd.h>

int main() {

  pid_t pid = fork();

  if (pid == 0) {

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

  } else {

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

  }

  return 0;

}
```

Hardik Vaskle, 23I4133

- exec(): Replaces the current process image with a new program image.
- wait(): Pauses the execution of the parent process until a child process has completed.
- exit(): Terminates the current process.

2. File Management System Calls

File management system calls handle operations on files such as opening, reading, writing, and closing.

- open(): Opens a file or creates one if it doesn't exist. Example:

```
#include <fcntl.h>
#include <unistd.h>
#include <stdio.h>
int main() {
    int fd = open("example.txt", O_CREAT | O_WRONLY, 0644);
    if (fd == -1) {
        perror("Error opening file");
        return 1;
    }
    close(fd);
    return 0;
}
```

- read(): Reads data from a file into a buffer.
- write(): Writes data into a file.
- close(): Closes the opened file descriptor.

3. Device Management System Calls

Device management system calls provide interaction with physical and virtual devices.

- ioctl(): Configures or controls devices. Example:

```
#include <sys/ioctl.h>
#include <unistd.h>
#include <stdio.h>
int main() {
```

Hardik vaskle, 23l4133

```
    int result = ioctl(0, 0); // Example usage
    printf("ioctl result: %d\n", result);
    return 0;
}
```

- read() and write(): Used to transfer data between a device and a buffer.

4. Network Management System Calls

Network management system calls enable processes to communicate over a network, facilitating data transmission and reception.

- socket(): Creates a socket for communication between devices. Example:

```
#include <sys/socket.h>
#include <stdio.h>
int main() {
    int sockfd = socket(AF_INET, SOCK_STREAM, 0);
    if (sockfd == -1) {
        perror("Error creating socket");
        return 1;
    }
    printf("Socket created successfully.\n");
    return 0;
}
```

- connect(): Connects a socket to a remote server.
- send(): Sends data through a connected socket.
- recv(): Receives data from a connected socket.

5. System Information Management System Calls

System information management system calls provide access to the system's state and configuration details.

- getpid(): Returns the process ID of the calling process. Example:

```
#include <sys/types.h>
#include <unistd.h>
```

Hardik vaskle, 231433

```
#include <stdio.h>
int main() {
    pid_t pid = getpid();
    printf("Process ID: %d\n", pid);
    return 0;
}
```

- getuid(): Retrieves the user ID of the calling process.
- gethostname(): Fetches the hostname of the system.
- sysinfo(): Provides information about the system (like uptime and memory usage).

Hardik vaskle, 23l4133