**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:

**Submitted by:** **Riya Rawat, 23I4060**

**🔹 1. Process Management System Calls**

These system calls manage the lifecycle of processes — creation, execution, waiting, and termination.

**• fork()**

* Creates a **new process** by duplicating the calling process.
* Returns:
  + 0 to child
  + PID of child to parent
  + -1 on error

Ex: pid\_t pid = fork();

if (pid == 0)

printf("Child Process\n");

else

printf("Parent Process, child PID = %d\n", pid);

**• exec()**

* Replaces the current process image with a **new process image**.
* Many variants: execl(), execp(), execv(), etc.

Ex: execl("/bin/ls", "ls", "-l", NULL);

**• wait()**

* Parent **waits for child process** to terminate.

Ex: int status;

wait(&status);

**• exit()**

* **Terminates the process** and returns status to the parent

Ex: exit(0); // successful exit

**🔹 2. File Management System Calls**

These handle low-level **file I/O** operations such as open, read, write, and close.

**• open()**

* Opens a file and returns a **file descriptor**.

Ex: int fd = open("file.txt", O\_RDONLY);

**• read()**

* Reads data from a file descriptor into a buffer.

Ex; char buffer[100];

read(fd, buffer, 100);

**• write()**

* Writes data from a buffer to a file descriptor

Ex: write(fd, "Hello", 5);  
**• close()**

* Closes the file descriptor

Ex: close(fd);  
  
  
**🔹 3. Device Management System Calls**

These system calls are used to **communicate with hardware devices**.

**• read() and write() (also used for devices)**

* These operate on device files in /dev (e.g., /dev/tty, /dev/sda).  
  Ex: int fd = open("/dev/tty", O\_RDONLY);
* read(fd, buffer, 100);
* **• ioctl()**
* Sends control commands to devices.

Ex: int fd = open("/dev/tty", O\_RDONLY);

ioctl(fd, TIOCGWINSZ, &w); // get terminal window size

**• select()**

* Monitors multiple file descriptors (including devices) for I/O readiness.

Ex: fd\_set rfds;

FD\_SET(fd, &rfds);

select(fd + 1, &rfds, NULL, NULL, NULL);  
  
**🔹 4. Network Management System Calls**

These are essential for **socket programming** and internet communication.

**• socket()**

* Creates a socket.

Ex: int sockfd = socket(AF\_INET, SOCK\_STREAM, 0);  
  
**• connect()**

* Connects the socket to a remote server

Ex: connect(sockfd, (struct sockaddr \*)&server\_addr, sizeof(server\_addr));

**• send()**

* Sends data over the socket.

Ex: send(sockfd, "Hello", strlen("Hello"), 0);

**• recv()**

* Receives data from the socket.

Ex: recv(sockfd, buffer, 1024, 0);

**🔹 5. System Information Management System Calls**

These calls retrieve information about the **process, user, and system**.

**• getpid()**

* Returns the **process ID** of the calling process.

Ex: printf("PID: %d\n", getpid());

**• getuid()**

* Returns the **user ID** of the calling process.  
  Ex: printf("UID: %d\n", getuid());

**• gethostname()**

* Gets the **hostname** of the system.

Ex: char hostname[1024];

gethostname(hostname, 1024);

**• sysinfo()**

* Retrieves **system statistics** (uptime, memory, load, etc.).

Ex: struct sysinfo info;

sysinfo(&info);

printf("Uptime: %ld seconds\n", info.uptime);