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

**Introduction-**

System calls act as the interface between a user application and the kernel (operating system). They enable programs to request services like process management, file operations, device handling, and network communication.

Linux provides a wide range of system calls categorized into different groups. Here we study five major categories.

1. Process Management System calls

**fork()-** Creates a child process by duplicating the parent process. Returns 0 (child) or child PID (parent).

**exec()-** Replaces the current process image with a new executable (e.g., execl, execvp).

**wait()** Suspends the parent until a child process exits.

**exit()-** Terminates the calling process with a status code.

#include <stdio.h>

#include <unistd.h>

#include <sys/wait.h>

int main() {

pid\_t pid = fork();

if (pid == 0) { // Child process

execl("/bin/ls", "ls", "-l", NULL); // Replace child with "ls -l"

} else if (pid > 0) { // Parent process

wait(NULL); // Wait for child to finish

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

} else {

perror("fork() failed");

}

return 0;

}

1. File Management System calls

Purpose: Manipulate files (open, read/write, close).  
Key System Calls:

* **open():** Opens a file, returns a file descriptor.
* **read():** Reads data from a file descriptor.
* **write():** Writes data to a file descriptor.
* **close():** Closes a file descriptor.

#include <fcntl.h>

#include <unistd.h>

#include <string.h>

int main() {

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

const char \*msg = "Hello, World!\n";

write(fd, msg, strlen(msg));

close(fd);

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

char buf[100];

read(fd, buf, sizeof(buf));

printf("Read: %s\n", buf);

close(fd);

return 0;

}

1. Device Management System calls

Purpose: Manage hardware devices (treated as files in Linux).  
Key System Calls:

* **ioctl():** Device-specific operations (e.g., adjusting terminal settings).
* **select():** Monitors multiple file descriptors for I/O readiness.

Example (Terminal Size via ioctl):

#include <sys/ioctl.h>

#include <unistd.h>

#include <stdio.h>

int main() {

struct winsize w;

ioctl(STDIN\_FILENO, TIOCGWINSZ, &w);

printf("Terminal size: %d rows x %d cols\n", w.ws\_row, w.ws\_col);

return 0;

}

1. Network Management System calls

Purpose: Create and manage network sockets.  
Key System Calls:

* **socket():** Creates a socket (e.g., AF\_INET, SOCK\_STREAM).
* **connect():** Connects to a remote server.
* **send()/recv():** Send/receive data over a socket.

**Example** (Simple TCP Client):

#include <sys/socket.h>

#include <netinet/in.h>

#include <arpa/inet.h>

#include <string.h>

int main() {

int sock = socket(AF\_INET, SOCK\_STREAM, 0);

struct sockaddr\_in server\_addr = {

.sin\_family = AF\_INET,

.sin\_port = htons(8080),

.sin\_addr.s\_addr = inet\_addr("127.0.0.1")

};

connect(sock, (struct sockaddr\*)&server\_addr, sizeof(server\_addr));

send(sock, "Hello Server!", 13, 0);

close(sock);

return 0;

}

1. System Information Management System calls

Purpose: Retrieve system/process metadata.  
Key System Calls:

* **getpid():** Returns the current process ID.
* **getuid():** Returns the user ID of the process owner.
* **gethostname():** Gets the system hostname.
* **sysinfo():** Returns system statistics (e.g., uptime).

#include <unistd.h>

#include <sys/sysinfo.h>

#include <stdio.h>

int main() {

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

printf("UID: %d\n", getuid());

char hostname[100];

gethostname(hostname, sizeof(hostname));

printf("Hostname: %s\n", hostname);

struct sysinfo info;

sysinfo(&info);

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

return 0;

}