Operating System

(4ITRC2)

IT IV Semester

Submitted by

HIMANSHU GAWALI

23I4025

Information Technology -A

Submitted to

Jasneet kaur

Department of Information Technology

Institute of Engineering and Technology

Devi Ahilya Vishwavidhyalaya, Indore (M.P.) India

(www.iet.dauniv.ac.in**)**

Session jan- may, 2025

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()

- The fork() system call creates a new child process by duplicating the calling process.

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;
}
```

exec()

- The exec() family of system calls replaces the current process image with a new process.

Example:

```
#include <stdio.h>
#include <unistd.h>

int main() {
   char *args[] = {"/bin/ls", "-l", NULL};
   execvp(args[0], args);
   return 0;
}
```

wait()

- The wait() system call makes a parent process wait for its child process to terminate.

Example:

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

int main() {
   pid_t pid = fork();
   if (pid > 0) {
      wait(NULL);
      printf("Child process terminated.\n");
   } else {
      printf("Child process executing.\n");
   }
   return 0;
```

```
}
```

exit()

- The exit() system call terminates a process.

Example:

```
#include <stdlib.h>

int main() {
   exit(0);
}
```

2. File Management System Calls

These system calls manage file operations such as opening, reading, writing, and closing files.

open()

- Opens a file and returns a file descriptor.

Example:

```
#include <fcntl.h>
#include <stdio.h>

int main() {
   int fd = open("example.txt", O_CREAT | O_WRONLY, 0644);
   if (fd < 0) {
      printf("Error opening file.\n");
   }
   return 0;
}
```

read()

- Reads data from a file.

Example:

```
#include <fcntl.h>
#include <unistd.h>
#include <stdio.h>

int main() {
   char buffer[100];
```

```
    int fd = open("example.txt", O_RDONLY);
    read(fd, buffer, sizeof(buffer));
    printf("%s\n", buffer);
    close(fd);
    return 0;
}
```

write()

- Writes data to a file.

Example:

```
#include <fcntl.h>
#include <unistd.h>

int main() {
    int fd = open("example.txt", O_WRONLY);
    write(fd, "Hello, World!", 13);
    close(fd);
    return 0;
}
```

close()

- Closes an open file.

Example:

```
#include <fcntl.h>
#include <unistd.h>
```

```
int main() {
    int fd = open("example.txt", O_RDONLY);
    close(fd);
    return 0;
}
```

3. Device Management System Calls.

These system calls manage device input and output operations.

read()

- Used to interact with devices like /dev/random (input) and /dev/null (output).

Example:

```
int fd = open("/dev/mydevice", O_RDONLY);
char buffer[100];
ssize_t bytesRead = read(fd, buffer, sizeof(buffer));
close(fd);
```

write()

- The write() function sends data from user space to a device.

Example:

```
int fd = open("/dev/mydevice", O_WRONLY);
char data[] = "Hello, Device!";
write(fd, data, strlen(data));
close(fd);
```

ioctl()

- Performs device-specific operations.
-

Example:

```
#include <stdio.h>
#include <fcntl.h>
#include <sys/ioctl.h>
```

```
int main() {
   int fd = open("/dev/tty", O_RDONLY);
```

```
    int status;
    ioctl(fd, FIONREAD, &status);
    printf("Bytes available: %d\n", status);
    close(fd);
    return 0;
}
```

select()

- Monitors multiple file descriptors.

Example:

```
#include <sys/select.h>
#include <stdio.h>
#include <unistd.h>

int main() {
    fd_set set;
    FD_ZERO(&set);
    FD_SET(0, &set);
    select(1, &set, NULL, NULL, NULL);
    printf("Input detected.\n");
    return 0;
}
```

4. Network Management System Calls.

These system calls manage network communication.

socket()

- Creates a socket.

Example:

```
#include <sys/socket.h>

int main() {
   int sockfd = socket(AF_INET, SOCK_STREAM, 0);
   return 0;
}
```

connect()

- Connects to a remote host.

Example:

```
#include <sys/socket.h>
#include <netinet/in.h>

int main() {
   int sockfd = socket(AF_INET, SOCK_STREAM, 0);
   struct sockaddr_in server;
   server.sin_family = AF_INET;
   server.sin_port = htons(8080);
   connect(sockfd, (struct sockaddr *)&server, sizeof(server));
   return 0;
}
```

send()

- Sends data over a network.

  Example:

```
#include <sys/socket.h>

int main() {
   int sockfd = socket(AF_INET, SOCK_STREAM, 0);
   char message[] = "Hello";
   send(sockfd, message, sizeof(message), 0);
   return 0;
}
```

recv()

- Receives data from a network.

  Example:

```
#include <sys/socket.h>

int main() {
   int sockfd = socket(AF_INET, SOCK_STREAM, 0);
   char buffer[1024];
   recv(sockfd, buffer, sizeof(buffer), 0);
   return 0;
}
```

5. System Information Management System Calls.

These system calls retrieve system-related information.

getpid()

- Gets the process ID.

Example:

```
#include <stdio.h>
#include <unistd.h>

int main() {
   printf("PID: %d\n", getpid());
   return 0;
}
```

getuid()

- Gets the user ID.

Example:

```
#include <stdio.h>
#include <unistd.h>

int main() {
   printf("UID: %d\n", getuid());
   return 0;
```

```
}
```

gethostname()

- Gets the hostname of the system.

Example:

```
#include <stdio.h>
#include <unistd.h>

int main() {
    char hostname[1024];
    gethostname(hostname, sizeof(hostname));
    printf("Hostname: %s\n", hostname);
    return 0;
}
```

sysinfo()

- Gets system information.

Example:

```
#include <stdio.h>
#include <sys/sysinfo.h>

int main() {
    struct sysinfo info;
    sysinfo(&info);
```

```
    printf("Uptime: %ld seconds\n", info.uptime);
    return 0;
}
```