# ASSIGNMENT-4

**Objective:**

To perform :-

1. Process Management System Calls
2. File Management System Calls
3. Device Management System Calls
4. Network Management System Calls
5. System Information Management System Calls

**1. Process Management System Calls**

These system calls are used for process creation, execution, synchronization, and termination.

* **fork()**
* Creates a new process by duplicating the calling process.
* Returns 0 in the child process and PID of the child in the parent process.

**Example:**

#include <unistd.h>

#include <stdio.h>

int main() {

pid\_t pid = fork();

if (pid == 0)

printf("Child Process\n");

else

printf("Parent Process\n");

return 0;

}

* **exec()**
* Replaces the current process image with a new one.
* Common variants include execl(), execp(), execv(), etc.

**Example:**

#include <unistd.h>

int main() {

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

return 0;

}

* **wait()**
* Makes the parent process wait until a child process terminates.

**Example:**

#include <sys/wait.h>

#include <unistd.h>

#include <stdio.h>

int main() {

if (fork() == 0)

execlp("ls", "ls", NULL);

else

wait(NULL);

return 0;

}

* **exit()**
* Terminates the calling process and returns status to the parent.

**Example:**

#include <stdlib.h>

int main() {

exit(0);

}

**2. File Management System Calls**

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

* **open()**
* Opens or creates a file and returns a file descriptor.

**Example:**

#include <fcntl.h>

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

* **read()**
* Reads data from a file descriptor.

**Example:**

#include <unistd.h>

char buffer[100];

read(fd, buffer, 100);

* **write()**
* Writes data to a file descriptor.

**Example:**

write(fd, "Hello", 5);

* **close()**
* Closes an opened file descriptor.

**Example:**

close(fd);

**3. Device Management System Calls**

These system calls are used to manage and control devices using file descriptors.

* **read() and write()**
* Used in device files like /dev/tty to interact with hardware in the same way as files.
* **ioctl()**
* Stands for "Input/Output Control".
* Used to configure or control device parameters.

**Example:**

#include <sys/ioctl.h>

#include <fcntl.h>

int main() {

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

int result;

ioctl(fd, TIOCMGET, &result);

close(fd);

return 0;

}

* **select()**
* Monitors multiple file descriptors to check if I/O is possible on any of them.

**Example:**

#include <sys/select.h>

fd\_set rfds;

FD\_ZERO(&rfds);

FD\_SET(0, &rfds); // Monitor stdin

select(1, &rfds, NULL, NULL, NULL);

**4. Network Management System Calls**

Used for socket programming and networking communication.

* **socket()**
* Creates a new socket.

**Example:**

#include <sys/socket.h>

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

* **connect()**
* Connects a socket to a remote address.

**Example:**

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

* **send()**
* Sends data over a socket.

**Example:**

send(sockfd, "Hello", 5, 0);

* **recv()**
* Receives data from a socket.

**Example:**

recv(sockfd, buffer, sizeof(buffer), 0);

**5. System Information Management System Calls**

These system calls fetch system-level information like process ID, user ID, uptime, etc.

* **getpid()**
* Returns the process ID of the calling process.

**Example:**

#include <unistd.h>

#include <stdio.h>

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

* **getuid()**
* Returns the user ID of the calling process.

**Example:**

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

* **gethostname()**
* Gets the hostname of the machine.

**Example:**

#include <unistd.h>

char hostname[1024];

gethostname(hostname, 1024);

* **sysinfo()**
* Returns system statistics such as uptime, total RAM, etc.

**Example:**

#include <sys/sysinfo.h>

struct sysinfo info;

sysinfo(&info);

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