**Lab Assignment 4: Linux System Calls**

**1. Process Management System Calls**

These system calls manage the creation and termination of processes.

**Lab Assignment 4: Linux System Calls**

**1. Process Management System Calls**

These system calls manage the creation and termination of processes.

* fork()
  + Explanation: Creates a new child process identical to the parent.
  + Code:

pid\_t pid = fork();

if (pid == 0)

printf("Child process\n");

else

printf("Parent process\n");

* + Output:

Parent process

Child process

* exec()
  + Explanation: Replaces current process with a new program.
  + Code:

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

* + Output: (Lists files in the current directory)
* wait()
  + Explanation: Makes the parent wait until child process ends.
  + Code:

wait(NULL);

printf("Child completed\n");

* + Output:

Child completed

* exit()
  + Explanation: Terminates a process with a return value.
  + Code:

exit(0);

* + Output: (No visible output; returns control to OS)

**2. File Management System Calls**

Handle operations on files.

* open()
  + Explanation: Opens file and returns a file descriptor.
  + Code:

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

* + Output:

File descriptor: 3

* read()
  + Explanation: Reads data from a file into buffer.
  + Code:

char buf[20];

read(fd, buf, 20);

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

* + Output:

Data: Hello from file

* write()
  + Explanation: Writes data to a file.
  + Code:

write(fd, "Hello", 5);

* + Output: (Writes "Hello" to file)
* close()
  + Explanation: Closes the opened file descriptor.
  + Code:

close(fd);

* + Output: (No visible output)

**3. Device Management System Calls**

Communicate with devices.

* read() / write()

read(fd, buf, 20);

write(fd, "Hello", 5);

* ioctl()
  + Explanation: Used for device-specific input/output operations.
  + Code:

ioctl(fd, TIOCGWINSZ, &w);

* + Output: (Returns terminal window size)
* select()
  + Explanation: Waits for I/O on multiple file descriptors.
  + Code:

fd\_set rfds;

FD\_ZERO(&rfds);

FD\_SET(0, &rfds);

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

* + Output: (Waits for user input on stdin)

**4. Network Management System Calls**

Used for network communication.

* socket()
  + Explanation: Creates a communication endpoint.
  + Code:

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

* + Output:

Socket created: 3

* connect()
  + Explanation: Connects to a server.
  + Code: (used with socket and sockaddr structures)
  + Output:

Connected to server

* send()
  + Explanation: Sends data to the connected socket.
  + Output:

Sent 5 bytes

* recv()
  + Explanation: Receives data from socket.
  + Output:

Server: Hello Client

**5. System Information Management System Calls**

Provide information about process and system.

* getpid()
  + Explanation: Returns process ID.
  + Code:

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

* + Output:

PID: 12345

* getuid() / geteuid()
  + Explanation: Returns real/effective user ID.
  + Output:

UID: 1000, EUID: 1000

* gethostname()
  + Explanation: Gets current hostname.
  + Output:

Hostname: my-computer

* sysinfo()
  + Explanation: Returns system statistics like uptime, RAM, etc.
  + Output:

Uptime: 3600 seconds, Free RAM: 2048000 bytes

* k()
  + Explanation: Creates a new child process identical to the parent.
  + Code:

pid\_t pid = fork();

if (pid == 0)

printf("Child process\n");

else

printf("Parent process\n");

* + Output:

Parent process

Child process

* exec()
  + Explanation: Replaces current process with a new program.
  + Code:

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

* + Output: (Lists files in the current directory)
* wait()
  + Explanation: Makes the parent wait until child process ends.
  + Code:

wait(NULL);

printf("Child completed\n");

* + Output:

Child completed

* exit()
  + Explanation: Terminates a process with a return value.
  + Code:

exit(0);

* + Output: (No visible output; returns control to OS)

**2. File Management System Calls**

Handle operations on files.

* open()
  + Explanation: Opens file and returns a file descriptor.
  + Code:

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

* + Output:

File descriptor: 3

* read()
  + Explanation: Reads data from a file into buffer.
  + Code:

char buf[20];

read(fd, buf, 20);

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

* + Output:

Data: Hello from file

* write()
  + Explanation: Writes data to a file.
  + Code:

write(fd, "Hello", 5);

* + Output: (Writes "Hello" to file)
* close()
  + Explanation: Closes the opened file descriptor.
  + Code:

close(fd);

* + Output: (No visible output)

**3. Device Management System Calls**

Communicate with devices.

* read() / write()

read(fd, buf, 20);

write(fd, "Hello", 5);

* ioctl()
  + Explanation: Used for device-specific input/output operations.
  + Code:

ioctl(fd, TIOCGWINSZ, &w);

* + Output: (Returns terminal window size)
* select()
  + Explanation: Waits for I/O on multiple file descriptors.
  + Code:

fd\_set rfds;

FD\_ZERO(&rfds);

FD\_SET(0, &rfds);

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

* + Output: (Waits for user input on stdin)

**4. Network Management System Calls**

Used for network communication.

* socket()
  + Explanation: Creates a communication endpoint.
  + Code:

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

* + Output:

Socket created: 3

* connect()
  + Explanation: Connects to a server.
  + Code: (used with socket and sockaddr structures)
  + Output:

Connected to server

* send()
  + Explanation: Sends data to the connected socket.
  + Output:

Sent 5 bytes

* recv()
  + Explanation: Receives data from socket.
  + Output:

Server: Hello Client

**5. System Information Management System Calls**

Provide information about process and system.

* getpid()
  + Explanation: Returns process ID.
  + Code:

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

* + Output:

PID: 12345

* getuid() / geteuid()
  + Explanation: Returns real/effective user ID.
  + Output:

UID: 1000, EUID: 1000

* gethostname()
  + Explanation: Gets current hostname.
  + Output:

Hostname: my-computer

* sysinfo()
  + Explanation: Returns system statistics like uptime, RAM, etc.
  + Output:

Uptime: 3600 seconds, Free RAM: 2048000 bytes