**LAB ASSIGNMENT 4**

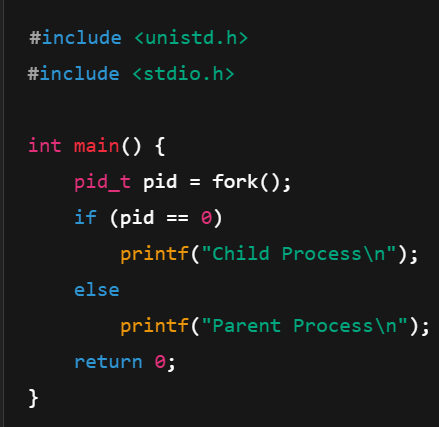
**OPERATING SYSTEMS**

1. **Process Management System Calls**

These system calls are related to creating, executing, and terminating processes in Linux.

1. **fork()**

* **Purpose**: Creates a **new child process**.
* **Returns**:
  + 0 in child
  + PID of child in parent
* **Syntax**: pid\_t fork();



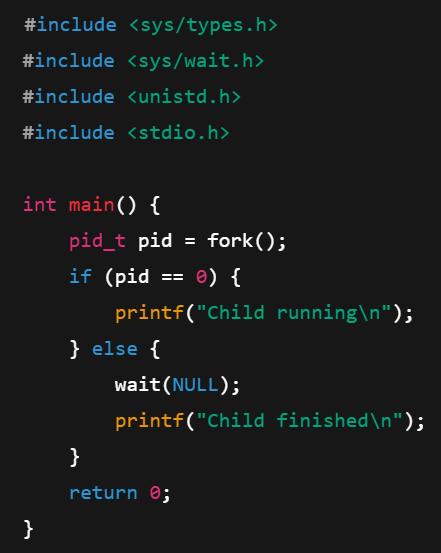
b) **exec()**

* **Purpose**: Replaces current process with a **new executable**.
* **Family**: execl(), execp(), execv() etc.
* **Syntax**: int execl(const char \*path, const char \*arg, ..., NULL);



c) **wait()**

* **Purpose**: Makes parent **wait** until child finishes.
* **Syntax**: pid\_t wait(int \*status);



d) **exit()**

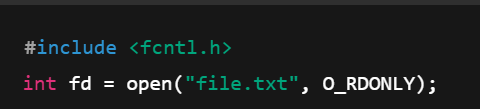
* **Purpose**: Terminates the process.
* **Syntax**: void exit(int status);



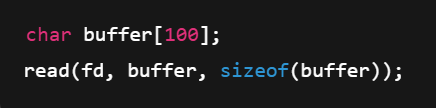
1. **File Management System Calls**

These are used to handle files in Linux.

* 1. **open()**
* **Purpose**: Opens a file for reading/writing.
* **Syntax**: int open(const char \*pathname, int flags);

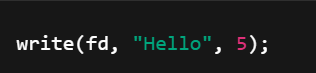


* 1. **read()**
* **Purpose**: Reads data from a file.
* **Syntax**: ssize\_t read(int fd, void \*buf, size\_t count);



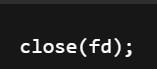
**c)write()**

* **Purpose**: Writes data to a file.
* **Syntax**: ssize\_t write(int fd, const void \*buf, size\_t count);



d) **close()**

* **Purpose**: Closes a file descriptor.
* **Syntax**: int close(int fd);



1. **Device Management System Calls**

These calls interact with **hardware devices** using device files.

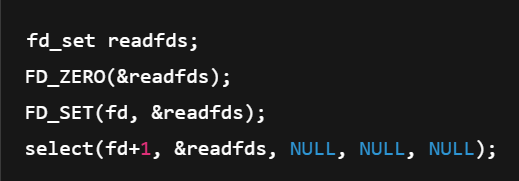
* 1. **read() and write()**

Used with device files like /dev/tty, /dev/sda, etc.

* 1. **ioctl()**
* **Purpose**: Device-specific input/output control operations.
* **Syntax**: int ioctl(int fd, unsigned long request, ...);



* 1. **select()**
* **Purpose**: Monitors multiple file descriptors to see if I/O is possible.
* **Syntax**: int select(int nfds, fd\_set \*readfds, ..., struct timeval \*timeout);



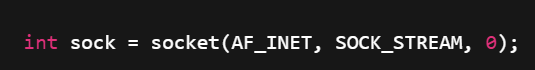
**4. Network Management System Calls**

Used for socket programming and network communication.

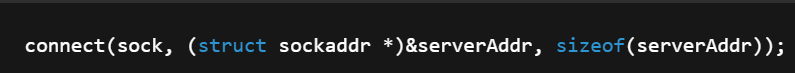
* 1. **socket()**

 **Purpose**: Creates a socket.

 **Syntax**: int socket(int domain, int type, int protocol);



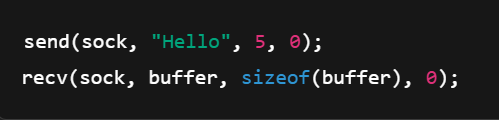
* 1. connect()
* **Purpose**: Connects to a remote server.
* **Syntax**: int connect(int sockfd, struct sockaddr \*addr, socklen\_t addrlen);



* 1. **send() and recv()**
* **Purpose**: Send/Receive data over sockets.
* **Syntax**:

i)ssize\_t send(int sockfd, const void \*buf, size\_t len, int flags);

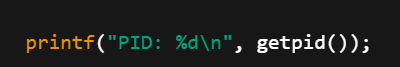
ii) ssize\_t recv(int sockfd, void \*buf, size\_t len, int flags);



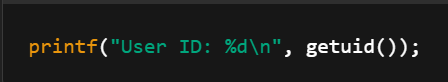
1. **System Information Management System Calls**

These calls retrieve system-related info like process ID, user ID, host info, etc.

* 1. **getpid()**
* **Returns**: Current process ID.
* **Syntax**: pid\_t getpid();

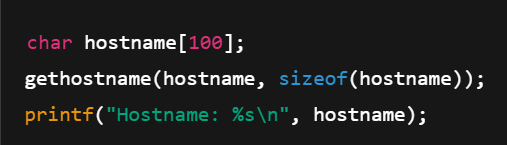


* 1. **getuid()**
* **Returns**: User ID of current process.
* **Syntax**: uid\_t getuid();



c) **gethostname()**

* **Purpose**: Gets the hostname of the machine.
* **Syntax**: int gethostname(char \*name, size\_t len);



* 1. **sysinfo()**
* **Purpose**: Gives info about memory, uptime, etc.
* **Syntax**: int sysinfo(struct sysinfo \*info);

