Experiment - 2

System calls of Linux operating system

7.

Aim:-

To study the various system calls of Linux operating system

Theory:-

1. Process management

→ fork() - creates a new process by duplicating the existing process

Syntax: pid_t fork(void);

→ execve() :- Loads and executes a new program in the current process.

Syntax: int execve(const char * filename, char* const argv[], char* const envp[]);

→ exit() :- Terminates the current process.

Syntax :- void exit(int status);

→ waitpid() :- waits for the termination of a specific child process.

Syntax :- pid_t waitpid (pid_t pid, int* status, int options);

→ getpid() :- returns the process ID of the calling process.

Syntax :- pid_t getpid(void);

2. File management

→ open() :- opens a file or device

Syntax :- int open (const char* pathname, int flags, mode_t mode);

→ read() :- Reads data from a file descriptor.

Syntax :- ssize_t read(int fd, void *buf, size_t count);

→ write() :- writes data to a file descriptor.

Syntax :- ssize_t write(int fd, const void *buf, size_t count);

→ close() :- closes a file descriptor.
Syntax :- int close (int fd);

3. Directory management.

→ mkdir() :- creates a new directory.
Syntax :- int mkdir (const char* pathname, mode_t mode);

→ rmdir() :- Removes an empty directory.
Syntax :- int rmdir (const char * pathname);

→ readdir() :- Read a directory stream.
Syntax :- struct dirent * readdir (DIR* dirp);

4. Interprocess communication.

→ pipe() :- creates an interprocess communication pipe.
syntax :- int pipe (int pipefd [2]);

→ shmget() :- Allocates a shared memory segment.
syntax :- int shmget (key_t key, size_t size, int shmflg);

→ shmat() :- Attaches a shared memory segment.
syntax :- void *shmat (int shmid, const void *shmaddr, int shmflg);

→ msgget() :- creates a message queue.
Syntax :- int msgget (key_t key, int msgflg);

5. Network communication.

→ socket() :- creates an endpoint for communication.
Syntax :- int socket (int domain, int type, int protocol);

→ bind() :- Binds a socket to a specific address and port.
syntax :- int bind (int sockfd, const struct sockaddr *addr, socklen_t addrlen);

→ listen() :- Marks a socket as passive, ready to accept incoming connections
Syntax :- int listen (int sockfd, int backlog);

Algorithm:-

1. Start the program.
2. Declare the variables pid, pid1 and pid2
3. call fork() system call to create process.
4. if pid == 0, exit
5. if pid != 0, get the process id using getpid().
6. print the process id.
7. stop the program.

Result:-

Successfully studied various system calls of Linux operating system.