Name: Jenil Kotadia

Roll No: 19/Div:2

Experiment no 5

Aim: Explore the system calls open, read, write, close, getuid, getgid, getegid, geteuid of Linux.

Objective: When a program in user mode requires access to RAM or a hardware resource, it must ask the kernel to provide access to that resource. This is done via a system call.

Theory:

getuid, geteuid - get user identity

getgid, getegid - get group identity

getuid() returns the real user ID of the calling process.

geteuid() returns the effective user ID of the calling process.

getgid() returns the real group ID of the calling process.

getegid() returns the effective group ID of the calling process.

All four functions shall always be successful and no return value is reserved to indicate an error.

Unix-like operating systems identify a user within the kernel by a value called a user identifier, often abbreviated to user ID or UID. The UID, along with the group identifier (GID) and other access control criteria, is used to determine which system resources a user can access.

Effective user ID

The effective UID (euid) of a process is used for most access checks. It is also used as the owner for files created by that process. The effective GID (egid) of a process also affects

access control and may also affect file creation, depending on the semantics of the specific kernel implementation in use and possibly the mount options used.

Open: Used to Open the file for reading, writing or both. Open() returns file descriptor 3 because when main process created, then fd 0, 1, 2 are already taken by stdin, stdout and stderr. So first unused file descriptor is 3 in file descriptor table.

int open(const char *pathname, int flags);

Parameters

Path : path to file which you want to use

use absolute path begin with "/", when you are not work in same directory of file.

Use relative path which is only file name with extension, when you are work in same directory of file.

flags : How you like to use

O_RDONLY: read only, O_WRONLY: write only,O_RDWR: read and write, O_CREAT: create file if it doesn't exist

Close: Tells the operating system you are done with a file descriptor and Close the file which pointed by fd.

int close (int fd);

Parameter

fd :file descriptor

Return

0 on success.

-1 on error.

read: Read data from one buffer to file descriptor, Read size bytes from the file specified by fd into the memory location.

size_t read (int fd, void* buf, size_t cnt);

Parameters

fd: file descripter

buf: buffer to read data from

cnt: length of buffer

Returns: How many bytes were actually read

return Number of bytes read on success

return 0 on reaching end of file

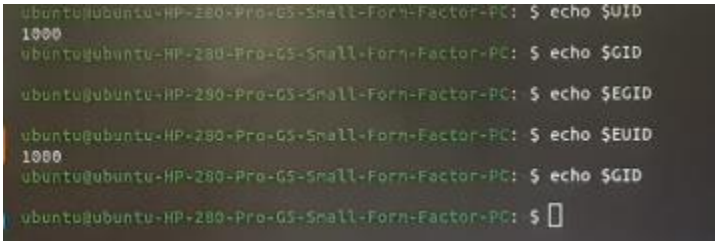
return -1 on error

return -1 on signal interrupt

Code:-

Output:-

Code:-



Conclusion:

Kernel mode: Operating system kernel has unrestricted access to hardware, executing privileged instructions.

User mode: Applications run with limited hardware access, requiring kernel for privileged operations.