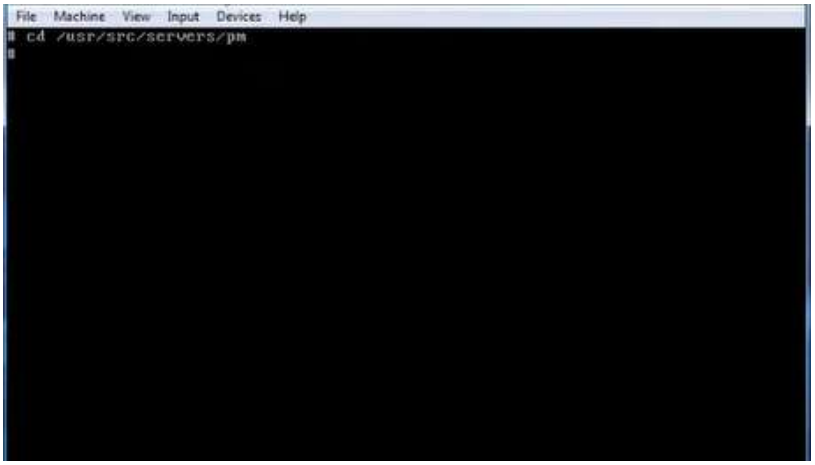
What is a System Call in MINIX 3?

A system call in MINIX 3 is a controlled interface that allows user-level programs to request services from the operating system's kernel. Since user applications cannot directly access hardware or critical system resources for safety and security reasons, they use system calls to perform tasks like file manipulation, process control, and communication with devices. In MINIX 3, system calls act as the bridge between user programs and the microkernel, enabling operations such as creating files, changing ownership (chown()), and managing processes while maintaining system stability and security.

Minix 3 Chown() System Call

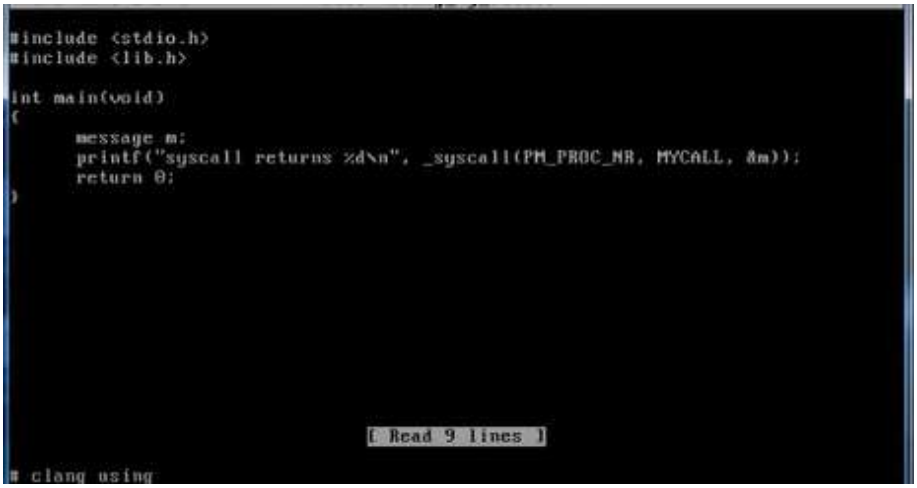
In Minix 3 OS, the chown() system call is used to change the ownership of files or directories. For example, to change the ownership of a file to the user **Hilin Yinager**, you first ensure the user exists and identify their user ID (UID) and group ID (GID). Using the chown() call, you provide the file path along with the UID and GID corresponding to Hilin Yinager. This updates the file's ownership, allowing the specified user and group to control access and permissions. The process typically involves executing the command with superuser privileges and verifying the ownership change using ls -l or similar commands in Minix.



```
/* This file contains the table used to map system call numbers onto the
 * routines that perform them.
 */

#define _TABLE

#include "pm.h"
#include <minix/callnr.h>
#include <signal.h>
#include "mproc.h"
#include "param.h"

int (*call_vec[])(void) = {
        no_sys,         /*  0 = unused  */
        do_exit,        /*  1 = exit    */
        do_fork,        /*  2 = fork    */
        no_sys,         /*  3 = read    */
        no_sys,         /*  4 = write   */
        no_sys,         /*  5 = open    */
        no_sys,         /*  6 = close   */
                                [ Read 131 lines ]
^G Get Help  ^O WriteOut  ^R Read File ^Y Prev Page ^K Cut Text  ^C Cur Pos
```

```
File  Machine  View  Input  Devices  Help
install ===> iso9660fs
    install  /sbin/isofs
install ===> mfs
    install  /sbin/mfs
install ===> pfs
    install  /usr/sbin/pfs
install ===> pm
    install  /usr/sbin/pm
install ===> procfs
    install  /sbin/procfs
install ===> rs
    install  /usr/sbin/rs
install ===> sched
    install  /usr/sbin/sched
install ===> vfs
    install  /usr/sbin/vfs
install ===> vm
    install  /usr/sbin/vm
install ===> devman
    install  /sbin/devman
install ===> hgfs
    install  /sbin/hgfs
install ===> vbfs
    install  /sbin/vbfs
#
```

```
        return 0;
}
/*===========================================
*             do_mycall
*===========================================*/
int do_mycall(void) {

    printf("wow! it really work.\n");
    return 0;

{




/*==========================
*       do_mysyscall
*==========================*/
 int do_mysyscall(void)
 {
    printf("This is my system call!\n");
                                [ Wrote 517 lines ]

# cd ..
# make:make install
```


#include <stdio.h>
#include <lib.h>
int main(void)
{
message m;
printf("syscall returns %d\n", _syscall(PM_PROC_NR, MYCALL, &m));
return 0;
}
[Read 9 lines]
clang using