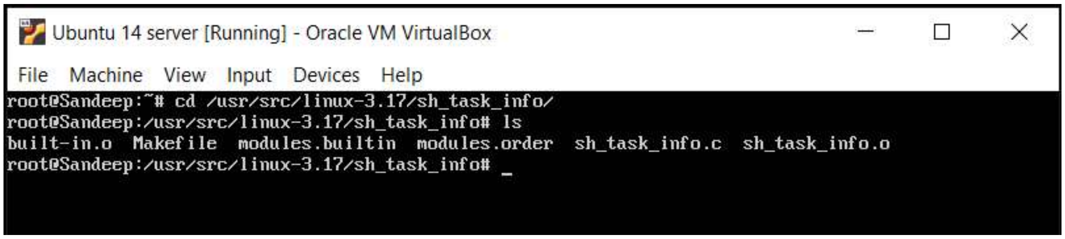
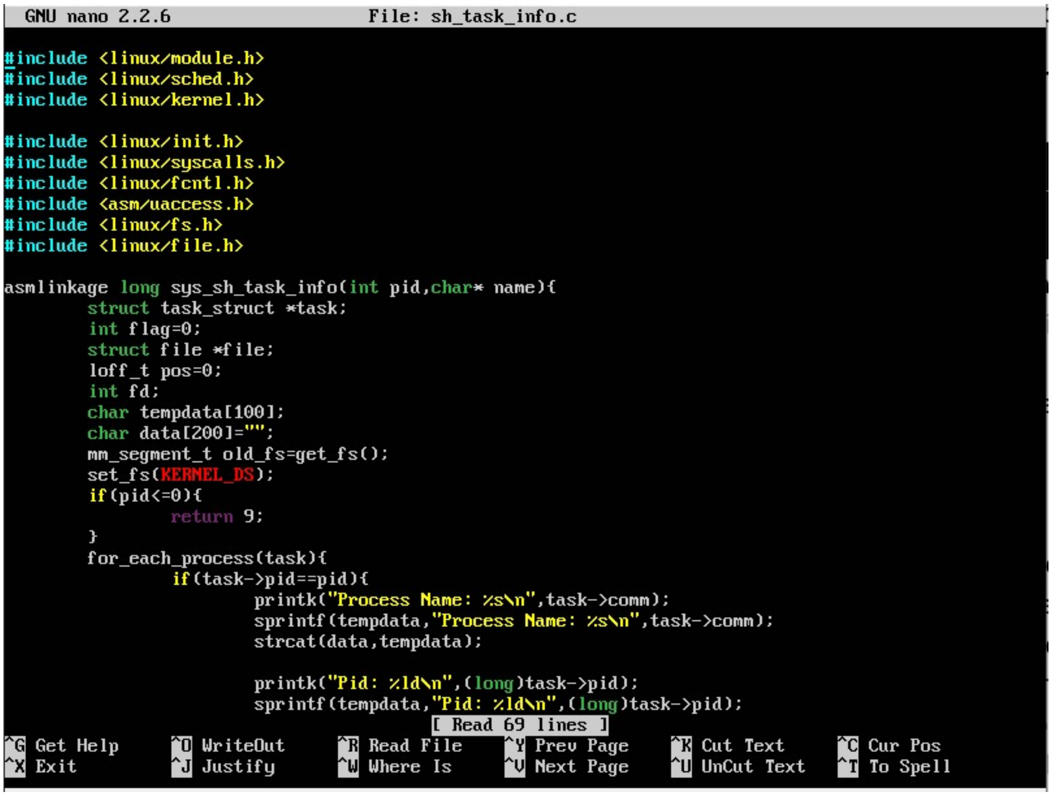
Write-up

1. Create a new folder named with the required name (i.e., sh_task_info) inside the kernel files.


Ubuntu 14 server [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help

```
root@Sandeep:~# cd /usr/src/linux-3.17/sh_task_info/
root@Sandeep:/usr/src/linux-3.17/sh_task_info# ls
built-in.o  Makefile  modules.builtin  modules.order  sh_task_info.c  sh_task_info.o
root@Sandeep:/usr/src/linux-3.17/sh_task_info# _
```

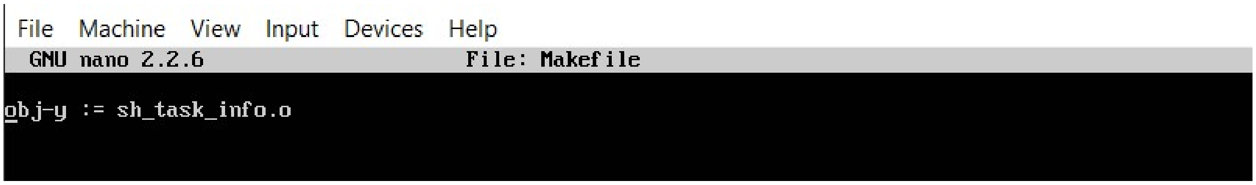
2. Create the C file which the required functionalities, i.e. to print the details of task struct into a file of a process which is the user provides PID. sh_task_info.c, Makefile



```
  GNU nano 2.2.6                    File: sh_task_info.c

#include <linux/module.h>
#include <linux/sched.h>
#include <linux/kernel.h>

#include <linux/init.h>
#include <linux/syscalls.h>
#include <linux/fcntl.h>
#include <asm/uaccess.h>
#include <linux/fs.h>
#include <linux/file.h>

asmlinkage long sys_sh_task_info(int pid,char* name){
        struct task_struct *task;
        int flag=0;
        struct file *file;
        loff_t pos=0;
        int fd;
        char tempdata[100];
        char data[200]="";
        mm_segment_t old_fs=get_fs();
        set_fs(KERNEL_DS);
        if(pid<=0){
                return 9;
        }
        for_each_process(task){
                if(task->pid==pid){
                        printk("Process Name: %s\n",task->comm);
                        sprintf(tempdata,"Process Name: %s\n",task->comm);
                        strcat(data,tempdata);

                        printk("Pid: %ld\n",(long)task->pid);
                        sprintf(tempdata,"Pid: %ld\n",(long)task->pid);
                                         [ Read 69 lines ]
```

3. In this file we need to add code to write to a file, all the details, I have used the sys_open() function to write to file.
4. Makefile contains only one line

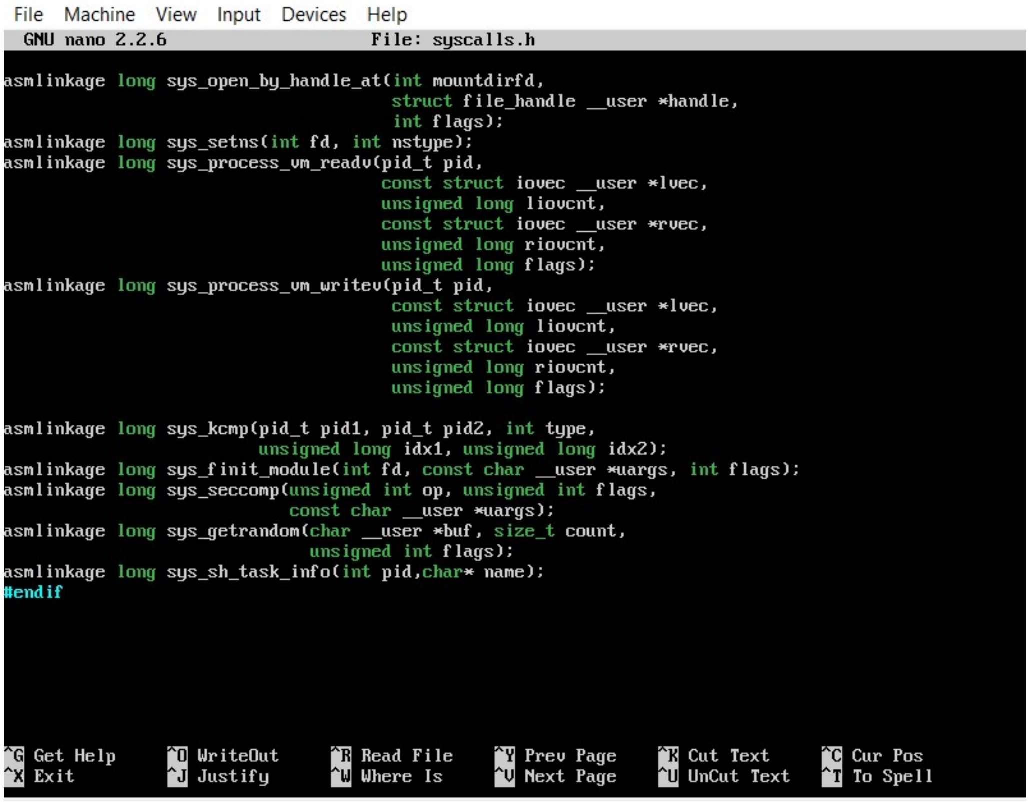
   Obj-y:=sh_task_info.o



```
 GNU nano 2.2.6                    File: Makefile

obj-y := sh_task_info.o
```

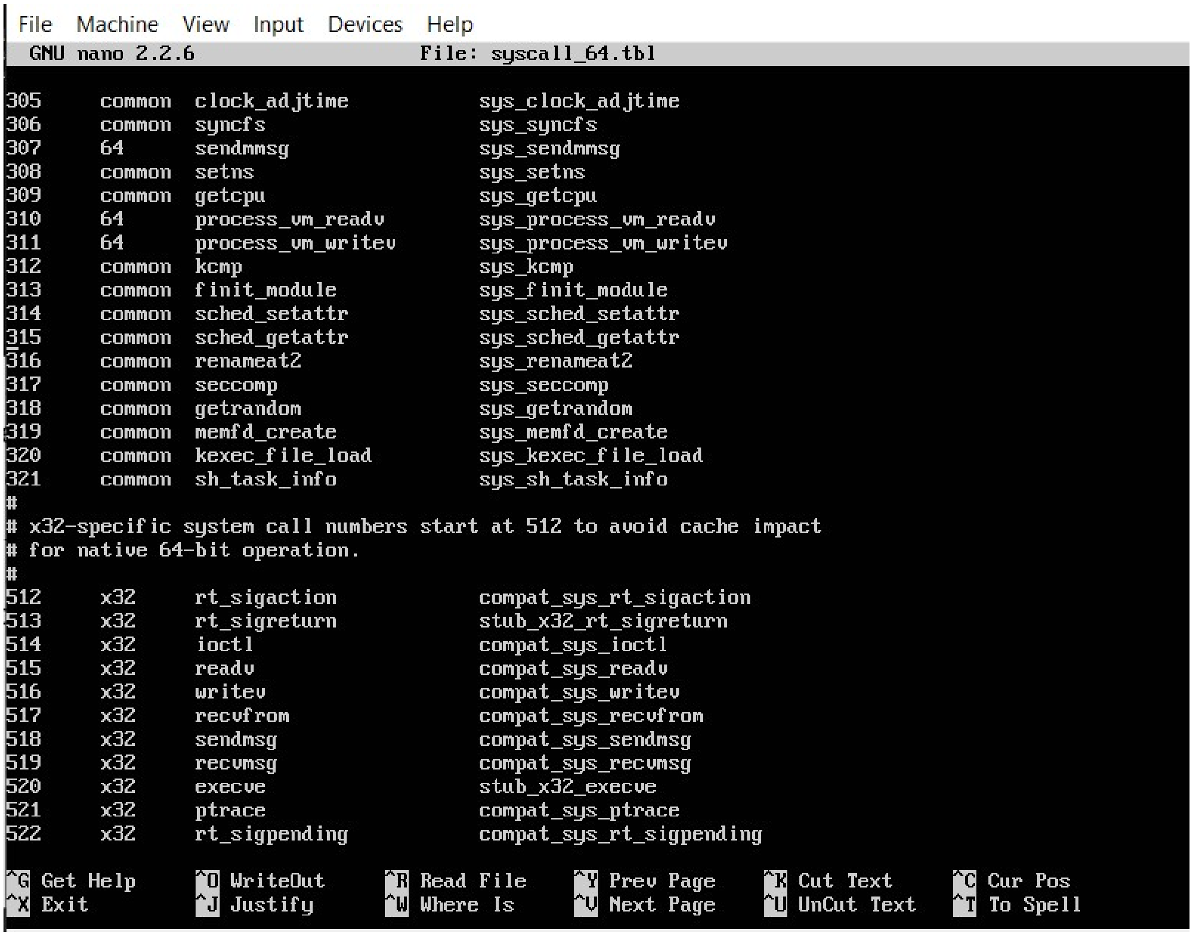
Sandeep Kumar singh
2018363

5. Sh_task_info.c contains only one function named as sys_sh_task_info() which takes two arguments PID and the custom name of file.
6. Then we need to change in the system call header file, its present in /include/Linux/syscalls.h, inside the kernel’s folder.
   We need to add the function’s definition inside it
   Asmlinkage long sys_sh_task_info(int PID, char * name);



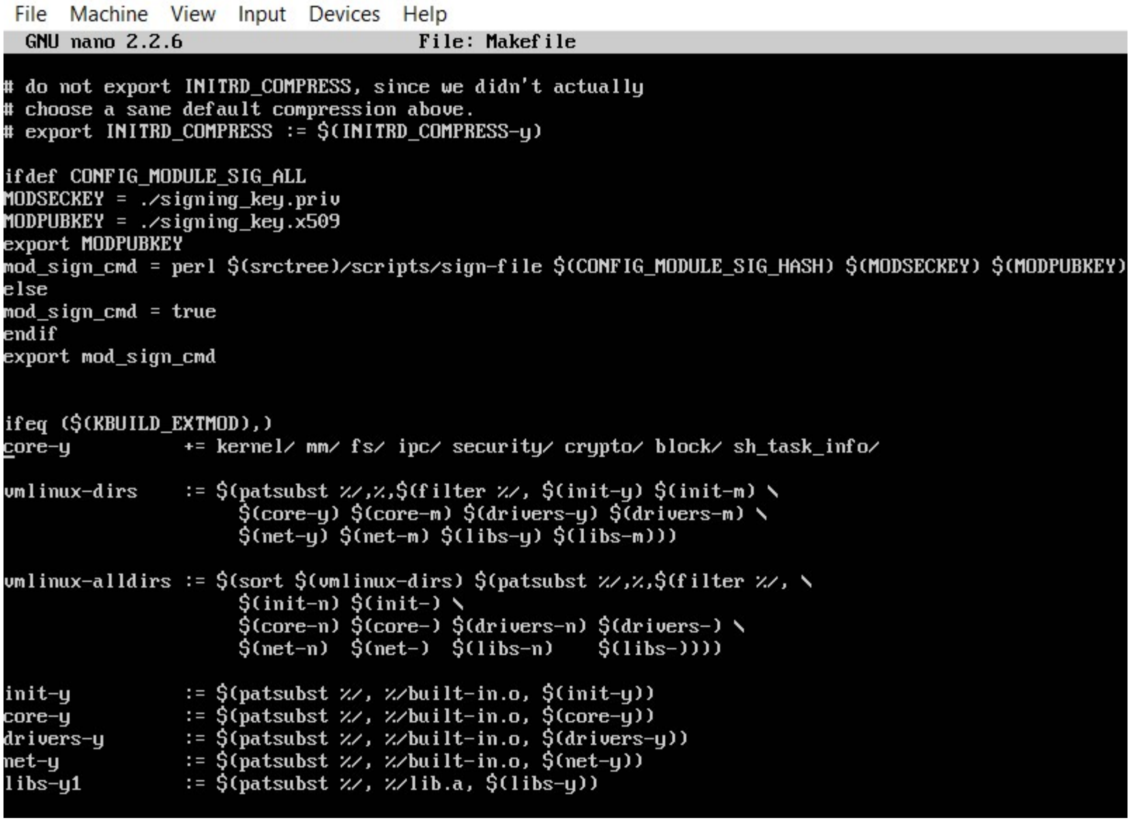
```
asmlinkage long sys_open_by_handle_at(int mountdirfd,
                                      struct file_handle __user *handle,
                                      int flags);
asmlinkage long sys_setns(int fd, int nstype);
asmlinkage long sys_process_vm_readv(pid_t pid,
                                     const struct iovec __user *lvec,
                                     unsigned long liovcnt,
                                     const struct iovec __user *rvec,
                                     unsigned long riovcnt,
                                     unsigned long flags);
asmlinkage long sys_process_vm_writev(pid_t pid,
                                      const struct iovec __user *lvec,
                                      unsigned long liovcnt,
                                      const struct iovec __user *rvec,
                                      unsigned long riovcnt,
                                      unsigned long flags);

asmlinkage long sys_kcmp(pid_t pid1, pid_t pid2, int type,
                         unsigned long idx1, unsigned long idx2);
asmlinkage long sys_finit_module(int fd, const char __user *uargs, int flags);
asmlinkage long sys_seccomp(unsigned int op, unsigned int flags,
                            const char __user *uargs);
asmlinkage long sys_getrandom(char __user *buf, size_t count,
                              unsigned int flags);
asmlinkage long sys_sh_task_info(int pid,char* name);
#endif
```

7. Now we need to change the system call table depending on the architecture(64 or 32 bit), this file is present in /arch/x86/syscalls/syscall_64.tbl, we need to add a line at the end of file
   321   common   sh_task_info   sys_sh_task_info



```
  GNU nano 2.2.6                   File: syscall_64.tbl

305     common  clock_adjtime           sys_clock_adjtime
306     common  syncfs                  sys_syncfs
307     64      sendmmsg                sys_sendmmsg
308     common  setns                   sys_setns
309     common  getcpu                  sys_getcpu
310     64      process_vm_readv        sys_process_vm_readv
311     64      process_vm_writev       sys_process_vm_writev
312     common  kcmp                    sys_kcmp
313     common  finit_module            sys_finit_module
314     common  sched_setattr           sys_sched_setattr
315     common  sched_getattr           sys_sched_getattr
316     common  renameat2               sys_renameat2
317     common  seccomp                 sys_seccomp
318     common  getrandom               sys_getrandom
319     common  memfd_create            sys_memfd_create
320     common  kexec_file_load         sys_kexec_file_load
321     common  sh_task_info            sys_sh_task_info
#
# x32-specific system call numbers start at 512 to avoid cache impact
# for native 64-bit operation.
#
512     x32     rt_sigaction            compat_sys_rt_sigaction
513     x32     rt_sigreturn            stub_x32_rt_sigreturn
514     x32     ioctl                   compat_sys_ioctl
515     x32     readv                   compat_sys_readv
516     x32     writev                  compat_sys_writev
517     x32     recvfrom                compat_sys_recvfrom
518     x32     sendmsg                 compat_sys_sendmsg
519     x32     recvmsg                 compat_sys_recvmsg
520     x32     execve                  stub_x32_execve
521     x32     ptrace                  compat_sys_ptrace
522     x32     rt_sigpending           compat_sys_rt_sigpending

^G Get Help     ^O WriteOut     ^R Read File    ^Y Prev Page    ^K Cut Text     ^C Cur Pos
^X Exit         ^J Justify      ^W Where Is     ^V Next Page    ^U UnCut Text   ^T To Spell
```

8. Now Makefile of kernel needs to be changed a little bit. We need to give the address of our system call files to it for compilation. For that, we need to append its address at the end of a line starting with
“core-y += kernel/ certs/ …..”



```
  GNU nano 2.2.6                    File: Makefile

# do not export INITRD_COMPRESS, since we didn't actually
# choose a sane default compression above.
# export INITRD_COMPRESS := $(INITRD_COMPRESS-y)

ifdef CONFIG_MODULE_SIG_ALL
MODSECKEY = ./signing_key.priv
MODPUBKEY = ./signing_key.x509
export MODPUBKEY
mod_sign_cmd = perl $(srctree)/scripts/sign-file $(CONFIG_MODULE_SIG_HASH) $(MODSECKEY) $(MODPUBKEY)
else
mod_sign_cmd = true
endif
export mod_sign_cmd


ifeq ($(KBUILD_EXTMOD),)
core-y          += kernel/ mm/ fs/ ipc/ security/ crypto/ block/ sh_task_info/

vmlinux-dirs    := $(patsubst %/,%,$(filter %/, $(init-y) $(init-m) \
                     $(core-y) $(core-m) $(drivers-y) $(drivers-m) \
                     $(net-y) $(net-m) $(libs-y) $(libs-m)))

vmlinux-alldirs := $(sort $(vmlinux-dirs) $(patsubst %/,%,$(filter %/, \
                     $(init-n) $(init-) \
                     $(core-n) $(core-) $(drivers-n) $(drivers-) \
                     $(net-n)  $(net-)  $(libs-n)    $(libs-))))

init-y          := $(patsubst %/, %/built-in.o, $(init-y))
core-y          := $(patsubst %/, %/built-in.o, $(core-y))
drivers-y       := $(patsubst %/, %/built-in.o, $(drivers-y))
net-y           := $(patsubst %/, %/built-in.o, $(net-y))
libs-y1         := $(patsubst %/, %/lib.a, $(libs-y))
```

9. Kernel Compilation:
    a. First, we need to have a .config inside the kernel's folder, for that either we can run "make oldconfig" or can copy the current config file from /boot directory
    b. Then we need to run the "make menuconfig" if we want to change any details of the kernel like appending the name to the kernel's name.
    c. Now we need to compile the kernel

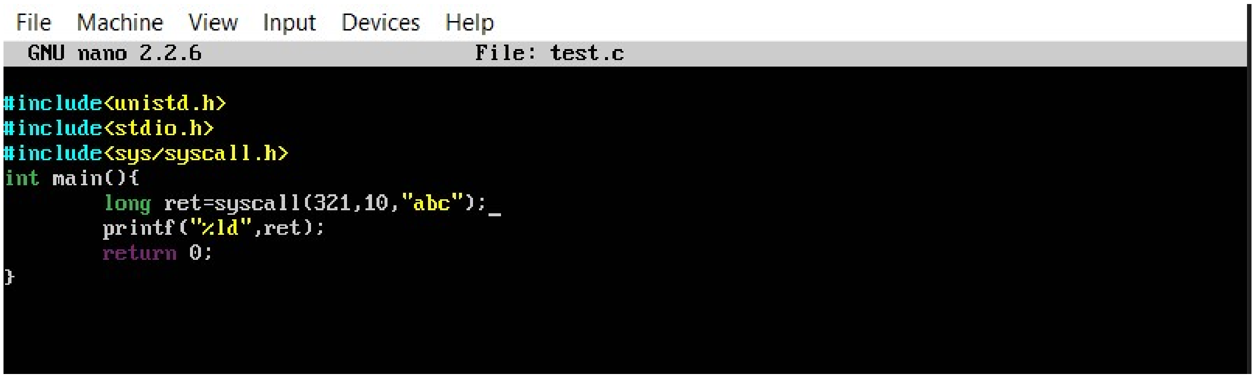
sudo make -j 4 && sudo make modules_install -j 4 && sudo make install

sudo update-grub

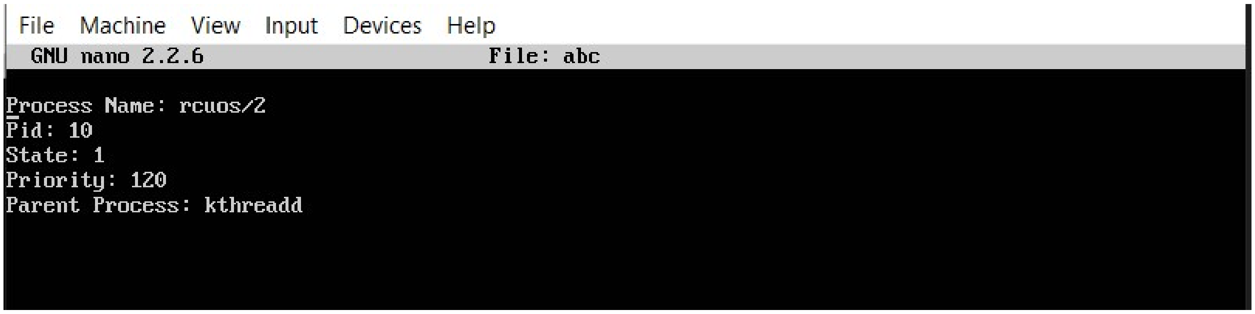
sudo reboot

then, boot into desired kernel by changing from advanced options in grub menu

10. Check the loaded kernel with "uname -a" command
11. Testing of System call:
    a. Create a test.c file and include the necessary header files
    b. Call the system call using the system call number and the PID and address of the desired file



```
  GNU nano 2.2.6                     File: test.c

#include<unistd.h>
#include<stdio.h>
#include<sys/syscall.h>
int main(){
        long ret=syscall(321,10,"abc");_
        printf("%ld",ret);
        return 0;
}
```



```
  GNU nano 2.2.6                     File: abc

Process Name: rcuos/2
Pid: 10
State: 1
Priority: 120
Parent Process: kthreadd
```

Error:

It returns 9 either when pid is less than 0 and or it is not currently assigned to anyone.

It return error when the file can’t be opened or written to.