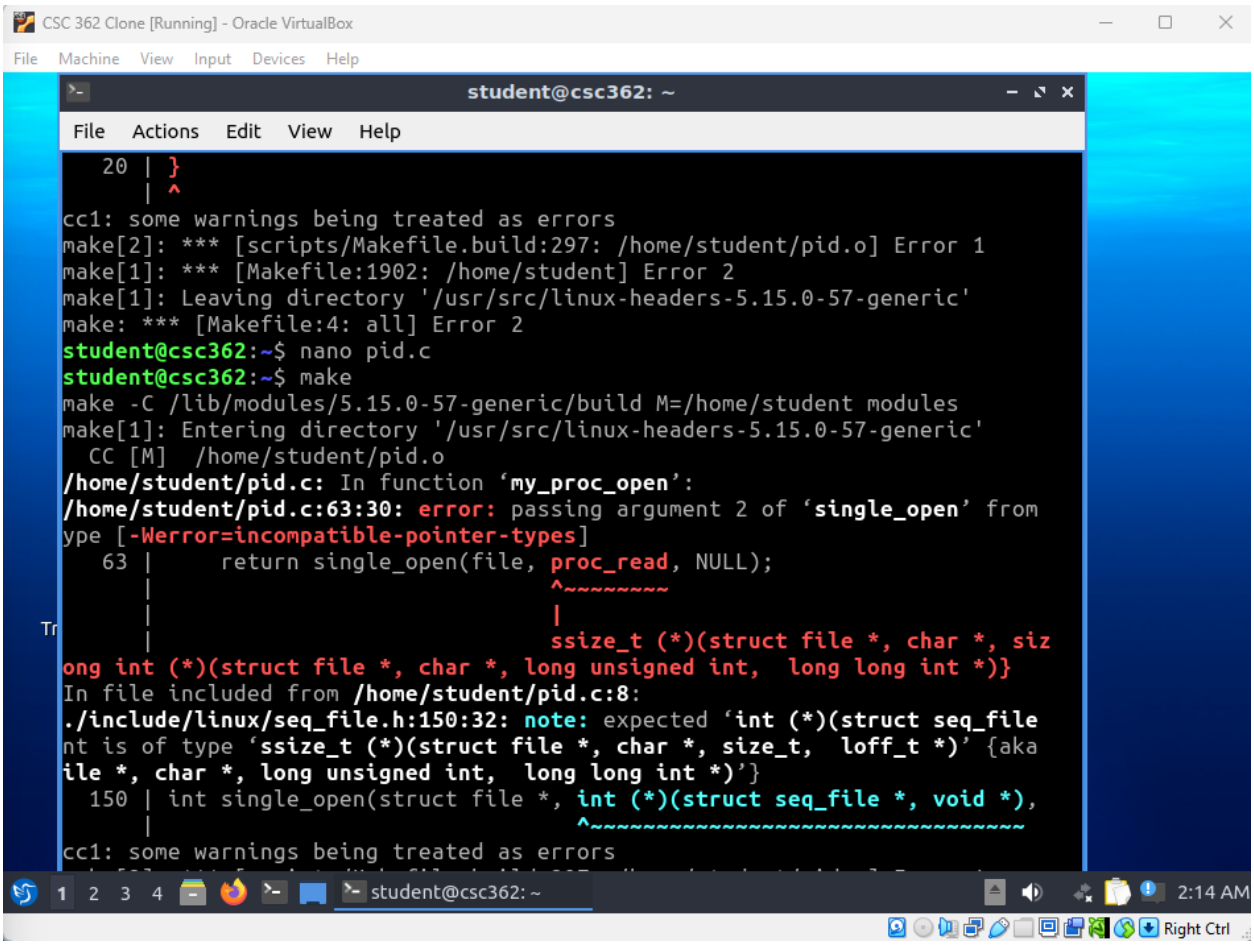
This project involved creating a Linux Kernel Module that allows users to read and write process information through the /proc file system. The module lets user-space applications write a process identifier (PID) to /proc/pid and read details about the process like its command, PID, and state.

During the development, I faced several challenges such as function pointer mismatches and compilation errors. These were caused by incorrect function setups and were fixed by adjusting the order of function declarations and ensuring proper function prototypes.

Error handling and security were major focuses to prevent system crashes and security risks. I paid close attention to managing memory safely and handling errors carefully to make sure the module did not cause any harm to the system or expose sensitive information.

Another challenge was making the kernel module easy for users to interact with. Kernel modules aren't usually built for direct interaction with users, but our user-space scripts bridged this gap effectively.

```
   7598 ?        00:00:00 snap
   8415 ?        00:00:00 gvfsd-computer
   9025 ?        00:00:00 gvfsd-metadata
  10888 ?        00:00:14 qterminal
  10909 ?        00:00:03 kworker/1:2-cgroup_destroy
  10922 pts/0    00:00:00 bash
  11507 ?        00:00:00 kworker/u4:0-ext4-rsv-conversion
  11754 ?        00:00:03 kworker/0:0-cgroup_destroy
  12349 ?        00:00:00 kworker/1:0-events
  12692 ?        00:00:00 kworker/u4:3-flush-8:0
  13046 ?        00:00:00 kworker/0:2-mm_percpu_wq
  13048 ?        00:00:00 kworker/0:1-events
  13062 ?        00:00:00 kworker/u4:1-events_unbound
  13063 ?        00:00:00 kworker/u4:2-events_unbound
  13064 ?        00:00:00 anacron
  13077 ?        00:00:00 kworker/1:1-cgroup_destroy
  13078 ?        00:00:00 kworker/1:3-events
  13088 ?        00:00:00 cupsd
  13089 ?        00:00:00 kworker/1:4-events
  13090 ?        00:00:00 cups-browsed
  13104 ?        00:00:00 kworker/0:3-events
  13167 pts/0    00:00:00 ps
student@csc362:~$ echo "11507"| sudo tee /proc/pid
11507
student@csc362:~$ cat /proc/pid
command = [kworker/u4:0] pid = [11507] state = [1026]
student@csc362:~$ nano pid.c
student@csc362:~$
```


CSC 362 Clone [Running] - Oracle VirtualBox
File Machine View Input Devices Help
student@csc362: ~
File Actions Edit View Help
20 | }
| ^
cc1: some warnings being treated as errors
make[2]: *** [scripts/Makefile.build:297: /home/student/pid.o] Error 1
make[1]: *** [Makefile:1902: /home/student] Error 2
make[1]: Leaving directory '/usr/src/linux-headers-5.15.0-57-generic'
make: *** [Makefile:4: all] Error 2
student@csc362:~$ nano pid.c
student@csc362:~$ make
make -C /lib/modules/5.15.0-57-generic/build M=/home/student modules
make[1]: Entering directory '/usr/src/linux-headers-5.15.0-57-generic'
CC [M] /home/student/pid.o
/home/student/pid.c: In function ‘my_proc_open’:
/home/student/pid.c:63:30: error: passing argument 2 of ‘single_open’ from
ype [-Werror=incompatible-pointer-types]
63 | return single_open(file, proc_read, NULL);
| ^~~~~~~~~
| |
| ssize_t (*)(struct file *, char *, siz
ong int (*)(struct file *, char *, long unsigned int, long long int *)}
In file included from /home/student/pid.c:8:
./include/linux/seq_file.h:150:32: note: expected ‘int (*)(struct seq_file
nt is of type ‘ssize_t (*)(struct file *, char *, size_t, loff_t *)’ {aka
ile *, char *, long unsigned int, long long int *)’}
150 | int single_open(struct file *, int (*)(struct seq_file *, void *),
| ^~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
cc1: some warnings being treated as errors
1 2 3 4
student@csc362: ~
2:14 AM
Right Ctrl