Assignment - 1

System Call for Linux written in C, compiling Linux source code

Part 1: Compiling the source code

The linux 3.16.74 source code is downloaded and it's extracted.

- cd to that directory

- do 'cp /boot/config-$(uname -r) .config' to copy the kernel's .config file then followed by 'make menuconfig'

- use 'sudo make -j 4 && sudo make modules_install -j 4 && sudo make install -j 4'.The kernel and modules are compiled.

- do 'update-initramfs -c -k 3.13.0' to update the kernel, and 'update-grub' to add it to the grub.

Part 2: Writing a System Call

- In the linux-3.16.74 dir, make a new dir(sh_task_info) and add the system call files there.

- Make a new file sh_task_info.c

- Write_to_file function takes file* and char * as arguments and uses write() to write data in it.

- In sys_sh_task_info(), a task_struct pointer task and struct file pointer file are created. For the process, it's attributes are printed on the console and simultaneously written to the file.

- A Makefile for compiling the above.

```
obj-y := sh_task_info.o
```

- This new directory has to be added to the Linux-3.16.74's Makefile by changing the line 'core -y += kernel/ mm/ fs/ ipc/ security/ crypto/ block/' to 'core -y += kernel/ mm/ fs/ ipc/ security/ crypto/ block/ sh_task_info/'.

- The pid for the new system call has to be added, in the arch/x86/syscalls/syscall_32.tbl. I added it at 355.

- Then, the function was included in the syscalls.h in include/ linux/ .

- It was recompiled again with the three parallel commands by 'sudo make -j 4 && sudo make modules_install -j 4 && sudo make install -j 4'.

- The system is restarted by ‘sudo reboot’ .

Part 3: Testing it with sample code

- The file test.c is the sample test code, compares the process IDs by using syscall. If the input process ID and the updated pid match, and there is no error in the given pid and the filename sh_task_info() is executed correctly.

- It should be run with 'gcc test.c '

- The executable is run with './a.out'

- If it executed successfully, we can check the log via 'dmesg'. The attributes corresponding to pid==1 are printed on the console.

- The process, it's process ID, process state are printed.

- Alternatively, we can check the log with 'cat data.txt'.

Part 4: Errors Handled

- 1. If the user doesn't enters an invalid pid such as a char or float

- 2. If the user enters pid <=0 , the function returns the errno 22 EINVAL invalid argument.

- 3. If the sys_open() for creating a file if it doesn't exist, with write access returns an int less than 0, the function returns the errno 21 EISDIR is a directory.

If there is no error "No space left on the device " in the part 2 step 9, then do the following : -

-The linux 3.16.74 source code is downloaded and it's extracted.

- cd to that directory

- do 'cp /boot/config-$(uname -r) .config' to copy the kernel's .config file then followed by 'make menuconfig'

- In the linux-3.16.74 dir, make a new dir(sh_task_info) and add the system call files there.

- Make a new file sh_task_info.c

- Write_to_file function takes file* and char * as arguments and uses write() to write data in it.

- In sys_sh_task_info(), a task_struct pointer task and struct file pointer file are created. For the process, it's attributes are printed on the console and simultaneously written to the file.

- A Makefile for compiling the above.

```
obj-y := sh_task_info.o
```

- This new directory has to be added to the Linux-3.16.74's Makefile by changing the line 'core -y += kernel/ mm/ fs/ ipc/ security/ crypto/ block/' to 'core -y += kernel/ mm/ fs/ ipc/ security/ crypto/ block/ sh_task_info/'.

- The pid for the new system call has to be added, in the arch/x86/syscalls/syscall_32.tbl. I added it at 355.

- Then, the function was included in the syscalls.h in include/ linux/ .

- It was recompiled again with the three parallel commands by 'sudo make -j 4 && sudo make modules_install -j 4 && sudo make install -j 4'.

- The system is restarted by ‘sudo reboot’ .