OPERATING SYSTEMS

CS 3003

IMPLEMENTATION OF A NEW SYSTEM CALL

Submitted By:

K.A.SIVA VARDHAN- B160333CS

P.SATYANARAYANA - B160340CS

B.MAHESWARARAO - B160349CS

R.VAMSI KRISHNA- B160109CS

ABSTRACT

This system call helps us to know the FILE NAME and PID of the runnable, terminated, sleeping and blocked processes along with total no. of such processes.

Along with this ,the user also get to know the information of the high priority, normal priority,and static priority of the processes.

Directory for new system call:

1. Create a new directory, say “hello” in the linux kernel version (i.e linux-4.18.7) and change to this directory.
2. Create a new file “hello.c” and write the source code for the system call in this file.
3. Create a Makefile in the same directory which contains the following code

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

This ensures that hello.c is compiled and included in the kernel source code.

Linking system call with kernel:

Now add the “hello” directory to the kernel Makefile.

The above link specifies the compiler that source files of our new system call can be found in the /hello directory.

Altering syscall_64.tbl:

To find out where this file is present, we can use the find command on the terminal from the linux-4.18.7 directory.

find -name syscall_64.tbl ### Should show the file's location In kernel-4.18.7, it is present in /arch/x86/entry/syscalls/syscall_64.tbl

Now, edit the file to include the new system call number and it's entry point.

Altering syscalls.h:

We use the find command to look for where the syscalls.h file is present.

find -name syscalls.h

In kernel-4.18.7, this is present in /include/linux/syscalls.h.

Add the following line at the end of the file(before the #endif) as shown

asmlinkage long sys_hello(void);

Recompile and Reboot:

To integrate the system call and to be able to actually use it, we will need to recompile the kernel by using following commands:

1)sudo make -j 4
2) sudo make modules_install -j 4
3) sudo make install -j 4
4)sudo update-grub

Once this is done, we restart the system.

Testing the system call:

To test the system call, we write a simple `test.c' function. Compile and execute this program. If it runs successfully, then, it should give the corresponding prompt and we can use `dmesg' to check the kernel log.

Functionality:

This system call helps us to know the FILENAME and PID of the runnable, terminated,sleeping and blocked processes along with their total count.

Along with this ,the user also get to know the information of the high priority ,normal priority,and static priority of the processes along with their count.