Eric Samuel Huddleston

3/7/2021

CS 4853 Section 001

Assignment 2: Adding A /proc Interface

First I start by creating a new directory for this assignment and a “test” directory, which is where we will code rdstp() so as to see the difference between the value for jiffies and the rdstp() value.

```
student@ubuntu:/home/a2-hpg103
```

```
drwxr-sr-x    3 student  student      4096 2021-02-11 16:06 test
```

Then I move the two starter files provided at the beginning of the assignment, jiffies.c and the Makefile, into my current working directory a2-hpg103.

```
-rw-r--r--    1 root     student      1493 2021-03-04 20:11 jiffies.c
-rw-r--r--    1 student  student       157 2021-03-04 20:14 Makefile
```

Then I add the required in the “TODO” parts of jiffies.c as shown below:

```
        //TODO: write the value of the jiffies global into the specified
        //buffer
        ret = sprintf(buffer, "jiffies=%lu\n", jiffies);
}
```

```
    //TODO:
    //utilize proc_dir_entry pointer to setup the interface/filelength;
    //utilize the THIS_MODULE object, which is generated automatically

    proc_file->read_proc = procfile_read;
    proc_file->owner = THIS_MODULE;
    proc_file->mode = S_IFREG | S_IRUGO;
    proc_file->uid = 0;
    proc_file->gid = 0;
    proc_file->size = 100;
```

I also modified existing parts of the code as shown below:

```
proc_file = create_proc_entry(procfs_name, 0644, NULL);
```

Additionally for the module cleanup function I added a print statement to make it easier for the user to see when the interface was removed.

```
void cleanup_module()
{
        remove_proc_entry(procfs_name, &proc_root);
        printk(KERN_INFO "/proc/%s removed\n", procfs_name);
}
```

Now I use the make command to compile jiffies.c as shown below:

```
student@ubuntu:/home/a2-hpg103 $ make
make -C /lib/modules/2.6.11/build M=/home/a2-hpg103 modules
make[1]: Entering directory `/home/student/linux-2.6.11'
  CC [M]  /home/a2-hpg103/jiffies.o
  Building modules, stage 2.
  MODPOST
  CC      /home/a2-hpg103/jiffies.mod.o
  LD [M]  /home/a2-hpg103/jiffies.ko
make[1]: Leaving directory `/home/student/linux-2.6.11'
```

Now I insert this module (jiffies.ko) as shown below:

```
student@ubuntu:/home/a2-hpg103 $ sudo insmod jiffies.ko
```

Then I use dmesg to check that the insert_module() function worked properly, as shown below:

```
student@ubuntu:/home/a2-hpg103 $ dmesg
```

```
NET: Registered protocol family 1
NET: Registered protocol family 10
IPv6 over IPv4 tunneling driver
NET: Registered protocol family 17
ACPI wakeup devices:

ACPI: (supports S0 S1 S4 S5)
EXT3-fs: INFO: recovery required on readonly filesystem.
EXT3-fs: write access will be enabled during recovery.
kjournald starting.  Commit interval 5 seconds
EXT3-fs: recovery complete.
EXT3-fs: mounted filesystem with ordered data mode.
VFS: Mounted root (ext3 filesystem) readonly.
Freeing unused kernel memory: 248k freed
Adding 380480k swap on /dev/hda5.  Priority:-1 extents:1
EXT3 FS on hda1, internal journal
e1000: eth0: e1000_watchdog: NIC Link is Up 1000 Mbps Full Duplex
Losing some ticks... checking if CPU frequency changed.
eth0: no IPv6 routers present
warning: many lost ticks.
Your time source seems to be instable or some driver is hogging interupts
rip __do_softirq+0x54/0xf0
jiffies: module license 'unspecified' taints kernel.
/proc/jiffies created
```

Now I can simply use dmesg -C to clear the ring buffer to get rid of the above warnings, which is shown below:

```
student@ubuntu:/home/a2-hpg103 $ sudo dmesg -c
```

Now if we ls in /proc/jiffies we can see the jiffies interface:

```
student@ubuntu:/home/a2-hpg103 $ ls /proc/jiffies
-r--r--r--    1 root     root          100 2021-03-04 20:56 /proc/jiffies
```

Here we can see that the call back function is being properly reached:

```
student@ubuntu:/home/a2-hpg103 $ cat /proc/jiffies
jiffies
student@ubuntu:/home/a2-hpg103 $ dmesg
procfile_read (/proc/jiffies) called
procfile_read (/proc/jiffies) called
procfile_read (/proc/jiffies) called
```

Now we can remove the module as shown below:

```
student@ubuntu:/home/a2-hpg103 $ sudo rmmod jiffies
student@ubuntu:/home/a2-hpg103 $ ls /proc/jiffies
ls: /proc/jiffies: No such file or directory
```

I can reinsert the module using this command:

```
student@ubuntu:/home/a2-hpg103 $ sudo insmod jiffies.ko
student@ubuntu:/home/a2-hpg103 $
```

Now, within the /a2-hpg103/test subdirectory I created another Makefile and a script to test my module and which will print the “jiffies” value out onto the screen, as shown below:

```
student@ubuntu:/home/a2-hpg103/test $ ls -l
total 20
drwxr-sr-x    3 student  student      4096 2021-03-04 22:03 .
drwxr-sr-x    4 student  student      4096 2021-03-04 20:50 ..
-rw-r--r--    1 student  student       157 2021-03-04 22:00 Makefile
-rwxr-xr-x    1 student  student       409 2021-03-04 22:03 testme.sh
drwxr-sr-x    2 student  student      4096 2021-02-11 16:06 .tmp_versions
```

Makefile:

```
obj-m += ../jiffies.o

all:
	make -C /lib/modules/$(shell uname -r)/build M=$(PWD) modules

clean:
	make -C /lib/modules/$(shell uname -r)/build M=$(PWD) clean
~
~
```

Bash script:

```
#!/bin/bash
echo \*\*\* Making the module...
make
echi \*\*\* Attempting to remove the old module, if it is present...
sudo rmmod -f jiffies
echo \*\*\* installing the module...
sudo insmod ../jiffies.ko
echo \*\*\* Printing file contents...
if [ ! -e "/proc/jiffies" ]
then
   echo \*\*\* "ERROR; /proc/jiffies does not exist, something has gone wrong..
"
else
   cat /proc/jiffies
   echo \*\*\ Success!
fi
```

Now I can simply test the module with the following command, as shown below:

```
student@ubuntu:/home/a2-hpg103/test $ ./testme.sh
*** Making the module...
make -C /lib/modules/2.6.11/build M=/home/a2-hpg103/test modules
make[1]: Entering directory `/home/student/linux-2.6.11'
  Building modules, stage 2.
  MODPOST
make[1]: Leaving directory `/home/student/linux-2.6.11'
*** Attempting to remove the old module, if it is present...
*** Installing the module...
*** Printing file contents...
jiffies=4312759870
** Success!
```

Now I coded the rdtscp file since it is not available in this version of Linux, and the code is as shown below:

```
#include <stdio.h>
#include <stdlib.h>

inline unsigned long long rdtscp() {
        unsigned int lo, hi;
        asm volatile (
                "rdtsc"
                : "=a"(lo), "=d"(hi) /* outputs */
                : "a"(0)             /* inputs  */
                : "%ebx", "%ecx");     /* clobbers*/
        unsigned long long retval = ((unsigned long long)lo) | (((unsigned long
long)hi) << 32);
        return retval;
}

int main(int argc, char ** argv) {
        unsigned long long timestamp;

        timestamp = rdtscp();

        printf("rdtscp=%llu\n", timestamp);

        return EXIT_SUCCESS;
}
```

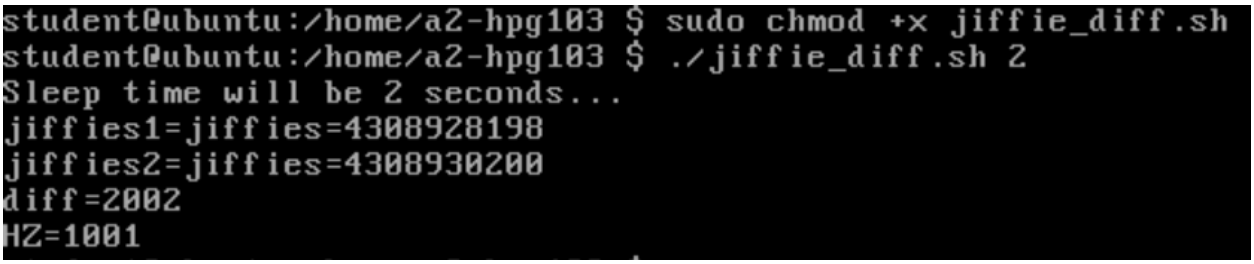
I can also compile and run rdtscp as shown below:

```
student@ubuntu:/home/a2-hpg103 $ gcc rdtscp.c -o rdtscp
```

```
student@ubuntu:/home/a2-hpg103 $ ./rdtscp
rdtscp=375739249695552
```

Now I create the two required user applications, as bash scripts, the first of these being jiffies-diff.sh, as shown below:

```
#!/bin/bash
SLEEP_TIME="$1"
echo "Sleep time will be $1 seconds..."
jiff1=$(cat /proc/jiffies)
echo jiffies1=$jiff1
sleep ${SLEEP_TIME}
jiff2=$(cat /proc/jiffies)
echo jiffies2=$jiff2
diff=$((jiff2-jiff1))
echo "diff=$diff"
HZ=$((diff/SLEEP_TIME))
echo "HZ=$HZ"
```



Here we can see that the script was successful in calculating the difference between the two jiffies values, and the HZ value.

4. **Finally, briefly explain what could be wrong with using the simple performance evaluation technique utilized in this assignment, particularly when requirements dictate differentiation of extremely small differences in runtimes between programs.**

This evaluation technique is not completely accurate because in the development of the scripts for testing, we use the sleep(n) function. Because the accuracy of the jiffies and HZ values increases as the sleep time increase, our measurement of the module and the code for the module cannot be accurate since when using this technique (providing the sleep time on the command line), it is difficult to provide sleep times through the command line that get closer to infinity over time.