

Trenton Bergdahl - 11768322

11/15/2023

Lab 4 Report

The implementation of my module was to refactor proc read and write. Proc read was refactored to display all process entries in the process list to the `/proc/kmlab/status` file. Proc write is designed to accept a string pid and convert it to an integer, then use the pid to get the CPU time of the process, and store both in a list entry. There is also a timer configured that will schedule a function called `work_function` that will traverse the list and update the list based off process completion. If a process isn't complete the work function will update its CPU time and if it is complete it will remove the process from the list. Screenshots of the outputs are provided below.

Output with multiple processes:

A terminal window with a dark background and light-colored text. The text shows the execution of a module and two processes. It starts with 'Skipping BTF generation for /home/trenton/CPT5360/Lab4/' and 'make[1]: Leaving directory '/usr/src/linux-headers-5.15''. Then it runs 'gcc -o userapp userapp.c', 'sudo insmod kmlab.ko', and starts two background processes: './userapp 10 &' and './userapp 15 &'. After a 'sleep 6', it runs 'cat /proc/kmlab/status' and shows two entries: '13084: CPU Time: 6844000000' and '13082: CPU Time: 7876000000'. Then it shows '[1] + Done' and './userapp 10'. Next, it runs 'cat /proc/kmlab/status' again and shows only one entry: '13084: CPU Time: 11932000000'. Then it shows '[2] + Done' and './userapp 15'. Finally, it runs 'cat /proc/kmlab/status' and shows an empty list, indicated by a single '\$' on a new line.

```
Skipping BTF generation for /home/trenton/CPT5360/Lab4/
make[1]: Leaving directory '/usr/src/linux-headers-5.15'
gcc -o userapp userapp.c
$ sudo insmod kmlab.ko
$ ./userapp 10 &
$ ./userapp 15 &
$ sleep 6
$ cat /proc/kmlab/status
13084: CPU Time: 6844000000
13082: CPU Time: 7876000000
[1] + Done                               ./userapp 10
$ cat /proc/kmlab/status
13084: CPU Time: 11932000000
[2] + Done                               ./userapp 15
$ cat /proc/kmlab/status
$ |
```

After inserting the two processes, they are both in the list, as shown by the first cat command. Then, process 1 completes, shown by the `[1] + Done`. The next call to cat only shows the second process, which makes sense, as the work function should have removed the first process from the list. Then the 2nd process finishes, and the third and final call to cat reflects this as the process is complete, so there should be no processes left in the list.

Output with single processes:

```
$ make
rm -f userapp *- *.ko *.o *.mod.c Module.symvers module
make -C /lib/modules/5.15.0-86-generic/build M=/home/tr
make[1]: Entering directory '/usr/src/linux-headers-5.1
  CC [M] /home/trenton/CPTS360/Lab4/kmlab.o
  MODPOST /home/trenton/CPTS360/Lab4/Module.symvers
  CC [M] /home/trenton/CPTS360/Lab4/kmlab.mod.o
  LD [M] /home/trenton/CPTS360/Lab4/kmlab.ko
  BTF [M] /home/trenton/CPTS360/Lab4/kmlab.ko
Skipping BTF generation for /home/trenton/CPTS360/Lab4/
make[1]: Leaving directory '/usr/src/linux-headers-5.15
gcc -o userapp userapp.c
$ sudo insmod kmlab.ko
$ ./userapp 10 &
$ sleep 6
$ cat /proc/kmlab/status
14840: CPU Time: 6892000000
$ cat /proc/kmlab/status
14840: CPU Time: 10280000000
[1] + Done                               ./userapp 10
$ cat /proc/kmlab/status
$
```

After inserting the process, a similar thing happens. It runs and is in the linked list, shown by the first call to cat. Next it completes ([1] + Done) and while the process is still in the list, the kernel timer hasn't interrupted it yet to remove the process. On the final call to cat, you can see the timer has interrupted the module and removed the process from the list since it is complete.