Isaac Dahle
11775278

Implementation Overview

Linked List:

One of the first things I implemented in this lab was all the stuff relating to linked lists. First, I added several of the helper functions that we used in class such as add_node, delete_node, and show list. Then I added the struct for the list head and a struct for my specific linked list like we discussed in class. The function I added for the linked list implementation was the delete_list() function. This function deletes all the nodes in the list and is used in the exit module to free all the nodes that remain in the kernel linked list.

Work Handler:

This function handles the updating of process CPU times for the processes that are stored in the kernel linked list. I did this through using the built in list_for_each_enty_safe_function() and then for each entry calling the function get_cpu_use() that was provided. Additionally, if the process had finished, I deleted it and freed it memory. Additionally, I used a spin lock to protect the kernel data structure as I am updating all the CPU times.

Kernel Timer:

In the kernel timer I schedule the next work handler and then use the mod_timer function to ensure that the timer will repeat every 5 seconds.

Proc File Write:

In this function the first thing I did is clear the proc buffer. Then I check the length of the proc buffer to prevent buffer overflow. Then I grab the PID that the user wrote to the proc buffer and add that process to the kerned buffer via using the add_node() function I wrote.

Proc File Read:

In this function the first thing I did is to clear the proc buffer again. Then I loop through the kernel linked list and add all of the process information stored in the linked list to the proc buffer variable. Then I copy this buffer variable to the proc file via copy_to_user. Additionally, I use spin locks while reading the data to prevent any issues if another function is trying to modify the linked list at the same time.

Module Init:

In this module I create the kernel linked list and then the proc directory and file. Additionally, I set up the timer and the workqueue.

Module Exit:

In this module I make sure to free all the memory being used in the kernel linked list and I remove the procfile and procfile directory. Additionally, I delete the kernel timer and then destroy the workqueue.

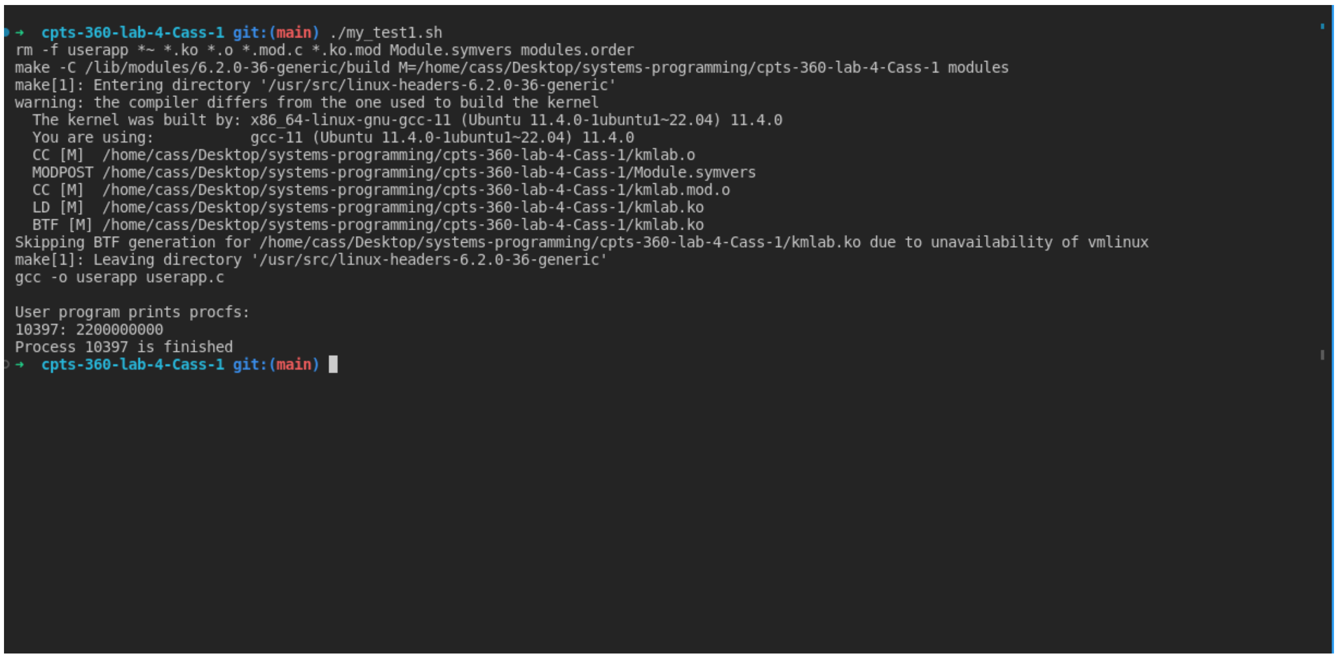
Testing:

I wrote two test cases to test this program. The first runs one process and the second runs two processes. My user application prints out information from the procfile as specified in the instructions.

General Overview:

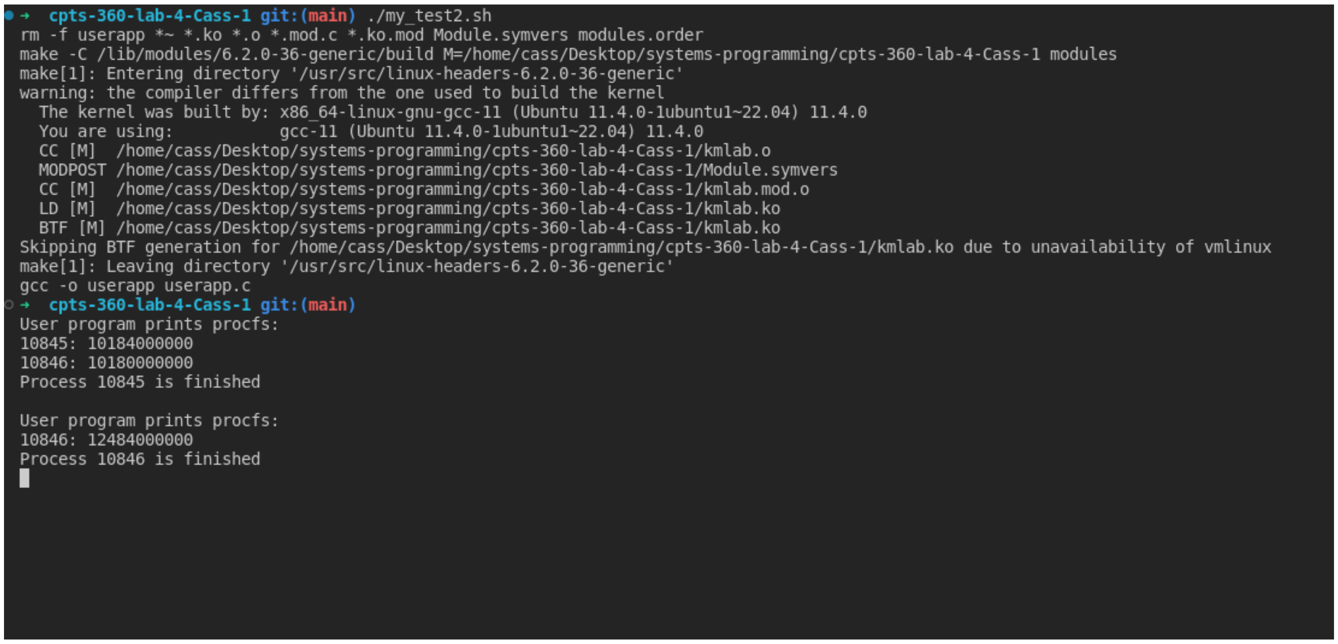
This module allows a user to write the PID number of a process to a specific procfile and then the kernel module will track that process. Whenever the user reads from the procfile they will see all the procfiles they have tracked and their CPU times. This is done by using a kernel linked list to store information on all the tracked processes and a kernel timer to periodically update the information on how long a process has been running. This information is then written to the procfile on-demand.

Screenshots of The Tests



```
➜  cpts-360-lab-4-Cass-1 git:(main) ./my_test1.sh
rm -f userapp *~ *.ko *.o *.mod.c *.ko.mod Module.symvers modules.order
make -C /lib/modules/6.2.0-36-generic/build M=/home/cass/Desktop/systems-programming/cpts-360-lab-4-Cass-1 modules
make[1]: Entering directory '/usr/src/linux-headers-6.2.0-36-generic'
warning: the compiler differs from the one used to build the kernel
  The kernel was built by: x86_64-linux-gnu-gcc-11 (Ubuntu 11.4.0-1ubuntu1~22.04) 11.4.0
  You are using:           gcc-11 (Ubuntu 11.4.0-1ubuntu1~22.04) 11.4.0
  CC [M]  /home/cass/Desktop/systems-programming/cpts-360-lab-4-Cass-1/kmlab.o
  MODPOST /home/cass/Desktop/systems-programming/cpts-360-lab-4-Cass-1/Module.symvers
  CC [M]  /home/cass/Desktop/systems-programming/cpts-360-lab-4-Cass-1/kmlab.mod.o
  LD [M]  /home/cass/Desktop/systems-programming/cpts-360-lab-4-Cass-1/kmlab.ko
  BTF [M] /home/cass/Desktop/systems-programming/cpts-360-lab-4-Cass-1/kmlab.ko
Skipping BTF generation for /home/cass/Desktop/systems-programming/cpts-360-lab-4-Cass-1/kmlab.ko due to unavailability of vmlinux
make[1]: Leaving directory '/usr/src/linux-headers-6.2.0-36-generic'
gcc -o userapp userapp.c

User program prints procfs:
10397: 2200000000
Process 10397 is finished
➜  cpts-360-lab-4-Cass-1 git:(main) 
```



```
➜  cpts-360-lab-4-Cass-1 git:(main) ./my_test2.sh
rm -f userapp *~ *.ko *.o *.mod.c *.ko.mod Module.symvers modules.order
make -C /lib/modules/6.2.0-36-generic/build M=/home/cass/Desktop/systems-programming/cpts-360-lab-4-Cass-1 modules
make[1]: Entering directory '/usr/src/linux-headers-6.2.0-36-generic'
warning: the compiler differs from the one used to build the kernel
  The kernel was built by: x86_64-linux-gnu-gcc-11 (Ubuntu 11.4.0-1ubuntu1~22.04) 11.4.0
  You are using:           gcc-11 (Ubuntu 11.4.0-1ubuntu1~22.04) 11.4.0
  CC [M]  /home/cass/Desktop/systems-programming/cpts-360-lab-4-Cass-1/kmlab.o
  MODPOST /home/cass/Desktop/systems-programming/cpts-360-lab-4-Cass-1/Module.symvers
  CC [M]  /home/cass/Desktop/systems-programming/cpts-360-lab-4-Cass-1/kmlab.mod.o
  LD [M]  /home/cass/Desktop/systems-programming/cpts-360-lab-4-Cass-1/kmlab.ko
  BTF [M] /home/cass/Desktop/systems-programming/cpts-360-lab-4-Cass-1/kmlab.ko
Skipping BTF generation for /home/cass/Desktop/systems-programming/cpts-360-lab-4-Cass-1/kmlab.ko due to unavailability of vmlinux
make[1]: Leaving directory '/usr/src/linux-headers-6.2.0-36-generic'
gcc -o userapp userapp.c
➜  cpts-360-lab-4-Cass-1 git:(main)
User program prints procfs:
10845: 10184000000
10846: 10180000000
Process 10845 is finished

User program prints procfs:
10846: 12484000000
Process 10846 is finished
```