CS 423 MP1

Group Members:

1) Furquan Shaikh(fmshaik2)
2) Gurmeet Singh(gurmeet2)
3) Puneet Chandra(pchandr2)

Implementation and Design decisions:

A) Initialization:

Following tasks are performed during mp1 kernel module initialization:

1) On init, mp1_kernel_mod creates a proc directory entry "mp1" and a proc entry under this directory "status". Thus, user processes can use the entry /proc/mp1/status to read from or write to.

2) Head for a link list which will hold a list of all the registered processes is initialized. Every entry in the list has the following members:
a) list head b) process id c) cpu time

3) A kernel thread mp1kt is created which will act as the bottom half of the interrupt handler. This thread is used to update the cpu times of every process which is registered with mp1 kernel module.

4) A semaphore mp1_sem is initialized to control access to the linked list. This is required since both kernel module and kernel thread can make updates to the linked list. Hence, synchronization is achieved between them using semaphore mp1_sem.

5) A timer mp1_timer is created, which will wake up the kernel thread every 5 seconds to update the cpu time of registered processes. This timer is set only when the first process is added to the link list and restarted only if there already exist processes on the list. So, if no processes exist on the list, the timer will not be restarted.

B) Proc entry updates:

1) When a write is made onto the proc entry -
pid of the process is copied from user and a new entry is initialized. We check if the link list is empty. If yes, mp1_timer is set to fire in 5 secs. The new entry is added to the list.

2) When a read is made on the proc entry -
Scan the linked list and send PID:CPU Time pairs to the user.

Both the proc entry operations on linked list are performed after locking the semaphore.

C) Kernel thread (Bottom half) :
The kernel thread acts as the bottom half of the interrupt handler. Thus, it sleeps in interruptible state until the top half comes and wakes it. On waking up it first checks if the exit module has asked the thread to quit. If yes, the kernel thread exits. Else, it locks the semaphore and traverses the linked list to update CPU time of every registered process. For the processes, whose CPU time is returned as 0, the process entry is removed from the linked list (since it means that the process has completed execution). Then, it restarts the timer only if the linked list is non-empty.

D) Timer handler (Top Half):
It just wakes up the sleeping kernel thread to perform its task

E) Clean-up of module:
It performs the following tasks:
1) Stop the timer

2) Remove the proc entries

3) Delete all the entries in linked list

4) Wake up the kernel thread and ask it to exit

F) User space application:
The user space application on starting execution registers itself with the kernel module by writing to the proc entry. It performs various factorial operations and at the end of it reads back from the proc entry.

<u>Submission files</u>:
1) mp1_kernel_mod.c : Kernel module implementation for mp1
2) mp1_user_app.c : User app implementation for mp1
3) mp1_given.h : Header file containing given code
4) Makefile : Makefile to compile kernel module and user application

<u>Compilation steps</u>:
```
make clean
make
```

<u>How to use</u>:
1) insmod mp1_kernel_mod.ko
2) ./mp1_user_app
3) dmesg shows mp1 specific logs with "mp1:" tag
4) rmmod mp1_kernel_mod

Multiple instances of mp1_user_app can be executed to check for multiple processes in linked list.