

CS 423 MP1 Group 9

Laurynas Tamulevicius tamulev2

Pranava Aditya paditya2

Vassil Mladenov mladenov2

David Lipowicz lipowicz2

1) In order to create Proc Filesystem entries we simply used `<linux/proc_fs.h>` API which provides methods `proc_mkdir` to create directory and `proc_create` to create entry - the method calls were placed at `mp1_init` function. Also API has `proc_remove` method which allows to delete created directory and entry, which was done in `mp1_exit`.

2) In order to create Linked list we used Linux kernel Linked List. The structure of node, which have fields: `list` (of type `list_head`), `pid` (integer) and `cputime` (long unsigned), was defined at line 30-35.

3) Regarding callback functions, we used `copy_from_user()` and `copy_to_user()` for obtaining/providing I/O from/to user. `mp1_write` creates a buffer that stores copied data. New node is created and its fields are set. Before adding the new node we acquire a interruptible mutex and after we add element the mutex is released. We used interruptible mutex as if adding node is asleep it should allow for an interrupting thread to pre-empt. In `mp1_read` we iterate through list and store information in the buf also before iterating we acquire interruptible mutex and after the information is stored we release the mutex and copy it to user.

4) We implemented a simple test user application (from step 9) (calculates factorial) which registers itself in the module and is stored in "userapp.c".

5) Regarding timer, we created a function `timerFun` which sets expiration to 5 seconds and starts timer at kernel (`add_timer`) also as timer is set we put work task (`wq`) in global workqueue using kernel method `schedule_work`. The `timerFun` is being invoked and its parameters are initialised at `mp1_init` function.

6) To implement workqueue we first setted up workqueue in `wq_fun` function and to initialise workqueue itself and submit a task to a workqueue we used `<linux/workqueue.h>` API macro `DECLARE_WORK`.

7) We update cpu times in workqueue function by acquiring mutex and then iterating over the linked list where the helper function was used. After cpu times are updated for all nodes the mutex is released.

8) Linked list, timer and proc file system are destroyed in `mp1_exit`. The static variables are destroyed using `module_exit` macro.

To run a program please navigate to main directory which contains Makefile and type:

```
make
```

```
sudo insmod mp1.ko
```

```
./userapp &          ###Comment### Do that several times to trigger the factorial computation
```

cat /proc/mp1/status **###Comment###** Several times and watch the numbers go up
sudo rmmod mp1 **###Comment###** And then verify that the proc entries are deleted

```
tamulev2@sp16-cs423-g09:~$ cd mp1/
tamulev2@sp16-cs423-g09:~/mp1$ make
rm -f userapp *~ *.ko *.o *.mod.c Module.symvers modules.order
make -C /lib/modules/3.19.0-25-generic/build M=/home/tamulev2/mp1 modules
make[1]: Entering directory `/usr/src/linux-headers-3.19.0-25-generic'
  CC [M]  /home/tamulev2/mp1/mp1.o
Building modules, stage 2.
MODPOST 1 modules
  CC      /home/tamulev2/mp1/mp1.mod.o
  LD [M]  /home/tamulev2/mp1/mp1.ko
make[1]: Leaving directory `/usr/src/linux-headers-3.19.0-25-generic'
gcc -o userapp userapp.c
tamulev2@sp16-cs423-g09:~/mp1$ sudo insmod mp1.ko
[sudo] password for tamulev2:
tamulev2@sp16-cs423-g09:~/mp1$ cat /proc/mp1/status
tamulev2@sp16-cs423-g09:~/mp1$ sudo rmmod mp1
tamulev2@sp16-cs423-g09:~/mp1$ cat /proc/mp1/status
cat: /proc/mp1/status: No such file or directory
tamulev2@sp16-cs423-g09:~/mp1$ sudo insmod mp1.ko
tamulev2@sp16-cs423-g09:~/mp1$ cat /proc/mp1/status
tamulev2@sp16-cs423-g09:~/mp1$ ./userapp & ./userapp &
[1] 1879
[2] 1880
tamulev2@sp16-cs423-g09:~/mp1$ cat /proc/mp1/status
PID: 1879, time: 480
PID: 1880, time: 479
tamulev2@sp16-cs423-g09:~/mp1$
```