

## 4.2 : Kernel Module:

```
[ 41.136484] char_device: major number is 243
[ 41.136492] use mknod /dev/char_device c 243 0" for device file
pi@raspberrypi:~/rpi-kernel/proj2/modules$
CTRL-A Z for help | 115200 8N1 | NOR | Minicom 2.7 | VT102 | Offline | ttyUSB0
```

```
prateek@prateek-XPS-13-9360: ~
pi@raspberrypi:~/rpi-kernel/proj2/apps/test_sys$ ./implement_sys
^C
pi@raspberrypi:~/rpi-kernel/proj2/apps/test_sys$ cat /dev/char_device c 243 0
```

```
[ 234.438436] char_device: opened device
[ 234.438489] char_device: reading from device
[ 234.438537] Name: implement_sys PID: [1109]
[ 238.957699] char_device: closed device
[ 244.318639] char_device: opened device
[ 244.318690] char_device: reading from device
[ 244.318739] Name: implement_sys PID: [1109]
pi@raspberrypi:~/rpi-kernel/proj2/apps/test_sys$
CTRL-A Z for help | 115200 8N1 | NOR | Minicom 2.7 | VT102
```

### Problem 1:

Concurrency and Parallelism are interrelated concepts. Concurrency means that 2 tasks A and B need to execute independently of each other. For instance if A starts executing, then B must start before A has finished its execution.

Parallelism is a way of accomplishing concurrency. Parallelism can be multiple CPUs working on different tasks at the same time. However, concurrency can also be accomplished by task switching. Task A can execute for a certain period and then the CPU can switch to another task 'B' and then switch back and forth between the two tasks. If the time slices spent in executing both tasks are small enough, it may appear to the user as if both tasks are being executed simultaneously.

Problem 2:

a.) No data points are lost in overhead as points are generated at intervals of  $10^{-3}$  seconds and the total overhead time taken to copy and make switches between kernel space and user space is  $0.2 \times 10^{-3} + 10^{-8}$  seconds which is less than the time it takes for the arrival of the next point.

b.) Total time taken to read a point =  $2 \times 100 \times 10^{-6} + 10 \times 10^{-9}$

Capacity of syscall (i.e maximum number of points it is able to read per second) =  $1 / (2 \times 100 \times 10^{-6} + 10 \times 10^{-9}) = 4999.75$

Points generated per second = 100,000.

Therefore, points lost per second =  $95000.249987501 = 95.000249988\%$  or .95000249987501

c.) A point arrives every  $10^{-5}$  seconds, and the buffer waits  $10^{-5} \times 1000$  seconds to fill up. It takes a total of  $2 \times 100 \times 10^{-6} + 1000 \times 10 \times 10^{-9}$  seconds to transfer data which causes it to miss 20.001 points. (1000 data points in buffer takes  $1000 \times 10$ ns to copy)

Therefore in  $10^{-5} \times 1000 + 2 \times 100 \times 10^{-6} + 1000 \times 10 \times 10^{-9}$  seconds, it misses 21 points and reads 1000 points.

In one second,

it misses = 95454.545454545 points

Ratio of missed to total points = 0.954545455

d.) The implementation can be improved by having a secondary buffer storing data points while one buffer transmits data to user space. The size of the buffer that transmits the data however has to be optimized since there is a tradeoff in terms of the time increase in copying the data point value as the size increases. Thus, a secondary buffer/memory space along with an optimized transmission buffer is an optimal solution.

**NOTE: PLEASE USE ARCH AND CROSS\_COMPILE FLAGS WHILE COMPILING char device driver with makefile.**