**IMPORTANT:**To receive full credit for this laboratory you must be able to demonstrate and explain all your implemented solutions in Minix3 to the TA (Happy Hour). The demonstration will be made during any lab hours to accommodate all students. This may be done before and after this laboratory deadline. Failure to do so will constitute a change in marks (up to removing all points) for this lab.

Implement a Kernel call, as instructed on the [*Add Kernel Call to Minix Tutorial*](https://online.upr.edu/pluginfile.php/2526214/question/questiontext/3256614/1/25795765/Add%20Kernel%20Call%20to%20Minix%20Tutorial%20-%20Updated.pdf)pdf, to allow management of a **tickets** attribute for the process.  Your code should have the following changes to facilitate the next exercise:

| **Names to be replaced** | |
| --- | --- |
| **Kernel Call Tutorial** | **This Exercise** |
| SYS\_SETEDF | SYS\_MNGTKT |
| do\_setedf | do\_kmngtkt |
| deadline | tickets |
| MYCALL | MNGTKT |
| do\_setedf.c | do\_kmngtkt.c |
| sys\_edf.c | sys\_tkt.c |
| sys\_edf | sys\_settkt |
| **System Call Tutorial** | **This Exercise** |
| do\_mycall | do\_mngtkt |
| mycall | set\_tickets |
| mycalllib.h | ticketslib.h |

Some troubleshooting:

* Be sure to create a snapshot of your Minix virtual machine before you compile.
* If you are having trouble booting (system won't load) after modifying and compiling the Kernel, **boot option 1,** go to /usr/src/releasetools and run **make clean**, then compile.
* If your system panics,**boot option 1**, fix your mistake, run make clean, compile again and try to boot option 2.
* If you get a "suberblock" error try resetting the virtual machine (on the VirtualBox menu bar: Machine > Reset) and booting option 1.
* You might want to try restarting the VirtualBox application as well as the VM.
* Sometimes the VM might go into panic (automatically resetting) if an exception happens when executing a kernel/system call, which makes it difficult to see error information before the VM restarts and overwrites it.  You might find useful the video capture feature of VirtualBox to save a video of the guest VM screen and play it back after the reset.
* If all of the above fails, you will need to restore from a snapshot.

# Exercise 1

## 40/40

According to the instructions ([Guide Lottery Scheduling](https://online.upr.edu/pluginfile.php/2526214/question/questiontext/3256614/2/25795762/lottery2.pdf)) of this lab, enter below your modification of the descending enqueue algorithm.

Example

if this function is invoked with the following order of values of rp-> tickets:

rp-> tickets=6

rp-> tickets=5

rp-> tickets=8

rp-> tickets=7

rp-> tickets=9

After aplying your algorithm, the rdy queue should be in descending order as follow:

rdy\_head->tickets=9

(rdy\_head->p\_nextready)->tickets=8

((rdy\_head->p\_nextready)->p\_nextready)->tickets=7

((rdy\_head->p\_nextready)->p\_nextready)->p\_nextready)->tickets=6

# Exercise 2

## 60/60

Next you must write your lottery algorithm in the indicated lines. The variable total\_tcks is already defined in glo.h and it already saves the total number of tickets. Similarly, cont\_wins is already defined globally and with an initial value of 0.

The rand () function works in the same way as the one shared in the PDF of the guide.

Example and expected resutls:

Assuming that we have in the ready queue:

rdy\_queue[q][0]=32

rdy\_queue[q][1]=19

rdy\_queue[q][2]=18

rdy\_queue[q][3]=17

rdy\_queue[q][4]=16

rdy\_queue[q][5]=15

(Note that the queue is already in Descending order)

As the winner proccess is selected randomly, we cant predict an exact order of execution. But for testing purposes let's assume that the first process (head) always win the lottery (like rand() is always returning 1). Then the execution order should be:

1)rdy\_queue[q][0]=32       //count\_wins=1

2)rdy\_queue[q][1]=19       //count\_wins=2

3)rdy\_queue[q][2]=15       // count\_wins is >= 2 we pick the process with fewer tickets to execute and we restart the counter

4)rdy\_queue[q][3]=18       //count\_wins=1

5)rdy\_queue[q][4]=16       // Againcount\_wins is >= 2 we pick the process with fewer tickets to execute

5)rdy\_queue[q][5]=17       //count\_wins=1