Lab 2: Modifying xv6 Scheduler

In this part, you will implement lottery scheduler and stride scheduler. Implement only basic operation (no ticket transfers, compensation tickets, etc...). It is up to you to design the interface to assign the number of tickets, but it should be sufficient to illustrate that the schedulers work correctly.

To get started, look at the file “proc.c”. In there you will find a simple round robin CPU scheduler implemented. You should change the logic there to use lottery and stride scheduling.

After you complete the schedulers, here is how you can demo your work for each one separately.

Run the following program with for 3 different ticket values:

```
#include "types.h"
#include "stat.h"
#include "user.h"
int main(int argc, char *argv[])
{
      FUNCTION_SETS_NUMBER_OF_TICKETS(30);    // write your own function here
      int i,k;
      const int loop=43000;
      for(i=0;i<loop;i++)
      {
            asm(nop);       //in order to prevent the compiler from optimizing the for loop
            for(k=0;k<loop;k++)
            {
                  asm(nop);
            }
      }
      exit();
}
```

Output should look like this:

for example: The program prog1.c has 30 tickets, add prog2.c with 20 tickets and prog3.c with 10 tickets. Run all of them simultaneously:

Input

$ prog1&;prog2&;prog3

Assuming that you run 3 test programs, each with a different number of tickets, when each of these programs finishes execution, your system call should print the number of times the processes was scheduled to run (number of ticks).

Output: (not necessarily in the same order)

Ticks value of each program

prog1 : xxx

prog2 : xxx

prog3 : xxx

Use the number of ticks to calculate the allocated ratio per process : (number of ticks per program)/(total number of ticks by all 3 programs)

Approximately prog1 ratio will be 1/2 , prog2 ratio will be 1/3 and prog3 will be 1/6

Compare your implementations of lottery scheduler and stride scheduler. Consult Section 5 of the stride scheduling paper on how to compare and evaluate these two schedulers. Try to reproduce Figure 8 and Figure 9 for your implementations.

What to submit:

You need to submit the following:

(1) Two patch files over the original XV6 source code: one for lottery scheduler, and one for stride scheduler. Hint: use "git diff" command to generate a patch file.

(2) A PDF document with detailed explanation what changes you have made, the comparison figures, and how you produce the figures.

Grades breakdown:

- Working lottery scheduler (with clear explanation): 40 %
- Working strider scheduler (with clear explanation): 40%
- Comparison figures and clear explanation how to produce them: 20%

Total: 100%