**Lab #2: Modifying xv6 Scheduler**

Submitted by: Priyanka Jadli (862325015) and Chirag Rajavat (862324954)

# **Demonstration Link**

<https://youtu.be/b168PkvQM6I>

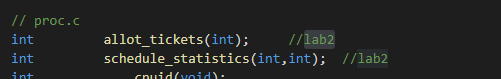
**List of files modified:**

1. kernel/defs.h
2. kernel/proc.c
3. kernel/proc.h
4. kernel/rand.c
5. kernel/rand.h
6. kernel/syscall.c
7. kernel/syscall.h
8. kernel/sysproc.c
9. user/prog1.c
10. user/prog2.c
11. user/prog3.c
12. user/user.h
13. user/usys.pl
14. Makefile

**Modification screenshots:**

1. kernel/defs.h

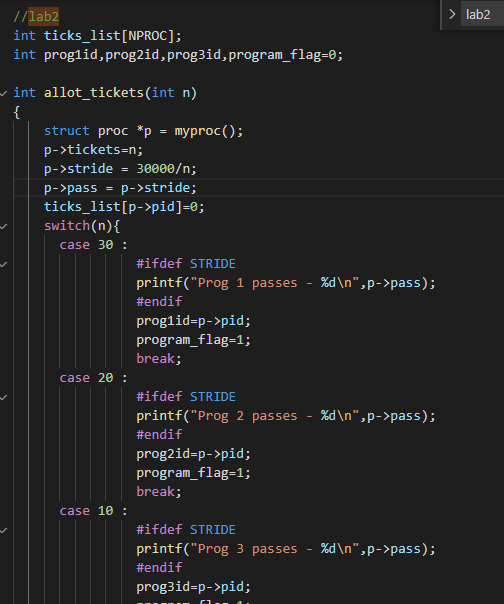
Added entries for kernel functions

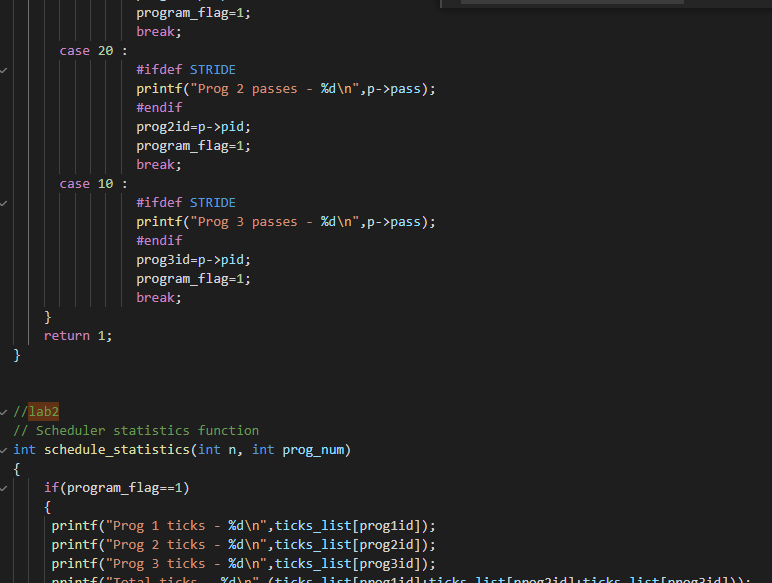


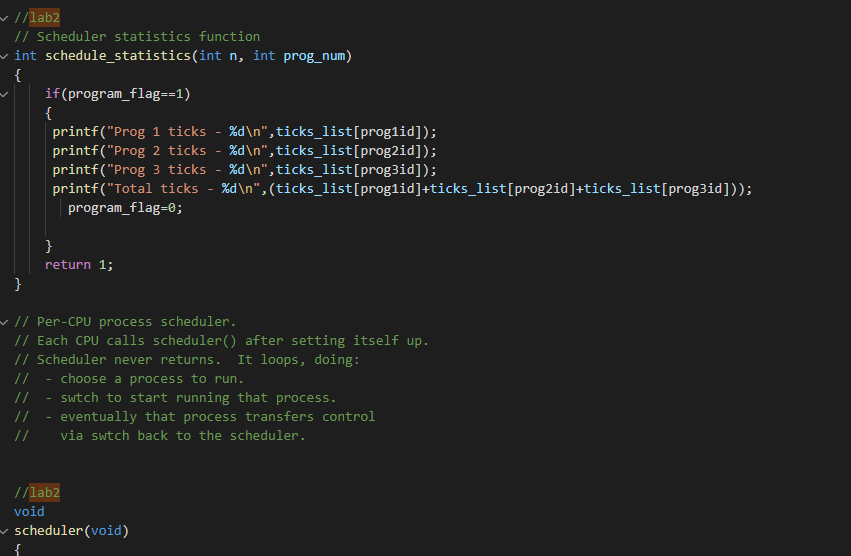
1. kernel/proc.c

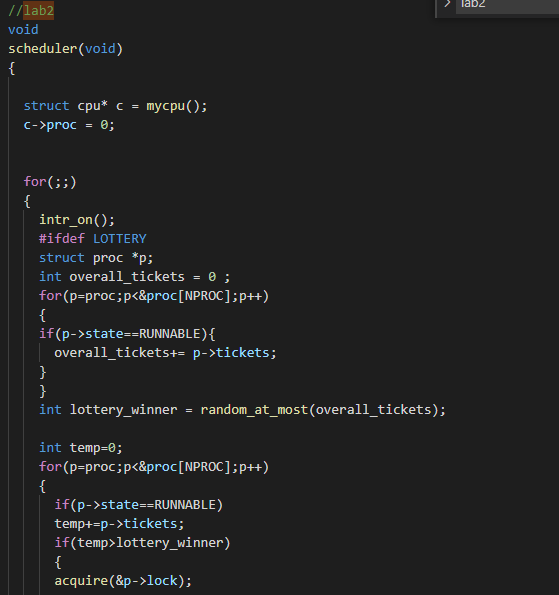
Included file rand.h to include random generator library in rand.c. Added implementation for lottery and stride scheduling. The allot\_tickets function allots tickets to the process. Sched\_statistics function prints the ticks and passes in case of stride.

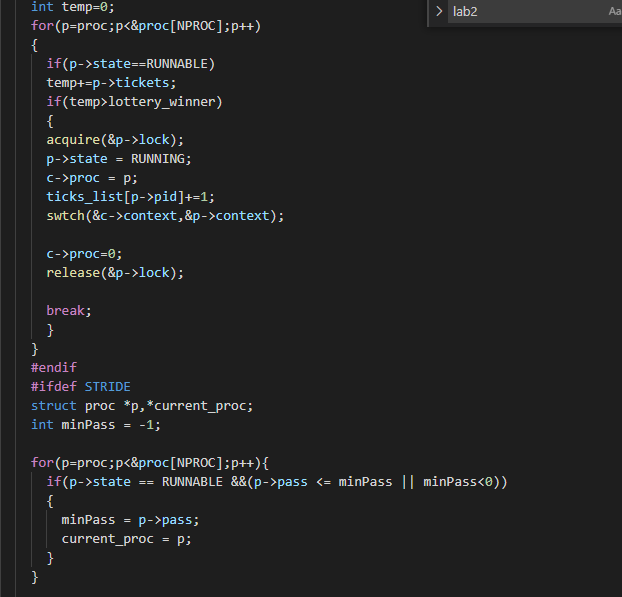


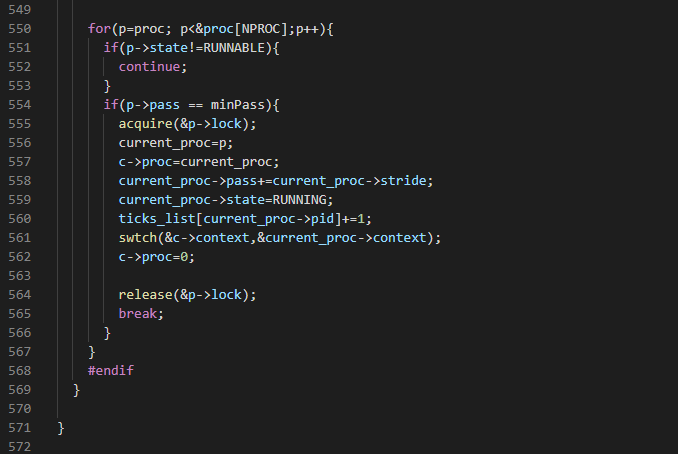






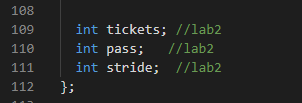






1. kernel/proc.h

Added fields tickets, pass, and stride to structure process.



1. kernel/rand.c

Added new file rand.c as a random generator

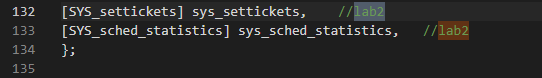
1. kernel/rand.h

Added new file rand.h as a header file for rand.c

1. kernel/syscall.c

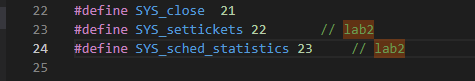
Added declaration for our both system calls





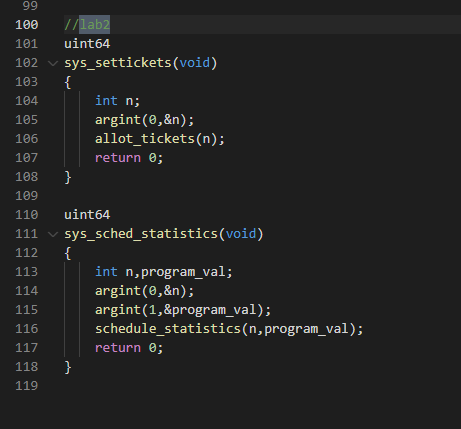
1. kernel/syscall.h

Assign a new number to our system call



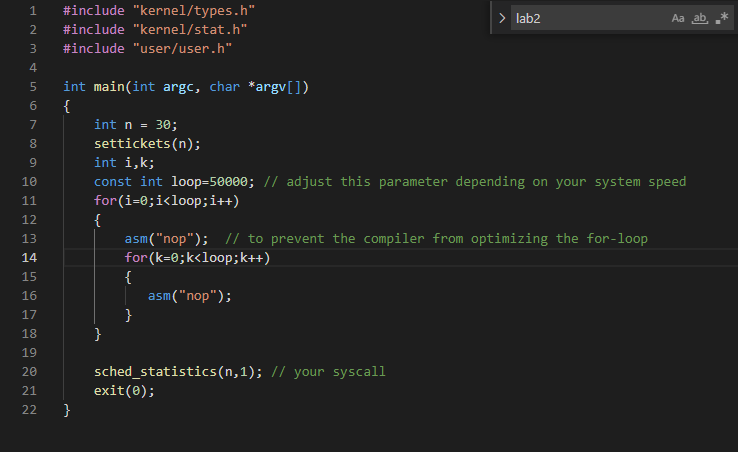
1. kernel/sysproc.c

Defined sys\_info syscall function and passed the argument to get\_sys\_info function for further processing



1. user/prog1.c

Added user programs to be run and test our scheduling algorithm. This file is replicated as prog2.c and prog3.c and allotted tickets 30, 20, and 10 respectively.



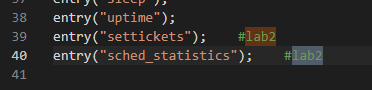
1. user/user.h

Added entry to update user syscall interface. This would be the function our user program can call.



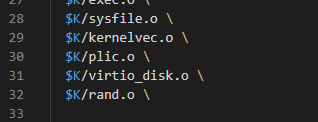
1. user/usys.pl

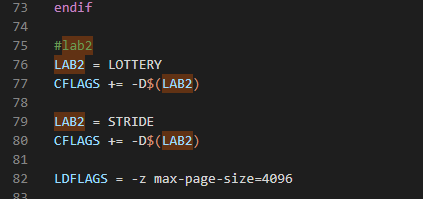
Added entry to update user syscall interface

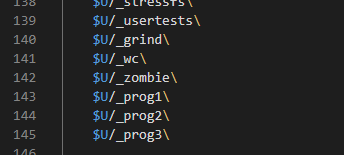


1. Makefile

Added rand.o object in kernel list to add the random generator library to kernel. We also add our user program to this file to make it available for xv6 source code for compilation. Changes also made for changing the lottery scheduling algorithm.





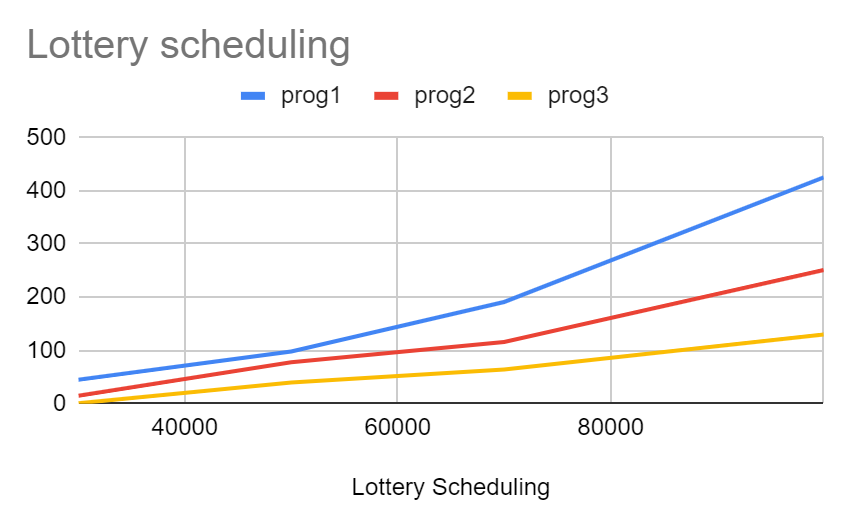
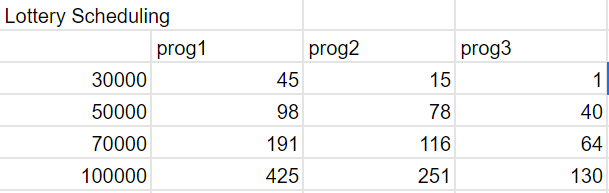


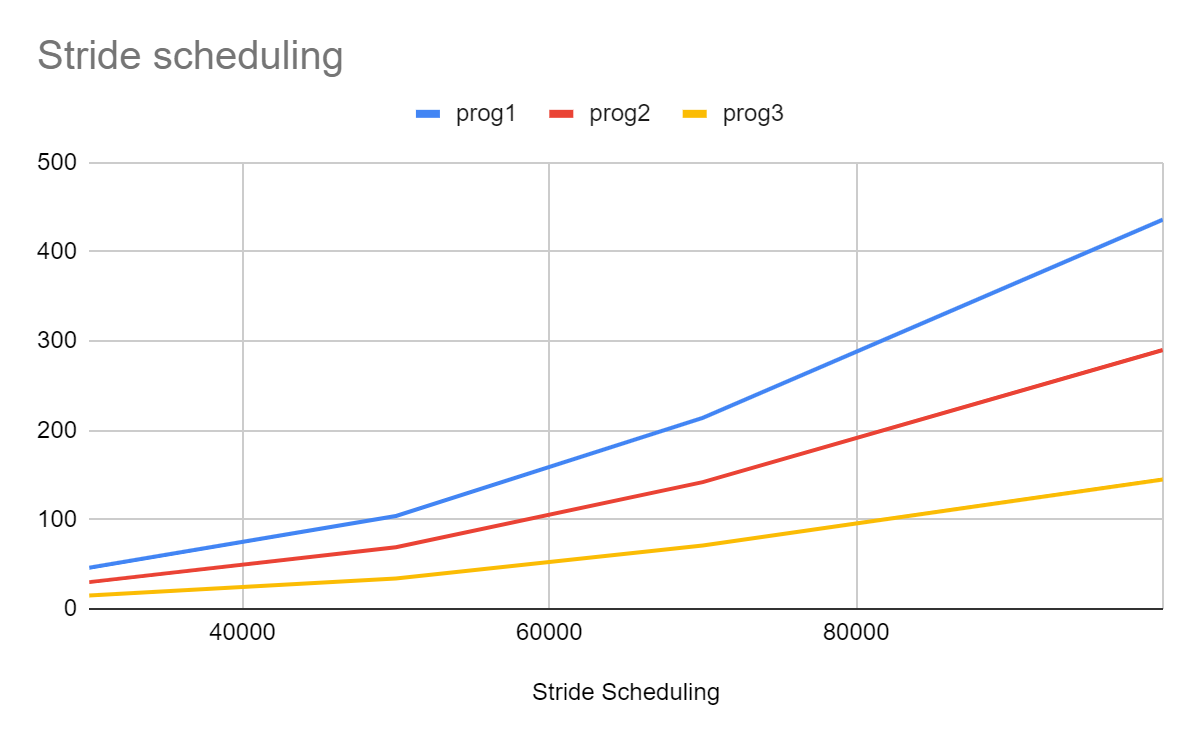
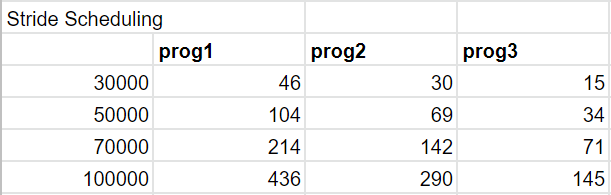
**Explanation of files and graph**

Xv6 implements a round-robin scheduling algorithm by default in proc.c in scheduler() function. We modified the proc.c to implement lottery and stride algorithms. First, we implemented the system calls to allocate tickets and get scheduled statistics. A random generator library is integrated into the kernel. The process structure is modified to contain the fields tickets, stride, and pass.

The function allot\_tickets calculates and allots the tickets and stride values in the process structure. We then modify the scheduler function. It generates a random number as a ticket and the process which contains that ticket value is chosen to be context switched. In the case of the stride scheduling algorithm, we calculate the stride passes, inversely proportional to the number of tickets allocated. The process with a minimum pass value is chosen to be executed and is context switched.

We ran both the scheduling algorithms for different no. of loop count, specifically, 30000, 50000, 70000, and 100000 and noted their tick counts. The graph was generated using the google sheets chart generation tool and is given below.



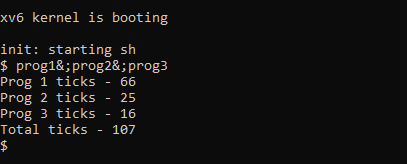


**Output screenshots:**

make clean;

Make qemu LAB2=LOTTERY

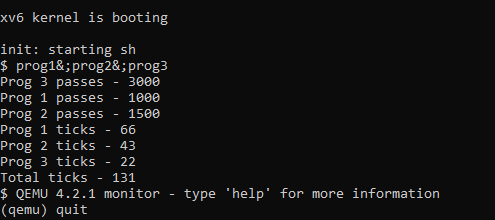
prog1&;prog2&;prog3



make clean;

Make qemu LAB2=STRIDE

prog1&;prog2&;prog3

****

**Work distribution**

The approach to lab work and distribution of work is as follows :

* Understanding lottery and stride scheduling algorithm - done together
* Discussion and coming up with the approach and pseudo code - done together
* Implementing lottery scheduling algorithm - Chirag Rajavat
* Implementing stride scheduling algorithm - Priyanka Jadli
* Video Making - Priyanka Jadli
* Report (Adding files modified and screenshots explanation) - Priyanka Jadli
* Report (Explanation) - Chirag Rajavat
* Report (Generating graph) - Priyanka Jadli