Goal:

Design a Fair Share Scheduling(FSS) algorithm where all users are assigned, in aggregate, equal time slices for their respective processes.

# Implementation:

The bulk of the modifications were done in sched.c. In the schuler\_tick() method, we modified the existing logic for determining whether or not a process needs to get its time slice replenished. If flag for FSS is set and the current process is not a root process, we calculate the time slice using our custom logic instead of the default logic.

The time slice is calculated by treating DEF\_TIMESLICE value as the time allocated to each user. Then all that we need to do to ensure fair scheduling is to divide DEF\_TIMESLICE by the number of processes of the current user and use that as the time slice for all processes of that user.

Two system calls were implemented to toggle the FSS algorithm and its profiling.

# Testing:

A test program was written which runs an infinite loop. Multiple instances of this program were run for 4 different users and the time slice allocated to all of these processes was logged to the kernel log. That log was then used to prepare a graph to see the share of CPU utilization of all the users and their processes.