Scheduling Report

Implementation

- MakeFile: CPU set to 1
- Issue with setting NCPU in param.h to lower than 3 (Not 100% sure why or how, but probably due to that being VCPU)
- Proc.c: changes to scheduler, wait, yield and allocproc off the top of my head
- Proc.h: changes to proc
- To implement syscalls
    - User.h
    - The perl file
    - Sysproc
- Everything inside #ifdef is new
- Runs selectively

RR

Default scheduler

Average run time: 17 ticks
Average wait time: 168 ticks

Lottery based scheduler

Every process is assigned 1 ticket on initialization.
A process might have more than 1 ticket and it has a higher chance of being selected if it has more.
Can be increased with set tickets syscall.
The random number is chosen by the “random.c”. Then, the process is set to running (and proc is switched) and swtch command is called to transfer control to the process.

If a process is considered the winner of the lottery, it is only if there are no other processes with the same number of tickets but an earlier arrival time

Average run time: 17 ticks
Average wait time: 163 ticks

Issue with prioritizing earlier arrival time

If there are a lot of long running processes, there is chance that later process will starve and never get a chance to execute.
Example
A, arrived at t=0s with 3 tickets,
X, arrived at t=0s with 2 tickets,
Y, arrived at t=0s with 1 tickets,
Z, arrived at t=0s with 5 tickets,
B, arrived at t=3s with 4 tickets,
C, arrived at t=4s with 3 tickets,

If max tickets is 5, then **process C will never get a chance to execute**.

Hence, the max tickets should be high and number of tickets should be reasonable unique.

Multilevel Feedback queue

There are 4 queues. Default is queue 0.

As fork can creates child, added to same queue.
Priority boosting every 48 ticks.

Average run time: 15 ticks
Average wait time: 162 ticks

Extra: Average rtime 16, wtime 165

MLFQ Analysis

The following is a timeline graph illustrating the queue position of various processes over time, with different processes color-coded. This shows how processes move across queues based on their CPU consumption and when priority boosts occur.

- **Y-axis**: Queue ID (higher values represent lower priority queues).
- **X-axis**: Time elapsed since the start of execution.
- **Color-coded lines**: Represent individual processes.

Modified scheduler to priority at each step
Due to large number of processes (10) newer reaches below queue 2 before priority boost