Operating Systems Lab (CS-UH 3010)

Spring 2024

Assignment 4: Lottery Scheduling Implementation

Deadline: April 25, 2024, 11:59 PM

Introduction:

In this lab, we'll examine a different type of scheduler known as a proportional-share scheduler, also sometimes referred to as a fair-share scheduler. Proportional-share is based around a simple concept: instead of optimizing for turnaround or response time, a scheduler might instead try to guarantee that each job obtains a certain percentage of CPU time. An excellent early example of proportional-share scheduling found in research is known as lottery scheduling; however, the idea is certainly older. The basic idea is quite simple: every so often, hold a lottery to determine which process should get to run next; processes that should run more often should be given more chances to win the lottery.

Underlying lottery scheduling is one very basic concept: tickets, which are used to represent the share of a resource that a process (or user or whatever) should receive. The percent of tickets that a process has represents its share of the system resource in question.

Let's look at an example. Imagine two processes, A and B, and further that A has 75 tickets while B has only 25. Thus, what we would like is for A to receive 75% of the CPU and B the remaining 25%. Lottery scheduling achieves this probabilistically (but not deterministically) by holding a lottery every so often (say, every time slice). Holding a lottery is straightforward: the scheduler must know how many total tickets there are (in our example, there are 100). The scheduler then picks a winning ticket, which is a number from 0 to 99.

Probably the most amazing thing about lottery scheduling is the simplicity of its implementation. All you need is a good random number generator to pick the winning ticket, a data structure to track the processes of the system (e.g., a linked list), and the total number of tickets.

Instructions:

The objective of this assignment is to implement a simple lottery scheduling simulation in C.

1. Study the concept of lottery scheduling and understand its basic principles.
2. Familiarize yourself with the provided code skeleton. Understand its structure and the purpose of each function.
3. Complete the implementation of the insert function to insert a new job into the linked list.
4. Implement the print_list function to print the contents of the linked list.
5. Modify the main function to accept command-line arguments for the seed and number of loops.
6. Implement the lottery scheduling logic in the main function to select a winner based on the number of tickets.
7. Ensure that the winner selection process does not result in an infinite loop and handles edge cases appropriately e.g. if the winning ticket is 0. You may start the ***winner_calculation*** from 1 instead of 0.
8. Test your implementation with different inputs to verify correctness and robustness.