Benjamin Patrick Schreiber 11790684 PA 2 CPU Schedulers

CPU Scheduler Implementation

Introduction

This C program imitates several CPU Schedulers: First Come First Serve, Shortest Job First, Round Robin. The program reads in an input file of the format: NUM_PROCESSES (A,B,C,M) (A,B,C,M) …

It then outputs statistics on the results of that scheduler: finishing time, cpu utilization, i/o utilization, throughput, average wait time, average turnaround time.

Set up for simulation

I began by modifying the template to create a more pleasing C experience. I removed all global variables and encapsulated them in a struct each scheduler function will return. I moved the random number functions into their own file, and created a C macro which when present makes the random number be exactly the same for each iteration, so as to not mess up any tests.

I modified the process struct by removing the list/queue fields. I do not use them in my algorithm. I then added an enum for the process status.

Then, all of the inputs need to be sorted. I created a cmpr_process function, and in every scheduler the first call I make is to qsort the input based on process arrival.

Scheduler Simulation

Now that the environment is set up, we can write our first function. All of my functions follow the same general template, with minor tweaks.

First Come First Serve

I wanted to create this algorithm to use constant space complexity, and operate in "one pass" per cycle. For first come first serve, it is incredibly important that when a running process is blocked, the process directly after it gets to be run. This relationship is circular. For example, if the processes are { p1, p2 }, and the p2 is running, and p1 is ready, then after p2 is done running p1 should run next.

To use this relationship, I turned array memory into a circular iterator with the circular_iterator macro. Whatever the address of the Running Process is, each cycle will begin there. Then, it will iterate until it finds a READY state process, even circularly to the beginning of the array again. This is always the "first come" process in First Come First Serve.

From there, the tie breakers are simple. A running process can turn into blocked or terminated, but nothing else in the same cycle.

Shortest Job First

SJF is another constant space one pass per cycle algorithm. Instead of circularly looping, we can simply iterate linearly with each cycle. However, at the end of the cycle, whichever process had the shortest time remaining, would be selected to run.

Round Robin

Round Robin is exactly the same as FCFS, however it uses a quantum. The quantum replaces the cpu burst / io burst patterns, and will always decrement the cpu burst a constant amount and always go into a blocked state afterwards.