REPORT

Name: Chris Banci
Date: March 31, 2017
Course: CS433 - Operating Systems
Assignment: 3 - CPU scheduling

Description:

This program is an implementation of a Discrete event simulation (DES), which is used to analyze the performance of various CPU scheduling algorithms such as First Come First Serve (FCFS) and Shortest Job First (SJF) . For this assignment, the discrete event simulation will run for a total duration time of five minutes and will report an analysis at the end of the simulation.

Implementation:

Using a discrete event simulator (DES), an event queue is used to jump forward in time when the next meaningful event occurs. As long as the event queue is not empty or the total duration of the simulation has not ended, it will continue to execute these events. An event contains simple information like execution time of event, type of event and the process that belongs to the event.

There are five types of events that are implemented for the system:

1) Process Arrival = This event represents the arrival of a new process in the ready queue. When handled, new processes are pushed into the ready queue and the cpu scheduler checks to see if the cpu is idle, then dispatches a process according to the selected cpu scheduling algorithm.

2) CPU Burst Completion = This event represents the completion of a processes burst time. When handled, it checks to see if the process's total duration time has been completed. If completed, it removes the process from the cpu. if not completed, it generates an I/O burst time and creates an I/O completion event. Lastly, it dispatches an process from the ready queue.

3) I/O Completion = This event represents the completion of a processes I/O burst time. When handled, it generates a cpu burst time for the process and creates a CPU Burst Completion event.Lastly, it dispatches an process from the ready queue.

4) Timer Completion = This event is used with preemptive scheduling algorithms for when a process's burst time exceeds the maximum time slice allowed for that process. When handled, it moves the current job from the cpu back to the ready_queue and updates the remaining cpu burst time of the event.

Two non-preemptive CPU scheduling algorithms have been implemented for this assignment:

1) First Come First Serve (FCFS) = In this CPU scheduling algorithm, the process that is dispatched from the ready queue first is the process that came in first.
2) Shortest Job First (SJF) = In this CPU scheduling algorithm, the process that is dispatched from the ready queue first is the process with the shortest CPU burst time.

Using this system, processes are in a constant cycle of being placed into the ready queue, dispatched to the CPU, and waits for I/O, until its process's total duration time completes. At the end of the simulation, statistics of each process is presented such as its arrival time, service time, turnaround time and wait time. These statistics are used to make an analysis report for the CPU scheduler selected.

Analysis:

FCFS scheduling

	10 process	**20 process**	**50 process**
Jobs Completed	4	7	3
CPU Utilization	64.30%	96.97%	97.10%
Throughput	0.013 jobs / s	0.023 jobs / sec	0.010 jobs / s
Avg Turnaround	50.018 s	120.736 s	204.932 sec
Avg Wait Time	45.364 s	89.538 s	136.909 sec

SJF scheduling

	10 process	**20 process**	**50 process**
Jobs Completed	4	6	4
CPU Utilization	64.30%	96.99%	97.12%
Throughput	0.013 jobs / s	0.020 jobs / s	0.013 jobs / s
Avg Turnaround	50.353 s	131.410 s	44.507 s
Avg Wait Time	44.916 s	58.540 s	31.395 s

According to the reports above, I have observed that the SJF scheduling algorithm generally produces lower turnaround times and waiting times than the FCFS scheduling algorithm. With 10 processes simulated, there is little to no difference, however with 20 processes simulated, the difference seems to level off. With 100 processes simulated, the difference becomes apparent that SJF algorithm is shown to be the more efficient one between the two scheduling algorithms. This observation makes sense because the SJF scheduling algorithm dispatches the process with the shortest CPU time burst, resulting in lower average wait time and average turnaround time. Lastly, I have observed that the number of jobs completed within the five minute duration of the simulation seems to plateau at 20 processes and decrease from there on out.

Included Files:

Source:

- main.cpp — // driver file to test out the simulation.
- event.cpp — // implementation file of an event used in an event queue
- process.cpp — // implementation file of a process used in a ready queue
- simulation.cpp — // implementation file of the DES simulation
- scheduler_FCFS.cpp — // implementation file of FCFS scheduler
- scheduler_SJF.cpp — // implementation file of SJF scheduler
- random.cpp — // used to generate random numbers, provided by course.

Header:

- event.h
- process.h
- simulation.h
- scheduler_FCFS.h
- scheduler_SJF.h
- random.h

Other:

- makefile — // used to compile the program into an executable.

How to run:

To compile this program, use the makefile which will compile the source files and create an executable called output.

To run the executable, enter ***./output <X> <Y>*** in the console.

- The argument ***<X>*** being the amount of processes to simulate.

- The argument ***<Y>*** being the CPU scheduling type
    - 1 = FCFS scheduler type
    - 2 = SJF scheduler type

Example:

```
./output 10 1          // This will simulate 10 processes using the FCFS scheduler.
./output 25 2          // This will simulate 20 processes using the SJF scheduler.
```

****If no arguments are passed, the program will prompt user for input****