Scheduling Simulator

Jon Bernal & Jerome Solomon
Te@m J^2

COEN283 Operating Systems
Summer 2018
Professor Amr Elkady

Rationale

Structurally, the project fits into the course very well because the class has an entire unit devoted to processes, their properties, and possible states and state transitions (ready/running/blocked); an entire unit on synchronization and resource management; and an entire unit on deadlocks and how to deal with them by detection, avoidance, and prevention. This project synthesizes many aspects of these lectures into a deeper and more complete understanding of the interactions between processes, scheduling algorithms, and resources, as well as the properties that give rise to deadlocks. This project also introduces three different algorithms that were not covered in class; one is purely theoretical and utilizes a variable time quantum, while the other two are actually implementable.

Process management is an essential part of how modern operating systems are able to provide the experience of concurrently running some number of processes, p, on some number of cores n, where $n < p$ (though in the common user's experience, $n << p$). According to the syllabus, the student must demonstrate an understanding of the interactions between the various components of an operating system as a course objective. The scope of the policies of these components, according to the course objectives, is generally introductory. The scope of the project that we are doing contains simulation and analysis of these same introductory policies. However, it will also cover some other scheduling algorithms not contained among these introductory policies. Inherent to the simulation of the scheduling process is the need to simulate limitations on access to resources, which naturally lends itself to exploration of deadlock detection and avoidance techniques.

As far as learning outcomes, the syllabus says that a student should be able to "apply the concepts in process management, such as allocating resources to processes, handling inter-process communication, multithreading, [and] deadlock detection and avoidance", as well as "understand basic OS functionality: CPU scheduling, [and] process management…". Our project definitely requires an understanding of basic OS functionality at the very minimum just to design the abstractions and perform the simulation, as well as an understanding of the performance measures used to describe various ways in which speedup may occur with a given scheduling algorithm.

Goals

The objectives of the project are:

- To accurately simulate the flow of processes between the ready, running, and blocked states and the requesting/allocation of resources by different processes
  - Process simulation goal reached by testing transition functions
  - Resource simulation goal reached by testing with known cyclical resource dependencies
- To output the scheduling as an annotated graph
  - Goal reached by testing transcripts generated from known data against known solutions.

Plan of Activities

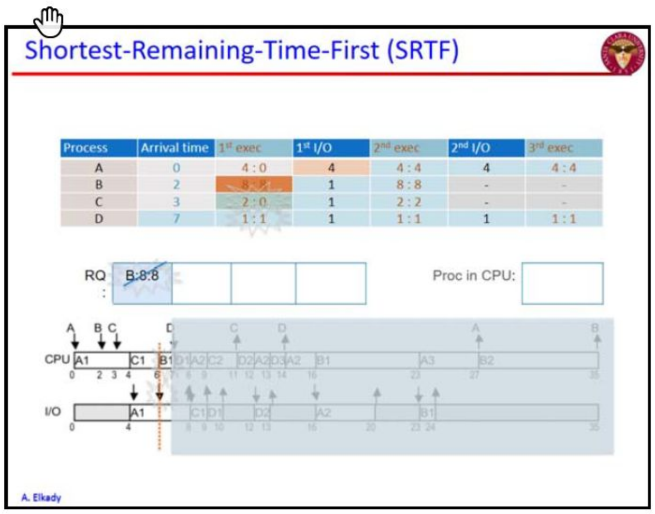
- Build simulator (Jon), finish by 8/2
  - Build Process generator
  - Build resource tracker
  - Build deadlock detection (Jon+Jerome)
  - Create data structures for each state
  - Create state transition functions (what happens at each state/how)
  - Test
- Implement algorithms (Jon+Jerome), finish by 8/10
  - Implement
  - Test
- Build different types of “loads” and run them through the simulator. (Jon+Jerome), finish by 8/12
  - Build loads
  - Run loads under various conditions
  - Analyze results
- Build visualizer (Jon+Jerome), finish by 8/17
  - Build transcript reader
  - Test
- Write paper (Jon+Jerome), finish by due date.

Scheduling Algorithms

- Scheduling Algorithms from the course:
  - First-Come First-Serve
  - Round Robin
  - Shortest Process First
  - Shortest Remaining Time First
  - Priority Scheduling
  - Static Priority Multi-Level Feedback Queues
  - Dynamic Priority Multi-Level Feedback Queues
- Additional Scheduling Algorithms
  - Fixed Priority Preemptive Scheduling (https://en.wikipedia.org/wiki/Fixed-priority_pre-emptive_scheduling)
  - Gang Scheduling (https://en.wikipedia.org/wiki/Gang_scheduling)
  - Prioritized Fair Round Robin with Variable Time Quantum (IEEExplore, 2015) (https://ieeexplore.ieee.org/document/7421020/)

Final Product Description

Our goal is to produce a scheduling tool that allows selection of the number of CPU cores & processes to be executed. The scheduling algorithm can also be selected. The simulator simulates the processes & produces an annotated graph similar to the output graph used by Professor Elkady in our class (picture below). We will also do a demonstration of the tool running in action (either showing an executable, or more likely a movie/video capture of the software running).



Shortest-Remaining-Time-First (SRTF)

Process	Arrival time	1st exec	1st I/O	2nd exec	2nd I/O	3rd exec
A	0	4 : 0	4	4 : 4	4	4 : 4
B	2	8 : 8	1	8 : 8	-	-
C	3	2 : 0	1	2 : 2	-	-
D	7	1 : 1	1	1 : 1	1	1 : 1

RQ: ~~B:8:8~~

Proc in CPU:

A. Elkady

To produce data and analyze it, we will create several runs of each scheduling algorithm & do a comparative analysis of each of the scheduling algorithms for our project presentation.

Tools

The tool will be developed in python. Some statistical functions and modelling will be provided by numpy and graphing will be done with matplotlib. The project is being developed within GitHub & we are using trello.com to track project development.

References

https://en.wikipedia.org/wiki/Fixed-priority_pre-emptive_scheduling
https://en.wikipedia.org/wiki/Gang_scheduling
https://ieeexplore.ieee.org/document/7421020/