Project 5 Report Jacob Galanopoulos

11/17/2019

Questions:

1. How did you separate scheduling mechanism from scheduling policies?

* The design that I implemented had the scheduling policies as their own sperate policy functions, FCFS\_policy(), RR\_policy(), and SRTF\_policy(). In each of those, I then sent it to the other functions for calculation of the different times/scheduling mechanisms, calcWaitTimewhich then invokes calcTurnAroundTime, find calcResponseTime, etc.

1. How did you implement the three scheduling algorithms?

* Each of the three policy functions has their own way of scheduling themselves. FCFS has its own management mechanism that automatically executes queued requests and processes by the order of their arrival. RR is assigned a fixed time slot in a cyclic way. SJF chooses the shortest arrived process and runs it in full.

1. How did I calculate waiting times?

* Waiting time is the amount of time a process has been waiting in the ready queue. findWaitingTime, for example was turnaround\_time – execution time.

1. How did I calculate response times?

* Response Time is the amount of time from when a request was submitted until the first response is produced. findResponseTime = start\_time – arrival\_time.

1. How did I calculate turnaround time?

* findTurnAroundTime is the amount of time to execute a particular process. findTurnaroundTime = finish\_time – arrival\_time

1. How did I implement the command-line parser?

* The task list that is file that is being read in through the main file with fscan. It is then filtered into an array called task\_array. That array is then sent to each of the policies to then calculate the scheduling.

Generality and Error Checking

1. How general is your solution?

* The solution I used is very general, all running is handled in policies and all the calculations are generalized rather than used for only one policy.

1. How easy would it be to add a new scheduling policy into your scheduler?

* Very simple. All that it would require is the add a new policy function, then add your policy to the list of options.

1. Does your program offer input error checking?

* Yes. In the code there are multiple error checking functionalities. Examples are the file opening and closing function. If the file is null or doesn’t have the correct format of data, the program will display a message. Another example would be if the file even opens or not, there’s displayed message stating that the file did not successfully open.

Miscellaneous factors

1. Is your code elegant?

* The code utilizes all resources and is as simple as possible. There could be better variable standardization, however I found it not worth it going over it again.

1. How innovative is your solution? Did you try any ideas not suggested on here?

* Yes, I believe the code solution is innovative and efficient. I went though several phases of pseudo code before completing the coding section for the project.

1. Did you document all outside sources?

Yes, I used in class resources that Qin provided us to complete the project