CSC 4320: Operating Systems Project 1

Kayla Walker, Morgan Warren

Hello Professor, we would like to say that our group member Aaron Reyes was extremely unhelpful with this assignment. We tried diligently to communicate and meet with him several times before, during, and after spring break to complete this assignment in a timely manner and he neglected to respond to us nor contribute any work of value to this assignment. Then at the last moment, without our knowledge, proceeded to submit an assignment and take all of the credit claiming that we did not help him at all and did not do any work. We did our best to include code, as well as conceptual knowledge of concepts to respond to these problems. Overall, we both found this group project experience to be extremely non productive and non conducive to applying the knowledge taught in this course. We plan to attend office hours on 3/28 to discuss this further.

In preparation for this experiment, our team used a processes generator to produce 250 processes with varying burst times and memory requirements.

Question 1:

First we created our six processors and opened the file that contains the code necessary to generate the processes. We made a Processes queue and added each line of the csv file to the queue. This way we can implement the FIFO algorithm by dequeuing to each processor. In the FIFO program, we used a for loop to add 42 processes to the first 4 processors and 41 to the last 2 processors. From there, we were stuck on finding out how to access the specific burst times from each processor. Had we figured it out, we would have computed the average turnaround time by adding up every burst time from a single processor and dividing it by the number of processes in that processor. For the wait time, we would take the turnaround time and subtract the final processes burst time, then divide that by the total number of processes in that processor. We would do that for each processor and use Pyplot to make histograms of or findings.

For SJF, we sorted the process csv file and ran the same FIFO algorithm. This would put processes with the least amount of CPU cycles at the top of the queue and they will be distributed first.

from queue import Queue

import csv

import queue

import numpy as np

processes = Queue()

pid = []

cycles = []

memory = []

p1 = []

p2 = []

p3 = []

p4 = []

p5 = []

p6 = []

turn\_time = 0

csvfile = open("processes.csv", "r")

file = csv.DictReader(csvfile)

for process in file:

pid.append(process["Process ID"])

cycles.append(process["CPU Cycles"])

memory.append(process["Memory Requirement"])

processes.put(process)

# FIFO program for question 1 part 1

def fifo():

for i in range(42):

p1.append(processes.get())

p2.append(processes.get())

p3.append(processes.get())

p4.append(processes.get())

for i in range(41):

p5.append(processes.get())

p6.append(processes.get())

wait\_time = 0

turn\_time = 0

for i, value in enumerate(p1):

turn\_time += int(value)

p1\_avg\_turn\_time = turn\_time / len(p1)

print("average wait time is: " + str(p1\_avg\_turn\_time))

Question 2:

Now with the new constraint of different processing cores, we would use the same logic from the SJF algorithm. We’d still sort the processes in the csv file first, then put the jobs with the lowest burst times on the 2 GHz processors (A,B,C) and the processes with high burst times will go to the 4GHz processors (D,E,F). Instead of separating the processes all at once, we’d focus on filling the 3 small processors first, then fill the fast ones.