Scheduler Design Plan

170D WOBC: Module L Exam I (A)

CW2 Kyle Spicer

Due Date: 6 November 2022

1 Project Summary

1.1 Description

In this exercise you will be given a sequence of corresponding job and arrival times. A job is therefore implicitly identified by this data and no other information is provided.

Design and develop a job scheduler application that schedules jobs with the primary goal of minimizing the average wait time.

Schedule a sequence of n jobs so that the queue length and average wait time are minimized.

1.2 Objective

The students are given several files: driver, scheduler.h, and scheduler_test.h. The objective is to create a scheduler.c to complete the function declarations within scheduler.h with an end-state of making the driver file functional.

2 Architecture

2.1 Directories

2.1.1 scheduler (top level directory)

- Makefile

2.1.2 src

provided by instructor: scheduler_driver.c, scheduler_test.h, scheduler.h

student created: scheduler.c

2.1.3 test

2.1.4 doc

design.pdf, writup.pdf, testplan.pdf, scheduler.1

2.2 Structures

2.3 scheduler

queue_t * queue
int * job_tracker
int job_id
int job_count
int finish_count
int sum_job_time;
int sum_arrival_time;
int sum_end_time;
int sum_wait_time;

2.4 queue

int head;
int tail;
int size;
int * data;

3 Program Flow

Students main objective
1. Utilize the scheduler.h and create a functional scheduler.c for the provided function prototypes
2. Complete two main exercises located within the scheduler driver program.

Utilizing the program
1. When the program is executed, a scheduler struct and queue are created to house arrays with job times.
2. The student is responsible for creating the scheduler, queue, as well as the functions to aid in completion.

Driver two main functions
schedule_static - all jobs arrived at t=0, meaning they were all processed in order.
1. First the scheduler was created from the original driver code.

2. When the function was called, the scheduler struct, total number of jobs, jobs array, and a few functions were passed into the static function.
3. I first set the schedulers job count equal to the total number of jobs sent in.
4. created memory to copy the original array and then sort the array from shortest to longest time.
5. Once sorted, I added the elements to my queue(array list) by index.
6. To use the queue as intended, I then dequeued the first job to be able to work with it.
7. I created variables to track the total wait time and to keep track of the current wait time.
8. My while loop kept track of progress and made sure each job was received and counted, prior to completion.
9. Called the report function, which reported the time elapsed and job wait time.
10. Then each job was processed after the allotted wait time and the report was sent after each job was completed.
11. After all jobs were processed and reported, then the average wait time was calculated for reference when called by the driver program.]

schedule_variable - all jobs arrived at various times. Adding complexity to the solution.
1. Similar to the static function, the same arguments were passed into the function, with exception to an additional array to track various wait times of each job.
2. This required a few more variables to track arrival times, index location for the arrays, and job total wait time.
3. Once a job was dequeued, it was processed and the next job had to be tracked according to its arrival time. This means tracking what time it came in and keeping track of how long the job had to wait to be processed, if an overlap of time occurred.
4. Once all jobs were complete, the average wait time was calculated as well.

4 Timeline to Completion

1. Try to dissect the 12 page exercise rubric.(T+0)
2. Attempt to read and understand the starter code provided.(T+0)
3. Prepare documentation and working directories. (T+0)
4. Prepare questions for clarification. (T+0)
5. Begin completing functions to step through each problem within the program. (T+1)
6. Try to have first of three program functions complete. (T+1)
7. Try to have second program function complete. (T+2)
8. Complete final function and unit testing. (T+2)
9. Complete required documentation for project. (T+2)

5 Extra Credit Items

- create manpage reverse.1