CS 3305A: Operating Systems
Department of Computer Science
Western University
Assignment 3
Fall 2018
Due Date: December 2, 2018

Purpose

The goals of this assignment are the following:

- Gain experience with *pthread_mutex_t*
- Understand how to handle critical sections in a multithreaded computing
- Understand void pointers and how to cast them
- Understand different scheduling algorithms
- Gain more experience with the C programming language

Specification for Resource Management Program

In this assignment you are to build on top of the resource management simulator you implemented in assignment 2. The simulator will utilize multithread programming to execute multiple tasks simultaneously while keeping track of the system's memory resources as well as allowing the user to select different scheduling algorithms to process incoming jobs. You will be provided with skeleton code that will initialize the system resources, provide a stack of jobs for you to run, and run those jobs. You will be tasked with adding mutecies to the critical sections in the code so that the code could run as expected. You will also need to complete the get_next_job() function to help you process the jobs in the order that is selected.

Your simulator must handle the following orders:

- First Come First Serve
- Last In First Out
- Shortest Job First
- Round Robin

Provided Files

- Your changes should only be inside the simulate.c, scheduler.c, files, and Makefile
- There are tests provided with the code. You are encouraged to run them and create your own tests as well

Use the following commands to compile and run the program:

Compile the program using:

```
make
```

Run the program using:

```
./myOS simulator input.txt
```

Test the program using:

```
make test
```

Definitions and Function footprints

```
job_t *get_next_job(int, d_linked_list_t*);
```

This is the footprint of the get_next_job function. It will receive an integer that will define the order that jobs will run in and all the jobs that you will need to run.

```
#define FCFS 0
#define LIFO 1
#define SJF 2
#define RR 3
```

Those definitions define all types of orders respectively:

- First Come First Serve
- Last In First Out
- Shortest Job First
- Round Robin

Structures and Typedefs

```
typedef struct
{
      struct d_node *head, *tail;
      int size;
} d_linked_list_t;
```

linked_stack_t defines the stack that will be used to hold the jobs.

```
void enqueue(d_linked_list_t*, void*);
void insert_prev(d_linked_list_t*, struct d_node*, void*);
void insert_next(d_linked_list_t*, struct d_node*, void*);
void* dequeue(d_linked_list_t*);
void* pop(d_linked_list_t*);
```

Hint: these functions will be useful in the get_next_job function

Sample Output

```
MODE: LIFO
Starting job #6
Allocating 30
Memory at 994
Starting job #5
Allocating 20
Memory at 974
Starting job #4
Allocating 140
Memory at 834
Starting job #3
Allocating 130
Memory at 704
Job #6 completed
Deallocating 30
Memory at 734
Job #5 completed
Job #4 completed
Starting job #2
Allocating 10
Memory at 724
Job #3 completed
Deallocating 20
Memory at 744
Starting job #1
Allocating 120
Memory at 624
Deallocating 130
Memory at 754
Deallocating 140
Memory at 894
Job #2 completed
Deallocating 10
Memory at 904
Job #1 completed
Deallocating 120
Memory at 1024
```

Marks Distribution

- Mutex usage in critical sections: 30 marks
- Implementation of get_next_job: 70 marks
    - First Come First Serve: 10 marks
    - First In First Out: 10 marks
    - Job First: 25 marks
    - Round Robin: 25 marks

Assignment Submission Guideline

- Must run on GAUL. Otherwise, you will receive a mark of zero. Refer to the website for further information
- Only .c, .h, and Makefiles. Marks will be deducted if other files are submitted
- DO NOT compress your submission

Resources

- https://www.cs.cmu.edu/afs/cs/academic/class/15492-f07/www/pthreads.html