Tuan Nguyen

**README**

In the project, we implemented a system which includes:

1) priority queue to represent the Job Scheduler

2) FIFO Queue to represent the CPU

3) FIFO Queue to represent two Disks: Disk1 , Disk2

-When it is running, the system is responsible for scheduling jobs in different components based on their event types and write all the events into a log file which also includes jobID and time of each events, we have 6 different types of events:

\*Arrive at CPU

\*Finish at CPU

\*Arrive at Disk

\*Finish at Disk 1/2

\*Exits the system

\*Simulation Finished

-At each iteration, a Job will be popped from the Scheduler. If event type is Arrive at CPU. The CPU also will create a new Job with event type Arrive and send it back to the Scheduler. The arrival time will be decided by generate a random number from a specified range and add it to the currentTime variable (We need to read the parameters that specify the range from a (file.txt): config file. The Job will go to the CPU to wait for processing. There are 2 cases:

    -If the CPU is busying processing other jobs:

we'll do nothing

    -Else we will pop the task at the top of the CPU queue. Create a new event with type equals Finish at CPU and determine their finished time the same way we did previously. We switch the CPU to busy state

\*If event type is Finish at CPU, we will do the following:

    -Set CPU state to not busy

    -with some probability p, we will determine whether a job should leave the system or will be sent to disk for further I/O processing

\*If event type is Arrive at Disk, we will select the Disk with less capacity and push the new job that was created previously to that disk. If the disk is of equal capacity, we select randomly. Also, we need to generate a Disk finished event from the Disk and push it to the Scheduler

\*If event type is Finish at Disk, we determine which disk was responsible for processing the job. Now the job is ready to go back to CPU for further processing (switch from blocked state to ready), we create a new event with type CPU Arrival and send it to the Scheduler, the time of that job was the current time.

**Main.c**

-initialize priority queue

-initialize constant variable (seed, initial time, finish time , etc. )

- add two events to priorty queue, job1 arrival and simulation finished

-begin processing, use while loop

while(eventNodes != 0 && currentTime < FIN\_TIME)

- pop events from event queue and place them inside cpu queue

-set current time

switch(task.type){

case ARRIVAL: process\_CPU(task); break; //arrive CPU

case FINISH: process\_CPU(task); break; //finish CPU

case DISK\_ARRIVAL: process\_DISK(task); break; //arrive Disk

case DISK\_FINISH: process\_DISK(task); break; //finish Disk

case SIMULATION\_END: break; //end simulation

}

**priorityqueue.c**

- typedef struct event{

int time;

int jobNo;

int type;

}event;

typedef struct node{

struct event job;

struct node \*next;

}node;

-function: priority queue add job

void p\_append(node\*\* head\_ref, event data, int\* total)

-function: priority queue to remove job

event p\_removeJob(node\*\* head\_ref, int\* total)

-function: create new job

event createEvent(int time, int jobNo, int type)

-function: create random job

int randomJob(int min, int max)

-function: put all value to out.txt file

void recordEvent(char \*line){

-function: fifo queue add job

void append(node\*\* head\_ref, event data, int\* total)

-function: fifo queue remove job

event removeJob(node\*\* head\_ref, int\* total)

**searchFile.c**

-read value from config.txt file, processing, and output to out.txt file