Name: Michal Hus
Date: 03/20/2016
Language: C++

CPU Scheduler Program and Report

TABLE OF CONTENTS

INTRODUCTION

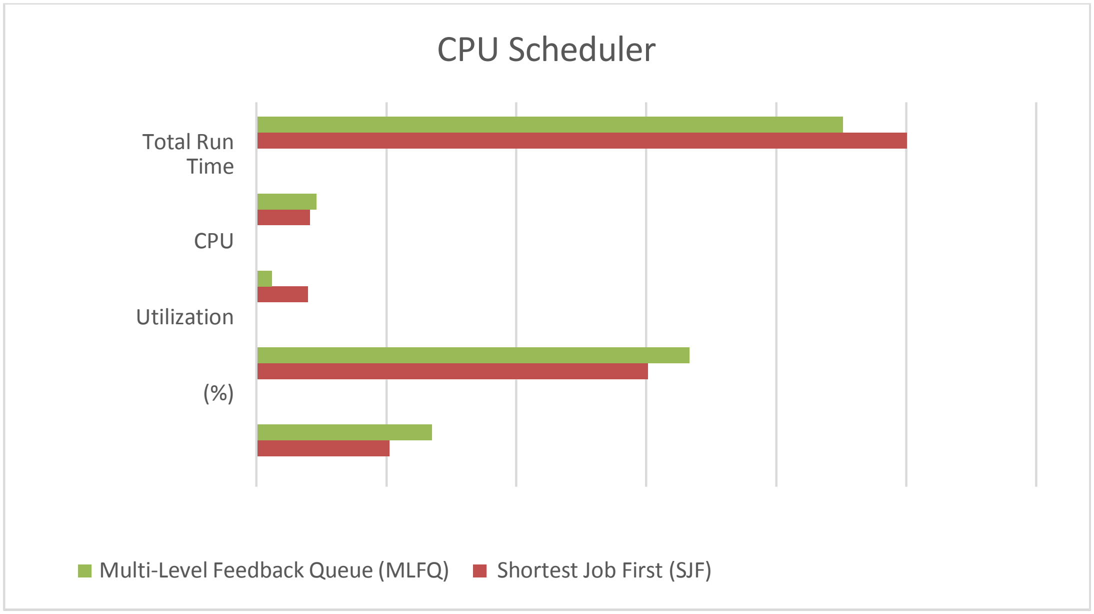
Computer Science industry is in high demands and in need of new intelligent, ambitious, and hardworking innovators. Companies try to look for individual with experience and proper education. Therefore, individuals that need to learn about the operating systems and algorithms used by them. In this report, one will find results of the simulations of SJF and MLFQ CPU scheduler.

MODIFICATIONS

Compare with previously mention methods and ideas, there were no major changed done to the implementation of this simulation. The simulation was write in C++ and was implemented with use of doubly linked-list that forms a circular queue.

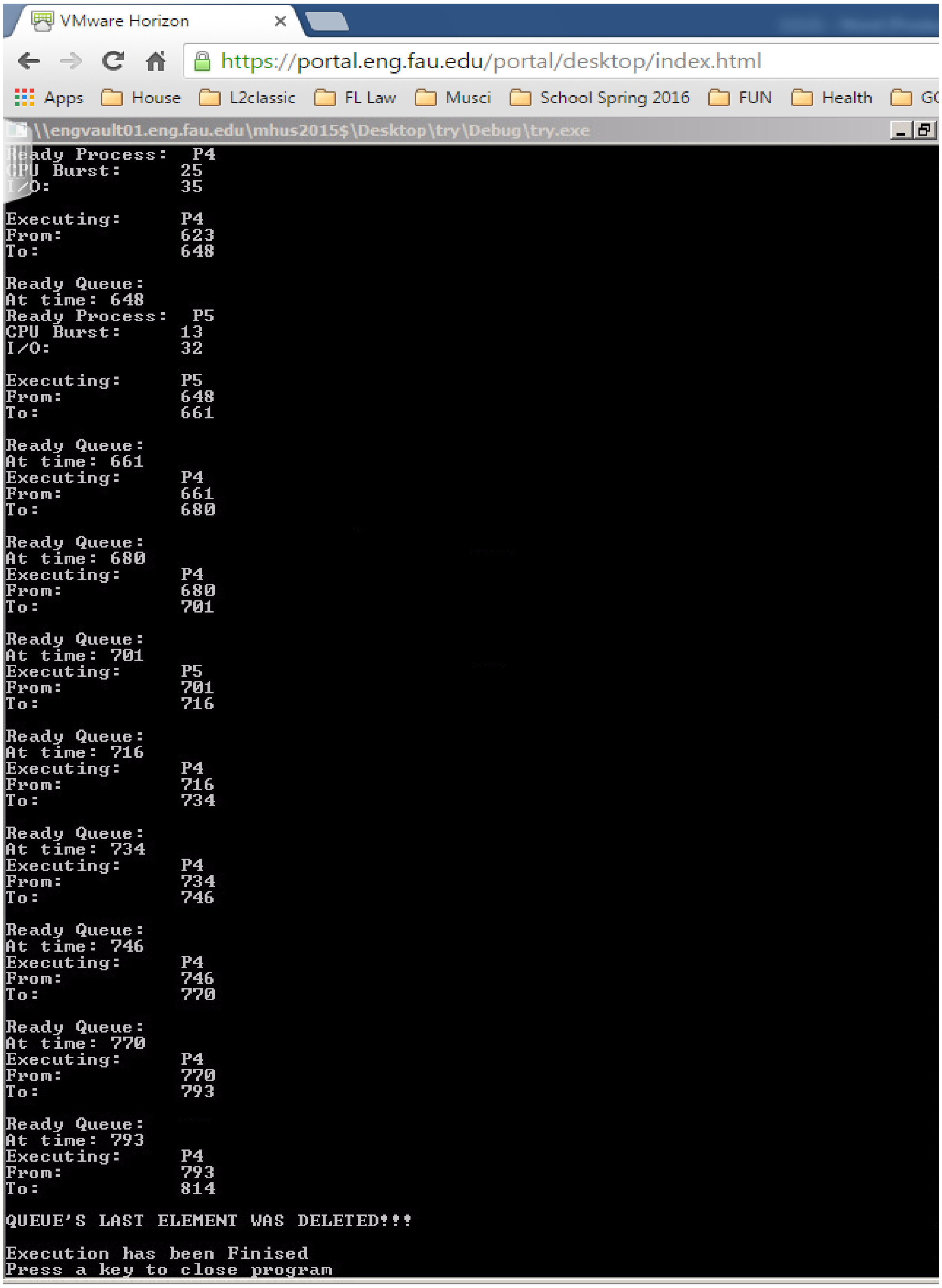
DISCUSSION

Shortest Job First (SJF)			Multi-Level Feedback Queue (MLFQ)		
WT	TT	RT	WT	TT	RT
196	594	82	232	629	27



PROGRAM OUTPUT

SJF


VMware Horizon
https://portal.eng.fau.edu/portal/desktop/index.html
Apps House L2classic FL Law Musci School Spring 2016 FUN Health GC
\\engvault01.eng.fau.edu\mhus2015$\Desktop\try\Debug\try.exe

```
Ready Process:  P4
CPU Burst:     25
I/O:           35

Executing:     P4
From:          623
To:            648

Ready Queue:
At time: 648
Ready Process:  P5
CPU Burst:     13
I/O:           32

Executing:     P5
From:          648
To:            661

Ready Queue:
At time: 661
Executing:     P4
From:          661
To:            680

Ready Queue:
At time: 680
Executing:     P4
From:          680
To:            701

Ready Queue:
At time: 701
Executing:     P5
From:          701
To:            716

Ready Queue:
At time: 716
Executing:     P4
From:          716
To:            734

Ready Queue:
At time: 734
Executing:     P4
From:          734
To:            746

Ready Queue:
At time: 746
Executing:     P4
From:          746
To:            770

Ready Queue:
At time: 770
Executing:     P4
From:          770
To:            793

Ready Queue:
At time: 793
Executing:     P4
From:          793
To:            814

QUEUE'S LAST ELEMENT WAS DELETED!!!

Execution has been Finised
Press a key to close program
```

SOURCE CODE

```
#include <iostream>

using namespace std;

#ifndef BQUEUE_H
#define BQUEUE_H

class bqnode
{
public:

        string proc;
        int CPU;
        int IO;
        int AT;                                         //data type input inside a node
        bqnode *prev, *next;                    //two pointers
};

class BQUEUE
{
public:
        BQUEUE();                                       //const
        ~BQUEUE();                                      //destructor
        BQUEUE(const BQUEUE &);                 //copy const
        void Enqueue(string, int, int, int);            //adds to end of the list
        void Dequeue();                         //removes form beginning of the list
        void Print();                           //prints content
        void Print(int);                        //prints content
        void run(int);
        bqnode * Search(int);
        void Out(bqnode *);

private:
        bqnode *front, *back;           // no need for front pointer?? as its equal to back->next due to
circular Q.
};

#endif
```

```
#include <iostream>
#include "BQUEUE.h"
#include <iomanip>
#include <string>
#include <climits>

using namespace std;
```

```
////////////////////////////////////////////////////////////////////////////////////////////////////////
//Function Name: BQUEUE()
//Precondition: N/A
//Postcondition: back set to 0.
//Description: Default constructor.
////////////////////////////////////////////////////////////////////////////////////////////////////////
BQUEUE::BQUEUE()
{
	back = NULL;
}

////////////////////////////////////////////////////////////////////////////////////////////////////////
//Function Name: ~BQUEUE()
//Precondition: BQUEUE needs to exist, be constructed.
//Postcondition: Deallocates memory.
//Description: Memory is given back (deallocated), prevents leakage.
////////////////////////////////////////////////////////////////////////////////////////////////////////
BQUEUE::~BQUEUE()
{
	while (back != 0)
	{
		Dequeue();
	}
}

////////////////////////////////////////////////////////////////////////////////////////////////////////
//Function Name: Enqueue(int num)
//Precondition: Needs a queue to be defined, consturcted.
//Postcondition:  Adds new element to the back of a queue.
//Description: Inserts a new element at the end of a queue.
////////////////////////////////////////////////////////////////////////////////////////////////////////
void BQUEUE::Enqueue(string process, int AT , int CPU, int IO)
{
	if (back==0)
	{
		back = new bqnode;
		back->proc = process;
		back->AT = AT;
		back->CPU = CPU;
		back->IO = IO;
		back->next = back;
		back->prev = back;
	}
	else
	{
		bqnode *f = back->next;
		back->next = new bqnode;
		back->next->next = f;
		back->next->prev = back;
		f->prev = back->next;
		back = back->next;
		back->proc = process;
		back->AT = AT;
		back->CPU = CPU;
		back->IO = IO;
		}
}
//////////////////////////////////////////////////////////////////////////////////////////////////////////
//Function Name: Dequeue()
//Precondition: needs to be construded, defined queue.
//Postcondition: Element removed from a front of a queue.
//Description: Removes the element at the front of a queue.
//////////////////////////////////////////////////////////////////////////////////////////////////////////
void BQUEUE::Dequeue()
{
if(back == 0)
{
	cout << "Can't remove more as Queue is EMPTY!!" << endl;

}

else if(back == back->next)
{
	back = 0;
	cout << "QUEUE'S LAST ELEMENT WAS DELETED!!!" << endl << endl;
}
else
{
	bqnode*f = back->next;
	f->next->prev = back;
	back->next = f->next;
	f->prev = f->next = 0;
	delete f;
}
}
//////////////////////////////////////////////////////////////////////////////////////////////////////////
//Function Name: Print()
//Precondition: queue need to be defiened before hand.
//Postcondition: displays items in a queue.
//Description: prints out a queue.
//////////////////////////////////////////////////////////////////////////////////////////////////////////

void BQUEUE::Print()
{
	if(back == 0)
	{
		cout << "EMPTY QUEUE" <<endl;
	}
	else
	{
		bqnode *f = back->next;			//pointer to front
				do
		{
			cout << left << setw(15) << f->proc
				<< setw(15) << left << f->AT
				<< setw(30) << left << f->CPU
				<< setw(10) << left << f->IO << endl;
			f = f->next;
		} while (f != back->next);
		cout << endl << endl;
		f=0;
```

```
                delete f;
        }
}
void BQUEUE::Print(int timer)
{
        if (back == 0)
        {
                cout << "EMPTY QUEUE" << endl;
        }
        else
        {
                bqnode *m = back->next;
                bqnode *n = m->next;
                cout << left << "Ready Queue:  " << endl << "At time: " << timer << endl;
                while (n != back)                       //chceks all DLL
                {
                        while (m->proc == n->proc && n != back)
                                                                // if same Procces change to next node
                        {
                                n = n->next;
                        }
                        if (m->AT <= timer)
                        {
                                cout << left << setw(16) << "Ready Process:  " << left << m->proc <<
endl
                                        << setw(16) << left << "CPU Burst: " << left << m->CPU <<
endl
                                        << setw(16) << left << "I/O:  " << left << m->IO << endl <<
endl;
                                m = n;
                        }
                        else
                        {
                                m = n;
                        }
                }
        }
}
void BQUEUE::run(int timer)
{
        //bqnode *f = back->next;                                       //pointer to front
        if (back == NULL)
        {
                return;
        }
        bqnode *x = Search(timer);
        if (x == 0)
        {
                timer++;
                run(timer);
        }
        if (back == NULL)
        {
                return;
        }
        // CONTEXT SWITCH
```

```
          Print(timer);
          cout << left << setw(15) << "Executing:  " << x->proc << endl
                    << setw(15) << left << "From: " << timer << endl
                    << setw(15) << left << "To:  " << timer + x->CPU << endl << endl;
          timer = timer + x->CPU;
          if (x->proc == x->next->proc)
          {
                    x->next->AT = timer + x->IO;                    //needs conditon to update
          }
          Out(x);                              //needs to deque only specified node not at begging.
                    if (back != NULL)
          {
                    run(timer);
          }
}
void BQUEUE::Out(bqnode * ptr)
{
          if (ptr == back->next)
          {
                    Dequeue();
          }
          else if (ptr == back)
          {
                    back->next->prev = back->prev;
                    back = back->prev;
                    back->next = ptr->next;
          }
          else
          {
                    ptr->prev->next = ptr->next;
                    ptr->next->prev = ptr->prev;
                    delete ptr;
          }
}
bqnode * BQUEUE::Search(int timer)                 //searches for process that can be used
{
          bqnode *m = back->next;
          bqnode *n = m->next;

          while (n != back->next)                    //chceks all DLL
          {
                    while (m->proc == n->proc && n != back)   // if same Procces change to next node
                    {
                              n = n->next;
                    }

                    if (m->CPU > n->CPU && n->AT <= timer )
                    {
                              m = n;
                    }
                    else if (n->CPU > m->CPU && m->AT <= timer)
                    {
                              n = n->next;
                    }
                    else if (n->CPU == m->CPU && m->AT <= timer && n->AT <= timer)
                    {
```

```
                if (m->AT < n->AT)
                {
                        n = n->next;
                }
                else if (m->AT > n->AT)
                {
                        m = n;
                }
                else
                {
                        n = n->next;
                }
            }
            else
            {
                n = n->next;
            }
        }
        if (n->CPU == m->CPU && m->AT <= timer && n->AT <= timer)
        {
            if (m->AT <= n->AT)
            {
                return m;
            }
            else
            {
                return n;
            }
        }
        else if (m->CPU < n->CPU && m->AT <= timer)  //error here due to at <=
        {
            return m;
        }
        else if (n->AT <= timer && m->CPU == n->CPU && m->AT > n->AT)
        {
            return n;
        }
        else
        {
            if (n->AT > timer)
            {
                return m;
            }
            else if (m->AT <= timer )
            {
                return n;
            }
            else
            {
                return 0;
            }
        }
}
```

```
#include <iostream>
#include "BQUEUE.h"
#include <climits>

using namespace std;

int main()
{
        int x = INT_MAX;
        int timer = 0;
        BQUEUE  P;

        P.Enqueue("P1", 0, 18, 41);                     //adds to queue
        P.Enqueue("P1", x, 16, 52);                     //adds to queue
        P.Enqueue("P1", x, 19, 31);                     //adds to queue
        P.Enqueue("P1", x, 14, 33);                     //adds to queue
        P.Enqueue("P1", x, 17, 43);                     //adds to queue
        P.Enqueue("P1", x, 19, 66);                     //adds to queue
        P.Enqueue("P1", x, 14, 39);                     //adds to queue
        P.Enqueue("P1", x, 17, 0);              //adds to queue

        P.Enqueue("P2", 0, 8, 32);              //adds to queue
        P.Enqueue("P2", x, 7, 42);              //adds to queue
        P.Enqueue("P2", x, 6, 27);              //adds to queue
        P.Enqueue("P2", x, 17, 41);                     //adds to queue
        P.Enqueue("P2", x, 7, 33);              //adds to queue
        P.Enqueue("P2", x, 11, 43);                     //adds to queue
        P.Enqueue("P2", x, 12, 32);                     //adds to queue
        P.Enqueue("P2", x, 14, 0);              //adds to queue

        P.Enqueue("P3", 0, 6, 51);              //adds to queue
        P.Enqueue("P3", x, 5, 53);              //adds to queue
        P.Enqueue("P3", x, 6, 46);              //adds to queue
        P.Enqueue("P3", x, 9, 32);              //adds to queue
        P.Enqueue("P3", x, 11, 52);                     //adds to queue
        P.Enqueue("P3", x, 4, 61);              //adds to queue
        P.Enqueue("P3", x, 8, 0);               //adds to queue
        P.Enqueue("P4", 0, 25, 35);
        P.Enqueue("P4", x, 19, 41);
        P.Enqueue("P4", x, 21, 45);
        P.Enqueue("P4", x, 18, 51);
        P.Enqueue("P4", x, 12, 61);
        P.Enqueue("P4", x, 24, 54);
        P.Enqueue("P4", x, 23, 61);
        P.Enqueue("P4", x, 21, 0);
        P.Enqueue("P5", 0, 15, 61);
        P.Enqueue("P5", x, 16, 52);
        P.Enqueue("P5", x, 15, 71);
        P.Enqueue("P5", x, 13, 41);
```

```
	P.Enqueue("P5", x, 15, 62);
	P.Enqueue("P5", x, 14, 31);
	P.Enqueue("P5", x, 14, 41);
	P.Enqueue("P5", x, 13, 32);
	P.Enqueue("P5", x, 15, 0);

	P.Enqueue("P6", 0, 6, 25);
	P.Enqueue("P6", x, 5, 31);
	P.Enqueue("P6", x, 6, 32);
	P.Enqueue("P6", x, 5, 41);
	P.Enqueue("P6", x, 4, 81);
	P.Enqueue("P6", x, 8, 39);
	P.Enqueue("P6", x, 11, 42);
	P.Enqueue("P6", x, 5, 0);

	P.Enqueue("P7", 0, 16, 38);
	P.Enqueue("P7", x, 17, 41);
	P.Enqueue("P7", x, 15, 29);
	P.Enqueue("P7", x, 14, 26);
	P.Enqueue("P7", x, 9, 32);
	P.Enqueue("P7", x, 5, 34);
	P.Enqueue("P7", x, 8, 26);
	P.Enqueue("P7", x, 6, 39);
	P.Enqueue("P7", x, 5, 0);

	P.Enqueue("P8", 0, 5, 52);
	P.Enqueue("P8", x, 4, 42);
	P.Enqueue("P8", x, 6, 31);
	P.Enqueue("P8", x, 7, 21);
	P.Enqueue("P8", x, 4, 43);
	P.Enqueue("P8", x, 5, 31);
	P.Enqueue("P8", x, 7, 32);
	P.Enqueue("P8", x, 6, 32);
	P.Enqueue("P8", x, 7, 41);
	P.Enqueue("P8", x, 4, 0);

	P.Enqueue("P9", 0, 11, 37);
	P.Enqueue("P9", x, 12, 41);
	P.Enqueue("P9", x, 6, 41);
	P.Enqueue("P9", x, 4, 48);
	P.Enqueue("P9", x, 6, 41);
	P.Enqueue("P9", x, 5, 29);
	P.Enqueue("P9", x, 4, 26);
	P.Enqueue("P9", x, 5, 31);
	P.Enqueue("P9", x, 3, 0);

	//P.Print();						//prints queue

	P.run(timer);

	cout << "Execution has been Finised" << endl;

	cout << "Press a key to close program";
	cin.get();
	return 0;
}
```