lab5

Bhupinder Singh

April 12, 2019

Contents

1 header.h

```
#include <iostream>


The purpose of this lab is to be familiar with queues.  The lab
uses the concept of queue as the base.  Queue is a data structure
designed to operate in FIFO (First in First out) context.  In queue
elements are inserted from rear end and get removed from front end.
Queue class is container adapter.  Container is an objects that hold
data of same type.  One common application of a queue is to simulate
some real-world process in order to better understand it.  For this
lab, we are modeling the management of blocks of storage used to store files.
```

```
This is the header file that i included in order to store the
maps in a queue.  We were given a choice to either store it in
a circular array or in a linked list.  I chose the linked list
   way as it seemed the most comprehensible to me.
   struct Node{
            std::string data = "";
            Node* next = nullptr; //basic Node declaration
   };

   struct Queue{     //declaring the queue
            void pop(Node*& head);
            void push(Node*& tail, Node*& head, std::string data);
   };
   the functions that take out the value from the queue
   void Queue::pop(Node*& head){
       if (head == nullptr) return;
       Node *temp = head;
       head = head -> next;
       delete temp;
   }
   the fucntion that adds into the queue when called
   void Queue::push(Node*& tail, Node*& head, std::string data){
            Node *newnode = new Node;
            newnode->data = data;
            newnode->next = nullptr;
            if(head==nullptr){head = newnode;}
            else{tail->next=newnode;
                    tail=newnode;}
   }
```

2 lab5.cpp

```
File block handler
Since files can be created and deleted, we use blocks of fixed size
to hold the data in the file.  We need to maintain a set of blocks
that are free to to used for a file, and when it is deleted, to add
those blocks to a queue of files waiting to have thier blocks
freed.  i.e.  There are 2 sets:  free blocks and used blocks.  a file
is a subset of the used blocks.  The queue has the files that are
waiting to be freed.

#include "header.h"
#include <config.h>
#include <FL/Fl_Cairo_Window.H>
#include <FL/Fl_Button.H>
#include <FL/Fl_Value_Input.H>
#include <set>
#include <algorithm>
#include <vector>
#include <map>
#include <sstream>
#include <cmath>
#include <string>

Fl_Input * inputFileName;
Fl_Value_Input * inputBlockNum;
typedef unsigned int BLOCKS;
const int N = 16;  for now, assume 16 blocks on disk
std::set<BLOCKS> allBlocks;
std::set<BLOCKS> usedBlocks;
std::set<BLOCKS> freeBlocks = {1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16};
```

```
int HEIGHT = 400;
int WIDTH = 400;
Queue erase;
Node* head = nullptr;
Node* tail = nullptr;

std::map<std::string,std::set<BLOCKS>> files;
most of the fltk and cairo code was given to us by the instructor
himself.
std::string displayState()
{
                std::cout << "Free Blocks" << std::endl;
                for(auto f: freeBlocks) std::cout << f << " ";
                std::cout << std::endl;
                std::cout << "Used Blocks" << std::endl;
                for(auto f: usedBlocks) std::cout << f << " ";
                std::cout << std::endl;
                std::cout << "File 1 Blocks" << std::endl;
                std::ostringstream oss;

                for(auto f: files)
                {
                        oss << f.first << ": ";
                        for(auto e: f.second)
                                oss << " " << e;
                }
                oss << std::endl;
                 std::cout << oss.str() << std::endl;
                 return oss.str();
}
```

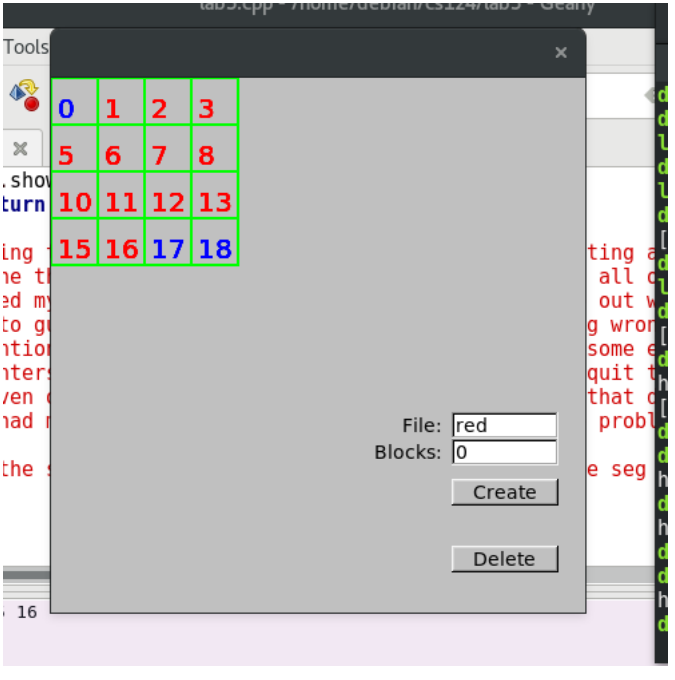
```
void createCB(void*,void*)
{
        std::cout << "Create File" << std::endl;
        Select Users numbers of blocks needed from free set
        Moves those to used set
        std::set<BLOCKS> f;
        std::set<BLOCKS>::iterator i;
        std::string fName = inputFileName -> value();
        int BLKS = 3; int found = 0;
        while(freeBlocks.empty() and found < BLKS)
        {
                i = freeBlocks.find((rand() % N)+1);
                if (i != freeBlocks.end())
                {
                        f.insert(*i);
                        usedBlocks.insert(*i);
                        freeBlocks.erase(*i);
                        found++;
                }
        }

        files[fName] = f;
        displayState();
}
void deleteCB(void*,void*) function that uses the pop fucntion to take away the value
{
        push this fule of blocks on the q (if not full)
        std::cout << "Delete File" << std::endl;
        std::string data = inputFileName->value();
        std::cout << std::endl;
        erase.push(tail, head, data);
        displayState();
```

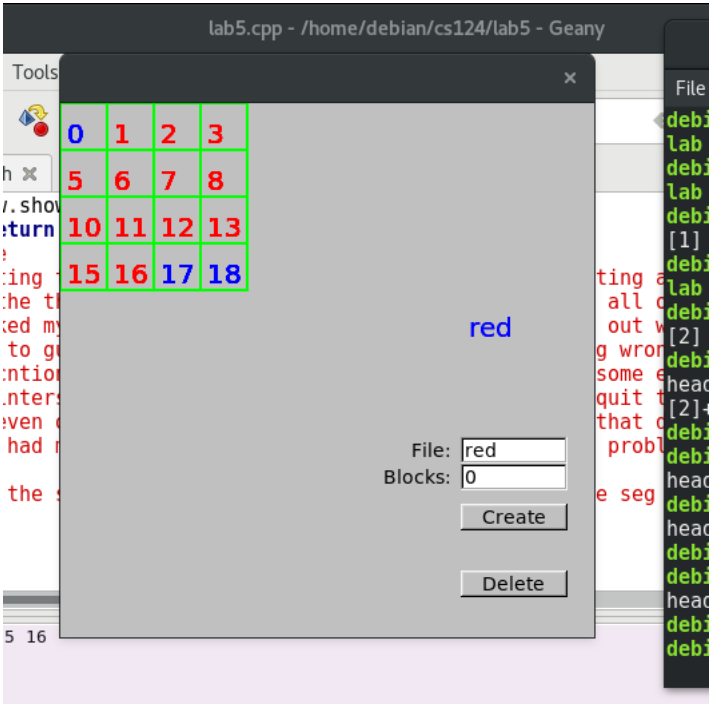
```
}

void callback(void*) function that outputs tick after 10 seconds.
{
                puts("TICK");
                Fl::repeat_timeout(10.0,callback);

}

void drawCB(Fl_Cairo_Window*, cairo_t *cr)
{
    std::cout<< "Drawing" <<std::endl;
    cairo_set_source_rgb(cr,1,0,0);    red green blue percentages
    //cairo_paint(cr);
    //const int N = 8;
    cairo_set_font_size (cr,20);
    const int s = 35;     //20 scale : pixels per unit
    const int offset = 5; //moving text away from edge
    int ROWS = std::sqrt(N); int COLS = ROWS;
    for(int i=0;i<COLS;i++)
    {
        for(int j=0;j<ROWS;j++)
        {
        cairo_set_source_rgb(cr,0,1,0);         //green
        cairo_rectangle(cr,i*s,j*s,s,s);
        cairo_stroke(cr);     //cairo_fill(cr);

        cairo_move_to(cr,i*s+offset,j*s+s-offset);
        int blockNumber = (i+j*(std::sqrt(N)+1));
        std::string b = std::to_string(blockNumber);
        if (freeBlocks.find(blockNumber) != freeBlocks.end())
            cairo_set_source_rgb(cr,1,0,0);         //red
```

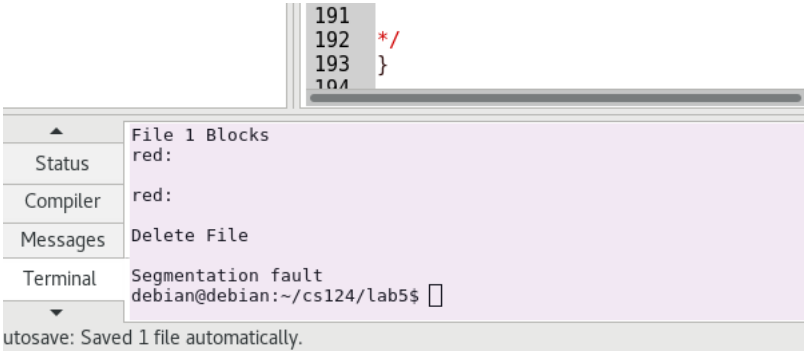
```
        else
            cairo_set_source_rgb(cr,0,0,1);         //blue
        cairo_show_text(cr,b.c_str());
        }
    }
    std::string f;
    cairo_rel_move_to(cr,170,20+20*files.size());
    for(auto e : files){
                if(e.first == inputFileName->value())
                {
                        f = e.first;
                        cairo_show_text(cr,f.c_str());
                        cairo_rel_move_to(cr,-20*f.length(),20);
                }}
    std::cout << displayState() << std::endl;
}

int main()
{
                //Fl::add_timeout(1.0,callback);

                std::set_union(usedBlocks.begin(), usedBlocks.end(),
                    freeBlocks.begin(),
                freeBlocks.end(),std::inserter(allBlocks,allBlocks.begin()))
                    ;

                displayState();
                Fl_Cairo_Window cw(WIDTH,HEIGHT);
                cw.set_draw_cb(drawCB);
                int x = 3*WIDTH/4; int y = 3*HEIGHT/4; int w = WIDTH/5;
                int h = HEIGHT/20;
                const char* tc = "Create";
```

```
Fl_Button bc(x,y,w,h,tc);
bc.callback((Fl_Callback*)createCB);

x = 3*WIDTH/4; y = 7*HEIGHT/8; w = WIDTH/5;
h = HEIGHT/20;
const char* td = "Delete";
Fl_Button bd(x,y,w,h,td);
bd.callback((Fl_Callback*)deleteCB);
inputFileName = new Fl_Input(x,y-100,w,h,"File: ");
inputBlockNum = new Fl_Value_Input(x,y-80,w,h,"Blocks: ");

cw.show();
return Fl::run();
```

After writing the create and the delete code. I kept on getting a segmentation fault.
I was at the the newark lab asking for help from the tutors all of friday.
I even asked my classmates for help and no one could figure out what the problem is.
If i were to guess, i am pretty sure that there is something wrong with my
delete fucntion. I tried going into gdb debugger as i have some experince with it, but
due to pointers and all, it was so confusing that i had to quit that option I have to
attend a even on saturday so i wont be able to do anything that day. I am pretty sure
that if i had more time, i would have eventually solved the problem.
Below are the screenshots of what worked so far and also the seg fault.




lab5.cpp - /home/debian/cs124/lab5 - Geany
Tools
File
0 1 2 3
5 6 7 8
10 11 12 13
15 16 17 18
red
File: red
Blocks: 0
Create
Delete


191
192 */
193 }
Status
Compiler
Messages
Terminal
File 1 Blocks
red:
red:
Delete File
Segmentation fault
debian@debian:~/cs124/lab5$
utosave: Saved 1 file automatically.

```
}
```