Jonathan Anthony

Homework 4

3/7/17

(15 points)
1.Consider the following snapshot of a system:

Process	Allocation	Max	Available
	A B C D	A B C D	A B C D
P0	0 0 1 2	1 0 1 2	2 5 2 0
P1	1 0 0 0	1 7 5 0	
P2	1 3 5 4	2 3 5 6	
P3	0 6 3 2	0 6 5 2	
P4	0 0 1 4	0 6 5 6	

Answer the following questions using the banker's algorithm.

a. What is the content of the matrix **Need**?
b. Is the system in a safe state? Why?
c. If a request from process P1 arrives for (0, 4, 2, 0), can the request be granted immediately?

1.a
Need

	A	B	C	D
P0	1	0	0	0
P1	0	7	5	0
P2	1	0	0	2
P3	0	0	2	0
P4	0	6	4	2

1.b
Yes, the system is in a safe state. There are available resources that is required to complete processes P1 and P3. When these processes complete, the resources will allow the completion of all remaining processes.

1.c

Yes, the state of the system would be as follows

Process	**Allocation**	**Need**	**Available**
	A B C D	**A B C D**	**A B C D**
P0	0 0 1 2	1 0 0 0	2 1 0 0
P1	1 4 2 0	0 3 3 0	
P2	1 3 5 4	1 0 0 2	
P3	0 6 3 2	0 0 2 0	
P4	0 0 1 4	0 6 4 2	

Complete P0:

Process	**Allocation**	**Need**	**Available**
	A B C D	**A B C D**	**A B C D**
P1	1 4 2 0	0 3 3 0	2 1 1 2
P2	1 3 5 4	1 0 0 2	
P3	0 6 3 2	0 0 2 0	
P4	0 0 1 4	0 6 4 2	

Compete P2:

Process	**Allocation**	**Need**	**Available**
	A B C D	**A B C D**	**A B C D**
P1	1 4 2 0	0 3 3 0	3 4 6 6
P3	0 6 3 2	0 0 2 0	
P4	0 0 1 4	0 6 4 2	

Complete P3:

Process	**Allocation**	**Need**	**Available**
	A B C D	**A B C D**	**A B C D**
P1	1 4 2 0	0 3 3 0	3 10 9 8
P4	0 0 1 4	0 6 4 2	

Complete P4:

Process	**Allocation**	**Need**	**Available**
	A B C D	**A B C D**	**A B C D**
P1	1 4 2 0	0 3 3 0	3 10 10 12

Complete P1:

Process	**Allocation**	**Need**	**Available**
	A B C D	**A B C D**	**A B C D**
P1			4 14 12 12

2. (15 points)
Consider a swapping system in which memory consists of the following hole sizes in memory order: 16K, 14K, 4K, 20K, 18K, 7K, 9K, 12K, and 15K. Which hole is taken for successive segment requests of

(a) 12K
(b) 10K
(c) 9K

for first fit? Now repeat the question for best fit, worst fit, and next fit.

First Fit:

H1	**H2**	**H3**	**H4**	**H5**	**H6**	**H7**	**H8**	**H9**
16	14	4	20	18	7	9	12	15

A ⇒H1

H1	**H2**	**H3**	**H4**	**H5**	**H6**	**H7**	**H8**	**H9**
4	14	4	20	18	7	9	12	15

B ⇒H2

H1	**H2**	**H3**	**H4**	**H5**	**H6**	**H7**	**H8**	**H9**
4	**4**	4	20	18	7	9	12	15

C ⇒H3

H1	**H2**	**H3**	**H4**	**H5**	**H6**	**H7**	**H8**	**H9**
4	4	4	**11**	18	7	9	12	15

Best Fit:

H1	**H2**	**H3**	**H4**	**H5**	**H6**	**H7**	**H8**	**H9**
16	14	4	20	18	7	9	12	15

A ⇒H8

H1	**H2**	**H3**	**H4**	**H5**	**H6**	**H7**	**H8**	**H9**
16	14	4	20	18	7	9	0	15

B ⇒H2

H1	**H2**	**H3**	**H4**	**H5**	**H6**	**H7**	**H8**	**H9**
16	**4**	4	20	18	7	9	0	15

C ⇒H7

H1	**H2**	**H3**	**H4**	**H5**	**H6**	**H7**	**H8**	**H9**
16	4	4	20	18	7	0	0	15

Worst Fit:

H1	**H2**	**H3**	**H4**	**H5**	**H6**	**H7**	**H8**	**H9**
16	14	4	20	18	7	9	12	15

A ⇒H4

H1	**H2**	**H3**	**H4**	**H5**	**H6**	**H7**	**H8**	**H9**
16	14	4	**8**	18	7	9	12	15

B ⇒H5

H1	**H2**	**H3**	**H4**	**H5**	**H6**	**H7**	**H8**	**H9**
16	14	4	8	**8**	7	9	12	15

C ⇒H1

H1	H2	H3	H4	H5	H6	H7	H8	H9
5	14	4	8	18	7	9	12	15

Next Fit:

H1	H2	H3	H4	H5	H6	H7	H8	H9
16	14	4	20	18	7	9	12	15

A ⇒H1

H1	H2	H3	H4	H5	H6	H7	H8	H9
4	14	4	20	18	7	9	12	15

B ⇒H2

H1	H2	H3	H4	H5	H6	H7	H8	H9
4	4	4	20	18	7	9	12	15

C ⇒H4

H1	H2	H3	H4	H5	H6	H7	H8	H9
4	4	4	11	18	7	9	12	15

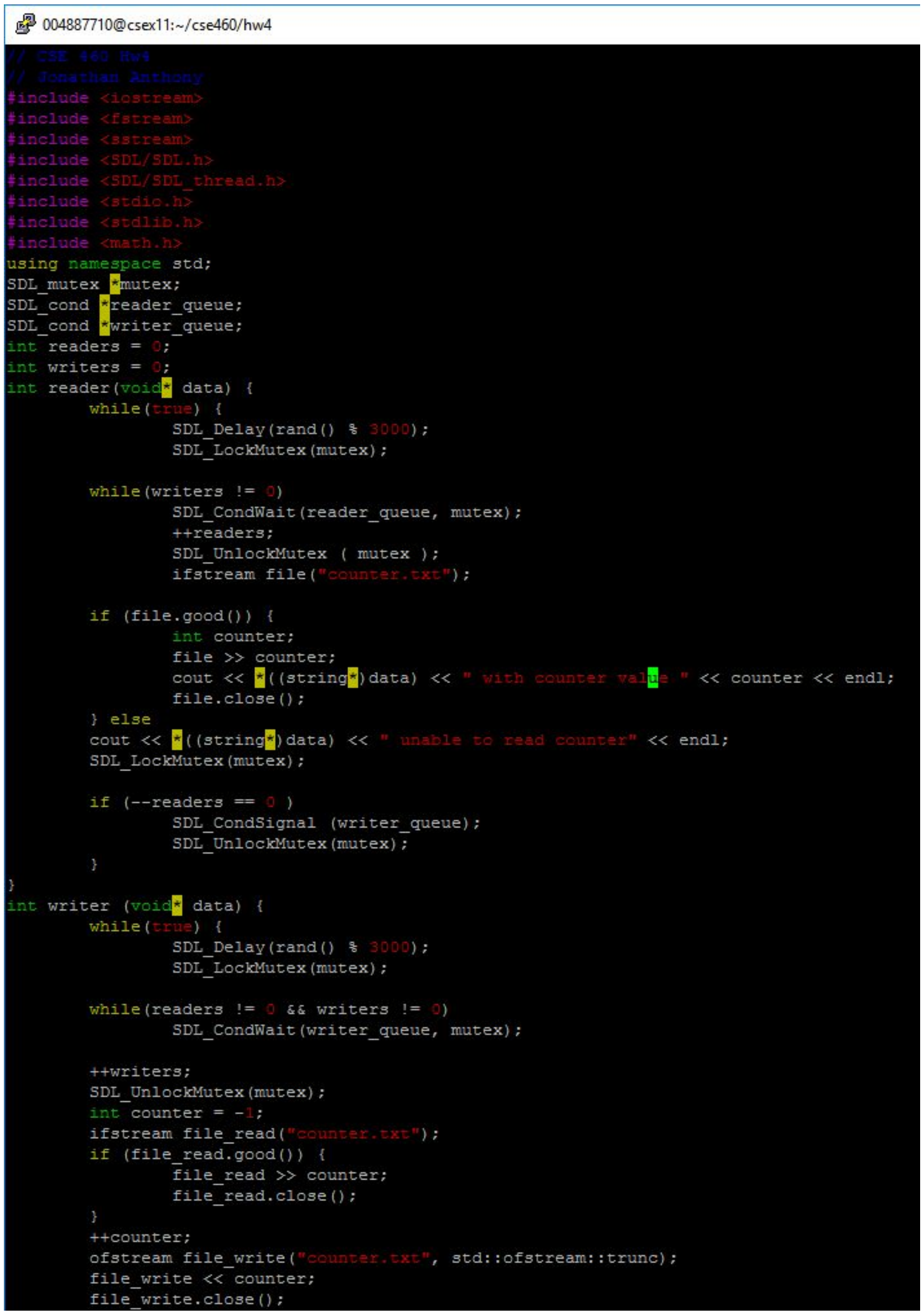
3.(20 points)
In the class we mentioned briefly the readers-writers problem with **writers priority**. The problem can be solved in guarded commands as follows:

```
• void reader()
  {
    when ( writers ==
  0 ) [
      readers++;
    ]

    //read

    [readers--;]
  }
```

```
• void writer()
  {
    [writers++;]
    when ( (readers == 0) && (active_writers ==
  0) )[
      active_writers++;
    ]

    //write

    [writers--;  active_writers--;]
  }
```

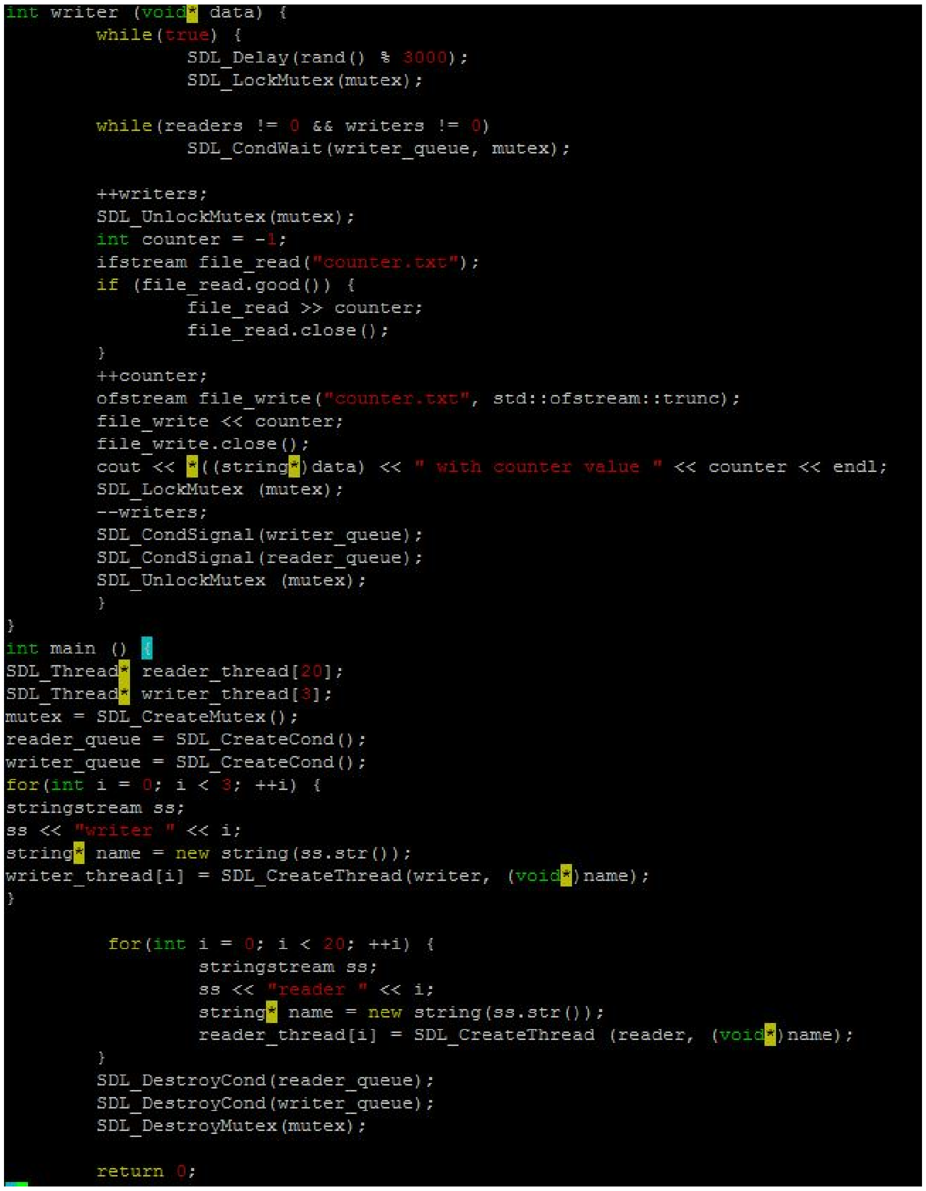
Here *writers* represents the number of threads that are either writing or waiting to write. The variable *active_writers* represents the number of threads (0 or 1) that are currently writing. Implement the solution using either POSIX threads or SDL threads. Again simulate the tasks by reading from and writing to a file named *counter.txt* as in problem 4 of Homework 3

3. Code:


004887710@csex11:~/cse460/hw4

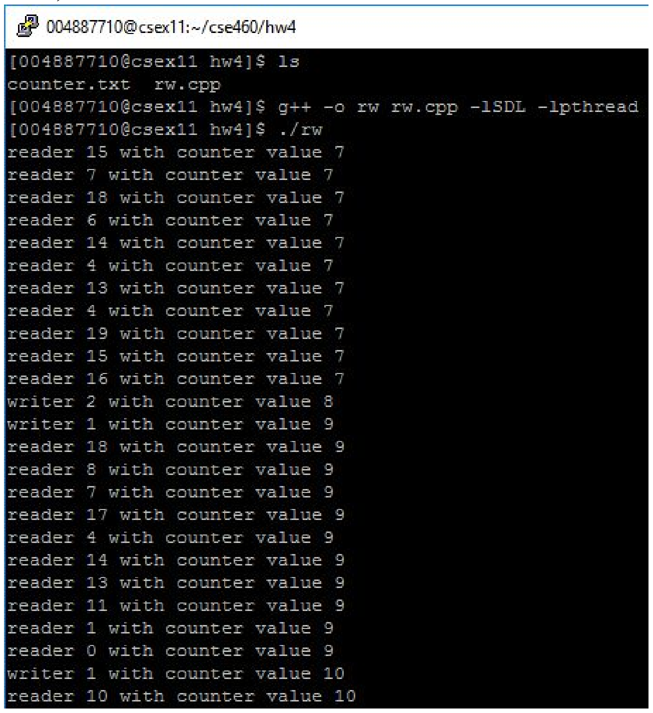
```
// CSE 460 Hw4
// Jonathan Anthony
#include <iostream>
#include <fstream>
#include <sstream>
#include <SDL/SDL.h>
#include <SDL/SDL_thread.h>
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
using namespace std;
SDL_mutex *mutex;
SDL_cond *reader_queue;
SDL_cond *writer_queue;
int readers = 0;
int writers = 0;
int reader(void* data) {
        while(true) {
                SDL_Delay(rand() % 3000);
                SDL_LockMutex(mutex);

        while(writers != 0)
                SDL_CondWait(reader_queue, mutex);
                ++readers;
                SDL_UnlockMutex ( mutex );
                ifstream file("counter.txt");

        if (file.good()) {
                int counter;
                file >> counter;
                cout << *((string*)data) << " with counter value " << counter << endl;
                file.close();
        } else
        cout << *((string*)data) << " unable to read counter" << endl;
        SDL_LockMutex(mutex);

        if (--readers == 0 )
                SDL_CondSignal (writer_queue);
                SDL_UnlockMutex(mutex);
        }
}
int writer (void* data) {
        while(true) {
                SDL_Delay(rand() % 3000);
                SDL_LockMutex(mutex);

        while(readers != 0 && writers != 0)
                SDL_CondWait(writer_queue, mutex);

        ++writers;
        SDL_UnlockMutex(mutex);
        int counter = -1;
        ifstream file_read("counter.txt");
        if (file_read.good()) {
                file_read >> counter;
                file_read.close();
        }
        ++counter;
        ofstream file_write("counter.txt", std::ofstream::trunc);
        file_write << counter;
        file_write.close();
```

3. Code cont.d:



```
int writer (void* data) {
        while(true) {
                SDL_Delay(rand() % 3000);
                SDL_LockMutex(mutex);

        while(readers != 0 && writers != 0)
                SDL_CondWait(writer_queue, mutex);

        ++writers;
        SDL_UnlockMutex(mutex);
        int counter = -1;
        ifstream file_read("counter.txt");
        if (file_read.good()) {
                file_read >> counter;
                file_read.close();
        }
        ++counter;
        ofstream file_write("counter.txt", std::ofstream::trunc);
        file_write << counter;
        file_write.close();
        cout << *((string*)data) << " with counter value " << counter << endl;
        SDL_LockMutex (mutex);
        --writers;
        SDL_CondSignal(writer_queue);
        SDL_CondSignal(reader_queue);
        SDL_UnlockMutex (mutex);
        }
}
int main () {
SDL_Thread* reader_thread[20];
SDL_Thread* writer_thread[3];
mutex = SDL_CreateMutex();
reader_queue = SDL_CreateCond();
writer_queue = SDL_CreateCond();
for(int i = 0; i < 3; ++i) {
stringstream ss;
ss << "writer " << i;
string* name = new string(ss.str());
writer_thread[i] = SDL_CreateThread(writer, (void*)name);
}

         for(int i = 0; i < 20; ++i) {
                 stringstream ss;
                 ss << "reader " << i;
                 string* name = new string(ss.str());
                 reader_thread[i] = SDL_CreateThread (reader, (void*)name);
        }
        SDL_DestroyCond(reader_queue);
        SDL_DestroyCond(writer_queue);
        SDL_DestroyMutex(mutex);

        return 0;
```

3. Output. (counter.txt is 07)



```
[004887710@csex11 hw4]$ ls
counter.txt  rw.cpp
[004887710@csex11 hw4]$ g++ -o rw rw.cpp -lSDL -lpthread
[004887710@csex11 hw4]$ ./rw
reader 15 with counter value 7
reader 7 with counter value 7
reader 18 with counter value 7
reader 6 with counter value 7
reader 14 with counter value 7
reader 4 with counter value 7
reader 13 with counter value 7
reader 4 with counter value 7
reader 19 with counter value 7
reader 15 with counter value 7
reader 16 with counter value 7
writer 2 with counter value 8
writer 1 with counter value 9
reader 18 with counter value 9
reader 8 with counter value 9
reader 7 with counter value 9
reader 17 with counter value 9
reader 4 with counter value 9
reader 14 with counter value 9
reader 13 with counter value 9
reader 11 with counter value 9
reader 1 with counter value 9
reader 0 with counter value 9
writer 1 with counter value 10
reader 10 with counter value 10
```