Dean Cosanella
Flavio dos Santos-Ross

CSE 460

Dr. Tong Lai Yu

Lab 8 Report

Dean Cosanella
Flavio dos Santos-Ross
CSE 460

1. Introduction to Thread Programming

Pthreads:

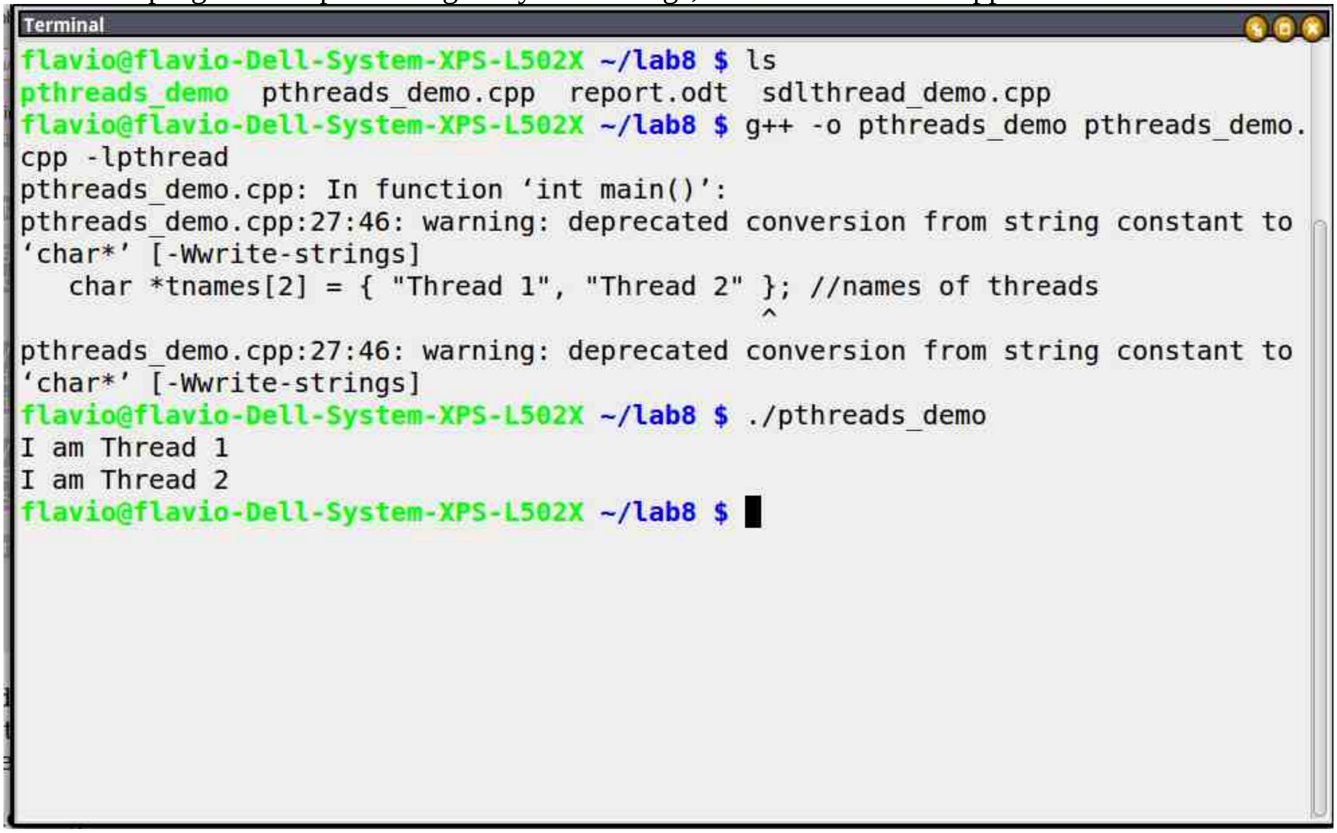
You must include **pthread.h** and link with **-l pthread**. The following code shows how to use **Pthreads**.

```
/*pthreads_demo.cpp
A very simple example demonstrating the usage of pthreads.
Compile: g++ -o pthreads_demo pthreads_demo.cpp -lpthread
Execute: ./pthreads_demo
*/

#include <pthread.h>
#include <stdio.h>

using namespace std;

//The thread
void *runner ( void *data )
{
  char *tname = ( char * )data;

  printf("I am %s\n", tname );

  pthread_exit ( 0 );
}

int main ()
{
  pthread_t id1, id2;           //thread identifiers
  pthread_attr_t attr1, attr2;  //set of thread attributes
  char *tnames[2] = { "Thread 1", "Thread 2" }; //names of threads
  //get the default attributes
  pthread_attr_init ( &attr1 );
  pthread_attr_init ( &attr2 );

  //create the threads
  pthread_create ( &id1, &attr1, runner, tnames[0] );
  pthread_create ( &id2, &attr2, runner, tnames[1] );

  //wait for the threads to exit
  pthread_join ( id1, NULL );
  pthread_join ( id2, NULL );

  return 0;
}
```

Dean Cosanella
Flavio dos Santos-Ross
CSE 460

Screenshot:

Note: The program compiles and gives you warnings, but it does what it suppose to:



```
flavio@flavio-Dell-System-XPS-L502X ~/lab8 $ ls
pthreads_demo  pthreads_demo.cpp  report.odt  sdlthread_demo.cpp
flavio@flavio-Dell-System-XPS-L502X ~/lab8 $ g++ -o pthreads_demo pthreads_demo.
cpp -lpthread
pthreads_demo.cpp: In function ‘int main()’:
pthreads_demo.cpp:27:46: warning: deprecated conversion from string constant to
‘char*’ [-Wwrite-strings]
   char *tnames[2] = { "Thread 1", "Thread 2" }; //names of threads
                                              ^
pthreads_demo.cpp:27:46: warning: deprecated conversion from string constant to
‘char*’ [-Wwrite-strings]
flavio@flavio-Dell-System-XPS-L502X ~/lab8 $ ./pthreads_demo
I am Thread 1
I am Thread 2
flavio@flavio-Dell-System-XPS-L502X ~/lab8 $ █
```

The above code can be explained as follows:

- **pthread_tid** to declare the identifiers for the threads we are going to create.
- **pthread_attr_t** to declare the attributes of the threads and set the attributes in the function call by **pthread_attr_init()**.
- **pthread_create()** is used to create a separate thread. In addition to passing the thread idenfifier and the attributes to the thread, we also pass the name of the function, **runner**, where the new thread will begin execution.
- **pthread_create()** in the example is a string parameter containing the name of the thread. At this point, the program has three threads: the initial parent thread in **main()** and two child threads in **runner()**. After creating the child threads, the **main()** thread will wait for the **runner()** threads to complete by calling **pthread_join()** function.

SDL Threads:

The mechanisms of using SDL Threads are basically the same as that of Pthreads. You start new threads with **SDL_CreateThread()** function, which returns a thread handle of type **SDL_Thread** for subsequent thread operations.

The above **Pthreads** example can be rewritten using SDL threads as follows:

```
/*
sdlthread_demo.cpp
A very simple example demonstrating the usage of sdl threads.
Compile:  g++ -o sdlthread_demo sdlthread_demo.cpp -lSDL -lpthread
Execute:  ./sdlthread_demo
*/

#include <SDL/SDL.h>
#include <SDL/SDL_thread.h>
#include <stdio.h>

using namespace std;

//The thread
int runner ( void *data )
{
  char *tname = ( char * )data;

  printf("I am %s\n", tname );
  return 0;
}

int main ()
{
  SDL_Thread *id1, *id2;                //thread identifiers
  char *tnames[2] = { "Thread 1", "Thread 2" }; //names of threads


  //create the threads
  id1 = SDL_CreateThread ( runner, tnames[0] );
  id2 = SDL_CreateThread ( runner, tnames[1] );

  //wait for the threads to exit
  SDL_WaitThread ( id1, NULL );
  SDL_WaitThread ( id2, NULL );

  return 0;
}
```

The **SDL_WaitThread()** function works in the same way as the Pthreads **pthread_join()**, which waits a thread to complete.

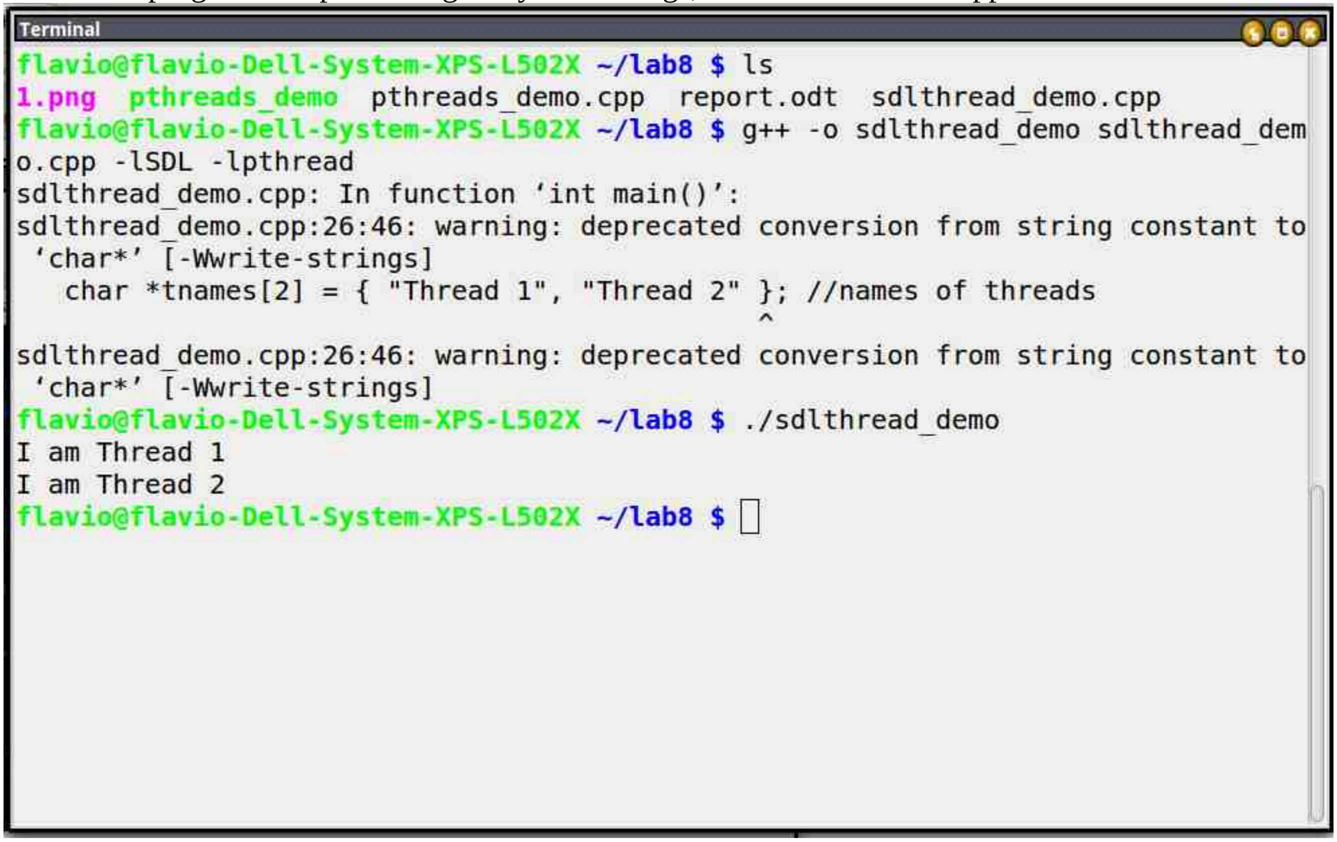
output after compilation:

```
flavio@flavio-Dell-System-XPS-L502X ~/lab8 $ g++ -o sdlthread_demo sdlthread_demo.cpp -lSDL -lpthread
flavio@flavio-Dell-System-XPS-L502X ~/lab8 $ ./sdlthread_demo
```

Dean Cosanella
Flavio dos Santos-Ross
CSE 460

I am Thread 1
I am Thread 2

Screenshot:
Note: The program compiles and gives you warnings, but it does what it suppose to:



```
flavio@flavio-Dell-System-XPS-L502X ~/lab8 $ ls
1.png  pthreads_demo  pthreads_demo.cpp  report.odt  sdlthread_demo.cpp
flavio@flavio-Dell-System-XPS-L502X ~/lab8 $ g++ -o sdlthread_demo sdlthread_dem
o.cpp -lSDL -lpthread
sdlthread_demo.cpp: In function ‘int main()’:
sdlthread_demo.cpp:26:46: warning: deprecated conversion from string constant to
 ‘char*’ [-Wwrite-strings]
   char *tnames[2] = { "Thread 1", "Thread 2" }; //names of threads
                                              ^
sdlthread_demo.cpp:26:46: warning: deprecated conversion from string constant to
 ‘char*’ [-Wwrite-strings]
flavio@flavio-Dell-System-XPS-L502X ~/lab8 $ ./sdlthread_demo
I am Thread 1
I am Thread 2
flavio@flavio-Dell-System-XPS-L502X ~/lab8 $ 
```

2. Modify the **pthreads.cpp** and **sdlthreads_demo.cpp** programs so that they run 3 threads (instead of two) and each thread runs a different function, displaying a different message:

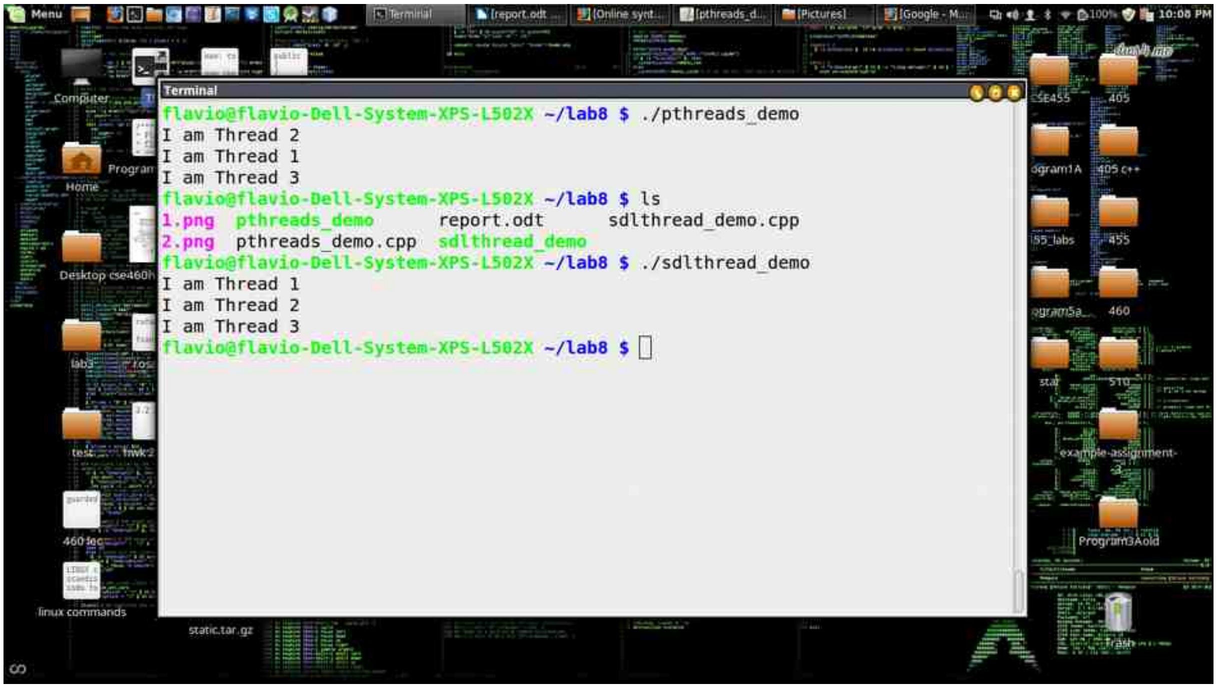
Code **sdlthreads_demo.cpp**: - See changes in yellow:

```
/*sdlthread_demo.cpp
A very simple example demonstrating the usage of sdl threads.
Compile:  g++ -o sdlthread_demo sdlthread_demo.cpp -lSDL -lpthread
Execute:  ./sdlthread_demo
*/

#include <SDL/SDL.h>
#include <SDL/SDL_thread.h>
#include <stdio.h>

using namespace std;
```

Dean Cosanella
Flavio dos Santos-Ross
CSE 460

```
//The thread
int runner ( void *data )
{
  char *tname = ( char * )data;

  printf("I am %s\n", tname );
  return 0;
}

int main ()
{
  SDL_Thread *id1, *id2, *id3;                                 //thread 
identifiers
  char *tnames[3] = { "Thread 1", "Thread 2", "Thread 3" };     //names of threads


  //create the threads
  id1 = SDL_CreateThread ( runner, tnames[0] );
  id2 = SDL_CreateThread ( runner, tnames[1] );
  id3 = SDL_CreateThread ( runner, tnames[2] );


  //wait for the threads to exit
  SDL_WaitThread ( id1, NULL );
  SDL_WaitThread ( id2, NULL );
  SDL_WaitThread ( id3, NULL );

  return 0;
}
```

output after compilation:



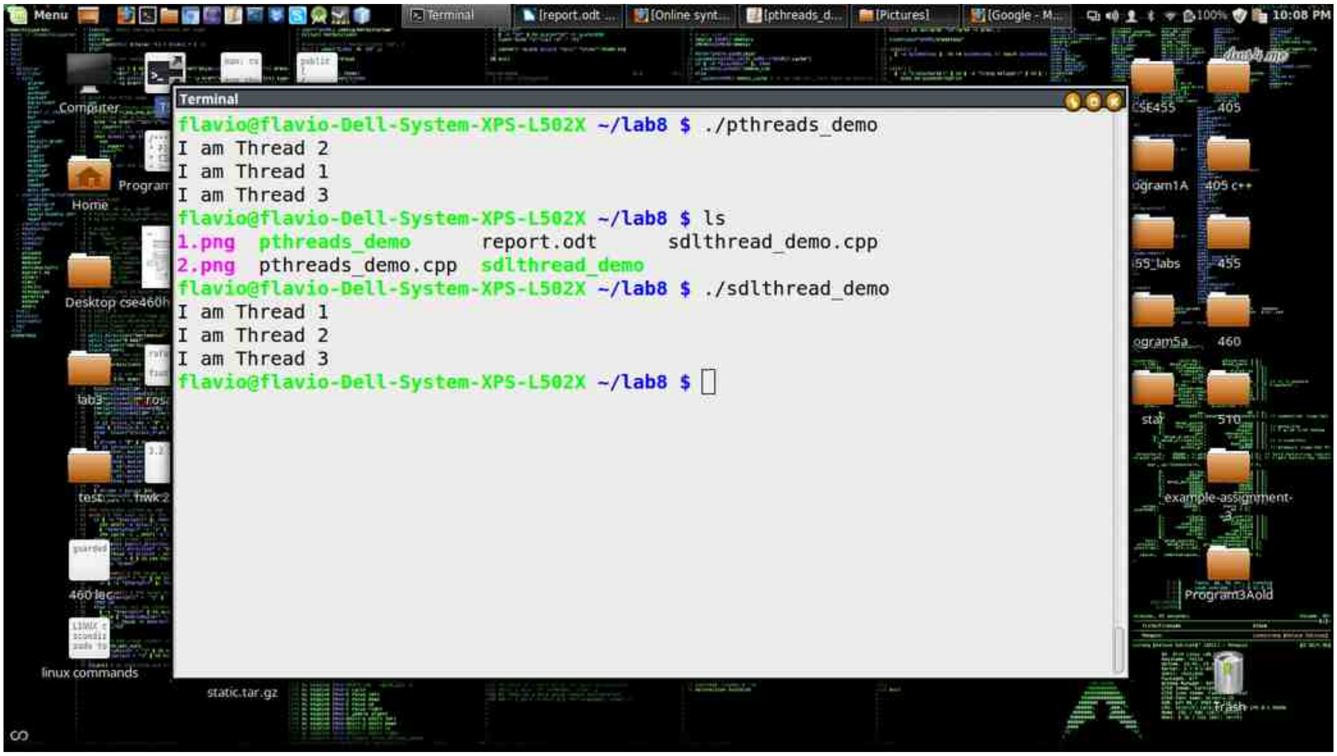
```
flavio@flavio-Dell-System-XPS-L502X ~/lab8 $ ./pthreads_demo
I am Thread 2
I am Thread 1
I am Thread 3
flavio@flavio-Dell-System-XPS-L502X ~/lab8 $ ls
1.png  pthreads_demo      report.odt      sdlthread_demo.cpp
2.png  pthreads_demo.cpp  sdlthread_demo
flavio@flavio-Dell-System-XPS-L502X ~/lab8 $ ./sdlthread_demo
I am Thread 1
I am Thread 2
I am Thread 3
flavio@flavio-Dell-System-XPS-L502X ~/lab8 $ 
```

Dean Cosanella
Flavio dos Santos-Ross
CSE 460

Code **pthreads.cpp**: - See changes in yellow:

```
/*pthreads_demo.cpp
A very simple example demonstrating the usage of pthreads.
Compile: g++ -o pthreads_demo pthreads_demo.cpp -lpthread
Execute: ./pthreads_demo
*/

#include <pthread.h>
#include <stdio.h>

using namespace std;

//The thread
void *runner ( void *data )
{
  char *tname = ( char * )data;

  printf("I am %s\n", tname );

  pthread_exit ( 0 );
}

int main ()
{
  pthread_t id1, id2, id3;             //thread identifiers
  pthread_attr_t attr1, attr2, attr3;  //set of thread attributes
  char *tnames[3] = { "Thread 1", "Thread 2", "Thread 3" }; //names of threads
  //get the default attributes
  pthread_attr_init ( &attr1 );
  pthread_attr_init ( &attr2 );
  pthread_attr_init ( &attr3 );
  //create the threads
  pthread_create ( &id1, &attr1, runner, tnames[0] );
  pthread_create ( &id2, &attr2, runner, tnames[1] );
  pthread_create ( &id3, &attr3, runner, tnames[2] );
  //wait for the threads to exit
  pthread_join ( id1, NULL );
  pthread_join ( id2, NULL );
  pthread_join ( id3, NULL );
  return 0;
}
```

Dean Cosanella
Flavio dos Santos-Ross
CSE 460



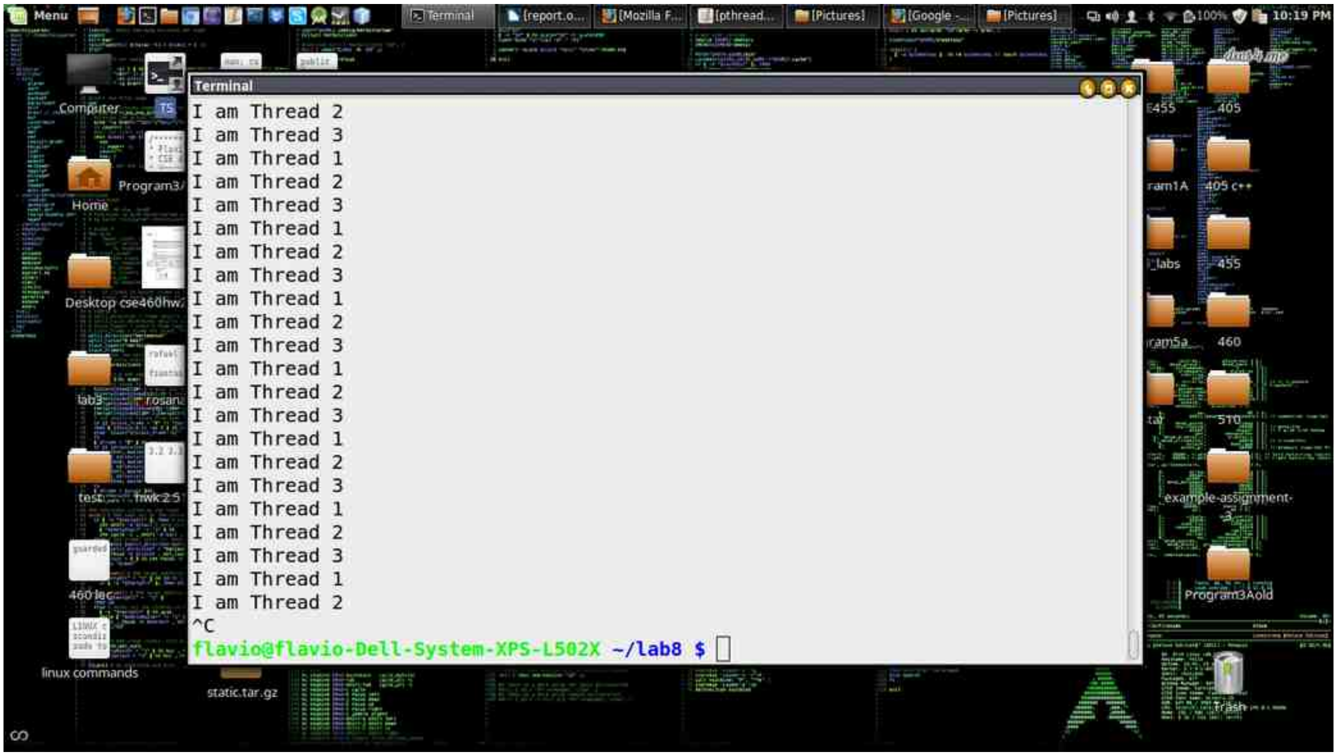
```
flavio@flavio-Dell-System-XPS-L502X ~/lab8 $ ./pthreads_demo
I am Thread 2
I am Thread 1
I am Thread 3
flavio@flavio-Dell-System-XPS-L502X ~/lab8 $ ls
1.png  pthreads_demo      report.odt      sdlthread_demo.cpp
2.png  pthreads_demo.cpp  sdlthread_demo
flavio@flavio-Dell-System-XPS-L502X ~/lab8 $ ./sdlthread_demo
I am Thread 1
I am Thread 2
I am Thread 3
flavio@flavio-Dell-System-XPS-L502X ~/lab8 $ 
```

Use SDL threads to create a thread that runs in an infinite loop after printing out a message:
pthreads_demo.cpp – See changes in yellow:

```
for (int i=0; i<1; i++)
{
        //create the threads
        pthread_create ( &id1, &attr1, runner, tnames[0] );
        pthread_create ( &id2, &attr2, runner, tnames[1] );
        pthread_create ( &id3, &attr3, runner, tnames[2] );

        //wait for the threads to exit
        pthread_join ( id1, NULL );
        pthread_join ( id2, NULL );
        pthread_join ( id3, NULL );

        i--;
 }
  return 0;
```

Dean Cosanella
Flavio dos Santos-Ross
CSE 460

Which gives an infinite loop after compiling:



```
I am Thread 2
I am Thread 3
I am Thread 1
I am Thread 2
I am Thread 3
I am Thread 1
I am Thread 2
I am Thread 3
I am Thread 1
I am Thread 2
I am Thread 3
I am Thread 1
I am Thread 2
I am Thread 3
I am Thread 1
I am Thread 2
I am Thread 3
I am Thread 1
I am Thread 2
I am Thread 3
I am Thread 1
I am Thread 2
^C
flavio@flavio-Dell-System-XPS-L502X ~/lab8 $
```

3. Use SDL threads to create a thread that runs in an infinite loop after printing out a message. The parent uses **SDL_KillThread**() to kill the thread; it then displays a message and terminates.

For this problem, we created a new file called sdlkill_demo.cpp. The code for this problem is shown below:

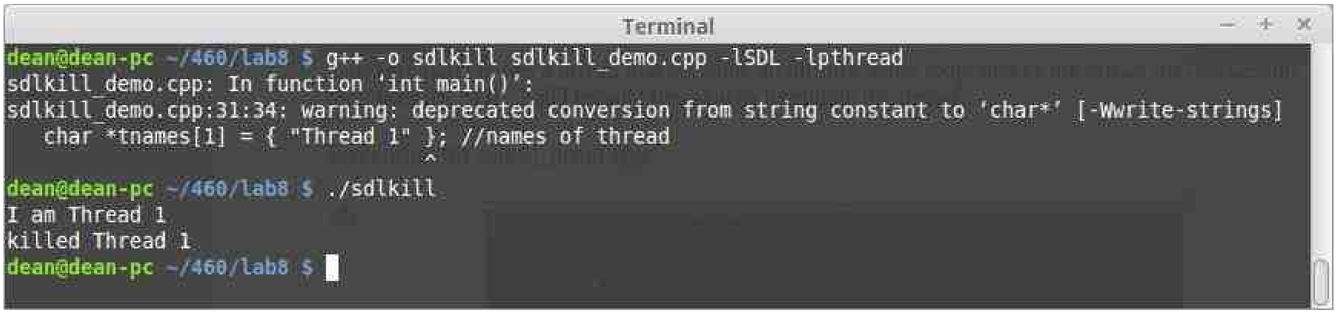
```
/*
File:     sdlkill_demo.cpp
Compile:  g++ -o sdlkill sdlkill_demo.cpp -lSDL -lpthread
Execute:  ./sdlkill_demo
*/

#include <SDL/SDL.h>
#include <SDL/SDL_thread.h>
#include <stdio.h>
#include <string>

using namespace std;

//The thread
int runner ( void *data )
{
  char *tname = ( char * )data;
```

Dean Cosanella
Flavio dos Santos-Ross
CSE 460

```
  printf("I am %s\n", tname );
  while (1)
  {
  }
  return 0;
}

int main ()
{
  SDL_Thread *id1;

  //thread identifier
  char *tnames[1] = { "Thread 1" }; //names of thread


  //create the thread
  id1 = SDL_CreateThread ( runner, tnames[0] );
  SDL_Delay(1000);
  SDL_KillThread(id1);
  printf("killed %s\n", tnames[0] );

  return 0;
}
```

This program creates a thread that contains an infinite while loop, delays the thread for one second, and then uses the SDL_KillThread() function to terminate the thread.

Screenshot for sdlkill_demo.cpp:



```
dean@dean-pc ~/460/lab8 $ g++ -o sdlkill sdlkill_demo.cpp -lSDL -lpthread
sdlkill_demo.cpp: In function 'int main()':
sdlkill_demo.cpp:31:34: warning: deprecated conversion from string constant to 'char*' [-Wwrite-strings]
   char *tnames[1] = { "Thread 1" }; //names of thread
                                 ^
dean@dean-pc ~/460/lab8 $ ./sdlkill
I am Thread 1
killed Thread 1
dean@dean-pc ~/460/lab8 $ 
```

4. Modify the **sdlthreads_demo.cpp** program so that it runs the two threads for about 10 seconds: Define a boolean global variable *quit* that is initialized to false in main(). Another thread called timer() is created; it sets *quit* to true, after running the two threads for 10 seconds. The **runner()** function should have a while loop that keeps printing out the thread name until *quit* is true; after printing out a name, it sleeps (SDL_Delay ()) for a random amount of time between 0 and 2 seconds. (Note that **SDL_Delay (1000)** makes the thread pause for 1 second.)

Dean Cosanella
Flavio dos Santos-Ross
CSE 460

For this problem, we created a thread timer which would cause a 10 second delay allowing the other two threads to run for 10 seconds before quitting. After the 10 second delay, we set the boolean value quit to true and exited the timer thread.

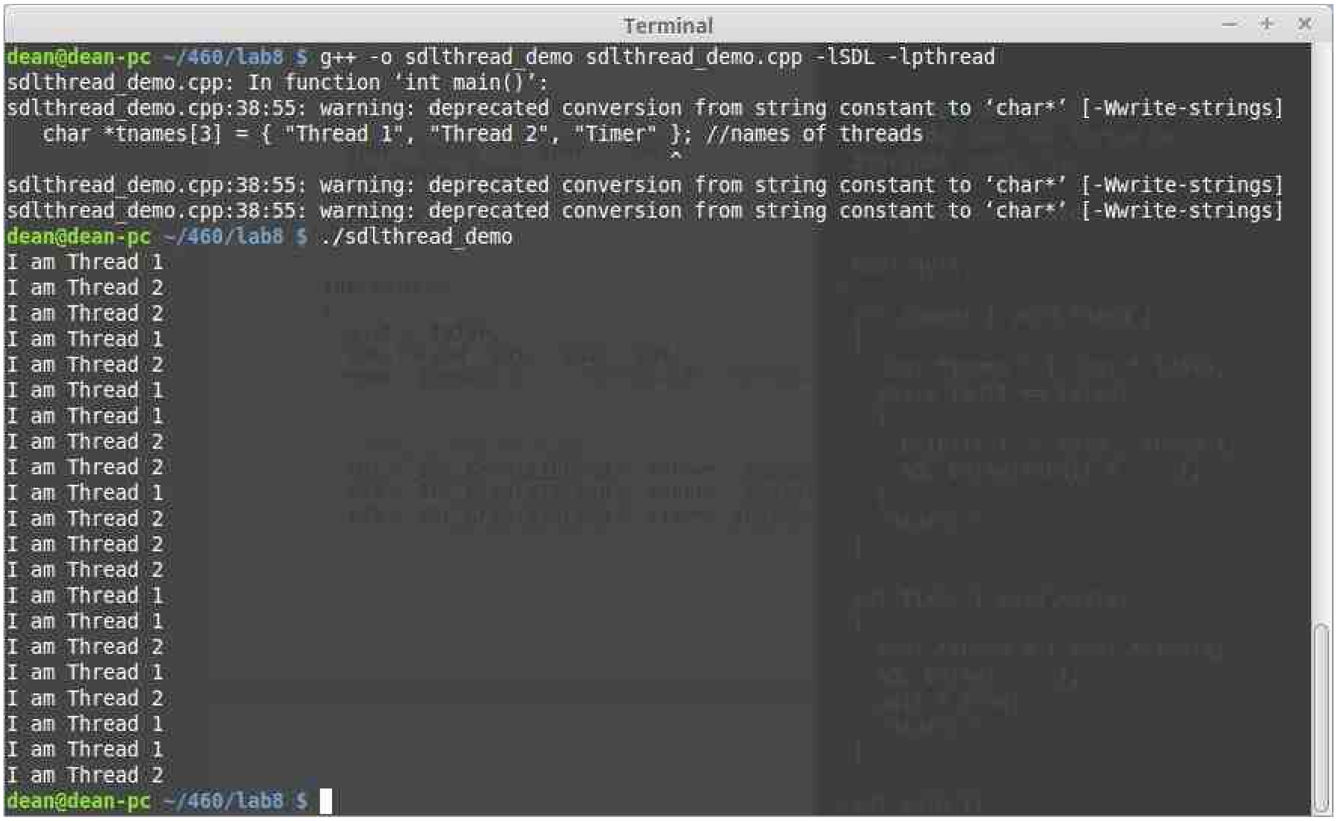
The code for this problem is shown below:

```
/*
sdlthread_demo.cpp
Compile:  g++ -o sdlthread_demo sdlthread_demo.cpp -lSDL -lpthread
Execute:  ./sdlthread_demo
*/

#include <SDL/SDL.h>
#include <SDL/SDL_thread.h>
#include <stdio.h>

using namespace std;

bool quit;

int runner ( void *data )
{
  char *tname = ( char * )data;
  while (quit == false)
  {
    printf("I am %s\n", tname );
    SDL_Delay(rand() % 2000);
  }
  return 0;
}

int timer ( void *data)
{
  char *tname = ( char * )data;
  SDL_Delay(10000);
  quit = true;
  return 0;
}

int main ()
{
  quit = false;
  SDL_Thread *id1, *id2, *id3;                //thread identifiers
  char *tnames[3] = { "Thread 1", "Thread 2", "Timer" }; //names of threads


  //create the threads
  id1 = SDL_CreateThread ( runner, tnames[0] );
  id2 = SDL_CreateThread ( runner, tnames[1] );
  id3 = SDL_CreateThread ( timer, tnames[2] );
```

Dean Cosanella
Flavio dos Santos-Ross
CSE 460

```
  //wait for the threads to exit
  SDL_WaitThread ( id1, NULL );
  SDL_WaitThread ( id2, NULL );
  SDL_WaitThread ( id3, NULL );

  return 0;
}
```

Screenshot of sdlthread_demo.cpp output:

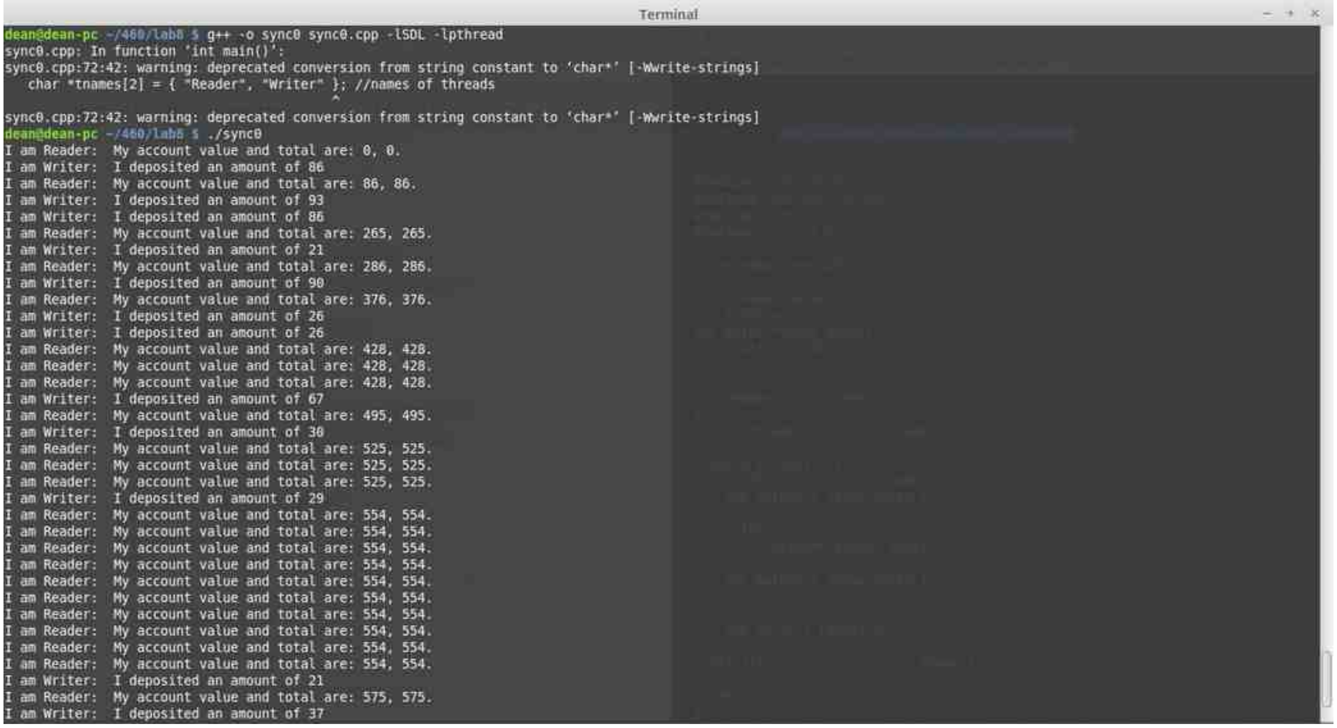


```
dean@dean-pc ~/460/lab8 $ g++ -o sdlthread_demo sdlthread_demo.cpp -lSDL -lpthread
sdlthread_demo.cpp: In function ‘int main()’:
sdlthread_demo.cpp:38:55: warning: deprecated conversion from string constant to ‘char*’ [-Wwrite-strings]
   char *tnames[3] = { "Thread 1", "Thread 2", "Timer" }; //names of threads
                                                       ^

sdlthread_demo.cpp:38:55: warning: deprecated conversion from string constant to ‘char*’ [-Wwrite-strings]
sdlthread_demo.cpp:38:55: warning: deprecated conversion from string constant to ‘char*’ [-Wwrite-strings]
dean@dean-pc ~/460/lab8 $ ./sdlthread_demo
I am Thread 1
I am Thread 2
I am Thread 2
I am Thread 1
I am Thread 2
I am Thread 1
I am Thread 1
I am Thread 2
I am Thread 2
I am Thread 1
I am Thread 2
I am Thread 2
I am Thread 2
I am Thread 1
I am Thread 1
I am Thread 2
I am Thread 1
I am Thread 2
I am Thread 1
I am Thread 1
I am Thread 2
dean@dean-pc ~/460/lab8 $
```

Dean Cosanella
Flavio dos Santos-Ross
CSE 460

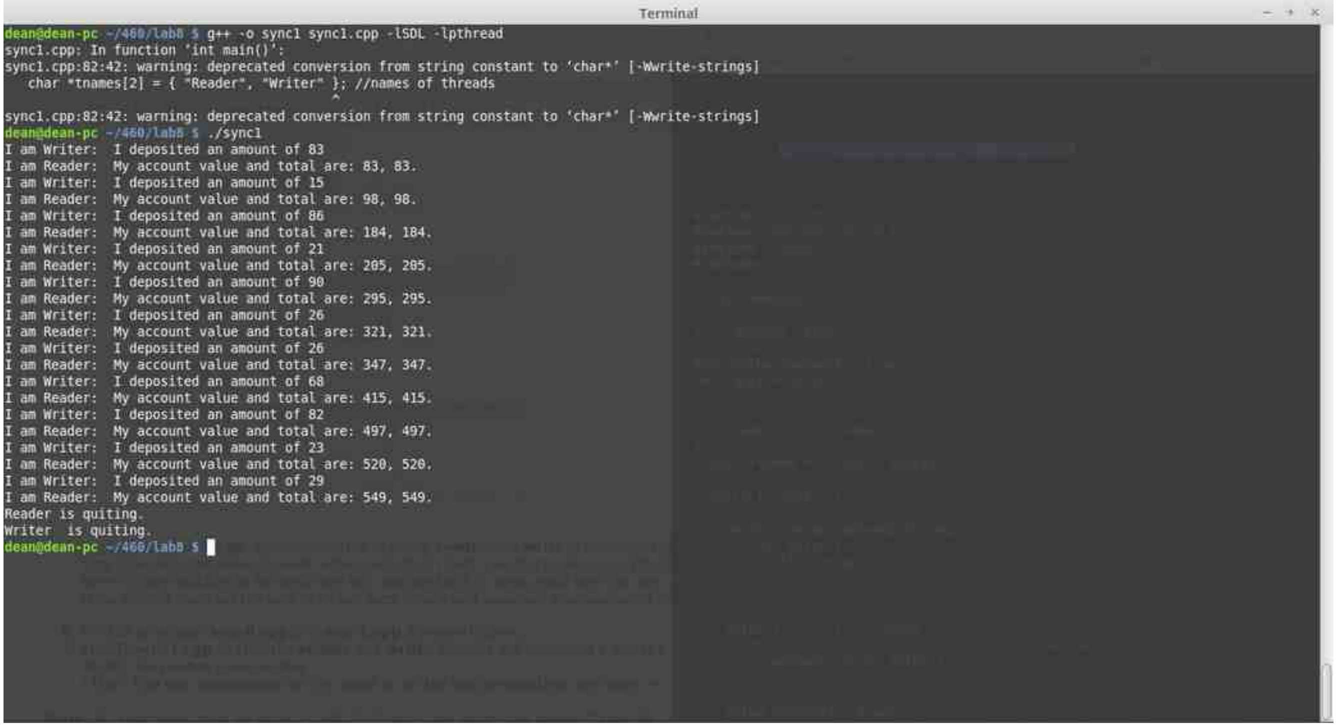
6. Try the programs **sync0.cpp** and **sync1.cpp** discussed above.

Screenshot of sync0.cpp output:



```
dean@dean-pc ~/460/lab8 $ g++ -o sync0 sync0.cpp -lSDL -lpthread
sync0.cpp: In function 'int main()':
sync0.cpp:72:42: warning: deprecated conversion from string constant to 'char*' [-Wwrite-strings]
   char *tnames[2] = { "Reader", "Writer" }; //names of threads
                                          ^
sync0.cpp:72:42: warning: deprecated conversion from string constant to 'char*' [-Wwrite-strings]
dean@dean-pc ~/460/lab8 $ ./sync0
I am Reader:  My account value and total are: 0, 0.
I am Writer:  I deposited an amount of 86
I am Reader:  My account value and total are: 86, 86.
I am Writer:  I deposited an amount of 93
I am Writer:  I deposited an amount of 86
I am Reader:  My account value and total are: 265, 265.
I am Writer:  I deposited an amount of 21
I am Reader:  My account value and total are: 286, 286.
I am Writer:  I deposited an amount of 90
I am Reader:  My account value and total are: 376, 376.
I am Writer:  I deposited an amount of 26
I am Writer:  I deposited an amount of 26
I am Reader:  My account value and total are: 428, 428.
I am Reader:  My account value and total are: 428, 428.
I am Reader:  My account value and total are: 428, 428.
I am Writer:  I deposited an amount of 67
I am Reader:  My account value and total are: 495, 495.
I am Writer:  I deposited an amount of 30
I am Reader:  My account value and total are: 525, 525.
I am Reader:  My account value and total are: 525, 525.
I am Reader:  My account value and total are: 525, 525.
I am Writer:  I deposited an amount of 29
I am Reader:  My account value and total are: 554, 554.
I am Reader:  My account value and total are: 554, 554.
I am Reader:  My account value and total are: 554, 554.
I am Reader:  My account value and total are: 554, 554.
I am Reader:  My account value and total are: 554, 554.
I am Reader:  My account value and total are: 554, 554.
I am Reader:  My account value and total are: 554, 554.
I am Reader:  My account value and total are: 554, 554.
I am Reader:  My account value and total are: 554, 554.
I am Reader:  My account value and total are: 554, 554.
I am Writer:  I deposited an amount of 21
I am Reader:  My account value and total are: 575, 575.
I am Writer:  I deposited an amount of 37
```

Screenshot of sync1.cpp output:



```
dean@dean-pc ~/460/lab8 $ g++ -o sync1 sync1.cpp -lSDL -lpthread
sync1.cpp: In function 'int main()':
sync1.cpp:82:42: warning: deprecated conversion from string constant to 'char*' [-Wwrite-strings]
   char *tnames[2] = { "Reader", "Writer" }; //names of threads
                                          ^
sync1.cpp:82:42: warning: deprecated conversion from string constant to 'char*' [-Wwrite-strings]
dean@dean-pc ~/460/lab8 $ ./sync1
I am Writer:  I deposited an amount of 83
I am Reader:  My account value and total are: 83, 83.
I am Writer:  I deposited an amount of 15
I am Reader:  My account value and total are: 98, 98.
I am Writer:  I deposited an amount of 86
I am Reader:  My account value and total are: 184, 184.
I am Writer:  I deposited an amount of 21
I am Reader:  My account value and total are: 205, 205.
I am Writer:  I deposited an amount of 90
I am Reader:  My account value and total are: 295, 295.
I am Writer:  I deposited an amount of 26
I am Reader:  My account value and total are: 321, 321.
I am Writer:  I deposited an amount of 26
I am Reader:  My account value and total are: 347, 347.
I am Writer:  I deposited an amount of 68
I am Reader:  My account value and total are: 415, 415.
I am Writer:  I deposited an amount of 82
I am Reader:  My account value and total are: 497, 497.
I am Writer:  I deposited an amount of 23
I am Reader:  My account value and total are: 520, 520.
I am Writer:  I deposited an amount of 29
I am Reader:  My account value and total are: 549, 549.
Reader is quiting.
Writer  is quiting.
dean@dean-pc ~/460/lab8 $
```

Dean Cosanella
Flavio dos Santos-Ross
CSE 460

7. Modify **sync1.cpp** so that the **reader** and **writer** threads are accessing a buffer (e.g. an array). When the buffer is full, the **writer** goes to sleep and when the buffer is empty, the **reader** goes to sleep.

For the modification on sync1.cpp, we created a new program called sync2.cpp. We created a vector which would act as our buffer and we created a mutex to help us achieve synchronization. When the buffer is full, the writer sleeps for 3 seconds and when the buffer is empty, the reader sleeps for 3 seconds.

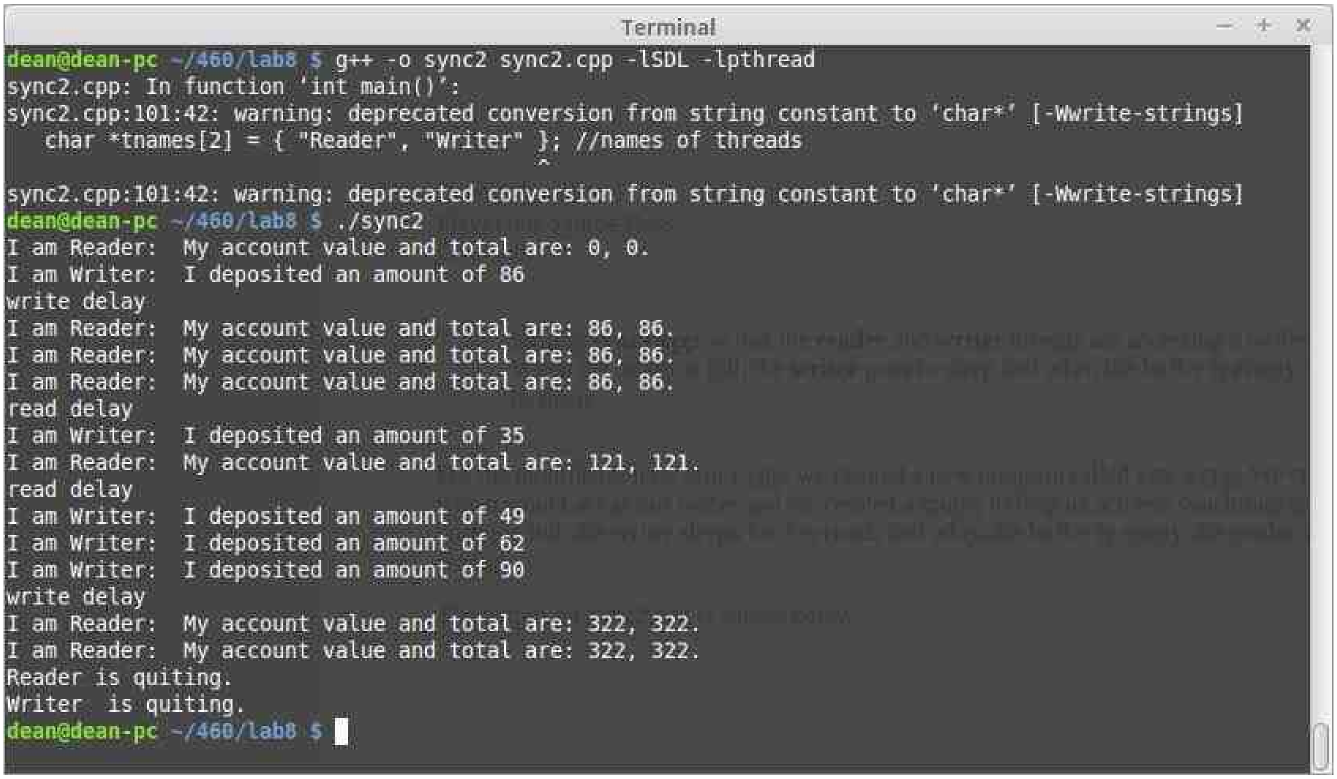
Code for sync2.cpp:

```
/*
sync2.cpp
Compile: g++ -o sync2 sync2.cpp -lSDL -lpthread
Execute: ./sync2
*/

#include <SDL/SDL.h>
#include <SDL/SDL_thread.h>
#include <stdio.h>
#include <stdlib.h>
#include <vector>

using namespace std;

int  account_value = 0;         //shared variable
int  total = 0;                 //shared variable
bool value_consumed = true;     //variable to control synchronization
bool quit = false;
SDL_mutex *value_mutex;
int buffer_size = 3;
vector<int> buffer(3);

//This thread reads account_value and total
int reader ( void *data )
{
  char *tname = ( char * )data;

  while ( !quit ) {
    if ( quit ) break;          //when you wake up
                                //  the world might have changed
   //now you can sefely access account_value and total
    printf("I am %s: ", tname );
    SDL_mutexP(value_mutex);
    printf(" My account value and total are: %d, %d.\n",
                account_value, total );
    buffer.pop_back();

    buffer_size--;

    if(buffer.empty())
    {
      SDL_mutexV(value_mutex);
      printf("read delay\n");
      SDL_Delay(3000);
    }

    //delay for a random amount of time
    SDL_Delay ( rand() % 1000 );
  }
  printf("%s is quiting.\n", tname );

  return 0;
}


//This thread  writes value
int writer ( void *data )
{
  char *tname = ( char * )data;

  while ( !quit ) {
    int a = rand() % 100;       //get a random number

    SDL_mutexP(value_mutex);

    if ( quit ) break;          //when you wake up,
                                //  the world might haved changed
    printf("I am %s: ", tname );
    account_value += a;
    total += a;
    printf(" I deposited an amount of %d\n", a );
    buffer.push_back(total);

    buffer_size++;

    if(buffer_size == 3)
    {
      SDL_mutexV(value_mutex);
      printf("write delay\n");
      SDL_Delay(3000);
    }

    //delay for a random amount of time
    SDL_Delay ( rand() % 2000 );
  }
  printf("%s  is quiting.\n", tname );

  return 0;
}

int main ()
{
  SDL_Thread *id1, *id2;                        //thread identifiers
  char *tnames[2] = { "Reader", "Writer" };     //names of threads
```

Dean Cosanella
Flavio dos Santos-Ross
CSE 460

```
  //create the threads
  id1 = SDL_CreateThread ( reader, tnames[0] );
  id2 = SDL_CreateThread ( writer, tnames[1] );

  //experiment with 10 seconds
  for ( int i = 0; i < 5; ++i )
      SDL_Delay ( 2000 );

  quit = true;                          //signal the threads to return
  //wait for the threads to exit
  SDL_WaitThread ( id1, NULL );
  SDL_WaitThread ( id2, NULL );

  return 0;
}
```

The output for sync2.cpp is shown below:



```
dean@dean-pc ~/460/lab8 $ g++ -o sync2 sync2.cpp -lSDL -lpthread
sync2.cpp: In function 'int main()':
sync2.cpp:101:42: warning: deprecated conversion from string constant to 'char*' [-Wwrite-strings]
   char *tnames[2] = { "Reader", "Writer" }; //names of threads
                                          ^
sync2.cpp:101:42: warning: deprecated conversion from string constant to 'char*' [-Wwrite-strings]
dean@dean-pc ~/460/lab8 $ ./sync2
I am Reader:  My account value and total are: 0, 0.
I am Writer:  I deposited an amount of 86
write delay
I am Reader:  My account value and total are: 86, 86.
I am Reader:  My account value and total are: 86, 86.
I am Reader:  My account value and total are: 86, 86.
read delay
I am Writer:  I deposited an amount of 35
I am Reader:  My account value and total are: 121, 121.
read delay
I am Writer:  I deposited an amount of 49
I am Writer:  I deposited an amount of 62
I am Writer:  I deposited an amount of 90
write delay
I am Reader:  My account value and total are: 322, 322.
I am Reader:  My account value and total are: 322, 322.
Reader is quiting.
Writer  is quiting.
dean@dean-pc ~/460/lab8 $ 
```

We learned a lot about threads in this lab. It is very interesting to see how two functions can be running concurrently and seeing the results on the screen. We are hoping we can learn more about threads so we can use them when we program outside of this class. Because we finished every part of this lab, we give ourselves a score of 20/20.