

Anthony Thap & Brian Ackley

25 February 2014

CSE460 Operating Systems

Winter 2014

Lab 8

- We did everything that was assigned and will receive full credit

```
1. /*
   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;
}

```

```

003404803@jb356-7:/students/csci/003404803/460/lab8
File Edit View Search Terminal Help
[003404803@jb356-7 003404803]$ ls
350      cse330      Firefox_wallpaper.png  read_write.h
455      cse350      fourbitadder           read_write.h~
455projects CSE366_Final2013.odt hw3a.odt               samp.cpp
460      cse460_lab5.odt lab2_460.odt           sample
Ant      cse460_lab5.pdf lab3_460.odt           Solutions Manual
a.out    Desktop     lab5.odt               Templates
cs330    Digital Logic lab3  Mano_Digital.Design.5E.pdf Videos
cs350    Documents   Music                  workspace
cse      Downloads   Pictures
cse202   Final CSE365.odt Public
[003404803@jb356-7 003404803]$ cd 460
[003404803@jb356-7 460]$ ls
a.out hw1 lab1 lab2 lab3 lab4 lab5 lab6 lab7 lab8
[003404803@jb356-7 460]$ cd lab8
[003404803@jb356-7 lab8]$ ls
pthreads_demo pthreads_demo.cpp pthreads_demo.cpp~
[003404803@jb356-7 lab8]$ ./pthreads_demo
I am Thread 1
I am Thread 2
[003404803@jb356-7 lab8]$

```

2. /*

pthread_demo.cpp

A very simple example demonstrating the usage of pthreads.

Compile: g++ -o pthread_demo pthread_demo.cpp -lpthread

Execute: ./pthread_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;
}

```

The screenshot shows a terminal window with the title bar '003404803@jb356-7:/students/csci/003404803/460/lab8'. The terminal content is as follows:

```

[003404803@jb356-7 003404803]$ cd 460
[003404803@jb356-7 460]$ ls
a.out hw1 lab1 lab2 lab3 lab4 lab5 lab6 lab7 lab8
[003404803@jb356-7 460]$ cd lab8
[003404803@jb356-7 lab8]$ ls
 pthreads_demo pthreads_demo2 pthreads_demo.cpp pthreads_demo.cpp~
[003404803@jb356-7 lab8]$ ./pthreads_demo2
I am CSE460
I am Winter 2014
[003404803@jb356-7 lab8]$

```

3. /*

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>
#include <pthread.h>

```

using namespace std;

//The thread

```

int runner2 ( void *data )
{
    char *tname = ( char * )data;
    while(1){
        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 ( runner2, tnames[0] );

void SDL_KillThread(SDL_Thread *id1);

printf("summer time");

//wait for the threads to exit

return 0;

}

```

```

003404803@jb358-8:/students/csci/003404803/460/lab8
File Edit View Search Terminal Help
I am Thread 1
I am Thread 1
I am Thread 1
I am Thread 1
I am Thread 1
I am Thread 1
I am Thread 1
I am Thread 1
I am Thread 1
I am Thread 1
I am Thread 1
I am Thread 1
I am Thread 1
I am Thread 1
Summer Time = I am Thread 1
I am Thread 1
I am Thread 1
I am Thread 1
I am Thread 1
I am Thread 1
I am Thread 1
I am Thread 1
I am Thread 1
I am Thread 1

```

/*

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;

bool quit = false;

//The thread

int runner ( void *data )

{

    char *tname = ( char * )data;

    int test = 0;

    while(test < 3)

    {

        SDL_Delay ( 1000 );

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

        test++;

    }

    return 0;

}

int main ()

{

    SDL_Thread *id1, *id2; //thread identifiers

    char *tnames[2] = { "Thread 1", "Thread 2" }; //names of threads

    for ( int i = 0; i < 5; ++i )

    {

        SDL_Delay ( 2000 );
```

```

id1 = SDL_CreateThread ( runner, tnames[0] );

id2 = SDL_CreateThread ( runner, tnames[1] );

}

quit = true;

//wait for the threads to exit

SDL_WaitThread ( id1, NULL );

SDL_WaitThread ( id2, NULL );

return 0;

}

```

```

003404803@jb358-8:/students/csci/003404803/460/lab8
File Edit View Search Terminal Help
I am Animal 2
I am Animal 1
I am Animal 2
I am Animal 2
I am Animal 1
I am Animal 1
I am Animal 2
I am Animal 1
I am Animal 2
I am Animal 2
I am Animal 1
I am Animal 1
I am Animal 2
I am Animal 2
I am Animal 1
I am Animal 1
I am Animal 2
I am Animal 2
I am Animal 1
I am Animal 1
I am Animal 2
I am Animal 1
I am Animal 2
I am Animal 1
I am Animal 2
I am Animal 1
I am Animal 2

```

4. /*

sync1.cpp

A simple example demonstrating the use of a global variable
to enforce strict alternation between two threads.

Compile: `g++ -o sync1 sync1.cpp -lSDL -lpthread`

Execute: `./sync1`

*/

`#include <SDL/SDL.h>`

```

#include <SDL/SDL_thread.h>

#include <stdio.h>

#include <stdlib.h>

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;

#define N 5 //number of slots in buffer

SDL_sem *mutex; //controls access to critical

SDL_sem *nempty; //counts number of empty slots

SDL_sem *nfilled; //counts number of filled slots

int buffer[N];

void insert_item ( int item )

{

    static int tail = 0;

    buffer[tail] = item;

    printf("\ninsert %d at %d", item, tail );

    tail = ( tail + 1 ) % N;

}

int remove_item ()

{

    static int head = 0;

    int item;

```

```

item = buffer[head];

printf("\nremove %d at %d", item, head );

head = ( head + 1 ) % N;

return item;

}

//This thread reads account_value and total

int reader ( void *data )

{

char *tname = ( char * )data;

while ( !quit ) {

while ( value_consumed && !quit )

SDL_Delay ( 20 ); //wait for new value

if ( quit ) break; //when you wake up

//now you can safely access account_value and total

account_value = remove_item ();

printf("I am %s: ", tname );

printf(" My account value and total are: %d, %d.\n",

account_value, total );

//tell writer that value has been read

value_consumed = true;

//delay for a random amount of time

SDL_Delay ( rand() % 1000 );

}

```



```

printf("%s is quitting.\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


//don't write until previous value has been read

while ( !value_consumed && !quit )

SDL_Delay ( 20 );

if ( quit ) break; //when you wake up,

printf("I am %s: ", tname );

insert_item ( a );

account_value += a;

total += a;

printf(" I deposited an amount of %d\n", a );

//tell reader new value is available

value_consumed = false;

//delay for a random amount of time

SDL_Delay ( rand() % 2000 );

}

```

```

printf("%s is quitting.\n", tname );

return 0;

}

int main ()

{

    SDL_Thread *id1, *id2; //thread identifiers

    char *tnames[2] = { "Reader", "Writer" }; //names of threads


    //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;

}

```