ESE-3025 Embedded Real Time Operating Systems

ASSIGNMENT 1

GROUP No. 2

Group Members	Student ID
Christy Rachel Philip	C0765535
Fahad Rahman	C0769871
James M Chacko	C0777192
Premil Presannan	C0777191
Vy Nguyen	C0776242

Introduction:

In this assignment, first we need to analyse two multithreading codes from the following link,

> https://github.com/takiszourntos/teaching/tree/master/lambton/2020/summer/ese3025/week_1/pthreads

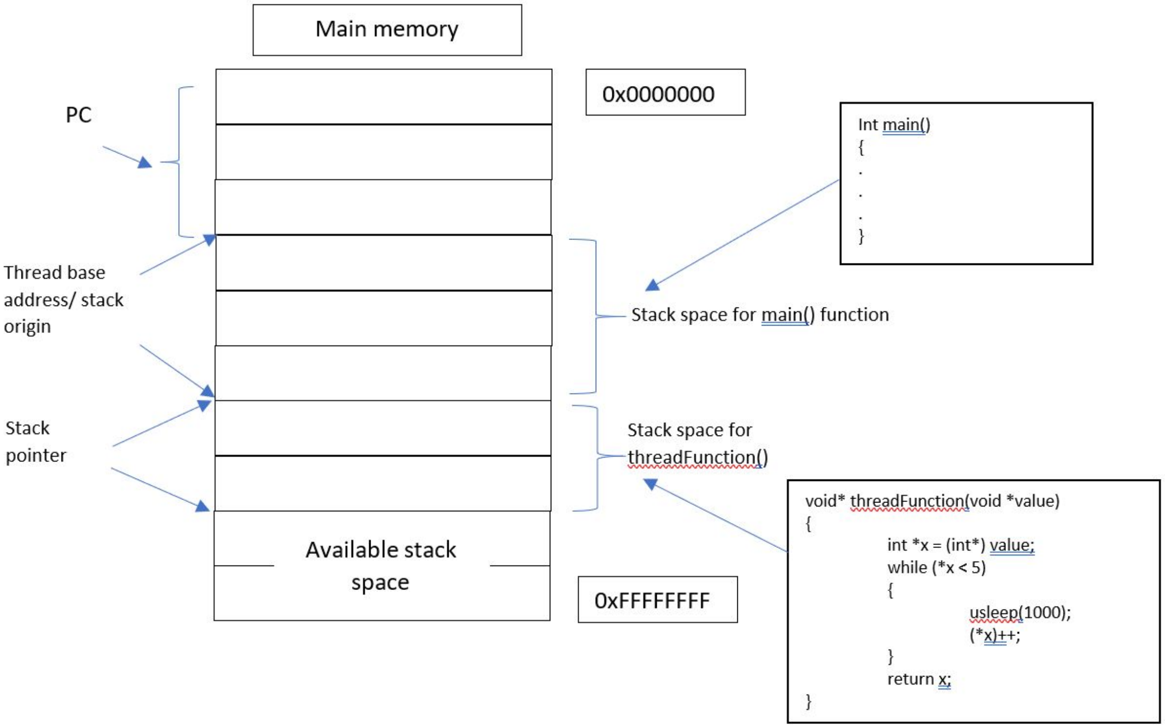
Then clone both the codes from the repository and compile it on both eclipse and beaglebone and compare the execution of both of them.

Discussion:

1. **How many threads are in Pthread0.c? Where is the stack space for threadFunction()? Where is the stack space for main()?**

Answer:

There are two threads in pthread_0.c. Stack spaces for both thread function and main are in the RAM during execution. The exact location cannot be premeditated as the system randomly allocates the location. The system allocates 1MB for 32 bit systems and 2MB for 64 bit systems. The two threads do not share the stack space.


Main memory
0x0000000
PC
Thread base address/ stack origin
Stack pointer
Stack space for main() function
Stack space for threadFunction()
Available stack space
0xFFFFFFFF

```
Int main()
{
.
.
.
}
```

```
void* threadFunction(void *value)
{
        int *x = (int*) value;
        while (*x < 5)
        {
                usleep(1000);
                (*x)++;
        }
        return x;
}
```

2. **Consider the program pthread1.c. a) Build the program and run it on your host machine as well as your beagle-bone. How do you compare the execution process on them? b) Given that the two thread functions are identical. Do you expect them to run an equal number of times? Why or Why not?**

Step 1:

Clone the code from the given repository, for that we use the command,

```
git clone https://github.com/takiszourntos/teaching
```

Step 2:

Build and run the code pthread_1 in **Eclipse.**

Different results from Eclipse:

OUTPUT 1

```
The value of x_1=0, x_2=0 and y=0

The value of x_1=4366, x_2=3194 and y=1

The value of x_1=7858, x_2=6667 and y=2

The value of x_1=11378, x_2=10053 and y=3

The value of x_1=14854, x_2=13425 and y=4

The value of x_1=18365, x_2=16838 and y=5

The value of x_1=21857, x_2=20273 and y=6

The value of x_1=25348, x_2=23648 and y=7

The value of x_1=28863, x_2=27069 and y=8

The value of x_1=32339, x_2=30455 and y=9

Final: x_1=37528, x_2=35552, y=10
```

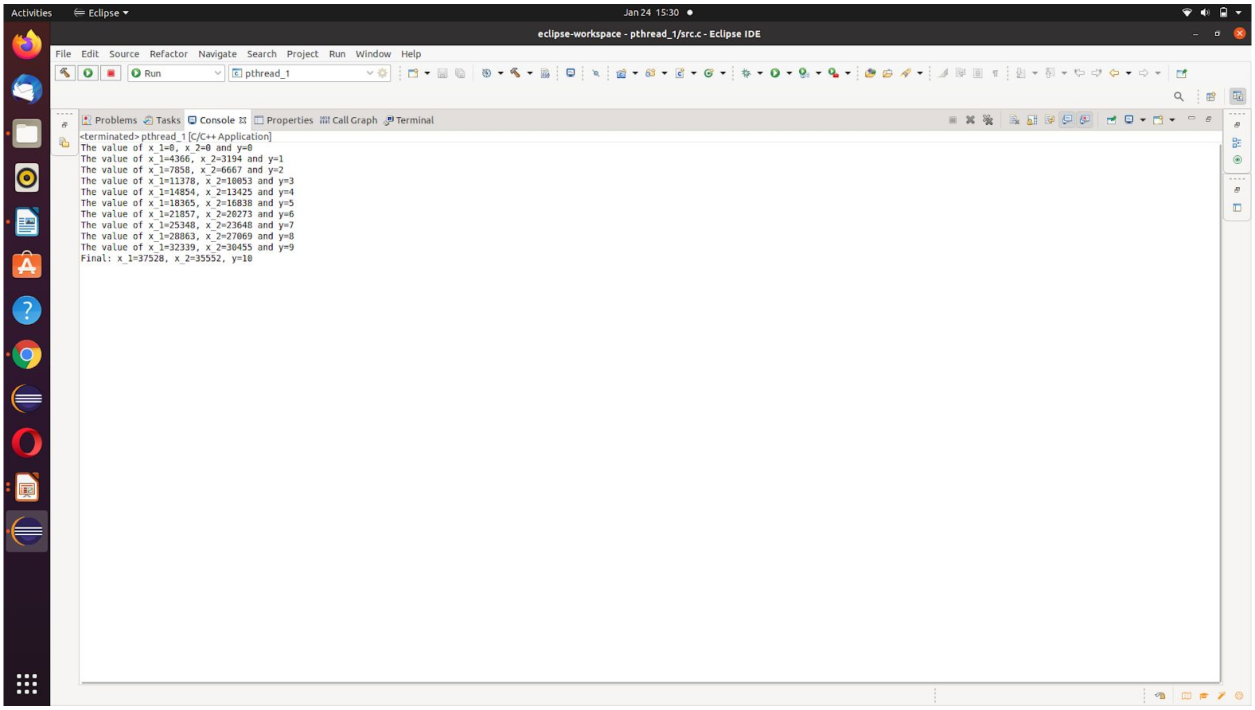
OUTPUT 2

```
The value of x_1=829, x_2=0 and y=0

The value of x_1=1767, x_2=6760 and y=1

The value of x_1=2635, x_2=12382 and y=2

The value of x_1=5689, x_2=14736 and y=3

The value of x_1=8655, x_2=16985 and y=4
The value of x_1=11578, x_2=19193 and y=5

The value of x_1=20254, x_2=25585 and y=6

The value of x_1=29371, x_2=32257 and y=7

The value of x_1=32397, x_2=34816 and y=8

The value of x_1=35301, x_2=37045 and y=9

Final: x_1=41491, x_2=42658, y=10
```

OUTPUT 3

```
The value of x_1=0, x_2=0 and y=0

The value of x_1=6742, x_2=4214 and y=1

The value of x_1=10951, x_2=7686 and y=2

The value of x_1=14986, x_2=11059 and y=3

The value of x_1=18664, x_2=14734 and y=4

The value of x_1=21957, x_2=17906 and y=5

The value of x_1=25108, x_2=20962 and y=6

The value of x_1=28240, x_2=23973 and y=7

The value of x_1=31429, x_2=27005 and y=8

The value of x_1=34669, x_2=30161 and y=9

Final: x_1=40210, x_2=35626, y=10
```

OUTPUT SCREENSHOT FOR PROOF:



Activities
Eclipse
Jan 24 15:30
eclipse-workspace - pthread_1/src.c - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Run
pthread_1
Problems
Tasks
Console
Properties
Call Graph
Terminal

```
<terminated> pthread_1 [C/C++ Application]
The value of x_1=0, x_2=0 and y=0
The value of x_1=4366, x_2=3194 and y=1
The value of x_1=7858, x_2=6667 and y=2
The value of x_1=11378, x_2=10053 and y=3
The value of x_1=14854, x_2=13425 and y=4
The value of x_1=18365, x_2=16838 and y=5
The value of x_1=21857, x_2=20273 and y=6
The value of x_1=25348, x_2=23648 and y=7
The value of x_1=28863, x_2=27069 and y=8
The value of x_1=32339, x_2=30455 and y=9
Final: x_1=37528, x_2=35552, y=10
```

Step 3:

Move pthread_1 to home in Ubuntu

Step 4:

Open linux terminal and give the following command to transfer the pthread_1 file to beaglebone using SFTP(Secure File Transfer Protocol)

```
sftp debian@192.168.7.2
```

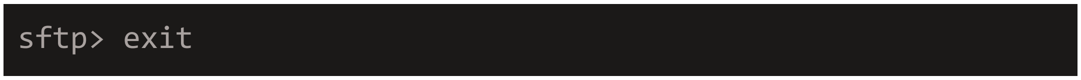
Step 5:

Then transfer the folder (pthread_1) to beaglebone by using put -r command:

```
sftp> put -r pthread_1
```

Step 6:

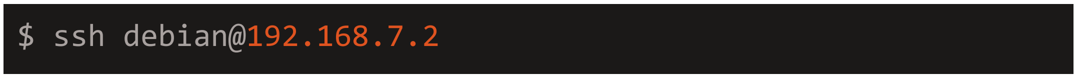
Then exit sftp using “exit”:



```
sftp> exit
```

Step 7:

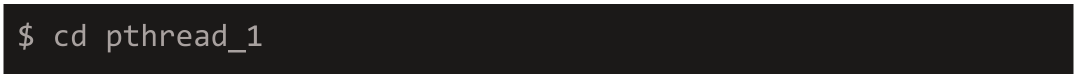
Switching to beaglebone with ssh(Secure Shell Protocol) :



```
$ ssh debian@192.168.7.2
```

Step 8:

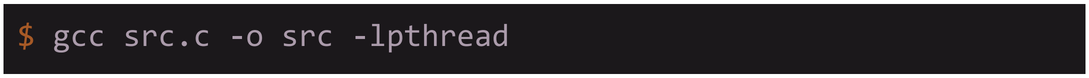
Then go to pthread_1 folder on your beaglebone:



```
$ cd pthread_1
```

Step 9:

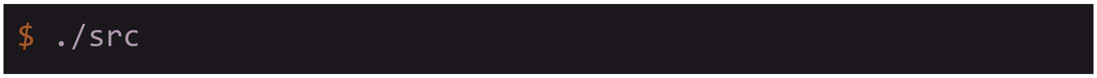
Compile the code on beaglebone:



```
$ gcc src.c -o src -lpthread
```

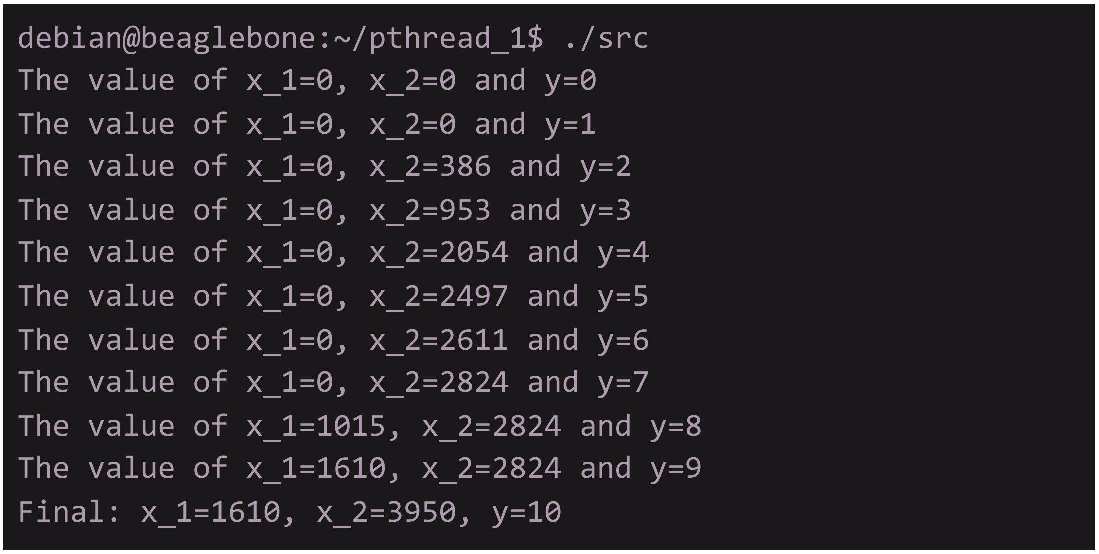
Step 10:

Run the code on beaglebone by calling it’s source file name with the following command:

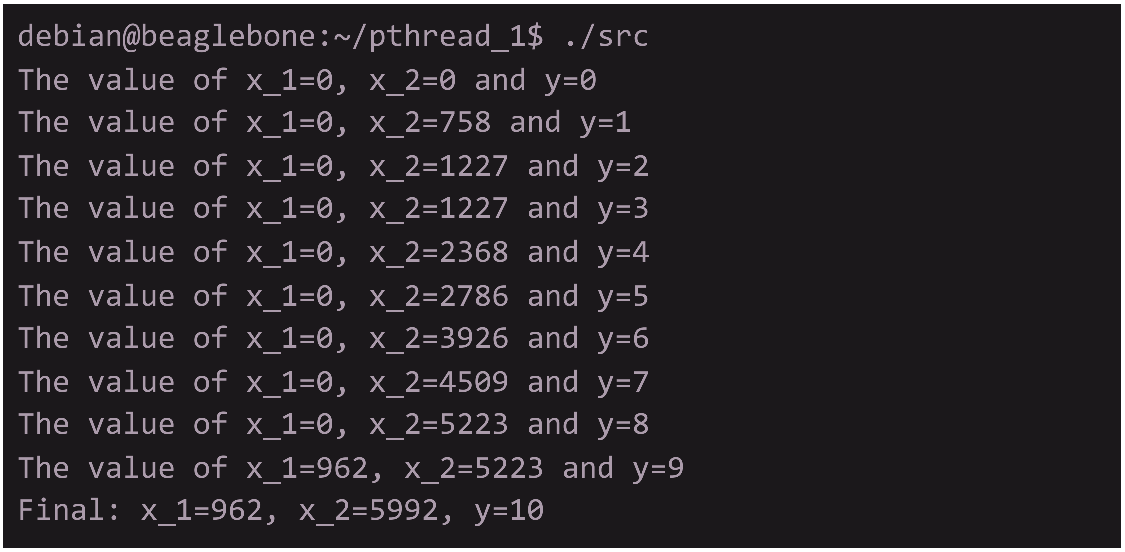


```
$ ./src
```

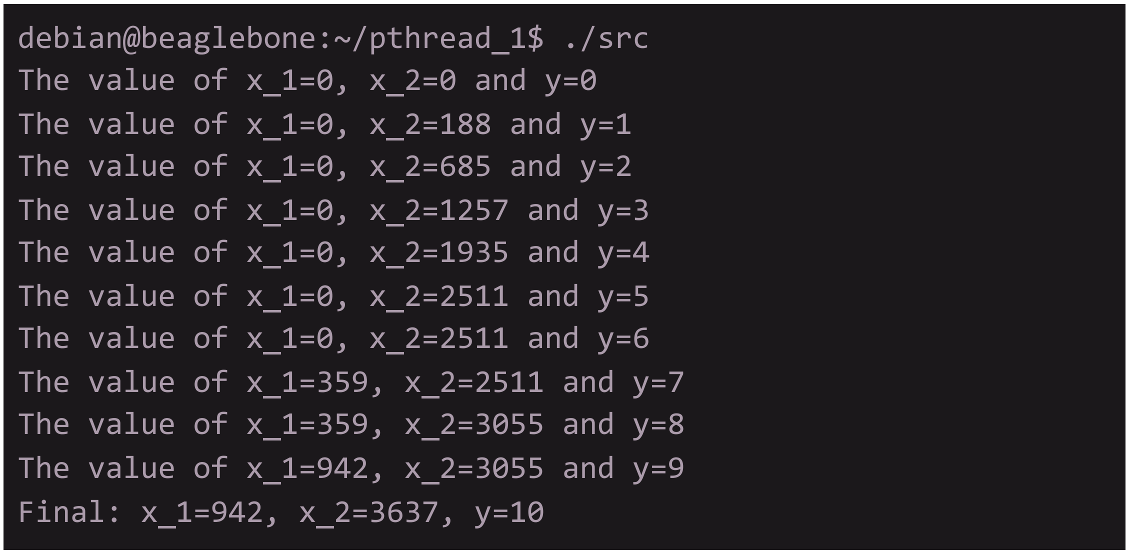
3 OUTPUTS FROM BEAGLEBONE:



```
debian@beaglebone:~/pthread_1$ ./src
The value of x_1=0, x_2=0 and y=0
The value of x_1=0, x_2=0 and y=1
The value of x_1=0, x_2=386 and y=2
The value of x_1=0, x_2=953 and y=3
The value of x_1=0, x_2=2054 and y=4
The value of x_1=0, x_2=2497 and y=5
The value of x_1=0, x_2=2611 and y=6
The value of x_1=0, x_2=2824 and y=7
The value of x_1=1015, x_2=2824 and y=8
The value of x_1=1610, x_2=2824 and y=9
Final: x_1=1610, x_2=3950, y=10
```

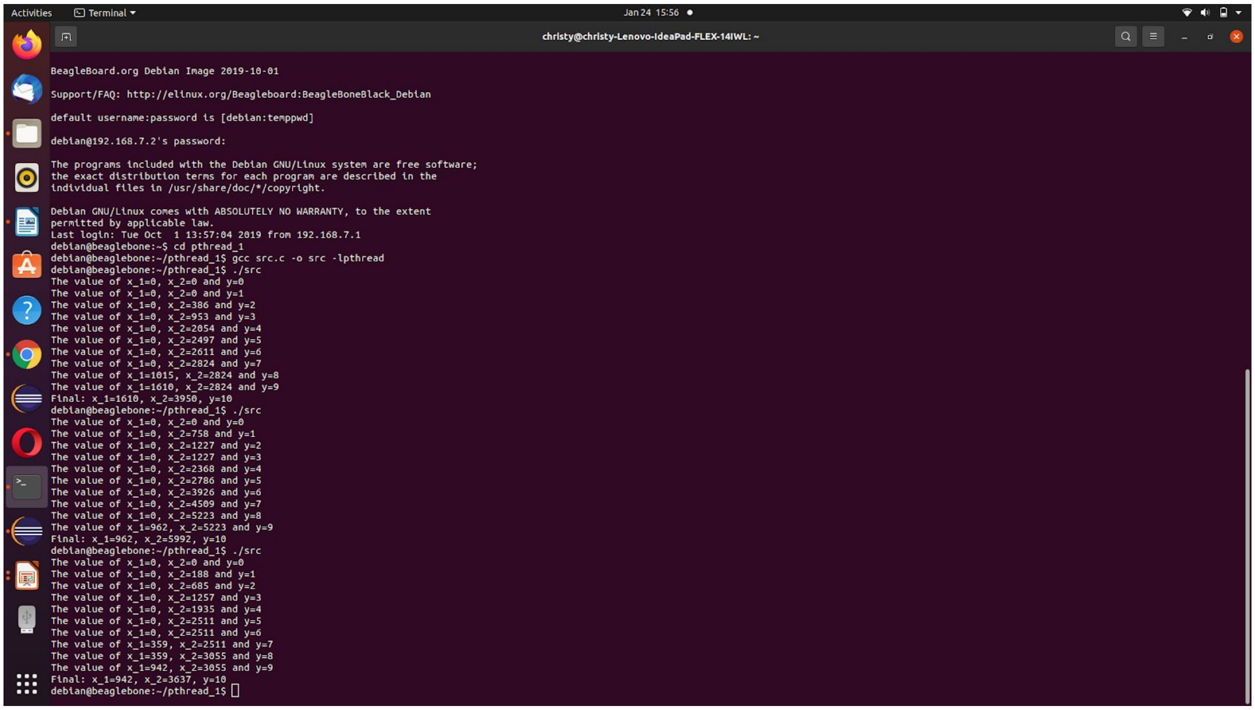


```
debian@beaglebone:~/pthread_1$ ./src
The value of x_1=0, x_2=0 and y=0
The value of x_1=0, x_2=758 and y=1
The value of x_1=0, x_2=1227 and y=2
The value of x_1=0, x_2=1227 and y=3
The value of x_1=0, x_2=2368 and y=4
The value of x_1=0, x_2=2786 and y=5
The value of x_1=0, x_2=3926 and y=6
The value of x_1=0, x_2=4509 and y=7
The value of x_1=0, x_2=5223 and y=8
The value of x_1=962, x_2=5223 and y=9
Final: x_1=962, x_2=5992, y=10
```



```
debian@beaglebone:~/pthread_1$ ./src
The value of x_1=0, x_2=0 and y=0
The value of x_1=0, x_2=188 and y=1
The value of x_1=0, x_2=685 and y=2
The value of x_1=0, x_2=1257 and y=3
The value of x_1=0, x_2=1935 and y=4
The value of x_1=0, x_2=2511 and y=5
The value of x_1=0, x_2=2511 and y=6
The value of x_1=359, x_2=2511 and y=7
The value of x_1=359, x_2=3055 and y=8
The value of x_1=942, x_2=3055 and y=9
Final: x_1=942, x_2=3637, y=10
```

OUTPUT SCREENSHOT FOR PROOF:



```
BeagleBoard.org Debian Image 2019-10-01

Support/FAQ: http://elinux.org/Beagleboard:BeagleBoneBlack_Debian

default username:password is [debian:temppwd]

debian@192.168.7.2's password:

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Tue Oct  1 13:57:04 2019 from 192.168.7.1
debian@beaglebone:~$ cd pthread_1
debian@beaglebone:~/pthread_1$ gcc src.c -o src -lpthread
debian@beaglebone:~/pthread_1$ ./src
The value of x_1=0, x_2=0 and y=0
The value of x_1=0, x_2=0 and y=1
The value of x_1=0, x_2=386 and y=2
The value of x_1=0, x_2=953 and y=3
The value of x_1=0, x_2=2054 and y=4
The value of x_1=0, x_2=2497 and y=5
The value of x_1=0, x_2=2611 and y=6
The value of x_1=0, x_2=2824 and y=7
The value of x_1=1015, x_2=2824 and y=8
The value of x_1=1610, x_2=2824 and y=9
Final: x_1=1610, x_2=3950, y=10
debian@beaglebone:~/pthread_1$ ./src
The value of x_1=0, x_2=0 and y=0
The value of x_1=0, x_2=758 and y=1
The value of x_1=0, x_2=1227 and y=2
The value of x_1=0, x_2=1227 and y=3
The value of x_1=0, x_2=2368 and y=4
The value of x_1=0, x_2=2786 and y=5
The value of x_1=0, x_2=3926 and y=6
The value of x_1=0, x_2=4509 and y=7
The value of x_1=0, x_2=5223 and y=8
The value of x_1=962, x_2=5223 and y=9
Final: x_1=962, x_2=5992, y=10
debian@beaglebone:~/pthread_1$ ./src
The value of x_1=0, x_2=0 and y=0
The value of x_1=0, x_2=188 and y=1
The value of x_1=0, x_2=685 and y=2
The value of x_1=0, x_2=1257 and y=3
The value of x_1=0, x_2=1935 and y=4
The value of x_1=0, x_2=2511 and y=5
The value of x_1=0, x_2=2511 and y=6
The value of x_1=359, x_2=2511 and y=7
The value of x_1=359, x_2=3055 and y=8
The value of x_1=942, x_2=3055 and y=9
Final: x_1=942, x_2=3637, y=10
debian@beaglebone:~/pthread_1$ 
```

After comparing the results from both beaglebone and eclipse, it is clear that it is getting different results when we compile each time. And based on the values of x1 and x2, it is very less in beaglebone than the host machine.

The two threads are identical but they may or may not run an equal number of times. And according to the outputs. They don’t run an equal number of times. Yes, the values of x1 and x2 were different. The values of x1 and x2 in BBB were much less as compared to the host. BBB has 1 core and a laptop has at least 2 cores. This is the reason why BBB was slow as compared to a laptop.

Conclusion:

In this assignment, first analysed two multithreading codes from the following link,

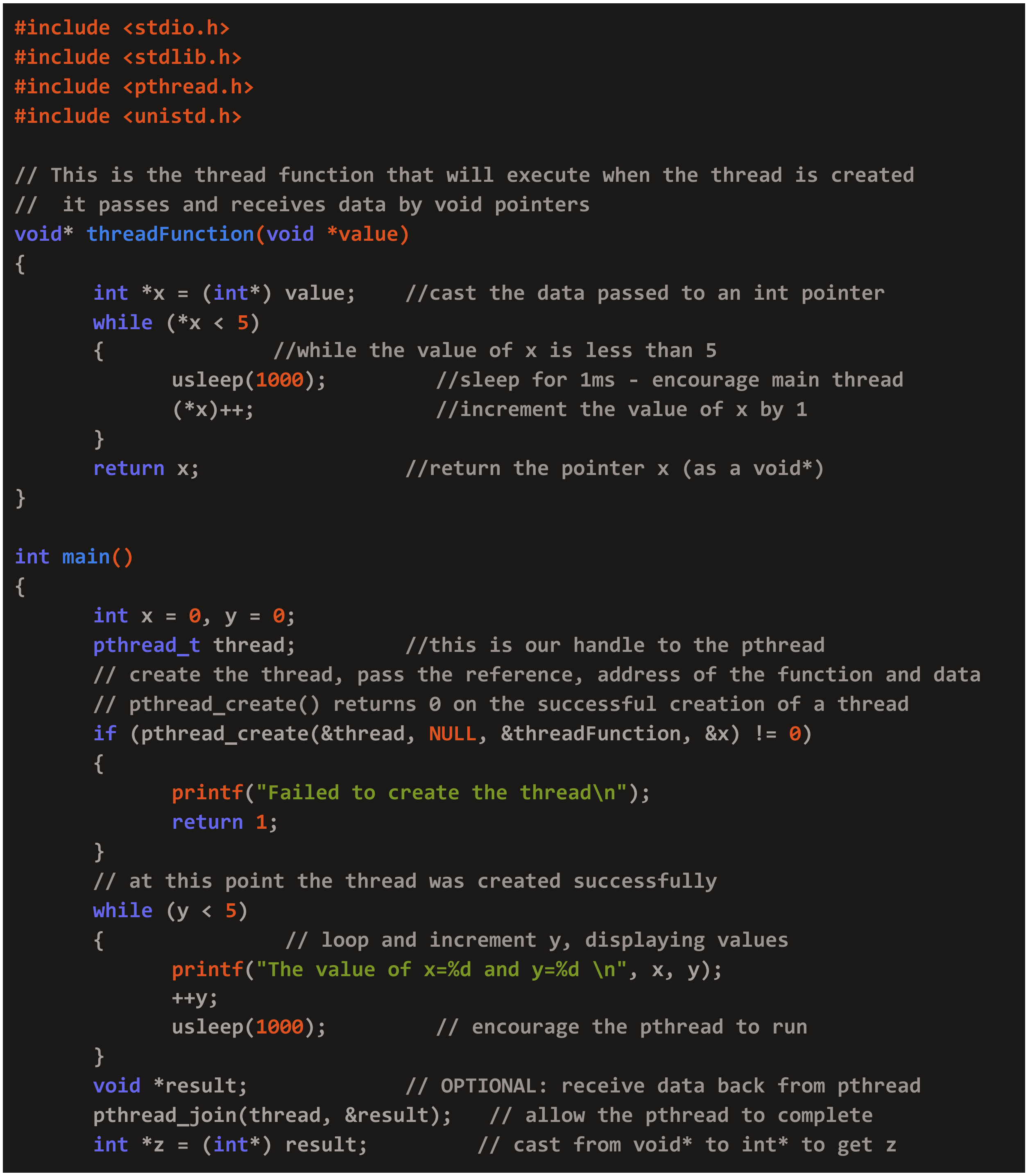
> https://github.com/takiszourntos/teaching/tree/master/lambton/2020/summer/ese3025/week_1/pthreads

Then cloned both the codes from the repository and compiled it on both eclipse and beaglebone.

After comparing both the results, it is found that for pthread_1, we are getting different results when we compile each time. And based on the values of x1 and x2, it is very less in beaglebone than the host machine.

APPENDIX

Pthread_0:

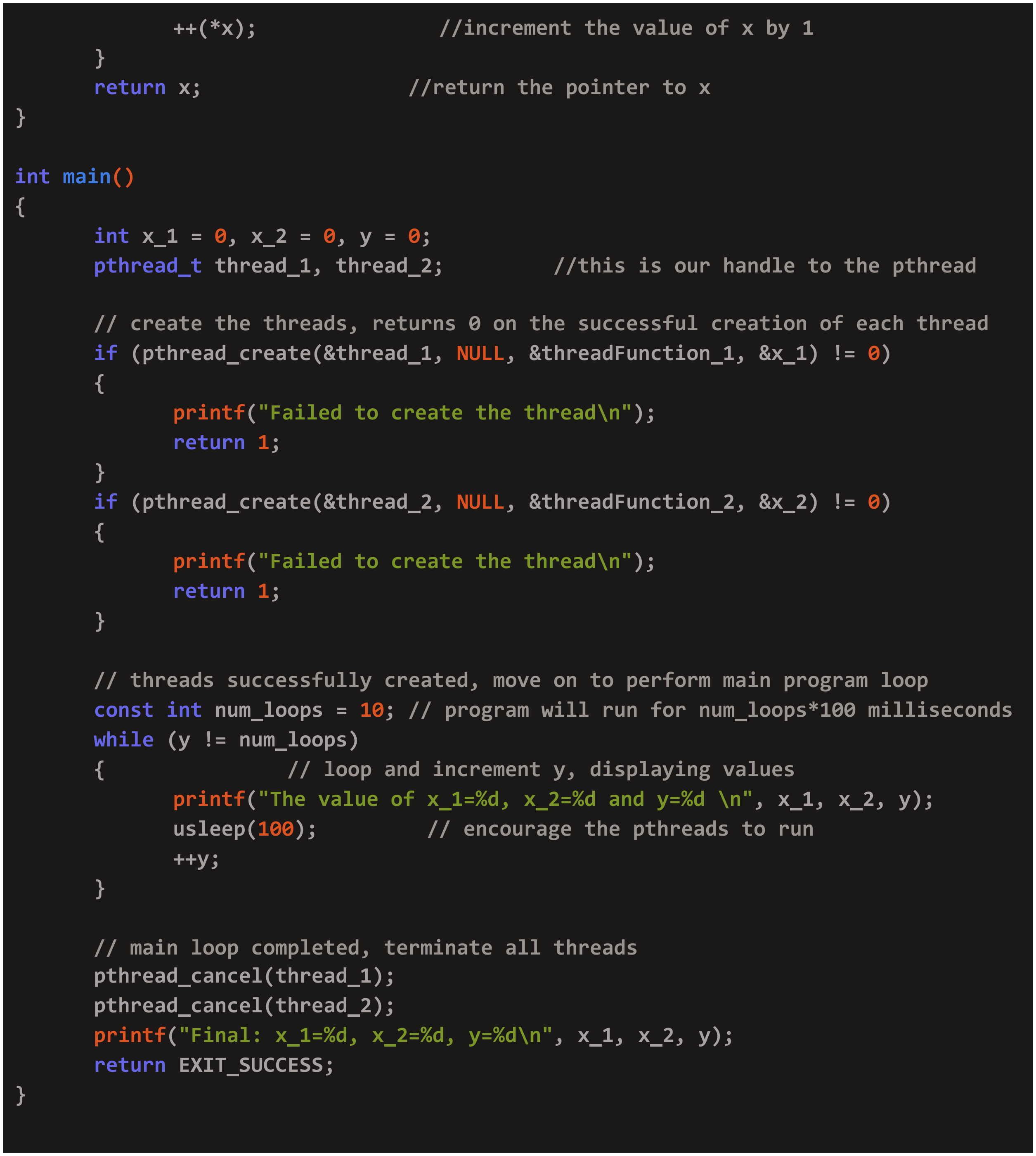


```c
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>
#include <unistd.h>

// This is the thread function that will execute when the thread is created
//  it passes and receives data by void pointers
void* threadFunction(void *value)
{
      int *x = (int*) value;    //cast the data passed to an int pointer
      while (*x < 5)
      {              //while the value of x is less than 5
            usleep(1000);         //sleep for 1ms - encourage main thread
            (*x)++;               //increment the value of x by 1
      }
      return x;                 //return the pointer x (as a void*)
}

int main()
{
      int x = 0, y = 0;
      pthread_t thread;         //this is our handle to the pthread
      // create the thread, pass the reference, address of the function and data
      // pthread_create() returns 0 on the successful creation of a thread
      if (pthread_create(&thread, NULL, &threadFunction, &x) != 0)
      {
            printf("Failed to create the thread\n");
            return 1;
      }
      // at this point the thread was created successfully
      while (y < 5)
      {               // loop and increment y, displaying values
            printf("The value of x=%d and y=%d \n", x, y);
            ++y;
            usleep(1000);         // encourage the pthread to run
      }
      void *result;             // OPTIONAL: receive data back from pthread
      pthread_join(thread, &result);   // allow the pthread to complete
      int *z = (int*) result;         // cast from void* to int* to get z
```

```
       printf("Final: x=%d, y=%d and z=%d\n", x, y, *z);
       return EXIT_SUCCESS;
}
```

Pthread_1:

```
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>
#include <unistd.h>

#define      MAXCOUNT     2147483647

// simple delay function
void short_delay(int count)
{
       for (int i = 0; i != count; ++i)
             ;
       return;
}

// This is the thread function that will execute when the thread is created
//  it passes and receives data by void pointers
void* threadFunction_1(void *value)
{
       int *x = (int*) value;    //cast the data passed to an int

       while (*x < MAXCOUNT)
       {
             short_delay(10);      //sleep for a short delay
             ++(*x);               //increment the value of x by 1
       }
       return x;                 //return the pointer to x
}

void* threadFunction_2(void *value)
{
       int *x = (int*) value;    //cast the data passed to an int

       while (*x < MAXCOUNT)
       {
             short_delay(10);      //sleep for a short delay
```



```
                ++(*x);             //increment the value of x by 1
        }
        return x;                //return the pointer to x
}

int main()
{
        int x_1 = 0, x_2 = 0, y = 0;
        pthread_t thread_1, thread_2;        //this is our handle to the pthread

        // create the threads, returns 0 on the successful creation of each thread
        if (pthread_create(&thread_1, NULL, &threadFunction_1, &x_1) != 0)
        {
                printf("Failed to create the thread\n");
                return 1;
        }
        if (pthread_create(&thread_2, NULL, &threadFunction_2, &x_2) != 0)
        {
                printf("Failed to create the thread\n");
                return 1;
        }

        // threads successfully created, move on to perform main program loop
        const int num_loops = 10; // program will run for num_loops*100 milliseconds
        while (y != num_loops)
        {               // loop and increment y, displaying values
                printf("The value of x_1=%d, x_2=%d and y=%d \n", x_1, x_2, y);
                usleep(100);         // encourage the pthreads to run
                ++y;
        }

        // main loop completed, terminate all threads
        pthread_cancel(thread_1);
        pthread_cancel(thread_2);
        printf("Final: x_1=%d, x_2=%d, y=%d\n", x_1, x_2, y);
        return EXIT_SUCCESS;
}
```