

**Embedded Real Time Operating Systems**

# ASSIGNMENT - 1

**FALL 2019**

**INSTRUCTOR: Takis**  **Zourntos**

# CLASS: ESE 3025

|  |  |  |
| --- | --- | --- |
| Student Name | Student ID | Signature\* |
| FRANCIS VIJITH A J | C0742980 | FV |
| VIJAYAPRAKASH | C0744450 | VI |
| PRATHYUSHA MUNI | C0743277 | PM |

*\*By signing above you attest that you have contributed to this submission and confirm that all work you have contributed to this submission is your own work. Any suspicion of copying or plagiarism in this work will result in an investigation of Academic Misconduct and may result in a “0” on the work, an “F” in the course, or possibly more severe penalties.*

**Assignment -1**

**pthreads**

**Introduction:**

Pthreads is one of the parallelization techniques, which are multi-threaded processes that were earlier called POSIX, later shortened to pthreads. The big advantage of pthreads is that they share address spaces, so can share memory variables. They can also create private memory using Thread Local Storage (TLS). The pthreads interface provides basics necessary to achieve synchronization: mutexes and condition variables.

Here, in the following assignment, we create two pthreads for calculating average of given integers in array and to bubble sort them.

**Code:**

**#include** <stdio.h>

**#include** <stdlib.h>

**#include** <pthread.h>

**#include** <unistd.h>

**#define** MAXINTS 10 // maximum number of integers that user may enter

/\*

\* global variables

\*/

**int** num\_ints=0; // current number of integers entered by user

**int** done=0; // flag indicating completion of user data entry

**float** average=0.0; // global storage for the average

/\*

\* function to display the set of numbers periodically

\*/

**void** **disp\_sorted**(**int** \*data)

{

**if** (num\_ints != 0)

{

printf("the current (sorted) integer set is: ");

**for** (**int** i=0; i!=num\_ints; ++i)

{

printf("%d ", data[i]);

}

printf("\n");

}

**return**; // return the pointer to x

}

/\*

\* thread function to display the average of the set of numbers periodically

\*/

**void** **disp\_avg**(**float** x)

{

printf("the current average is: %f\n", x);

**return**;

}

/\*

\* thread function computes average of current set

\*/

**void** \***avg\_thread**(**void** \*arr)

{

**int** \*x = (**int** \*) arr; // cast the passed data to integer type

**int** sum; // storage for the sum

**while** (done==0)

{

sum=0;

// add up the integer set

**for** (**int** i=0; i!=num\_ints; ++i)

{

sum += x[i];

}

// calculate average, if possible

**if** (num\_ints !=0)

{

average = (**float**) (sum/num\_ints);

}

}

**return** x; // return the set, unchanged

}

/\*

\* thread function sorts data set in its current set

\*/

**void** \***bubb\_sort\_thread**(**void** \*arr)

{

**int** \*x = (**int** \*) arr; // cast the passed data to integer type

**int** swaps; // a flag indicating if a swap has occurred (if none occur, this set is sorted)

**int** temp;

//while (num\_ints != MAXINTS)

**while** (done==0)

{

**if** (num\_ints != 0)

{

swaps = 1;

**while** (swaps==1)

{

swaps = 0; // if a swap occurs, this flag toggled

**for** (**int** i=0; i!=(num\_ints-1); ++i)

{

**if** (x[i] > x[i+1]) // elements must be in ascending order, if not ...

{ // ... then swap!

temp = x[i];

x[i] = x[i+1];

x[i+1] = temp;

swaps = 1; // a swap has occurred

}

}

}

}

}

**return** x; // return the set, sorted

}

**int** **main**()

{

pthread\_t thread\_calc\_1; // our handle for the averaging thread

pthread\_t thread\_calc\_2; // our handle for the sorting thread

**int** set[MAXINTS]; // storage for our numbers

// initialize set to zero

**for** (**int** i=0; i != MAXINTS; ++i)

{

set[i]=0;

}

// create threads

pthred\_create(&thread\_calc\_1, NULL, &avg\_thread, (**void**\*)set)

pthred\_create(&thread\_calc\_2, NULL, &bubb\_sort\_thread, (**void**\*)set)

// main loop (get user input)

**int** i;

**int** count = 0;

**while**(i < MAXINTS)

{

scanf("%d",&i);

set[num\_ints] = i;

printf("the sorted array is:\n");

disp\_sorted(set);

prinf("the average of the array is:\n");

disp\_avg(average);

count++;

}

done=1; // user has finished entering data

**return** 0;

}

**Conclusion:**

In the above function, we’ve created an array where we can enter up to 10 integers, and two pthread functions are created where one thread calculates the average of given integers and other thread bubble sorts the array parallelly.