#include <pthread.h>

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

#include "log.h"

int logindex = 0;

int \*logi = &logindex;

// mutex for logging functions required for thread-safe code

pthread\_mutex\_t tlock = PTHREAD\_MUTEX\_INITIALIZER;

int no\_of\_threads = 0;

#define ARRAYSIZE 1000000

int no\_of\_items = 0;

//#1

//TO DO: Here go your global variables

//TO DO: declare and initialize global mutex

Declare ‘arr’ with ‘ARRAYSIZE’;

Declare and initialize ’global sum’;

Declare and initialize ‘sum\_mutex’(see Assignment 2)

void init\_arr() {

srand(time(NULL));

long i;

for(i = 0; i < ARRAYSIZE; i++) {

arr[i] = (long) (rand() % 10);

}

}

// function doit ----------------------------------------

void\* doit(void \*params) {

//#2

// TO DO: here goes your local variables

// TO DO: get from param your thread index

Declare ‘from’, ‘to’;

Declare and initialize ‘local sum’;

Declare a local variable ‘thread index’ to save ‘\*params’;

//#3

// TO DO: find your slice of the array -- from and to

// TO DO: display the message about which part of the array this threads does

// TO DO: log the message about which part of the array this threads does

from = ‘thread index’ \* no\_of\_items;

to = from + no\_of\_items - 1;

if(‘**to’** exceeds the end of the array)

{‘**to’** has to be trimmed;}

msg("the message");

Msg("the message");

//example: ’Thread 1 processing the array from 333334 to 666667’

//#4

// TO DO: sum up your portion of the array arr[]

// TO DO: display the message about your summation

// TO DO: log the message about your summation

Sum up elements in ‘arr’ from ‘from’ to ‘to’ and save it to ‘local sum’;

msg("the message");

Msg("the message");

//example: ’ Thread 2 summation is 1497599’

//#5

// TO DO: lock your global mutex

// TO DO: update the global sum

// TO DO: unlock your global mutex

// TO DO: exit the thread

lock ‘sum\_mutex’(See Assignment2);

‘global sum’ = ‘global sum’ + ‘local sum’;

Unlock ‘sum\_mutex’;

exit() the current thread;

}//end doit

// function main ----------------------------------------

int main(int argc,char \*argv[]) {

//#6

// TO DO: here go your local variables

//TO DO: check argc, argv if error, display message and terminate the program

//TO DO: set the number of threats to be dispatched

//TO DO: check it is between 2 and 10 inclusive, if not error and terminate

Declare ‘thread index’ and ‘tid’;

if(argc != 2){

msg("the message");

msg\_exit("the message");

}

//example:

//a sample of the output for ‘assgn3’

//‘wrong number of command line arguments’

//‘usage - assgn3 <number of threads>’

no\_of\_threads = ‘the scend command line argument’;

//don’t forget date type conversion

if (no\_of\_threads is not is between 2 and 10 inclusive)

{msg("the message");

msg\_exit("the message");}

//example:

// a sample of the output for assgn3 30

//wrong number of threads, must be 2 - 10

//usage - assgn3 <number of threads>

create\_log("assgn3.log");

init\_arr();

//#7

//TO DO: determine no\_of\_items each thread is supposed to sum up

//TO DO: create dynamically array to hold the index of each dispatched thread

//TO DO: create dynamically array to hold thread id of each dispatched thread

if(ARRAYSIZE is divisible by no\_of\_threads)

{ no\_of\_items = ARRAYSIZE/no\_of\_threads;}

else{ no\_of\_items = 1+(ARRAYSIZE/no\_of\_threads);}

‘thread index’= ‘using malloc() or calloc()’;

‘tid’ = ‘using malloc() or calloc()’;

//#8

//TO DO: in a loop, dispatch each thread to execute doit() (see pthread\_create())

//TO DO: and display and log message "dispatched thread XXX with tid = YYY"

for(){

‘thread index’ assignment;

pthread\_create();

msg("the message");

Msg("the message");

//example:

//’dispatched thread 1 with tid = 47362362755392’

}

//#9

//TO DO: wait for all threads to complete

//TO DO: display message "Overall sum = XXX"

for(traverse all threads){

using pthread\_join();

}

msg("the message");

//example:

//’Overall sum = 4498934’

//#10

//TO DO: sum up the array arr[]

//TO DO: and display the message "Check sum = XXX"

for(){

Sum up elements in ‘arr’ and save it to ‘local sum’;

}

msg("the message");

//example:

// Check sum = 4498934’

return 0;

}//end main