# CSC 410

## Assignment: Thread 1

Due: Monday 9/11 8pm

Submit: Text to D2L dropbox containing your code [[ NOT a file ]]

(nicely formatted)

& sample outputs

In your home directory, assignt1.c as your final code

Your task, as described in class on 8/30

- Create 8 threads [[ easy to change 2 to 12 ]]

- can be hard coded, input, whatever; just easy to change

- Each thread with tid 1 to 8 ( when 8 threads )

Odd threads: such at tid=5

- compute tid factorial 1\*2\*3\*4\*5

Even threads: such as tid=6

- compute tid summation 1+2+3+4+5+6

Testing: threads can print

Final: all output from main thread

- Main thread computes sum of the other thread values

- report each value & sum

Sample Testing with 8 threads:

A screenshot of a computer

Description automatically generated

Sum is confirmed,

Testing with 6 threads:

A screen shot of a computer program

Description automatically generated

Code:

#include <stdio.h>

#include <pthread.h>

#include <stdlib.h>

//HERE IS THE DEFINITION FOR NUMBER OF THREADS TO BE USED IN MAIN

#define NUMTHREADS 8

void \*oddthreads\_factorial(void \*thread\_id)

{

//FOR ODD THREADS, THE FACTORIAL OF THE THREAD ID PASSED IS FOUND USING A FOR LOOP, MULTIPLYING EACH ITERATION TOGETHER

//THE RESULT IS PASSED IN THE PTHREAD\_EXIT FUNCTION

long factorial = (long) thread\_id;

long factorial\_return = 1;

//printf("Thread %d\n",factorial);

for(int i=1; i<= factorial; i++)

{

factorial\_return = factorial\_return \* i;

}

//printf("Thread %ld factorial: %ld\n",factorial,factorial\_return);

pthread\_exit( (void\*) factorial\_return);

}

void \*eventhreads\_sum(void \*thread\_id)

{

//FOR EVEN THREADS, THE SUMMATION OF EACH THREAD ID PASSED IS FOUND USING A LOOP, ADDING EACH ITERATION TOGETHER

//THE RESULT IS PASSED IN THE PTHREAD\_EXIT FUNCTION

long thread\_number = (long) thread\_id;

long summation\_return = 0;

for(int i=1; i<= thread\_number; i++)

{

summation\_return = summation\_return + i;

}

//printf("Thread %ld summation: %ld\n",thread\_id,summation\_return);

pthread\_exit( (void\*) summation\_return);

}

int main()

{

pthread\_t thread\_ids[NUMTHREADS];

long factorial\_sum;

long summation\_sum;

long current\_thread =1;

long status;

long running\_sum\_main = 0;

//INITIALIZE THREADS, DETERMINE EVEN AND ODD USING MOD 2

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

{

//CURRENT THREAD IS TRACKED USING LOCAL VARIABLE, COULD HAVE USED i+1 BUT WANTED AN INT FOR THE LOOP WHICH WOULD HAVE ISSUES TYPE CASTING

//CURRENT THREAD IS PASSED TO THE THREADS

if(current\_thread % 2 == 1)

{

pthread\_create(&thread\_ids[i], NULL, oddthreads\_factorial, (void \*) current\_thread);

} else

{

pthread\_create(&thread\_ids[i], NULL, eventhreads\_sum, (void \*) current\_thread);

}

current\_thread = current\_thread +1;

}

//JOIN THREADS

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

{

//RESULT OF THREADS IS TRACKED BY STATUS WHICH IS OUTPUTTED TO THE CONSOLE EACH JOIN

//IT IS ALSO ADDED TO THE RUNNING\_SUM VARIABLE WHICH TRACKS THE SUM OF ALL THREADS OUTPUT

pthread\_join(thread\_ids[i], (void \*) &status);

printf("Value from thread %ld is: %ld\n",i+1,status);

running\_sum\_main = running\_sum\_main + status;

}

printf("Sum of factorials and summations for %ld threads is: %ld\n",NUMTHREADS,running\_sum\_main);

}