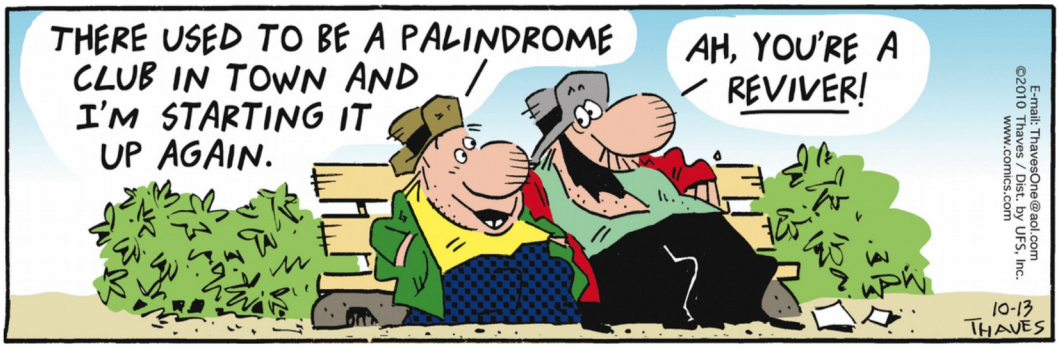
CS 218 – Assignment #12

Purpose: Become more familiar with operating system interaction, threading, and race conditions.
Due: Tuesday (4/09), by **2:30 PM** (all sections)
Points: 80 40 for program and 40 for write-up
Scoring will include functionality, documentation, coding style, and write-up

Assignment:



In recreational number theory, a palindromic number[1] is a number that remains the same when its digits are reversed. For example, 16461 is a palindromic number since when it is reversed is the same.

Write an assembly language to find the count palindromic numbers between 1 and a user provided limit (inclusive). For example, between 1 and 1,000,000 (402854_{12}) there are exactly 1,998 ($11X6_{12}$) palindromic numbers. In order to improve performance, the program should use threads to perform some of the computations in parallel.

The program should read the thread count option and number limit in Dozenal from the command line in the following format; "./palNums <-t1|-t2|-t3|-t4> -l <dozenalNumber>". For example:

```
./palNums -t4 -l 84
```

Submission:
When complete, submit:

- A copy of the **source file** via the class web page (assignment submission link) by 2:30 PM. ***Assignments received after the due date/time will not be accepted.***
- Submit a final write-up (PDF format).

Main
The provided main template, when completed, will perform some basic actions including:

- Displaying applicable header/status messages, read/validate command line arguments, and initiate thread calls as appropriate based on thread count.
- Additionally, the main will call some support functions (which you need to create);
  - Function ***getArguments()*** to read and verify command line arguments.
  - Function ***int2dozenal()*** (can change name) to convert an integer into an ASCII/Dozenal string.
  - Function ***dozenal2int()*** to convert an ASCII/Dozenal string to integer.

1 For more information, refer to: https://en.wikipedia.org/wiki/Palindromic_number

Thread Function:

- Create a thread function, ***findPalNums()***, that performs the following operations:
  - Obtain the next number to check (via global variable, ***idxCounter***)
  - While the next number is ≤ ***numberLimit*** (globally available);
    - Check for palindromic number.
    - If palindromic number, increment the *palCount* variable and store number.

Incrementing, obtaining, and checking the next number, ***idxCounter*** variable, must be performed as a critical section (i.e., locked and unlocked using the provided *spinLock1()* and *spinUnlock1()* functions). Once a palindromic number has been found, the count of palindromic should be incremented. For this we can use the lock prefix on the increment instruction to ensure it Is performed atomically (i.e., the instruction can not be interrupted corrupted by another thread).

It is recommended to complete and test the program initially with only the single sequential thread before testing the parallel threads. In order to debug, you may wish to temporarily insert a direct call (non-threaded) to the ***findPalNums()*** function.

Thread Functions

The following are the function calls required for thread management (create and wait for completion).

```
;  pthread_create(&pthreadID0, NULL, &findPalNums, NULL);
        mov     rdi, pthreadID0
        mov     rsi, NULL
        mov     rdx, findPalNums
        mov     rcx, NULL
        call    pthread_create

;  pthread_join (pthreadID0, NULL);
        mov     rdi, qword [pthreadID0]
        mov     rsi, NULL
        call    pthread_join
```

The required data structure for ***pthreadID0*** and the others are in the provided template.

Results

When the program is working, complete additional timing and testing actions as follows;

- Use the provided script file to execute and time the working program.
- Compute the speed-up[2] factor from the base sequential execution and the parallel execution times. Use the following formula to calculate the speed-up factor:

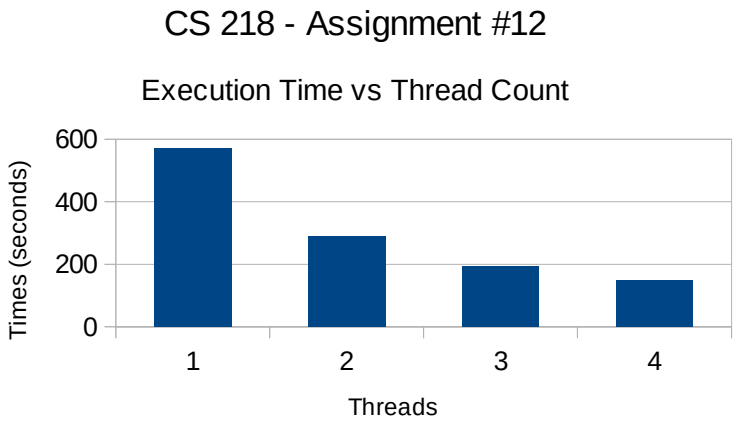
$$SpeedUp = \frac{ExecTime_{sequential}}{ExecTime_{parallel}}$$

- Remove the locking calls (the spinLock calls and the 'lock') and re-execute the program using a limit of 2,500,000,000 ($5992E3314_{12}$) for 1 thread and then 4 threads.
  - The Unix time command (as per asst #11B) should be used to obtain the execution times. Use the "real" time.

2 For more information, refer to: https://en.wikipedia.org/wiki/Speedup

- Create a final write-up including a copy of the program output from the timing script file and an explanation of the results. The explanation must address
  - the final results for 1, 2, 3, and 4 thread executions, including final results (count of palindromic numbers) and the execution times for both each.
  - the speed-up factor from 1 thread to 2, 3, and 4 threads (via the provided formula)
  - simple chart plotting the execution time (in seconds) vs thread count (see example below).
  - the difference with and without the locking calls for the parallel execution
    - explain specifically what caused the difference

  The explanation part of the write-up (not including the output and timing data) should be less than ~300 words. Overly long explanations will be not be scored.



Example Execution:

The following is an example execution for the sequential version. Note the "**ed-vm%**" is the prompt.

```
ed-vm%
ed-vm% ./palNums -t -l 84
Error, invalid thread count specifier.
ed-vm%
ed-vm% ./palNums -t -l 84
Error, invalid thread count specifier.
ed-vm%
ed-vm% ./palNums -t1 -l 0
Error, limit invalid.
ed-vm%
ed-vm%
ed-vm% ./palNums -t1 -l 84
*****************************************
Palindromic Numbers Program

--------------------------------------
Start Counting
 ...Thread starting...

Palindromic Numbers: 16


Completed.
ed-vm%
ed-vm%
ed-vm%
```

```
ed-vm% ./palNums -t4 -l e765x6628
*****************************************
Palindromic Numbers Program

-------------------------------------
Start Counting
 ...Thread starting...
 ...Thread starting...
 ...Thread starting...
 ...Thread starting...

Palindromic Numbers: 7297X


Completed.
ed-vm%
ed-vm% ./palNums -t4 -l xxxxxxxx
******************************************
Palindromic Numbers Program

-------------------------------------
Start Counting
 ...Thread starting...
 ...Thread starting...
 ...Thread starting...
 ...Thread starting...

Palindromic Numbers: 244X7


Completed.
ed-vm%
ed-vm% ./palNums -t4 -l 1811628
******************************************
Palindromic Numbers Program

-------------------------------------
Start Counting
 ...Thread starting...
 ...Thread starting...
 ...Thread starting...
 ...Thread starting...

Palindromic Numbers: 357X


Completed.
ed-vm%
ed-vm% ./palNums -t3 -l 5954
******************************************
Palindromic Numbers Program

-------------------------------------
Start Counting
 ...Thread starting...
 ...Thread starting...
 ...Thread starting...

Palindromic Numbers: 146


Completed.
ed-vm%
```