Computer Science 360 – Introduction to Operating Systems
Fall 2013

Assignment #1

Due: Friday, October 11th, 12:30 pm by submission to conneX
(no late submissions accepted)

Programming Platform

For this assignment **your code must work on machines in ECS 242**. You need not be present in the room and you can remotely login to these machines from elsewhere on campus or from home—this was described at the September 20th tutorial, and slides for that tutorial are available at our conneX course site. (The ECS 242 lab is heavily used this fall semester and so drop-in access is difficult.) You may already have access to your own Unix system (e.g., Linux, Mac OS X) yet we recommend you work on ECS 242 machines rather than try to complete the assignment on your machine for uploading to UVic. Bugs in systems programming tend to be platform-specific and something that works perfectly at home may end up crashing on a different hardware configuration.

Individual work

This assignment is to be completed by each individual student (i.e., no group work). Naturally you will want to discuss aspects of the problem with fellow students, and such discussion is encouraged. However, sharing of code is strictly forbidden. If you are still unsure about what is permitted or have other questions regarding academic integrity, please direct them as soon as possible to the instructor. (Code-similarity tools will be used to analyze submitted programs.)

Parts of this assignment

1. Write a C program implementing a solution to the “palindrome dictionary” problem.
2. Write a multi-threaded version in C of your “palindrome dictionary” problem solution.
3. Prepare a one-page summary of your work, including timings taken for executions of each program.
4. Demonstrate your work for credit and explain your chosen data structures, algorithms, synchronization constructs etc. that appear in the source code of your solution.

Part 1: Write an unthreaded C solution to the “palindrome dictionary” problem

Palindromes are words or phrases that spell the same way backwards and forwards. **For this problem, however, your task is somewhat different.** Your program is to analyze a dictionary of words and output those words having their reverse in the dictionary. (Note: The dictionary of words will be provided to you and contains both proper and common nouns.)

Some words are clearly their own palindromes and will be output by your program. Examples are:

- anna
- ewe
- reviver
- radar

Each word in this group will be output to stdout on its own line exactly once.

Other words have their reverse in the dictionary. Examples are:

- enamor & romane
- leper & repel
- mood & doom

Each word in a pair will be output by your program (i.e., both “enamor” and “romane”) on their own line exactly once.

Both sets of words (i.e., words from each group) are to be output together in alphabetical order. For example, the words above would be output as:

```
anna
doom
enamor
ewe
leper
mood
radar
repel
reviver
romane
```

Your solution to this first part of the assignment is *single threaded program* (i.e., unthreaded as the only thread is the main thread) named **palin.c**. Please keep all code within a single source file. The dictionary of words is in a text file posted with the assignment (i.e., in an archive named **words.zip**, with uncompressed file named **words**).

Part 2: Write a multithreaded C solution to the “palindrome dictionary” problem

Once you are finished Part 1 you will have a solution to the problem that takes a specific amount of time to run. You are now to convert that solution into a multi-threaded solution.

One approach is to create a single thread for each letter of the alphabet. These threads will run the same function, but the function will be parameterized on the letter. In this way you can ensure each thread looks for palindromes—as described in part 1—through a single range of the word lists (i.e., that part of the list containing words beginning with that letter). **You must ensure that all race conditions are handled via the use of pthreads semaphore synchronization operations.** (You may only use semaphores.) However, you must be parsimonious in the use of such operations—you do not want to simply convert your multithreaded program into one that behaves essentially as a single-threaded program (i.e., marks will be deducted for such approaches).

Your output for this program must match that for Part 1. The program is to be named **threaded_palin.c**. Please keep all code in a single source-code file. You must use the same dictionary of words as in part 1.

For both parts 1 and 2:

- Use dynamic memory where possible. A better solution will not use a static array to store lists of words (and some marks will be given for the quality of your solutions).
- Busy waiting is strictly forbidden. In technical terms, this means you may not have any loops which have either empty or null-effect bodies.
- Result words are to be output in alphabetical order, one per line.
- As mentioned above, some marks will be given for the quality of your solution. For example, programs consisting of only two or three functions will not contain proper decompositions of the problem and will not receive full marks.
- All input is to be from stdin and all output is to stdout. During evaluation your code will be invoked using the command “cat words | ./palin” or “cat words | ./threaded_palin”.

Part 3: One page summary

On a single page describe briefly the algorithms used in each of the programs, the race conditions identified, and your approach to handling these conditions. You must also include five timings for each of the programs along with some explanation of the data. (Use "time" to obtain these timings.) A very brief explanation of the timings should accompany the numbers. *The summary must be either (a) plain text or (b) a PDF document.* (Please—no Microsoft Word, OpenOffice, Pages, etc. documents.)

Part 4: Evaluation

Students will demonstrate their solution to a member of the teaching team. The demo will be the basis for your assignment evaluation and grade. Demo signup sheets will be posted nearer to the assignment due date.

What you must submit via conneX

- Code for **palin.c**
- Code for **threaded_palin.c**
- One-page summary

Evaluation

Our grading scheme is relatively simple.

- "A" grade: An exceptional submission demonstrating creativity and initiative. The palindrome solutions run without any problems.

- "B" grade: A submission completing the requirements of the assignment. The palindrome solutions run without any problems.

- "C" grade: A submission completing most of the requirements of the assignment. The palindrome solutions run with some problems.

- "D" grade: A serious attempt at completing requirements for the assignment. The palindrome solutions run with quite a few problems.

- "F" grade: Either no submission given, or submission represents very little work.