CSCI 360 - Lab 2 Report

Sami Al-Qusus

Sept 24, 2018

Assignment Goals:

- Experimenting with threads.

Assignment Instructions (Dr. LoPinto):

- Use the example thread program discussed in class to experiment with threads.
- Get the program to compile and run with pthread_join commented out.
- Run the program a few times and record your observations.
- Next, uncomment the pthread_join statment and run the program many times.
    - Compare the results.
    - Summarize your observations and try to explain the differences.
- Explain the code (use the man pages to explore the arguments used and their alternatives.) Include your makefile, code, and a report in PDF format.

What I did:

- Went over the code provided to me line by line:
    - Looked up the functions, variables or anything new to me in the code.
    - Then made comments beside it, to remember what it does.
- Ran the program multiple times with `pthread_join` function commented out and recorded my observations.
- Then ran the program multiple times with the `pthread_join` function and noted the different results.
- At the end summarized my observations and understandings.

Observation:

- When we run the program with pthread_join commented out, we just get an output
    - `Test Threads`
    - I think the reason is that the main() function returns before the thread terminates.

- When we run the program with pthread_join, we see that the sender function is executed as a thread when its name is specified in pthread_create(), because pthread_join blocks the calling thread until the thread with identifier equal to the first argument terminates and we get the output:
    - Test Threads
    - Thread: Count is 0
    - Thread: Count is 1
    - Thread: Count is 2
    - Thread: Count is 3
    - Thread: Count is 4
    - Thread: Count is 5
    - Thread: Count is 6
    - Thread: Count is 7
    - Thread: Count is 8
    - Thread: Count is 9
    - Thread: Count is 10
    - Thread: Count is 11
    - Thread: Count is 12
    - Thread: Count is 13
    - Thread: Count is 14
    - Thread: Count is 15
    - Thread: Count is 16
    - Thread: Count is 17
    - Thread: Count is 18
    - Thread: Count is 19

What I learnt:

- How to declare a thread with `pthread_t`.
    - We need this pthread library #include <pthread.h>
- Learnt about using void as a variable declaration.
- How to create a thread using `pthread_create`
    - Function to create a thread
    - Takes 4 arguments
    - First argument is a pointer to thread_id which is set by this function
    - Second argument specifies attributes
    - Value is NULL, then default attributes are used
    - Third argument is name of function to be executed for the thread to be created
    - Fourth argument is used to pass arguments to thread

- Learnt that `pthread_join` is kind of equivalent to wait() for processes.
    - A call to pthread_join blocks the calling thread until the thread with identifier equal to the first argument terminates.
- "Threads are not independent of one another like processes as a result threads shares with other threads their code section, data section and OS resources like open files and signals. But, like process, a thread has its own program counter (PC), a register set, and a stack space."
- From searching around I also found out that pthread_detach() allows the thread's storage to be cleaned up after the thread terminates.

Credits:

MODERN OPERATING SYSTEMS

- Modern Operating Systems 3rd ed.
- Dr. LoPinto lecture notes.
- https://www.geeksforgeeks.org/multithreading-c-2/
- https://stackoverflow.com/questions/1064640/do-i-have-to-pthread-join-each-thread-i-create