Programming Project 2: The Dining Philosophers

Due: Monday 10/19/2015 at 11:59:59 PM

15 points

This is an INDIVIDUAL assignment

This project is designed to give you practice using pthreads in C while employing mutexes and avoiding deadlock.

You are to implement one of the solutions discussed in class for avoiding deadlock in the Dining Philosopher's problem. To remind you, these solutions avoid deadlock by either breaking the Hold and Wait condition (using a semaphore) or the Circular Wait condition (by ordering the resources). In this implementation, you will represent the chopsticks (or forks, or spoons, or whatever) as `mutexes`. You will design your program to show that it can avoid deadlock by allowing the philosophers (represented by threads) to think for an arbitrary amount of time, as specified on the *command line*.

To do this, you must employ the arguments passed to the main function: `int argc, char* argv[]`. Now that you understand types and pointers in C, you may recognize that the int is an int, and the char*[] is actually an array of character arrays (or... strings!). If I were to compile my program, and it has the name `driver`, if I run `./driver asdf qwert` from the command line, `argc` would be 3, and `argv` would contain `["driver", "asdf", "qwert"]`. As such, if I were to run `./driver 1000`, `argc` would be 2, and `argv[1]` would be `"1000"`.

In order to get this to be an integer value, we must use the `atoi` function. This function takes a null terminated character array, and parses a number out of it. I won't go into the specifics of the error modes on it, but you can feel free to investigate that if you are curious. So, you can determine the number of times the philosophers must loop eating and thinking with the following code:

```
int numloops;
if (argc < 2)
{
    printf("You must specify a number of times to run the loop");
    exit(1); // stop if they didn't give us a number
}
numloops = atoi(argv[1]);
```

You should then send this value (and an id) to each philosopher. When a philosopher is created, you should output a message indicating this. Every time a philosopher thinks, they should print out `"Philosopher %d thinking; count = %d", thread_id, count`, or some other equally enlightening message (see the sample output). C does not provide us with a think() function. However, as we all know, a good meal makes us sleepy. Thus, to simulate thinking, the philosophers will call the `usleep` function, provided in the `unistd.h` header file. You should call it with an argument of 3 (sleep for 3 microseconds, or `usleep(3)`). You should also use this `usleep` function when a philosopher eats (and print an appropriate message).

When a philosopher has eaten and thought/slept the appropriate number of times, they should print `"Philosopher %d is leaving the table"` and exit the thread.

Your solution is to work for the case of 4 philosophers and 4 eating implements. As such, you will have 4 threads (besides main) and 4 mutexes. If you choose the semaphore solution, you must have a semaphore as well (that should go without saying).

Bonus point: For an extra bonus point, give every philosopher a unique name (an array of strings would serve you well), and print those instead of the philosopher's ID.

Coding Standards and Readability All source files you submit should be readable and follow formatting standards similar to those expressed in other courses. I will not specify that you must indent four or two or eight spaces, or always use Next-Line style braces. However, your code must be consistently formatted throughout.

Comments For every function you write, you must specify the following information above the function prototype:

```
/*
 * brief: a brief description of the purpose of the function
 * return: a description of the return value, if there is one
 */
```

Additionally, the top of every source file (`.c` and `.h`) must contain the following header block:

```
/*
 * Name:
 * Course:
 * Description:
 * Date:
 * By submitting this assignment, I acknowledge that:
    - I have not and will not share this code at any time with any present or future
         students in the course without express permission by the instructor under any
         circumstances
    - I have not copied or otherwise obtained this code from an outside source
    - I am claiming that I am responsible for the authorship of this code
    - Failing to adhere to the above statement can be construed as academic dishonesty
          and will be treated as such
 */
```

You are, of course, responsible for filling in your name, date, course, and description.

Submission: You are to submit your source code (a single file is fine for this assignment) and a makefile that will build the assignment. Remember that when building your code on a linux machine, you will require the `-lpthread` flag when calling gcc. So... `gcc -lpthread -g philosophers.c -o philosophers` will compile my solution correctly.

Grading: You will be graded on the following criteria:

Running to Completion: If your code does not run to completion cleanly (crashes, etc), it will be graded at a harsh penalty (50% of earned grade)	
Compiling: If your code does not compile on a modern version of `gcc`, you will receive zero points for the assignment. You only receive full points in this category if your code compiles cleanly (without errors or warnings)	**3 pts.**
Memory Management: You will only get full points in this category if your code does not trigger any memory errors or memory leaks in `valgrind`. Deductions will be based on the number and severity of errors and leaks reported.	**4 pts.**
Correct Output	**4 pts.**
Readability: Full points will only be awarded if the code is arranged consistently and cleanly, and follows the commenting standards outlined in this assignment and in class. Additionally, if the appropriate file comment headers are not included in every file, this assignment will receive zero points.	**4 pts**

Late Policy: On this and all future assignments, the first 24 hours after the due date is a 50% penalty (half your earned grade, not a flat 50% off the top). After the second 24 hours, no grade will be assigned to the submission.

Sample output: Command run: `./philosophers 5`

```
Philosopher 0 joined the table
Philosopher 0 is eating
Philosopher 0 joined the table
Philosopher 0 joined the table
Philosopher 2 is eating
Philosopher 0 joined the table
Philosopher 0 is pondering the meaning of life; count = 0
Philosopher 2 is pondering the meaning of life; count = 0
Philosopher 1 is eating
Philosopher 3 is eating
Philosopher 1 is pondering the meaning of life; count = 0
Philosopher 3 is pondering the meaning of life; count = 0
Philosopher 0 is eating
Philosopher 2 is eating
Philosopher 0 is pondering the meaning of life; count = 1
Philosopher 2 is pondering the meaning of life; count = 1
Philosopher 1 is eating
Philosopher 3 is eating
Philosopher 1 is pondering the meaning of life; count = 1
Philosopher 3 is pondering the meaning of life; count = 1
Philosopher 2 is eating
Philosopher 0 is eating
Philosopher 0 is pondering the meaning of life; count = 2
Philosopher 2 is pondering the meaning of life; count = 2
Philosopher 1 is eating
Philosopher 3 is eating
Philosopher 3 is pondering the meaning of life; count = 2
Philosopher 1 is pondering the meaning of life; count = 2
Philosopher 2 is eating
Philosopher 0 is eating
Philosopher 0 is pondering the meaning of life; count = 3
Philosopher 2 is pondering the meaning of life; count = 3
Philosopher 1 is eating
Philosopher 3 is eating
Philosopher 1 is pondering the meaning of life; count = 3
Philosopher 3 is pondering the meaning of life; count = 3
Philosopher 0 is eating
Philosopher 2 is eating
Philosopher 0 is pondering the meaning of life; count = 4
Philosopher 2 is pondering the meaning of life; count = 4
Philosopher 3 is eating
Philosopher 1 is eating
Philosopher 0 is leaving the table
Philosopher 1 is pondering the meaning of life; count = 4
Philosopher 2 is leaving the table
Philosopher 3 is pondering the meaning of life; count = 4
Philosopher 1 is leaving the table
Philosopher 3 is leaving the table
```