CS 105
Dining Philosophers Lab

See class calendar for submission and due dates

December 17, 2012

12:08 AM

Introduction and Goals

You are to implement a solution to the Dining Philosopher's problem. Your solution is to work in the following way:

- You are free to use attached the solution, but you MUST document each and every line of the solution.
- Each philosopher should be a process. Use *N* as the number of philosophers, *N* should be an input parameter to your solution.
- You are to use POSIX semaphores.
- You are free to use shared memory (a good choice) for your critical sections.
- You are to collaborate in groups of two.
- Add to each philosopher a count of the number of times he has eaten.
- When each philosopher has eaten the *max* number of times (*max* provided as an input parameter) the philosopher stops eating (his process dies).
- The time period (*etime* for each eating session is also an input parameter (fixed for all philosophers).

While this is a neat problem, as written no one knows what the heck is going on. Thus you /textbfMUST have another process, the **Waiter**, who monitors the Philosopher's table. Because he wants to get a good tip, every 2 seconds the **Waiter** looks at the table and determines, i.e., prints out in a horizontal format:

- Printout #
- How many Philosophers are left.
- How many Philosophers are eating.
- How many Philosophers are thinking.
- The current values in the state table for each Philosopher, e.g., Hungry, Thinking, Eating, or Gone.

Termination
When all the Philosophers are done eating the **Waiter**, prints out one more time and the program dies.

Input Parameters

There are 4 parameters to the program (in order):

- How many Philosophers are coming to dinner, defaults to 5.
- How long (in seconds) each Philosopher eats, defaults to 6.
- How many times each Philosopher eats, defaults to 4.
- How long (in seconds) each Philosopher thinks, defaults to 3.

Semaphore Notes:

Submission.
See the Lab web page for specific instructions for submitting this assignment.
What to Submit:

- *diningphils.c*
  file that is your implementation and describes in detail your overall implementation:
  - Your **well commented** dining philosophers code
  - including the number of semaphores, and the use of each semaphore,
- *output.h*
  Sample run with the default input values,.

References

- Chapter 3, *UNIX Network Programming*, W. Richard Stevens.
- *The Little Book of Semaphores*, Allen B. Downey.

Notes:

If you decide to not implement shared memory with a dynamic size array, then use a max size of 10 philosophers. 5 extra points for using a dynamic sized array. You must also indicate this choice in the documentation comments at the front of your solution code.

We reserve the right to change the problem statement when someone demonstrates the ambiguity of said problem statement.

```
#define N
#define LEFT  (i+N-1)%N
#define RIGHT (i+1)%N
#define THINKING 0;  #define HUNGRY  1; #define EATING   2;
typedef int semaphore;
int state(N);
semaphore mutex = 1;
semaphore s[N];

void phil(int i)
{
   while(TRUE)
   {
      think();
      takeForks(i);
      eat();
      putForks(i);
   }
}

void takeForks(int i)
{
   down(mutex);
   state[i] = HUNGRY;
   test(i);
   down(s[i]);
}

void putForks(int i)
{
   down(mutex);
   state[i] = THINKING;
   test(LEFT);
   test(RIGHT);
   up(mutex);
}

void test(int i)
{
   if (state[i] == HUNGRY && state[LEFT] != EATING && state[RIGHT] != EATING)
   {
      state[i] = EATING;
      up(state[i]);
   }
}
```