Project 5: More Synchronization

CS 411

Introduction

Continue using your Project 4 repository for these two programs.

Dining Philosophers Variation

The "obvious" solution to the Dining Philosophers problem for five philosophers suffers from deadlock:

```
while (1) {
   think();
   wait(left_chopstick);
   wait(right_chopstick);
   eat();
   signal(left_chopstick);
   signal(right_chopstick);
}
```

Model the chopsticks using binary semaphores. Eliminate the circular wait property necessary for a deadlock condition by using a counting semaphore to ensure that no more than four philosophers are trying to eat at any time. Keep in mind that the Project 4 Dining Philosophers program used a mutex and condition variables, so you will be starting this program from scratch, because of its use of semaphores.

Philosophers should alternate between thinking for 1 millisecond and eating for 1 second. (See the documentation in Section 3 of the Linux man pages for `usleep()` and `sleep()`. Carefully note the units of these function's parameters.) Each philosopher should announce its state changes (thinking, hungry, and eating) and thread ID via `printf()` calls. Do remember to unbuffer I/O. See example output below.

Pascal Pizza Party Protocol

The MaCS Department is throwing its annual Pascal Pizza Party, with a twist. A pizza maker brings out a single 42 slice pizza at a time, and then goes into a back room to continue devising evil programming projects for his operating systems students. The students at the party alternate between talking and eating. 40 of the students take a single slice of pizza at a time when they eat. The 41st student takes two slices at a time and the 42nd student takes three slices at a time. If either of these two latter students don't find enough available slices when they want to eat, they wait for the next pizza to appear. The student who takes the last slice of pizza notifies the pizza maker to produce the next pizza.

Write a program to solve this problem. You may use Pthreads semaphores, mutexes, and condition variables as you see fit. Hint: You'll probably want a single condition variable, used by all of the student threads. See `pthread_cond_broadcast()`.

Students alternate between talking (a random amount of time between 0 and 1 seconds; use the delay function from Project 4 and a *unique* random seed for each student), standing in the pizza line, and eating (0 delay). The student thread function should take as a parameter the number of slices to take. Students announce how many slices they want when they enter the line, that they are waiting if there aren't enough slices for them currently, the number of slices they took, and the number remaining, and when they request another pie from the pizza maker. The pizza maker announces when he has provided a fresh pie. The pizza maker thread function should take as a parameter the number of slices in a pie. Again, do unbuffer I/O. See example output below.

Example Dining Philosophers Output

```
Phil 3 thinking
Phil 1 thinking
Phil 4 thinking
Phil 0 thinking
Phil 2 thinking
Phil 3 hungry
Phil 3 eating
Phil 1 hungry
Phil 1 eating
Phil 0 hungry
Phil 4 hungry
Phil 2 hungry
Phil 3 thinking
Phil 1 thinking
Phil 2 eating
Phil 0 eating
Phil 3 hungry
Phil 1 hungry
Phil 0 thinking
Phil 4 eating
Phil 1 eating
Phil 2 thinking
Phil 0 hungry
Phil 2 hungry
Phil 4 thinking
Phil 3 eating
Phil 1 thinking
Phil 0 eating
Phil 4 hungry
Phil 1 hungry
Phil 3 thinking
Phil 0 thinking
Phil 4 eating
Phil 2 eating
Phil 3 hungry
Phil 0 hungry
Phil 4 thinking
Phil 2 thinking
Phil 1 eating
Phil 3 eating
Phil 4 hungry
Phil 2 hungry
Phil 3 thinking
Phil 1 thinking
Phil 2 eating
Phil 0 eating
Phil 3 hungry
Phil 1 hungry
Phil 0 thinking
Phil 4 eating
Phil 1 eating
Phil 2 thinking
Phil 0 hungry
Phil 2 hungry
Phil 4 thinking
Phil 3 eating
Phil 1 thinking
Phil 0 eating
Phil 4 hungry
Phil 1 hungry
Phil 3 thinking
Phil 0 thinking
Phil 4 eating
Phil 2 eating
Phil 3 hungry
Phil 0 hungry
Phil 4 thinking
Phil 2 thinking
Phil 1 eating
Phil 3 eating
Phil 4 hungry
Phil 2 hungry
```

Example Pascal Pizza Party Protocol Output

```
Want 1 slices.
Gotta wait.
Want 1 slices.
Gotta wait.
Want 1 slices.
Gotta wait.
Want 1 slices.
Gotta wait.
Want 1 slices.
Gotta wait.
```

```
Want 1 slices.
Gotta wait.
Want 1 slices.
Gotta wait.
Want 1 slices.
Gotta wait.
Want 1 slices.
Gotta wait.
Want 1 slices.
Gotta wait.
Want 1 slices.
Gotta wait.
Want 1 slices.
Gotta wait.
Want 1 slices.
Gotta wait.
Want 1 slices.
Gotta wait.
Want 1 slices.
Gotta wait.
Want 1 slices.
Gotta wait.
Want 1 slices.
Gotta wait.
Want 1 slices.
Gotta wait.
Want 1 slices.
Gotta wait.
Want 1 slices.
Gotta wait.
Serving a fresh pie.
Got my 1 slices.  41 remain
Got my 1 slices.  40 remain
Want 1 slices.
Got my 1 slices.  39 remain
Want 1 slices.
Got my 1 slices.  38 remain
Want 1 slices.
Got my 1 slices.  37 remain
Got my 1 slices.  36 remain
Got my 1 slices.  35 remain
Got my 1 slices.  34 remain
Got my 1 slices.  33 remain
Want 1 slices.
Got my 1 slices.  32 remain
Got my 1 slices.  31 remain
Got my 1 slices.  30 remain
Want 1 slices.
```

```
Got my 1 slices.  29 remain
Got my 1 slices.  28 remain
Want 1 slices.
Got my 1 slices.  27 remain
Got my 1 slices.  26 remain
Got my 1 slices.  25 remain
Got my 1 slices.  24 remain
Got my 1 slices.  23 remain
Want 1 slices.
Got my 1 slices.  22 remain
Want 1 slices.
Got my 1 slices.  21 remain
Want 1 slices.
Got my 1 slices.  20 remain
Want 1 slices.
Got my 1 slices.  19 remain
Got my 1 slices.  18 remain
Got my 1 slices.  17 remain
Got my 1 slices.  16 remain
Got my 1 slices.  15 remain
Want 1 slices.
Got my 1 slices.  14 remain
Want 1 slices.
Got my 1 slices.  13 remain
Got my 1 slices.  12 remain
Got my 1 slices.  11 remain
Got my 1 slices.  10 remain
Want 1 slices.
Got my 1 slices.  9 remain
Want 2 slices.
Got my 2 slices.  7 remain
Want 2 slices.
Got my 2 slices.  5 remain
Want 3 slices.
Got my 3 slices.  2 remain
Want 3 slices.
Gotta wait.
Want 1 slices.
Got my 1 slices.  1 remain
Want 1 slices.
Got my 1 slices.  0 remain
Oh, Mr. Pizza Maker...
Serving a fresh pie.
Want 2 slices.
Got my 2 slices.  40 remain
Want 2 slices.
Got my 2 slices.  38 remain
Want 1 slices.
```

```
Got my 1 slices.  37 remain
Want 1 slices.
Got my 1 slices.  36 remain
Want 1 slices.
Got my 1 slices.  35 remain
Want 1 slices.
Got my 1 slices.  34 remain
Want 1 slices.
Got my 1 slices.  33 remain
Want 1 slices.
Got my 1 slices.  32 remain
Got my 3 slices.  29 remain
Want 3 slices.
Got my 3 slices.  26 remain
Want 1 slices.
Got my 1 slices.  25 remain
Want 1 slices.
Got my 1 slices.  24 remain
Want 1 slices.
Got my 1 slices.  23 remain
Want 1 slices.
Got my 1 slices.  22 remain
Want 1 slices.
Got my 1 slices.  21 remain
Want 1 slices.
Got my 1 slices.  20 remain
Want 1 slices.
Got my 1 slices.  19 remain
Want 1 slices.
Got my 1 slices.  18 remain
Want 1 slices.
Got my 1 slices.  17 remain
Want 1 slices.
Got my 1 slices.  16 remain
Want 1 slices.
Got my 1 slices.  15 remain
Want 1 slices.
Got my 1 slices.  14 remain
Want 1 slices.
Got my 1 slices.  13 remain
Want 1 slices.
Got my 1 slices.  12 remain
Want 1 slices.
Got my 1 slices.  11 remain
Want 1 slices.
Got my 1 slices.  10 remain
Want 1 slices.
Got my 1 slices.  9 remain
```

```
Want 1 slices.
Got my 1 slices.  8 remain
Want 1 slices.
Got my 1 slices.  7 remain
Want 1 slices.
Got my 1 slices.  6 remain
Want 1 slices.
Got my 1 slices.  5 remain
Want 1 slices.
Got my 1 slices.  4 remain
Want 1 slices.
Got my 1 slices.  3 remain
Want 1 slices.
Got my 1 slices.  2 remain
Want 1 slices.
Got my 1 slices.  1 remain
Want 1 slices.
Got my 1 slices.  0 remain
Oh, Mr. Pizza Maker...
Serving a fresh pie.
Want 1 slices.
Got my 1 slices.  41 remain
```