Concurrent Programming
Lecture 26

12th December 2003

On the traffic signal project

I'd like to offer a few observations on this project, having less to do with concurrent programming and more to do with general programming practice.

How did I approach this problem:

- Thread per signal plus a sensor thread
- Single signal thread is also workable - as you found out. But maybe there are better ways to conceive the single thread.
- Start by thinking of a thread per signal. That thread is going to go through red/green/yellow cycles. Only one can be green at a time and the rules are really all about how long to stay green and how long to stay yellow. You have to stay red until all the other lights are red. And you need a rule for choosing the next light to turn green when you turn red.
- With this we can think about what is basically a FSM: green/red/yellow. Transition out of green is governed by: passage of time and arrival of certain sensors; the same for transition from yellow, but with different rules.
- Structure of the solution

```
while (1) {
   /* green */
   while StayGreen (...) {}
   change to yellow
   while StayYellow(...) {}
   change to red
   set up for next green (different signal)
}
```

- I find it very helpful to separate the code like this: the overall logic is just a sequence of green/yellow/red and that is made very apparent by this loop. The time of the transition out of green (and out of yellow) is a somewhat messy calculation but it is moved out of the main loop and is made self-contained.

- adding the train: the rules are sufficiently different for the train that I put in a separate set of states to handle things from the time of the TA sensor to the time of the TD sensor (including the transition back to normal operation).

Other observations:

- It is really important to not duplicate code: you can quickly end up with an unmanageable mess and the more copies there are the worse it gets. Just think about what happens if you find a mistake and have to make a change: you have to identify N places and make the change correctly at all of them.
- I really encourage you to seek a degree of elegance in all the code that you write. Have a focus: generalized solution, reusable solution, fast solution, clear solution, etc. Know which is important (or at least take a stand on what is important).