Programming Assignment 2

CMPE 250, Data Structures and Algorithms, Fall 2015

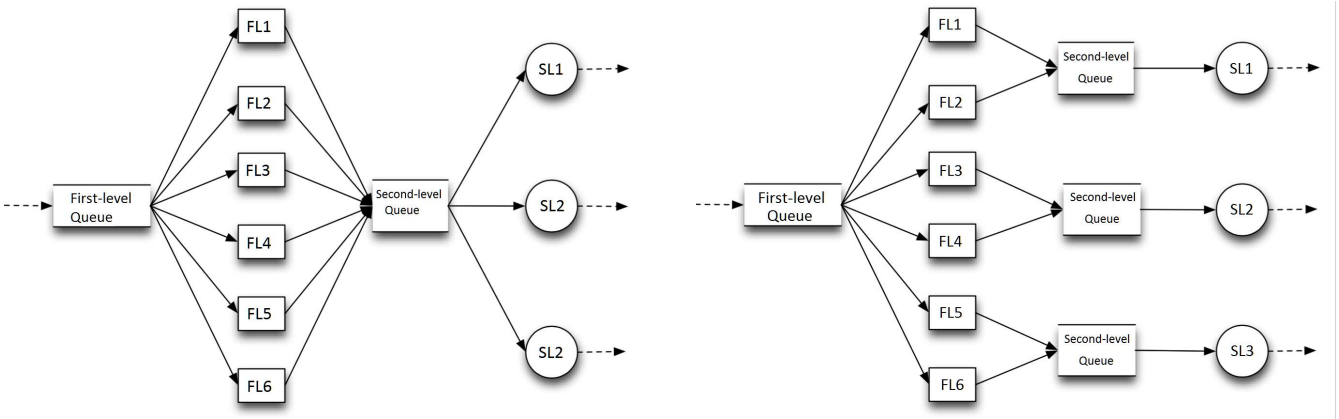
Instructor: A. T. Cemgil
TA's: Çağatay Yıldız, Atakan Arıkan, Barış Kaya

Due: December 11, 2015, 23:54 Sharp

Discrete Event Simulation

Assume that we want to simulate the working flow of a cargo company. This company has two types of units in their work flow: first-level units that collect and process the package information and second-level units that put the given packet into the correct truck. Whenever a new package comes, first-level units will get the package, process the related information and pass the package onto the second-level units.

These units may be connected to each other within different layouts. For the sake of simplicity, in this project, we are interested in only two types of layouts. Figure 1(a) shows the first type of layout where there's a single queue for the packages that passed the first-level processing and ready to begin with second-level processing. Figure 1(b) shows the other type of layout where there is a single queue for every 2 first-level units and 1 second-level units



(a) Layout 1: Single second-level queue

(b) Layout 2: Multiple second-level queues

Figure 1: Possible factory layouts

In discrete event simulations, you typically have events of different types and you observe (and update) the system while events are handled. Here is a good reading on DSE. In this project, you are going to simulate these two different layouts and collect the following statistics:

1. Average turnaround time
2. Maximum length of first-level queue
3. Maximum length of second-level queue
4. Average wait time of all packages
5. The longest wait time of all packages
6. Total running time of the cargo office

Input/Output File Format

As input, you are going to have the number of first level units, followed by their processing times, then the number of second-level units, followed by their processing times and finally, the number of packets and their entrance times. The input file has the following format:

1. First line is the *number of first-level units (N).*
2. Next N lines contain process time information of the first-level units.
3. Next line contains the *number of second-level units (M).*
4. Next M lines contain process time information of the second-level units.
5. Next line contains the *number of cargo packets (L)*
6. Remaining lines contain arrival times of packets.

The output file has the following format.

1. Average turnaround time, which is equal to
   the time that the package is sent to the truck - the arrival time of the package
2. Maximum length of first-level queue
3. Maximum length of second-level queue
4. Average wait time of all packages
5. Longest wait time of all packages
6. The last line is the total running time of the cargo office, which is equal to
   the time when the last order is packed - the arrival time of the first package

You are going to print out these values again, for the second factory layout in the same file. Do not forget to put an empty line between them. An example of exact input and output files can be seen in Table 1.

Sample Input File	Sample Output File
6	4.5
1.2	0
2.2	1
0.9	0.2
3.3	0.8
0.7	8.9
1.5	
3	a
3.0	b
3.4	c
3.2	d
4	e
0.0	f
1.0	
2.0	
3.0	

Table 1: Sample Input and Output files

Implementation Details

1. The number of second-level units are always half of the first-level units.

2. Whenever a unit finishes its job, it immediately fetches the package waiting in the queue connected to the unit. If that queue is empty, the unit goes idle.

3. Whenever more than 1 idle units are available, the unit with the **smallest** ID gets the job. For example, if a package arrives to the cargo office and $FL1$ and $FL5$ are idle, $FL1$ starts processing.

4. As always, your program will be compiled with `make` command. Therefore, if you add new files, you have to update `Makefile` accordingly so that your code auto-compiles. Note that it is fine to code in a single file in this project (Because I am assuming you already know what `.h` and `.cpp` files are and how to use those)

5. I will execute your program with `./project2 inputFile outputFile` command. So, use command line arguments to your main function accordingly.

Project Guidelines

- **Warning:** All source codes are checked automatically for similarity with other submissions and also the submissions from previous years. Make sure you write and submit your own code.
- Your program will be graded based the correctness of your output and the clarity of the source code. Correctness of your output will be tested automatically so make sure you stick with the format described above.
- There are several issues that makes a code piece 'quality'. In our case, you are expected to use C++ as powerful, steady and flexible as possible. Use mechanisms that affects these issues positively.
- Make sure you document your code with necessary inline comments, and use meaningful variable names. Do not over-comment, or make your variable names unnecessarily long.
- Try to write as efficient (both in terms of space and time) as possible. Informally speaking, try to make sure that your program completes in meaningful amount of time.