

```
1
2  /*
3   *
4   * Simulation_Run of A Single Server Queueing System
5   *
6   * Copyright (C) 2014 Terence D. Todd Hamilton, Ontario, CANADA,
7   * todd@mcmaster.ca
8   *
9   * This program is free software; you can redistribute it and/or modify it
10  * under the terms of the GNU General Public License as published by the Free
11  * Software Foundation; either version 3 of the License, or (at your option)
12  * any later version.
13  *
14  * This program is distributed in the hope that it will be useful, but WITHOUT
15  * ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or
16  * FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for
17  * more details.
18  *
19  * You should have received a copy of the GNU General Public License along with
20  * this program. If not, see <http://www.gnu.org/licenses/>.
21  *
22  */
23
24  /*****
25
26  #include <stdlib.h>
27  #include <stdio.h>
28  #include <math.h>
29  #include "output.h"
30  #include "simparameters.h"
31  #include "packet_arrival.h"
32  #include "cleanup_memory.h"
33  #include "trace.h"
34  #include "main.h"
35
36  *****/
37
38  /*
39   * main.c declares and creates a new simulation_run with parameters defined in
40   * simparameters.h. The code creates a fifo queue and server for the single
41   * server queueing system. It then loops through the list of random number
42   * generator seeds defined in simparameters.h, doing a separate simulation_run
43   * run for each. To start a run, it schedules the first packet arrival
44   * event. When each run is finished, output is printed on the terminal.
45   */
46
47  int
48  main(void)
49  {
50      Simulation_Run_Ptr simulation_run;
51      Simulation_Run_Data data;
52
53      /*
54       * Declare and initialize our random number generator seeds defined in
```

```
55     * simparameters.h
56     */
57
58     unsigned RANDOM_SEEDS[] = {RANDOM_SEED_LIST, 0};
59     unsigned random_seed;
60     int j=0;
61
62     /*
63      * Loop for each random number generator seed, doing a separate
64      * simulation_run run for each.
65      */
66
67     while ((random_seed = RANDOM_SEEDS[j++]) != 0) {
68
69         simulation_run = simulation_run_new(); /* Create a new simulation run. */
70
71         /*
72          * Set the simulation_run data pointer to our data object.
73          */
74
75         simulation_run_attach_data(simulation_run, (void *) & data);
76
77         /*
78          * Initialize the simulation_run data variables, declared in main.h.
79          */
80
81         data.blip_counter = 0;
82         data.arrival_count = 0;
83         data.number_of_data_packets_processed = 0;
84         data.number_of_voice_packets_processed = 0;
85         data.accumulated_data_packet_delay = 0.0;
86         data.accumulated_voice_packet_delay = 0.0;
87         data.random_seed = random_seed;
88
89         /*
90          * Create the packet buffer and transmission link, declared in main.h.
91          */
92
93         data.buffer = fifoqueue_new();
94         data.link    = server_new();
95
96         /*
97          * Set the random number generator seed for this run.
98          */
99
100        random_generator_initialize(random_seed);
101
102        /*
103         * Schedule the initial data and voice packet arrival for the current clock
104         * time (= 0).
105         */
106
107        schedule_packet_arrival_event(simulation_run,
108                                     simulation_run_get_time(simulation_run));
```

```
109     schedule_voice_packet_arrival_event(simulation_run,
110                                           simulation_run_get_time(simulation_run));
111
112     /*
113      * Execute events until we are finished.
114      */
115
116     while((data.number_of_data_packets_processed +
117 data.number_of_voice_packets_processed) < RUNLENGTH) {
118         simulation_run_execute_event(simulation_run);
119     }
120
121     /*
122      * Output results and clean up after ourselves.
123      */
124
125     output_results(simulation_run);
126     cleanup_memory(simulation_run);
127 }
128
129 getchar(); /* Pause before finishing. */
130 return 0;
131 }
132
133
134
135
136
137
138
139
140
141
142
143
```