```
 1: // $Id: semaphilo.cpp,v 1.22 2014-06-04 12:13:15-07 - - $
 2:
 3: //
 4: // Implementation of semaphores.
 5: // Solution to Dining Philosophers problem.
 6: //
 7:
 8: #include <condition_variable>
 9: #include <iomanip>
10: #include <iostream>
11: #include <mutex>
12: #include <random>
13: #include <string>
14: #include <thread>
15: using namespace std;
16:
17: //
18: // Timer.
19: //
20: class elapsed_time {
21:    private:
22:       using clock = chrono::high_resolution_clock;
23:       clock::time_point start {clock::now()};
24:    public:
25:       double elapsed_nanoseconds() {
26:          clock::time_point now = clock::now();
27:          return chrono::duration_cast<chrono::nanoseconds> (now - start)
28:                 .count() / 1e9;
29:       }
30: } timer;
31:
```

```
32:
33: //
34: // Printer for synchronized output accepts a variable number
35: // of arguments.
36: //
37:
38: class synch_printer {
39:    private:
40:       mutex out_mutex;
41:       ostream& out;
42:       void print_();
43:       template <typename Head, typename... Tail>
44:       void print_ (const Head& head, Tail... tail);
45:    public:
46:       synch_printer (ostream& out): out(out){}
47:       template <typename... Type>
48:       void print (Type... params);
49: };
50:
51: void synch_printer::print_() {
52: }
53:
54: template <typename Head, typename... Tail>
55: void synch_printer::print_ (const Head& head, Tail... tail) {
56:    out << head;
57:    print_ (tail...);
58: }
59:
60: template <typename... Type>
61: void synch_printer::print (Type... params) {
62:    unique_lock<mutex> lock {out_mutex};
63:    out << setw(9) << setprecision(6) << fixed
64:        << timer.elapsed_nanoseconds() << " ";
65:    print_ (params...);
66:    out << endl << flush;
67: }
68:
```

```
 69:
 70: //
 71: // class semaphore
 72: //    count > 0: number to be let through the gate
 73: //    count = 0: gate is locked, nobody waiting
 74: //    count < 0: gate locked with -count threads waiting
 75: // semaphore lock(1) is a mutex
 76: //
 77:
 78: class semaphore {
 79:    private:
 80:       mutex lock;
 81:       condition_variable cond;
 82:       int count{};
 83:    public:
 84:       semaphore (int count = 0);
 85:       void down(); // Dijkstra's P ("proberen", Dutch "try")
 86:       void up();   // Dijkstra's V ("verhogen", Dutch "raise")
 87: };
 88:
 89: semaphore::semaphore (int count): count(count) {
 90: }
 91:
 92: void semaphore::down() {
 93:    unique_lock<mutex> ulock {lock};
 94:    --count;
 95:    if (count < 0) cond.wait (ulock);
 96: }
 97:
 98: void semaphore::up() {
 99:    unique_lock<mutex> ulock {lock};
100:    if (count < 0) cond.notify_one();
101:    ++count;
102: }
103:
```

```
104:
105: class fork_manager {
106:    private:
107:       enum fork_state {THINKING = 0, HUNGRY, EATING};
108:       semaphore lock {1};
109:       const size_t size;
110:       vector<semaphore> waiting;
111:       vector<fork_state> state;
112:    private:
113:       size_t left (size_t id) { return (id + size - 1) % size; }
114:       size_t right (size_t id) { return (id + 1) % size; }
115:       void check (size_t id);
116:    public:
117:       fork_manager (size_t size);
118:       void take (size_t id);
119:       void put (size_t id);
120: };
121:
122: fork_manager::fork_manager (size_t size): size(size),
123:               waiting (size), state(size, THINKING) {
124: }
125:
126: void fork_manager::check (size_t id) {
127:    if (state[id] == HUNGRY and state[left(id)] != EATING
128:                            and state[right(id)] != EATING) {
129:       state[id] = EATING;
130:       waiting[id].up();
131:    }
132: }
133:
134: void fork_manager::take (size_t id) {
135:    lock.down();
136:    state[id] = HUNGRY;
137:    check (id);
138:    lock.up();
139:    waiting[id].down();
140: }
141:
142: void fork_manager::put (size_t id) {
143:    lock.down();
144:    state[id] = THINKING;
145:    check (left (id));
146:    check (right (id));
147:    lock.up();
148: }
149:
```

```
150:
151: //
152: // Code to simulate one Dining Philosopher.
153: //
154:
155: using normal_dist = normal_distribution<double>;
156: using rand_engine = default_random_engine;
157: using rand_gen = decltype (bind (normal_dist{}, rand_engine{}));
158:
159: rand_gen make_rand (double mean, double stdev) {
160:    auto seed = chrono::system_clock::now().time_since_epoch().count();
161:    rand_engine engine {seed};
162:    normal_dist distribution (mean, stdev);
163:    rand_gen rand = bind (distribution, engine);
164:    return rand;
165: }
166:
167: void perform (synch_printer& mcout, rand_gen& rand, size_t cycle,
168:               const string& ident, const string& activity) {
169:    long delay = rand();
170:    mcout.print (ident, activity, " ", cycle, " (", delay, ")");
171:    this_thread::sleep_for (chrono::milliseconds (delay));
172: }
173:
174: void philosopher (size_t id, const string& name,
175:                   size_t cycles, double mean, double stdev,
176:                   fork_manager& forks, synch_printer& mcout) {
177:    rand_gen rand = make_rand (mean, stdev);
178:    string ident = "(" + to_string(id) + ") " + name + " is ";
179:    mcout.print (ident, "STARTING");
180:    for (size_t cycle = 0; cycle < cycles; ++cycle) {
181:       perform (mcout, rand, cycle, ident, "thinking");
182:       mcout.print (ident, "hungry ", cycle);
183:       forks.take (id);
184:       perform (mcout, rand, cycle, ident, "eating");
185:       forks.put (id);
186:    }
187:    mcout.print (ident, "FINISHED");
188: }
189:
```

```
190:
191: //
192: // Host at the Symposium.
193: //
194: // In ancient Greece, the symposium (Greek ######### sympósion,
195: // from ######### sympínein, "to drink together") was a drinking
196: // party.  Literary works that describe or take place at a
197: // symposium include two Socratic dialogues, Plato's Symposium and
198: // Xenophon's Symposium, as well as a number of Greek poems such
199: // as the elegies of Theognis of Megara.
200: //
201:
202: int main() {
203:    synch_printer mcout {cout};
204:    mcout.print ("Symposium STARTING");
205:    vector<string> names = {
206:       "Pythagoras",
207:       "Socrates",
208:       "Plato",
209:       "Aristotle",
210:       "Zeno",
211:    };
212:    fork_manager forks {names.size()};
213:    vector<thread> philos;
214:    for (size_t id = 0; id < names.size(); ++id) {
215:       philos.push_back (thread (philosopher, id, ref (names[id]),
216:                                 5, 1000, 250,
217:                                 ref (forks), ref (mcout)));
218:    }
219:    for (auto& phil: philos) phil.join();
220:    mcout.print ("Symposium FINISHED");
221:    return 0;
222: }
223:
224: //TEST// semaphilo >semaphilo.out 2>&1
225: //TEST// mkpspdf semaphilo.ps semaphilo.cpp* semaphilo.out
226:
```

```
    1: @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ mkc: starting semaphilo.cpp
    2: semaphilo.cpp:
    3:      $Id: semaphilo.cpp,v 1.22 2014-06-04 12:13:15-07 - - $
    4: g++ -g -O0 -Wall -Wextra -std=gnu++11 semaphilo.cpp -o semaphilo -lglut
-lGLU -lGL -lX11 -lm -lrt
    5: rm -f semaphilo.o
    6: @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ mkc: finished semaphilo.cpp
```

```
  1:  0.000005 Symposium STARTING
  2:  0.001190 (0) Pythagoras is STARTING
  3:  0.001289 (2) Plato is STARTING
  4:  0.001354 (2) Plato is thinking 0 (1187)
  5:  0.001388 (1) Socrates is STARTING
  6:  0.001409 (1) Socrates is thinking 0 (703)
  7:  0.001433 (0) Pythagoras is thinking 0 (637)
  8:  0.001468 (3) Aristotle is STARTING
  9:  0.001528 (3) Aristotle is thinking 0 (933)
 10:  0.001554 (4) Zeno is STARTING
 11:  0.001582 (4) Zeno is thinking 0 (1223)
 12:  0.638537 (0) Pythagoras is hungry 0
 13:  0.638578 (0) Pythagoras is eating 0 (979)
 14:  0.704488 (1) Socrates is hungry 0
 15:  0.934618 (3) Aristotle is hungry 0
 16:  0.934662 (3) Aristotle is eating 0 (588)
 17:  1.188519 (2) Plato is hungry 0
 18:  1.224734 (4) Zeno is hungry 0
 19:  1.522758 (3) Aristotle is thinking 1 (555)
 20:  1.522807 (2) Plato is eating 0 (1046)
 21:  1.617665 (0) Pythagoras is thinking 1 (799)
 22:  1.617701 (4) Zeno is eating 0 (1068)
 23:  2.077874 (3) Aristotle is hungry 1
 24:  2.416759 (0) Pythagoras is hungry 1
 25:  2.568970 (2) Plato is thinking 1 (872)
 26:  2.569012 (1) Socrates is eating 0 (1128)
 27:  2.685863 (4) Zeno is thinking 1 (592)
 28:  2.685891 (3) Aristotle is eating 1 (905)
 29:  3.277906 (4) Zeno is hungry 1
 30:  3.441069 (2) Plato is hungry 1
 31:  3.591004 (4) Zeno is eating 1 (768)
 32:  3.591045 (3) Aristotle is thinking 2 (950)
 33:  3.697158 (1) Socrates is thinking 1 (673)
 34:  3.697200 (2) Plato is eating 1 (981)
 35:  4.359104 (4) Zeno is thinking 2 (1300)
 36:  4.359135 (0) Pythagoras is eating 1 (721)
 37:  4.370256 (1) Socrates is hungry 1
 38:  4.541139 (3) Aristotle is hungry 2
 39:  4.678281 (2) Plato is thinking 2 (610)
 40:  4.678329 (3) Aristotle is eating 2 (1008)
 41:  5.080245 (0) Pythagoras is thinking 2 (1008)
 42:  5.080287 (1) Socrates is eating 1 (976)
 43:  5.288387 (2) Plato is hungry 2
 44:  5.659250 (4) Zeno is hungry 2
 45:  5.686498 (3) Aristotle is thinking 3 (1196)
 46:  5.686549 (4) Zeno is eating 2 (1039)
 47:  6.056378 (1) Socrates is thinking 2 (1481)
 48:  6.056399 (2) Plato is eating 2 (1136)
 49:  6.088413 (0) Pythagoras is hungry 2
 50:  6.725713 (4) Zeno is thinking 3 (949)
 51:  6.725748 (0) Pythagoras is eating 2 (1049)
 52:  6.882674 (3) Aristotle is hungry 3
 53:  7.192532 (2) Plato is thinking 3 (1046)
 54:  7.192581 (3) Aristotle is eating 3 (893)
 55:  7.537529 (1) Socrates is hungry 2
 56:  7.674818 (4) Zeno is hungry 3
 57:  7.774931 (0) Pythagoras is thinking 3 (1164)
 58:  7.774964 (1) Socrates is eating 2 (1386)
```

```
 59:  8.085671 (3) Aristotle is thinking 4 (1193)
 60:  8.085725 (4) Zeno is eating 3 (1308)
 61:  8.238712 (2) Plato is hungry 3
 62:  8.939092 (0) Pythagoras is hungry 3
 63:  9.161135 (1) Socrates is thinking 3 (822)
 64:  9.161196 (2) Plato is eating 3 (615)
 65:  9.278847 (3) Aristotle is hungry 4
 66:  9.393907 (4) Zeno is thinking 4 (1270)
 67:  9.393966 (0) Pythagoras is eating 3 (816)
 68:  9.776293 (2) Plato is thinking 4 (1045)
 69:  9.776341 (3) Aristotle is eating 4 (325)
 70:  9.983247 (1) Socrates is hungry 3
 71: 10.101434 (3) Aristotle is FINISHED
 72: 10.210066 (0) Pythagoras is thinking 4 (1238)
 73: 10.210123 (1) Socrates is eating 3 (1068)
 74: 10.664083 (4) Zeno is hungry 4
 75: 10.664102 (4) Zeno is eating 4 (1202)
 76: 10.821456 (2) Plato is hungry 4
 77: 11.278285 (1) Socrates is thinking 4 (776)
 78: 11.278308 (2) Plato is eating 4 (1203)
 79: 11.448249 (0) Pythagoras is hungry 4
 80: 11.866242 (4) Zeno is FINISHED
 81: 11.866291 (0) Pythagoras is eating 4 (944)
 82: 12.054365 (1) Socrates is hungry 4
 83: 12.481445 (2) Plato is FINISHED
 84: 12.810390 (0) Pythagoras is FINISHED
 85: 12.810439 (1) Socrates is eating 4 (1031)
 86: 13.841611 (1) Socrates is FINISHED
 87: 13.841778 Symposium FINISHED
```