

```
1  Sarah Whelan
2  slw96
3  October 1, 2015
4  EECS 338
5  Assignment 2
6
7  1. Priority-based Searchers/Inserters/Deleters Problem without starvation
8
9  nonbinary semaphore mutex = 1;
10 nonbinary semaphore sWait = 0;
11 nonbinary semaphore iWait = 0;
12 nonbinary semaphore dWait = 0;
13
14 int sPassingCount = 0;
15 int sStarvationCount = 0;
16
17 int sWaitCount = 0;
18 int iWaitCount = 0;
19 int dWaitCount = 0;
20
21 int iStarvationCount = 0;
22 int dStarvationCount = 0;
23
24 boolean sBlocked = false;
25 boolean iBlocked = false;
26
27 boolean insertingDuringStarveFlag = false;
28 boolean inserting = false;
29 boolean deleting = false;
30
31 enum {SEARCHER, INSERTER, DELETER} PROCESS_TYPE;
32
33 process searcher(L, item){
34     wait(mutex);
35
36     // Handle entering starvation mode if a new searcher tries to search
37     if(sStarvationCount == 10){
38         if(sPassingCount == 0){
39             if((iWaitCount > 0) || (dWaitCount > 0)){
40                 sBlocked = true;
41                 if(iWaitCount > 0){
42                     iStarvationCount = iWaitCount;
43                     if(inserting){
44                         // If an inserter is inserting when starvation mode is entered
45                         // let it finish inserting and don't call signal on iWait
46                         insertingDuringStarveFlag = true;
47                     } else {
48                         sStarvationCount = 0;
49                         signal(iWait);
50                     }
51                 } else {
52                     iBlocked = true;
53                     sStarvationCount = 0;
54                     dStartvationCount = dWaitCount;
```

```

55         signal(dWait);
56     }
57     sWaitCount++;
58     // Don't call signal on mutex as an inserter or a deleter was called
    that will signal mutex
59     wait(sWait);
60     sWaitCount--;
61 } else {
62     // Starvation mode not entered as there were no waiting inserters or
    deleters
63     sStarvationCount = 0;
64     if(sWaitCount > 0){
65         sWaitCount++;
66         signal(mutex);
67         wait(sWait);
68         sWaitCount--;
69     }
70 }
71 } else {
72     sWaitCount++;
73     signal(mutex);
74     wait(sWait);
75     sWaitCount--;
76 }
77 // In starvation mode or a deleter is deleting block more searchers
78 } else if(sBlocked || deleting){
79     sWaitCount++;
80     signal(mutex);
81     wait(sWait);
82     sWaitCount--;
83 }
84 sPassingCount++;
85 sStarvationCount++;
86 signal(mutex);
87
88 LOG-START-TIME(PROCESS_TYPE.SEARCHER, item);
89 SEARCH-AND-LOG-RESULTS(L, item);
90
91 wait(mutex);
92 sPassingCount--;
93
94 // Handle entering starvation mode
95 if(sStarvationCount == 10 && sPassingCount == 0){
96     if((iWaitCount > 0) || (dWaitCount > 0)){
97         sBlocked = true;
98         if(iWaitCount > 0){
99             iStarvationCount = iWaitCount;
100             sStarvationCount == 0;
101             if(inserting){
102                 insertingDuringStarveFlag = true;
103             } else {
104                 signal(iWait);
105             }
106         } else {

```

```

107         iBlocked = true;
108         sStarvationCount == 0;
109         dStarvationCount = dWaitCount;
110         signal(dWait);
111     }
112     } else {
113         // Starvation mode not entered as there are no waiting inserters or deleters
114         sStarvationCount == 0;
115         if(sWaitCount > 0 && !sBlocked){
116             signal(sWait);
117         } else if(!inserting && iWaitCount > 0 && !deleting){
118             signal(iWait);
119         } else if(!deleting && dWaitCount > 0 && !inserting && sPassingCount < 0){
120             signal(dWait);
121         } else {
122             signal(mutex);
123         }
124     }
125 } else {
126     if(sWaitCount > 0 && !sBlocked){
127         signal(sWait);
128     } else if(!inserting && iWaitCount > 0 && !deleting){
129         signal(iWait);
130     } else if(!deleting && dWaitCount > 0 && !inserting && sPassingCount < 0){
131         signal(dWait);
132     } else {
133         signal(mutex);
134     }
135 }
136 }
137
138 process inserter(L, item){
139     wait(mutex);
140     if(inserting || deleting || iBlocked){
141         iWaitCount++;
142         signal(mutex);
143         wait(iWait);
144         iWaitCount--;
145     }
146     inserting = true;
147     signal(mutex);
148
149     LOG-START-TIME(PROCESS_TYPE.INSERTER, item);
150     INSERT-AND-LOG-RESULTS(L, item);
151
152     wait(mutex);
153     inserting = false;
154     if(sBlocked && !iBlocked){
155         if(insertingDuringStarveFlag){
156             // If this inserter was the one that was inserting when starvation mode
157             // was set don't decrement starvation count and let the waiting inserters go
158             insertingDuringStarveFlag = false;
159             if(iWaitCount > 0){
160                 signal(iWait);

```

```

161         }
162     } else if(iStarvationCount > 0){
163         iStarvationCount--;
164         if(iStarvationCount == 0){
165             iBlocked = true;
166             if(dWaitCount > 0){
167                 dStarvationCount = dWaitCount;
168                 signal(dWait);
169             }
170         } else {
171             signal(iWait);
172         }
173     }
174 } else if(sWaitCount > 0 && !deleting){
175     signal(sWait);
176 } else if(iWaitCount > 0 && !inserting && !deleting && !iBlocked){
177     signal(iWait);
178 } else if(dWaitCount > 0 && !inserting && !deleting && sPassingCount <= 0){
179     signal(dWait);
180 } else {
181     signal(mutex);
182 }
183 }
184
185 process deleter(L, item){
186     wait(mutex);
187     if(inserting || deleting || sPassingCount > 0){
188         dWaitCount++;
189         signal(mutex);
190         wait(dWait);
191         dWaitCount--;
192     }
193     deleting = true;
194     signal(mutex);
195
196     LOG-START-TIME(PROCESS_TYPE.DELETER, item);
197     DELETE-AND-LOG-RESULTS(L, item);
198
199     wait(mutex);
200     deleting = false;
201     if(sBlocked && iBlocked && dStarvationCount > 0){
202         dStarvationCount--;
203         if(dStarvationCount == 0){
204             sBlocked = false;
205             iBlocked = false;
206             if(sWaitCount <= 0 && iWaitCount <= 0){
207                 if(dWaitCount > 0){
208                     signal(dWait);
209                 } else {
210                     signal(mutex);
211                 }
212             } else if(sWaitCount > 0){
213                 signal(sWait);
214             } else if(iWaitCount > 0){

```

```

215         signal(iWait);
216     }
217 } else {
218     signal(dWait);
219 }
220 } else {
221     if(sWaitCount > 0){
222         signal(sWait);
223     } else if(iWaitCount > 0 && !inserting){
224         signal(iWait);
225     } else if(dWaitCount > 0){
226         signal(dWait);
227     } else {
228         signal(mutex);
229     }
230 }
231 }
232
233 public void LOG-START-TIME(PROCESS_TYPE, item){
234     //Logs that the process' PROCESS_TYPE, pid, and item along with the start time
235 }
236
237 public void SEARCH-AND-LOG-RESULTS(L, item){
238     // Searches through the linked list L for item
239     // if found log success
240     // if !found log failure
241 }
242
243 public void INSERT-AND-LOG-RESULTS(L, item){
244     // Attempts to insert the item into the linked list L
245     // if item doesn't already exist insert and log success
246     // if item already exists don't insert and log failure
247 }
248
249 public void DELETE-AND-LOG-RESULTS(L, item){
250     // Attempts to delete item from linked list L
251     // if item already exists delete and log success
252     // if item doesn't already exist don't delete and log failure
253 }
254
255 2. Four-of-a-Kind Problem
256
257 // Indexed from zero indicating it is p0's turn first
258 boolean[] turn = {true, false, false, false};
259 boolean gameWon = false;
260 nonbinary semaphore mutex = 1;
261 process player(i, L){
262     // access to gameWon can violate mutual exclusion as the
263     // race condition to restart the loop does not lead to livelocks
264     while(!gameWon){
265         wait(mutex);
266         if(turn[i]){
267             if(HAS-FOUR-OF-KIND(L)){
268                 gameWon = true;

```

```
269         } else {
270             DISCARD-CARD(i, L);
271             PICK-UP-CARD((i+1)mod(4), L);
272             if(HAS-FOUR-OF-KIND(L)){
273                 gameWon = true;
274             } else {
275                 turn[i] = false;
276                 turn[(i+1)mod(4)] = true;
277             }
278         }
279     }
280     signal(mutex);
281 }
282 return;
283 }
284
285 public boolean HAS-FOUR-OF-KIND(L){
286     // Returns if list of cards L is four of a kind
287 }
288
289 public void DISCARD-CARD(i, L){
290     // Chooses a card from L to discard and places it on the ith pile
291 }
292
293 public void PICK-UP-A-CARD(i, L){
294     // Picks up a card from the ith pile and places it into L
295 }
```