Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет
имени Н. Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н. Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления (ИУ)»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии (ИУ7)»

ОТЧЕТ

Лабораторная работа №2

по курсу «Математические основы верификации ПО»

на тему: «Моделирование гонки процессов»

Студент ИУ7-42М (Группа) _____ (Подпись, дата) К.Э. Ковалец (И. О. Фамилия)

Преподаватель _____ (Подпись, дата) О.В. Кузнецова (И. О. Фамилия)

2025 г.

1 Выполнение лабораторной работы

1.1 Задание

Необходимо описать взаимодействие двух процессов, работающих с одними данными. Затем место возникновения гонки необходимо дополнить мьютексами.

Отчёт должен содержать:

— описание модели взаимодействия процессов;

— демонстрация логов SPIN, в которых видна гонка;

— описание модели с мьютексом;

— результат корректного взаимодействия процессов — логи SPIN;

— выводы по работе.

1.2 Описание модели взаимодействия процессов

Модель взаимодействия процессов описывает два параллельных процесса, которые работают с общей переменной `shared_data`. Один процесс (`increase_counter`) увеличивает значение этой переменной на `INCREMENT`, другой (`reduce_counter`) уменьшает.

В цикле, который выполняется `REPETIONS_NUMBER` раз, оба процесса считывают текущее значение `shared_data` в переменную `temp`, изменяют ее на `INCREMENT` и записывают обратно в `shared_data`.

В данной модели возникает гонка, так как оба процесса одновременно обращаются к общей переменной `shared_data` без синхронизации, что может привести к некорректным результатам.

Код программы с гонкой процессов на языке Promela приведен в листинге 1.1.

Листинг 1.1 — Пример программы на языке Promela с гонкой процессов

```
#define REPETIONS_NUMBER 2
#define INCREMENT 1

int shared_data = 0;

proctype increase_counter() {
  int temp;
  int i = 0;
  do
  :: i < REPETIONS_NUMBER ->
    temp = shared_data;
    temp = temp + INCREMENT;
    shared_data = temp;
    i++;
  :: else -> break;
  od
}

proctype reduce_counter() {
  int temp;
  int i = 0;
  do
  :: i < REPETIONS_NUMBER ->
    temp = shared_data;
    temp = temp - INCREMENT;
    shared_data = temp;
    i++;
  :: else -> break;
  od
}

init {
  run increase_counter();
  run reduce_counter();
  (_nr_pr == 1) -> printf("shared_data = %d\n", shared_data);
}
```

1.3 Демонстрация логов SPIN, в которых видна гонка

Демонстрация логов SPIN, в которых видна гонка, приведена в листинге 1.2. Можно заметить, что оба процесса одновременно считывают одно и то же начальное значение переменной и затем независимо изменяют его. В итоге каждый процесс записывает своё значение обратно в переменную `shared_data`, что приводит к некорректному результату.

Листинг 1.2 — Демонстрация логов SPIN, в которых видна гонка

```
Starting increase_counter with pid 1
  1:        proc  0 (:init::1) race.pml:33 (state 1)        [(run
  ↪ increase_counter())]
                shared_data = 0
  2:        proc  1 (increase_counter:1) race.pml:10 (state 1)        [((i<2))]
                shared_data = 0
Starting reduce_counter with pid 2
  3:        proc  0 (:init::1) race.pml:34 (state 2)        [(run
  ↪ reduce_counter())]
                shared_data = 0
  4:        proc  1 (increase_counter:1) race.pml:11 (state 2)        [temp =
  ↪ shared_data]
                shared_data = 0
  5:        proc  1 (increase_counter:1) race.pml:12 (state 3)        [temp =
  ↪ (temp+1)]
                shared_data = 0
  6:        proc  2 (reduce_counter:1) race.pml:23 (state 1)        [((i<2))]
                shared_data = 0
  7:        proc  1 (increase_counter:1) race.pml:13 (state 4)        [shared_data
  ↪ = temp]
                shared_data = 1
  8:        proc  1 (increase_counter:1) race.pml:14 (state 5)        [i = (i+1)]
                shared_data = 1
  9:        proc  1 (increase_counter:1) race.pml:17 (state 9)        [.(goto)]
                shared_data = 1
 10:        proc  1 (increase_counter:1) race.pml:10 (state 1)        [((i<2))]
                shared_data = 1
 11:        proc  2 (reduce_counter:1) race.pml:24 (state 2)        [temp =
 ↪ shared_data]
                shared_data = 1
 12:        proc  2 (reduce_counter:1) race.pml:25 (state 3)        [temp =
 ↪ (temp-1)]
                shared_data = 1
```

Продолжение листинга 1.2

```
13:        proc  1 (increase_counter:1) race.pml:11 (state 2)        [temp =
↪ shared_data]
                shared_data = 1
14:        proc  2 (reduce_counter:1) race.pml:26 (state 4)        [shared_data =
↪ temp]
                shared_data = 0
15:        proc  2 (reduce_counter:1) race.pml:27 (state 5)        [i = (i+1)]
                shared_data = 0
16:        proc  2 (reduce_counter:1) race.pml:30 (state 9)        [.(goto)]
                shared_data = 0
17:        proc  1 (increase_counter:1) race.pml:12 (state 3)        [temp =
↪ (temp+1)]
                shared_data = 0
18:        proc  1 (increase_counter:1) race.pml:13 (state 4)        [shared_data
↪ = temp]
                shared_data = 2
19:        proc  1 (increase_counter:1) race.pml:14 (state 5)        [i = (i+1)]
                shared_data = 2
20:        proc  1 (increase_counter:1) race.pml:17 (state 9)        [.(goto)]
                shared_data = 2
21:        proc  1 (increase_counter:1) race.pml:15 (state 6)        [else]
                shared_data = 2
22:        proc  2 (reduce_counter:1) race.pml:23 (state 1)        [((i<2))]
                shared_data = 2
23:        proc  2 (reduce_counter:1) race.pml:24 (state 2)        [temp =
↪ shared_data]
                shared_data = 2
24:        proc  2 (reduce_counter:1) race.pml:25 (state 3)        [temp =
↪ (temp-1)]
                shared_data = 2
25:        proc  1 (increase_counter:1) race.pml:15 (state 7)        [goto :b0]
                shared_data = 2
26:        proc  2 (reduce_counter:1) race.pml:26 (state 4)        [shared_data =
↪ temp]
                shared_data = 1
27:        proc  2 (reduce_counter:1) race.pml:27 (state 5)        [i = (i+1)]
                shared_data = 1
28:        proc  2 (reduce_counter:1) race.pml:30 (state 9)        [.(goto)]
                shared_data = 1
29:        proc  2 (reduce_counter:1) race.pml:28 (state 6)        [else]
                shared_data = 1
30:        proc  2 (reduce_counter:1) race.pml:28 (state 7)        [goto :b1]
                shared_data = 1
31:        proc  0 (:init::1) race.pml:35 (state 3)        [((_nr_pr==1))]
                shared_data = 1
```

Продолжение листинга 1.2

```
      shared_data = 1
 32:        proc  0 (:init::1) race.pml:35 (state 4)        [printf('shared_data =
 ↪ %d\\n',shared_data)]
                shared_data = 1
3 processes created
```

1.4 Описание модели с мьютексом

Модель с мьютексом добавляет синхронизацию между процессами, чтобы избежать гонки процессов. В данной модели используется мьютекс для обеспечения взаимного исключения при доступе к общей переменной `shared_data`.

В цикле, который выполняется `REPETIONS_NUMBER` раз, оба процесса по очереди блокирует доступ к критической секции, считывают текущее значение `shared_data` в переменную `temp`, изменяют `temp` на `INCREMENT`, записывают обратно в `shared_data` и разблокируют критическую секцию.

Использование мьютекса гарантирует, что только один процесс в каждый момент времени может изменять значение `shared_data`, что предотвращает возникновение гонки процессов и обеспечивает корректное их взаимодействие.

Код программы с мьютексом на языке Promela приведен в листинге 1.3.

Листинг 1.3 — Пример программы на языке Promela с мьютексом

```
#define REPETIONS_NUMBER 2
#define INCREMENT 1

int shared_data = 0;
show byte mutex = 0;

#define lock(mutex)         \
  do                        \
  :: atomic {               \
    if                      \
    :: mutex == 0 -> {      \
      mutex = 1;            \
      break;                \
    }                       \
    :: else -> skip;        \
    fi                      \
  }                         \
  od
```

Продолжение листинга 1.3

```
19
20  #define unlock(mutex)       \
21    mutex = 0;
22
23  proctype increase_counter() {
24    int temp;
25    int i = 0;
26    do
27    :: i < REPETIONS_NUMBER ->
28      lock(mutex);
29      temp = shared_data;
30      temp = temp + INCREMENT;
31      shared_data = temp;
32      unlock(mutex);
33      i++;
34    :: else -> break;
35    od
36  }
37
38  proctype reduce_counter() {
39    int temp;
40    int i = 0;
41    do
42    :: i < REPETIONS_NUMBER ->
43      lock(mutex);
44      temp = shared_data;
45      temp = temp - INCREMENT;
46      shared_data = temp;
47      unlock(mutex);
48      i++;
49    :: else -> break;
50    od
51  }
52
53  init {
54    run increase_counter();
55    run reduce_counter();
56    (_nr_pr == 1) -> printf("shared_data = %d\n", shared_data);
57  }
```

1.5 Результат корректного взаимодействия процессов

Результат корректного взаимодействия процессов приведен в листинге 1.4. Можно заметить, что оба процесса последовательно получают доступ к критической секции (поочередно считывают, изменяют и записывают значение переменной `shared_data`), что гарантирует корректное взаимодействие процессов.

Листинг 1.4 — Результат корректного взаимодействия процессов — логи SPIN

```
Starting increase_counter with pid 1
  1:        proc  0 (:init::1) mutex.pml:54 (state 1)        [(run
  ↪ increase_counter())]
                shared_data = 0
                mutex = 0
  2:        proc  1 (increase_counter:1) mutex.pml:27 (state 1)        [((i<2))]
                shared_data = 0
                mutex = 0
Starting reduce_counter with pid 2
  3:        proc  0 (:init::1) mutex.pml:55 (state 2)        [(run
  ↪ reduce_counter())]
                shared_data = 0
                mutex = 0
  4:        proc  1 (increase_counter:1) mutex.pml:29 (state 12)        [.(goto)]
                shared_data = 0
                mutex = 0
  5:        proc  1 (increase_counter:1) mutex.pml:28 (state
  ↪ 2)        [((mutex==0))]
                shared_data = 0
                mutex = 0
  6:        proc  1 (increase_counter:1) mutex.pml:28 (state 3)        [mutex = 1]
                shared_data = 0
                mutex = 1
  7:        proc  2 (reduce_counter:1) mutex.pml:42 (state 1)        [((i<2))]
                shared_data = 0
                mutex = 1
  8:        proc  2 (reduce_counter:1) mutex.pml:44 (state 12)        [.(goto)]
                shared_data = 0
                mutex = 1
  9:        proc  2 (reduce_counter:1) mutex.pml:43 (state 6)        [else]
                shared_data = 0
                mutex = 1
 10:        proc  2 (reduce_counter:1) mutex.pml:43 (state 7)        [(1)]
```

Продолжение листинга 1.4

```
                shared_data = 0
                mutex = 1
  11:       proc  2 (reduce_counter:1) mutex.pml:43 (state 6)        [else]
                shared_data = 0
                mutex = 1
  12:       proc  2 (reduce_counter:1) mutex.pml:43 (state 7)        [(1)]
                shared_data = 0
                mutex = 1
  13:       proc  2 (reduce_counter:1) mutex.pml:43 (state 6)        [else]
                shared_data = 0
                mutex = 1
  14:       proc  2 (reduce_counter:1) mutex.pml:43 (state 7)        [(1)]
                shared_data = 0
                mutex = 1
  15:       proc  2 (reduce_counter:1) mutex.pml:43 (state 6)        [else]
                shared_data = 0
                mutex = 1
  16:       proc  2 (reduce_counter:1) mutex.pml:43 (state 7)        [(1)]
                shared_data = 0
                mutex = 1
  17:       proc  2 (reduce_counter:1) mutex.pml:43 (state 6)        [else]
                shared_data = 0
                mutex = 1
  18:       proc  2 (reduce_counter:1) mutex.pml:43 (state 7)        [(1)]
                shared_data = 0
                mutex = 1
  19:       proc  2 (reduce_counter:1) mutex.pml:43 (state 6)        [else]
                shared_data = 0
                mutex = 1
  20:       proc  2 (reduce_counter:1) mutex.pml:43 (state 7)        [(1)]
                shared_data = 0
                mutex = 1
  21:       proc  2 (reduce_counter:1) mutex.pml:43 (state 6)        [else]
                shared_data = 0
                mutex = 1
  22:       proc  2 (reduce_counter:1) mutex.pml:43 (state 7)        [(1)]
                shared_data = 0
                mutex = 1
  23:       proc  1 (increase_counter:1) mutex.pml:28 (state 13)        [break]
                shared_data = 0
                mutex = 1
  24:       proc  2 (reduce_counter:1) mutex.pml:43 (state 6)        [else]
                shared_data = 0
                mutex = 1
  25:       proc  2 (reduce_counter:1) mutex.pml:43 (state 7)        [(1)]
```

Продолжение листинга 1.4

```
                shared_data = 0
                mutex = 1
  26:       proc  2 (reduce_counter:1) mutex.pml:43 (state 6)        [else]
                shared_data = 0
                mutex = 1
  27:       proc  2 (reduce_counter:1) mutex.pml:43 (state 7)        [(1)]
                shared_data = 0
                mutex = 1
  28:       proc  1 (increase_counter:1) mutex.pml:29 (state 14)        [temp =
  ↪ shared_data]
                shared_data = 0
                mutex = 1
  29:       proc  2 (reduce_counter:1) mutex.pml:43 (state 6)        [else]
                shared_data = 0
                mutex = 1
  30:       proc  2 (reduce_counter:1) mutex.pml:43 (state 7)        [(1)]
                shared_data = 0
                mutex = 1
  31:       proc  1 (increase_counter:1) mutex.pml:30 (state 15)        [temp =
  ↪ (temp+1)]
                shared_data = 0
                mutex = 1
  32:       proc  2 (reduce_counter:1) mutex.pml:43 (state 6)        [else]
                shared_data = 0
                mutex = 1
  33:       proc  2 (reduce_counter:1) mutex.pml:43 (state 7)        [(1)]
                shared_data = 0
                mutex = 1
  34:       proc  1 (increase_counter:1) mutex.pml:31 (state
  ↪ 16)        [shared_data = temp]
                shared_data = 1
                mutex = 1
  35:       proc  2 (reduce_counter:1) mutex.pml:43 (state 6)        [else]
                shared_data = 1
                mutex = 1
  36:       proc  2 (reduce_counter:1) mutex.pml:43 (state 7)        [(1)]
                shared_data = 1
                mutex = 1
  37:       proc  2 (reduce_counter:1) mutex.pml:43 (state 6)        [else]
                shared_data = 1
                mutex = 1
  38:       proc  2 (reduce_counter:1) mutex.pml:43 (state 7)        [(1)]
                shared_data = 1
                mutex = 1
```

Продолжение листинга 1.4

```
117   39:        proc  1 (increase_counter:1) mutex.pml:32 (state 17)        [mutex =
      ↪ 0]
118                   shared_data = 1
119                   mutex = 0
120   40:        proc  2 (reduce_counter:1) mutex.pml:43 (state
      ↪ 2)        [((mutex==0))]
121                   shared_data = 1
122                   mutex = 0
123   41:        proc  2 (reduce_counter:1) mutex.pml:43 (state 3)        [mutex = 1]
124                   shared_data = 1
125                   mutex = 1
126   42:        proc  2 (reduce_counter:1) mutex.pml:43 (state 13)        [break]
127                   shared_data = 1
128                   mutex = 1
129   43:        proc  2 (reduce_counter:1) mutex.pml:44 (state 14)        [temp =
      ↪ shared_data]
130                   shared_data = 1
131                   mutex = 1
132   44:        proc  2 (reduce_counter:1) mutex.pml:45 (state 15)        [temp =
      ↪ (temp-1)]
133                   shared_data = 1
134                   mutex = 1
135   45:        proc  2 (reduce_counter:1) mutex.pml:46 (state 16)        [shared_data
      ↪ = temp]
136                   shared_data = 0
137                   mutex = 1
138   46:        proc  2 (reduce_counter:1) mutex.pml:47 (state 17)        [mutex = 0]
139                   shared_data = 0
140                   mutex = 0
141   47:        proc  2 (reduce_counter:1) mutex.pml:48 (state 18)        [i = (i+1)]
142                   shared_data = 0
143                   mutex = 0
144   48:        proc  2 (reduce_counter:1) mutex.pml:51 (state 22)        [.(goto)]
145                   shared_data = 0
146                   mutex = 0
147   49:        proc  1 (increase_counter:1) mutex.pml:33 (state 18)        [i =
      ↪ (i+1)]
148                   shared_data = 0
149                   mutex = 0
150   50:        proc  2 (reduce_counter:1) mutex.pml:42 (state 1)        [((i<2))]
151                   shared_data = 0
152                   mutex = 0
153   51:        proc  2 (reduce_counter:1) mutex.pml:44 (state 12)        [.(goto)]
154                   shared_data = 0
155                   mutex = 0
```

Продолжение листинга 1.4

```
156   52:        proc  1 (increase_counter:1) mutex.pml:36 (state 22)        [.(goto)]
157                   shared_data = 0
158                   mutex = 0
159   53:        proc  1 (increase_counter:1) mutex.pml:27 (state 1)        [((i<2))]
160                   shared_data = 0
161                   mutex = 0
162   54:        proc  2 (reduce_counter:1) mutex.pml:43 (state
      ↪ 2)        [((mutex==0))]
163                   shared_data = 0
164                   mutex = 0
165   55:        proc  2 (reduce_counter:1) mutex.pml:43 (state 3)        [mutex = 1]
166                   shared_data = 0
167                   mutex = 1
168   56:        proc  1 (increase_counter:1) mutex.pml:29 (state 12)        [.(goto)]
169                   shared_data = 0
170                   mutex = 1
171   57:        proc  1 (increase_counter:1) mutex.pml:28 (state 6)        [else]
172                   shared_data = 0
173                   mutex = 1
174   58:        proc  1 (increase_counter:1) mutex.pml:28 (state 7)        [(1)]
175                   shared_data = 0
176                   mutex = 1
177   59:        proc  2 (reduce_counter:1) mutex.pml:43 (state 13)        [break]
178                   shared_data = 0
179                   mutex = 1
180   60:        proc  1 (increase_counter:1) mutex.pml:28 (state 6)        [else]
181                   shared_data = 0
182                   mutex = 1
183   61:        proc  1 (increase_counter:1) mutex.pml:28 (state 7)        [(1)]
184                   shared_data = 0
185                   mutex = 1
186   62:        proc  1 (increase_counter:1) mutex.pml:28 (state 6)        [else]
187                   shared_data = 0
188                   mutex = 1
189   63:        proc  1 (increase_counter:1) mutex.pml:28 (state 7)        [(1)]
190                   shared_data = 0
191                   mutex = 1
192   64:        proc  2 (reduce_counter:1) mutex.pml:44 (state 14)        [temp =
      ↪ shared_data]
193                   shared_data = 0
194                   mutex = 1
195   65:        proc  1 (increase_counter:1) mutex.pml:28 (state 6)        [else]
196                   shared_data = 0
197                   mutex = 1
198   66:        proc  1 (increase_counter:1) mutex.pml:28 (state 7)        [(1)]
```

Продолжение листинга 1.4

```
199                 shared_data = 0
200                 mutex = 1
201     67:     proc  2 (reduce_counter:1) mutex.pml:45 (state 15)      [temp =
        ↪  (temp-1)]
202                 shared_data = 0
203                 mutex = 1
204     68:     proc  1 (increase_counter:1) mutex.pml:28 (state 6)      [else]
205                 shared_data = 0
206                 mutex = 1
207     69:     proc  1 (increase_counter:1) mutex.pml:28 (state 7)      [(1)]
208                 shared_data = 0
209                 mutex = 1
210     70:     proc  1 (increase_counter:1) mutex.pml:28 (state 6)      [else]
211                 shared_data = 0
212                 mutex = 1
213     71:     proc  1 (increase_counter:1) mutex.pml:28 (state 7)      [(1)]
214                 shared_data = 0
215                 mutex = 1
216     72:     proc  1 (increase_counter:1) mutex.pml:28 (state 6)      [else]
217                 shared_data = 0
218                 mutex = 1
219     73:     proc  1 (increase_counter:1) mutex.pml:28 (state 7)      [(1)]
220                 shared_data = 0
221                 mutex = 1
222     74:     proc  1 (increase_counter:1) mutex.pml:28 (state 6)      [else]
223                 shared_data = 0
224                 mutex = 1
225     75:     proc  1 (increase_counter:1) mutex.pml:28 (state 7)      [(1)]
226                 shared_data = 0
227                 mutex = 1
228     76:     proc  1 (increase_counter:1) mutex.pml:28 (state 6)      [else]
229                 shared_data = 0
230                 mutex = 1
231     77:     proc  1 (increase_counter:1) mutex.pml:28 (state 7)      [(1)]
232                 shared_data = 0
233                 mutex = 1
234     78:     proc  2 (reduce_counter:1) mutex.pml:46 (state 16)      [shared_data
        ↪  = temp]
235                 shared_data = -1
236                 mutex = 1
237     79:     proc  2 (reduce_counter:1) mutex.pml:47 (state 17)      [mutex = 0]
238                 shared_data = -1
239                 mutex = 0
240     80:     proc  1 (increase_counter:1) mutex.pml:28 (state
        ↪  2)       [((mutex==0))]
```

Продолжение листинга 1.4

```
241                 shared_data = -1
242                 mutex = 0
243     81:         proc  1 (increase_counter:1) mutex.pml:28 (state 3)        [mutex = 1]
244                 shared_data = -1
245                 mutex = 1
246     82:         proc  1 (increase_counter:1) mutex.pml:28 (state 13)        [break]
247                 shared_data = -1
248                 mutex = 1
249     83:         proc  2 (reduce_counter:1) mutex.pml:48 (state 18)        [i = (i+1)]
250                 shared_data = -1
251                 mutex = 1
252     84:         proc  1 (increase_counter:1) mutex.pml:29 (state 14)        [temp =
        ↪  shared_data]
253                 shared_data = -1
254                 mutex = 1
255     85:         proc  2 (reduce_counter:1) mutex.pml:51 (state 22)        [.(goto)]
256                 shared_data = -1
257                 mutex = 1
258     86:         proc  2 (reduce_counter:1) mutex.pml:49 (state 19)        [else]
259                 shared_data = -1
260                 mutex = 1
261     87:         proc  2 (reduce_counter:1) mutex.pml:49 (state 20)        [goto :b2]
262                 shared_data = -1
263                 mutex = 1
264     88:         proc  1 (increase_counter:1) mutex.pml:30 (state 15)        [temp =
        ↪  (temp+1)]
265                 shared_data = -1
266                 mutex = 1
267     89:         proc  1 (increase_counter:1) mutex.pml:31 (state
        ↪  16)        [shared_data = temp]
268                 shared_data = 0
269                 mutex = 1
270     90:         proc  1 (increase_counter:1) mutex.pml:32 (state 17)        [mutex =
        ↪  0]
271                 shared_data = 0
272                 mutex = 0
273     91:         proc  1 (increase_counter:1) mutex.pml:33 (state 18)        [i =
        ↪  (i+1)]
274                 shared_data = 0
275                 mutex = 0
276     92:         proc  1 (increase_counter:1) mutex.pml:36 (state 22)        [.(goto)]
277                 shared_data = 0
278                 mutex = 0
279     93:         proc  1 (increase_counter:1) mutex.pml:34 (state 19)        [else]
280                 shared_data = 0
```

Продолжение листинга 1.4

```
                mutex = 0
 94:        proc  1 (increase_counter:1) mutex.pml:34 (state 20)        [goto :b0]
                shared_data = 0
                mutex = 0
 95:        proc  0 (:init::1) mutex.pml:56 (state 3)        [((_nr_pr==1))]
                shared_data = 0
                mutex = 0
     shared_data = 0
 96:        proc  0 (:init::1) mutex.pml:56 (state 4)        [printf('shared_data
 ↪  = %d\\n',shared_data)]
                shared_data = 0
                mutex = 0
3 processes created
```

1.6 Вывод

В ходе выполнения лабораторной работы была разработана модель взаимодействия двух процессов, работающих с общей переменной. Была продемонстрирована гонка процессов, возникающая при одновременном доступе к критической секции без синхронизации. Для предотвращения гонки процессов была разработана модель с использованием мьютекса, который обеспечивает взаимное исключение при доступе к общей переменной. Результаты работы программ были продемонстрированы с помощью логов SPIN.