|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **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»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Лабораторная работа № \_\_**2**\_\_**

**Дисциплина Математические основы верификации ПО**

|  |  |
| --- | --- |
| **Тема Моделирование гонки процессов**  **Студент \_Брянская Е.В.\_\_\_\_\_\_\_\_**  **Группа \_ИУ7-41М\_\_\_\_\_\_\_\_\_\_\_\_\_**  **Оценка (баллы) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**  **Преподаватель \_Кузнецова О.В.** |  |

Москва

2024 г

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

Фрагмент кода

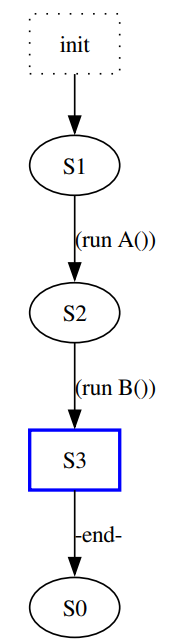
|  |
| --- |
| int x = 1024;  proctype A() {  int tmp  tmp = x  tmp++  x = tmp  printf("x=%d\n", x)  }  proctype B() {  int tmp  tmp = x  tmp--  x = tmp  printf("x=%d\n", x)  }  init {  run A()  run B()  } |

В программе запускаются два процесса, изменяющие значение глобальной переменной x. Один из процессов должен увеличивать на 1 значение, другой уменьшать.

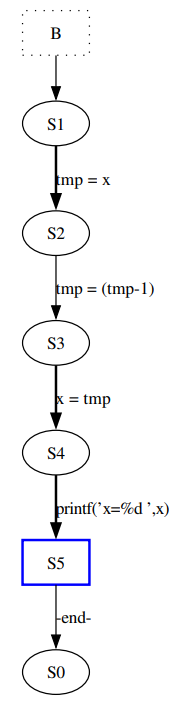
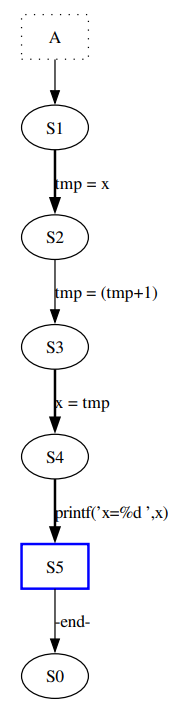
Ожидаемый конечный результат выполнения программы без возникновения гонки – 1024.

Граф переходов между состояниями модели

Общий граф переходов выглядит следующим образом:



Детализированный граф:



Демонстрация работы программы

|  |
| --- |
| $ pc\_spin651/spin.exe -p labs/lab02/lab02.pml  0: proc - (:root:) creates proc 0 (:init:)  Starting A with pid 1  1: proc 0 (:init::1) creates proc 1 (A)  1: proc 0 (:init::1) labs/lab02/lab02.pml:22 (state 1) [(run A())]  2: proc 1 (A:1) labs/lab02/lab02.pml:5 (state 1) [tmp = x]  Starting B with pid 2  3: proc 0 (:init::1) creates proc 2 (B)  3: proc 0 (:init::1) labs/lab02/lab02.pml:23 (state 2) [(run B())]  4: proc 2 (B:1) labs/lab02/lab02.pml:14 (state 1) [tmp = x]  5: proc 1 (A:1) labs/lab02/lab02.pml:6 (state 2) [tmp = (tmp+1)]  6: proc 1 (A:1) labs/lab02/lab02.pml:7 (state 3) [x = tmp]  x=1025  7: proc 1 (A:1) labs/lab02/lab02.pml:9 (state 4) [printf('x=%d\\n',x)]  8: proc 2 (B:1) labs/lab02/lab02.pml:15 (state 2) [tmp = (tmp-1)]  9: proc 2 (B:1) labs/lab02/lab02.pml:16 (state 3) [x = tmp]  x=1023  10: proc 2 (B:1) labs/lab02/lab02.pml:18 (state 4) [printf('x=%d\\n',x)]  10: proc 2 (B:1) terminates  10: proc 1 (A:1) terminates  10: proc 0 (:init::1) terminates  3 processes created |

По логу видно, что оба процесса одновременно сохранили значение переменной x в промежуточную переменную tmp и далее выполнили необходимые операции, записав итоговый результат в глобальную переменную x. Таким образом, был затёрт промежуточный результат, который никак не был использован вторым процессом.

Итого, можно сделать вывод о наличия гонки в системе.

Модель с использованием мьютекса

Решить проблему с гонкой может использование мьютексов, которые предотвращают одновременное вхождение процессов в критическую секцию.

Для этого в коде была заведена глобальная переменная mutex, которая увеличивается на 1 в случае, если процесс находится в критической секции и уменьшает на 1 в случае её освобождения.

|  |
| --- |
| int x = 1024, mutex = 0;  proctype A() {  atomic {  (mutex == 0) -> mutex++  }  int tmp  tmp = x  tmp++  x = tmp  printf("x=%d\n", x)  mutex--  }  proctype B() {  atomic {  (mutex == 0) -> mutex++  }  int tmp  tmp = x  tmp--  x = tmp  printf("x=%d\n", x)  mutex--  }  init {  run A()  run B()  } |

Демонстрация работы программы

|  |  |
| --- | --- |
| 1 | $ pc\_spin651/spin.exe -p labs/lab02/lab02.pml  0: proc - (:root:) creates proc 0 (:init:)  Starting A with pid 1  1: proc 0 (:init::1) creates proc 1 (A)  1: proc 0 (:init::1) labs/lab02/lab02.pml:34 (state 1) [(run A())]  Starting B with pid 2  2: proc 0 (:init::1) creates proc 2 (B)  2: proc 0 (:init::1) labs/lab02/lab02.pml:35 (state 2) [(run B())]  3: proc 2 (B:1) labs/lab02/lab02.pml:20 (state 1) [((mutex==0))]  4: proc 2 (B:1) labs/lab02/lab02.pml:20 (state 2) [mutex = (mutex+1)]  5: proc 2 (B:1) labs/lab02/lab02.pml:23 (state 3) [tmp = 0]  6: proc 2 (B:1) labs/lab02/lab02.pml:23 (state 4) [tmp = x]  7: proc 2 (B:1) labs/lab02/lab02.pml:24 (state 5) [tmp = (tmp-1)]  8: proc 2 (B:1) labs/lab02/lab02.pml:25 (state 6) [x = tmp]  x=1023  9: proc 2 (B:1) labs/lab02/lab02.pml:27 (state 7) [printf('x=%d\\n',x)]  10: proc 2 (B:1) labs/lab02/lab02.pml:29 (state 8) [mutex = (mutex-1)]  10: proc 2 (B:1) terminates  11: proc 1 (A:1) labs/lab02/lab02.pml:5 (state 1) [((mutex==0))]  12: proc 1 (A:1) labs/lab02/lab02.pml:5 (state 2) [mutex = (mutex+1)]  13: proc 1 (A:1) labs/lab02/lab02.pml:8 (state 3) [tmp = 0]  14: proc 1 (A:1) labs/lab02/lab02.pml:8 (state 4) [tmp = x]  15: proc 1 (A:1) labs/lab02/lab02.pml:9 (state 5) [tmp = (tmp+1)]  16: proc 1 (A:1) labs/lab02/lab02.pml:10 (state 6) [x = tmp]  x=1024  17: proc 1 (A:1) labs/lab02/lab02.pml:12 (state 7) [printf('x=%d\\n',x)]  18: proc 1 (A:1) labs/lab02/lab02.pml:14 (state 8) [mutex = (mutex-1)]  18: proc 1 (A:1) terminates  18: proc 0 (:init::1) terminates  3 processes created |
| 2 | $ pc\_spin651/spin.exe -p labs/lab02/lab02.pml  0: proc - (:root:) creates proc 0 (:init:)  Starting A with pid 1  1: proc 0 (:init::1) creates proc 1 (A)  1: proc 0 (:init::1) labs/lab02/lab02.pml:34 (state 1) [(run A())]  2: proc 1 (A:1) labs/lab02/lab02.pml:5 (state 1) [((mutex==0))]  3: proc 1 (A:1) labs/lab02/lab02.pml:5 (state 2) [mutex = (mutex+1)]  4: proc 1 (A:1) labs/lab02/lab02.pml:8 (state 3) [tmp = 0]  5: proc 1 (A:1) labs/lab02/lab02.pml:8 (state 4) [tmp = x]  6: proc 1 (A:1) labs/lab02/lab02.pml:9 (state 5) [tmp = (tmp+1)]  7: proc 1 (A:1) labs/lab02/lab02.pml:10 (state 6) [x = tmp]  x=1025  8: proc 1 (A:1) labs/lab02/lab02.pml:12 (state 7) [printf('x=%d\\n',x)]  9: proc 1 (A:1) labs/lab02/lab02.pml:14 (state 8) [mutex = (mutex-1)]  9: proc 1 (A:1) terminates  Starting B with pid 1  10: proc 0 (:init::1) creates proc 1 (B)  10: proc 0 (:init::1) labs/lab02/lab02.pml:35 (state 2) [(run B())]  11: proc 1 (B:1) labs/lab02/lab02.pml:20 (state 1) [((mutex==0))]  12: proc 1 (B:1) labs/lab02/lab02.pml:20 (state 2) [mutex = (mutex+1)]  13: proc 1 (B:1) labs/lab02/lab02.pml:23 (state 3) [tmp = 0]  14: proc 1 (B:1) labs/lab02/lab02.pml:23 (state 4) [tmp = x]  15: proc 1 (B:1) labs/lab02/lab02.pml:24 (state 5) [tmp = (tmp-1)]  16: proc 1 (B:1) labs/lab02/lab02.pml:25 (state 6) [x = tmp]  x=1024  17: proc 1 (B:1) labs/lab02/lab02.pml:27 (state 7) [printf('x=%d\\n',x)]  18: proc 1 (B:1) labs/lab02/lab02.pml:29 (state 8) [mutex = (mutex-1)]  18: proc 1 (B:1) terminates  18: proc 0 (:init::1) terminates  3 processes created |

Выше приведён лог программы с использованием мьютексов. Приведено оба случая выполнения двух процессов:

1. сначала В, потом А
2. сначала А, потом В

**Вывод**

В результате выполнения работы был рассмотрен случай гонки и способ её предотвращения с помощью мьютексов.