|  |  |
| --- | --- |
| **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 ИУ «международных образовательных программ»

КАФЕДРА ИУ-7 «Программное обеспечение эвм и информационные технологии»

**ОТЧЕТ**

**по Лабораторной работе №2**

**по курсу**

**«Математические основы верификации ПО»**

**Тема**

**«Моделирование гонки процессов»**

Студент \_\_\_ИУ7-и42М\_\_\_\_ \_\_\_Абди О.У.\_\_\_\_

(Группа) (И.О.Фамилия)

Преподаватель \_\_\_\_Кузнецова О.В.\_\_\_\_

(И.О.Фамилия)

*2024г.*

**Задание**

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

**Моделирование гонки процессов**

Моделирование гонки процессов: Демонстрирует параллельный доступ к общей переменной 'balance' двумя процессами, что может привести к гонкам и непредсказуемому поведению.

Листинг 1 — Моделирование гонки процессов.

|  |
| --- |
| #define N 10  int balance = 500;  active proctype Deposit() {  byte i;  do  :: i < N ->  balance = balance + 100;  printf("Deposit: Depositing $100. New balance: $%d\n", balance);  i = i + 1;  od;  }  active proctype Withdraw() {  byte i;  do  :: i < N ->  balance = balance - 50;  printf("Withdraw: Withdrawing $50. New balance: $%d\n", balance);  i = i + 1;  od;  } |

Результат работы модели приведен в листинге 2.

Листинг 2 — Результат моделирования.

|  |
| --- |
| 0: proc - (:root:) creates proc 0 (Deposit)  0: proc - (:root:) creates proc 1 (Withdraw)  1: proc 0 (Deposit:1) race.pml:8 (state 1) [((i<10))]  2: proc 1 (Withdraw:1) race.pml:18 (state 1) [((i<10))]  3: proc 0 (Deposit:1) race.pml:9 (state 2) [balance = (balance+100)]  Deposit: Depositing $100. New balance: $600  4: proc 0 (Deposit:1) race.pml:10 (state 3) [printf('Deposit: Depositing $100. New balance: $%d\\n',balance)]  5: proc 1 (Withdraw:1) race.pml:19 (state 2) [balance = (balance-50)]  6: proc 0 (Deposit:1) race.pml:11 (state 4) [i = (i+1)]  Withdraw: Withdrawing $50. New balance: $550  8: proc 1 (Withdraw:1) race.pml:20 (state 3) [printf('Withdraw: Withdrawing $50. New balance: $%d\\n',balance)]  9: proc 0 (Deposit:1) race.pml:8 (state 1) [((i<10))]  10: proc 0 (Deposit:1) race.pml:9 (state 2) [balance = (balance+100)]  Deposit: Depositing $100. New balance: $650  11: proc 0 (Deposit:1) race.pml:10 (state 3) [printf('Deposit: Depositing $100. New balance: $%d\\n',balance)]  12: proc 1 (Withdraw:1) race.pml:21 (state 4) [i = (i+1)]  14: proc 0 (Deposit:1) race.pml:11 (state 4) [i = (i+1)]  15: proc 1 (Withdraw:1) race.pml:18 (state 1) [((i<10))]  16: proc 1 (Withdraw:1) race.pml:19 (state 2) [balance = (balance-50)]  Withdraw: Withdrawing $50. New balance: $600  17: proc 1 (Withdraw:1) race.pml:20 (state 3) [printf('Withdraw: Withdrawing $50. New balance: $%d\\n',balance)]  19: proc 1 (Withdraw:1) race.pml:21 (state 4) [i = (i+1)]  20: proc 0 (Deposit:1) race.pml:8 (state 1) [((i<10))]  22: proc 1 (Withdraw:1) race.pml:18 (state 1) [((i<10))]  23: proc 0 (Deposit:1) race.pml:9 (state 2) [balance = (balance+100)]  24: proc 1 (Withdraw:1) race.pml:19 (state 2) [balance = (balance-50)]  Withdraw: Withdrawing $50. New balance: $650  25: proc 1 (Withdraw:1) race.pml:20 (state 3) [printf('Withdraw: Withdrawing $50. New balance: $%d\\n',balance)]  Deposit: Depositing $100. New balance: $650  26: proc 0 (Deposit:1) race.pml:10 (state 3) [printf('Deposit: Depositing $100. New balance: $%d\\n',balance)]  27: proc 0 (Deposit:1) race.pml:11 (state 4) [i = (i+1)]  29: proc 0 (Deposit:1) race.pml:8 (state 1) [((i<10))]  30: proc 0 (Deposit:1) race.pml:9 (state 2) [balance = (balance+100)]  Deposit: Depositing $100. New balance: $750  31: proc 0 (Deposit:1) race.pml:10 (state 3) [printf('Deposit: Depositing $100. New balance: $%d\\n',balance)]  32: proc 0 (Deposit:1) race.pml:11 (state 4) [i = (i+1)]  33: proc 1 (Withdraw:1) race.pml:21 (state 4) [i = (i+1)]  35: proc 0 (Deposit:1) race.pml:8 (state 1) [((i<10))]  36: proc 0 (Deposit:1) race.pml:9 (state 2) [balance = (balance+100)]  Deposit: Depositing $100. New balance: $850  38: proc 0 (Deposit:1) race.pml:10 (state 3) [printf('Deposit: Depositing $100. New balance: $%d\\n',balance)]  39: proc 1 (Withdraw:1) race.pml:18 (state 1) [((i<10))]  40: proc 0 (Deposit:1) race.pml:11 (state 4) [i = (i+1)]  42: proc 1 (Withdraw:1) race.pml:19 (state 2) [balance = (balance-50)]  43: proc 0 (Deposit:1) race.pml:8 (state 1) [((i<10))]  Withdraw: Withdrawing $50. New balance: $800  44: proc 1 (Withdraw:1) race.pml:20 (state 3) [printf('Withdraw: Withdrawing $50. New balance: $%d\\n',balance)]  45: proc 1 (Withdraw:1) race.pml:21 (state 4) [i = (i+1)]  47: proc 0 (Deposit:1) race.pml:9 (state 2) [balance = (balance+100)]  Deposit: Depositing $100. New balance: $900  48: proc 0 (Deposit:1) race.pml:10 (state 3) [printf('Deposit: Depositing $100. New balance: $%d\\n',balance)]  49: proc 1 (Withdraw:1) race.pml:18 (state 1) [((i<10))]  50: proc 0 (Deposit:1) race.pml:11 (state 4) [i = (i+1)]  52: proc 0 (Deposit:1) race.pml:8 (state 1) [((i<10))]  53: proc 0 (Deposit:1) race.pml:9 (state 2) [balance = (balance+100)]  54: proc 1 (Withdraw:1) race.pml:19 (state 2) [balance = (balance-50)]  Deposit: Depositing $100. New balance: $950  55: proc 0 (Deposit:1) race.pml:10 (state 3) [printf('Deposit: Depositing $100. New balance: $%d\\n',balance)]  Withdraw: Withdrawing $50. New balance: $950  56: proc 1 (Withdraw:1) race.pml:20 (state 3) [printf('Withdraw: Withdrawing $50. New balance: $%d\\n',balance)]  57: proc 1 (Withdraw:1) race.pml:21 (state 4) [i = (i+1)]  59: proc 0 (Deposit:1) race.pml:11 (state 4) [i = (i+1)]  61: proc 1 (Withdraw:1) race.pml:18 (state 1) [((i<10))]  62: proc 1 (Withdraw:1) race.pml:19 (state 2) [balance = (balance-50)]  63: proc 0 (Deposit:1) race.pml:8 (state 1) [((i<10))]  64: proc 0 (Deposit:1) race.pml:9 (state 2) [balance = (balance+100)]  Deposit: Depositing $100. New balance: $1000  65: proc 0 (Deposit:1) race.pml:10 (state 3) [printf('Deposit: Depositing $100. New balance: $%d\\n',balance)]  Withdraw: Withdrawing $50. New balance: $1000  66: proc 1 (Withdraw:1) race.pml:20 (state 3) [printf('Withdraw: Withdrawing $50. New balance: $%d\\n',balance)]  67: proc 1 (Withdraw:1) race.pml:21 (state 4) [i = (i+1)]  68: proc 0 (Deposit:1) race.pml:11 (state 4) [i = (i+1)]  71: proc 0 (Deposit:1) race.pml:8 (state 1) [((i<10))]  72: proc 1 (Withdraw:1) race.pml:18 (state 1) [((i<10))]  73: proc 0 (Deposit:1) race.pml:9 (state 2) [balance = (balance+100)]  74: proc 1 (Withdraw:1) race.pml:19 (state 2) [balance = (balance-50)]  Withdraw: Withdrawing $50. New balance: $1050  75: proc 1 (Withdraw:1) race.pml:20 (state 3) [printf('Withdraw: Withdrawing $50. New balance: $%d\\n',balance)]  76: proc 1 (Withdraw:1) race.pml:21 (state 4) [i = (i+1)]  Deposit: Depositing $100. New balance: $1050  78: proc 0 (Deposit:1) race.pml:10 (state 3) [printf('Deposit: Depositing $100. New balance: $%d\\n',balance)]  79: proc 0 (Deposit:1) race.pml:11 (state 4) [i = (i+1)]  81: proc 0 (Deposit:1) race.pml:8 (state 1) [((i<10))]  82: proc 1 (Withdraw:1) race.pml:18 (state 1) [((i<10))]  83: proc 0 (Deposit:1) race.pml:9 (state 2) [balance = (balance+100)]  Deposit: Depositing $100. New balance: $1150  84: proc 0 (Deposit:1) race.pml:10 (state 3) [printf('Deposit: Depositing $100. New balance: $%d\\n',balance)]  85: proc 0 (Deposit:1) race.pml:11 (state 4) [i = (i+1)]  86: proc 1 (Withdraw:1) race.pml:19 (state 2) [balance = (balance-50)]  Withdraw: Withdrawing $50. New balance: $1100  88: proc 1 (Withdraw:1) race.pml:20 (state 3) [printf('Withdraw: Withdrawing $50. New balance: $%d\\n',balance)]  89: proc 1 (Withdraw:1) race.pml:21 (state 4) [i = (i+1)]  91: proc 1 (Withdraw:1) race.pml:18 (state 1) [((i<10))]  92: proc 1 (Withdraw:1) race.pml:19 (state 2) [balance = (balance-50)]  Withdraw: Withdrawing $50. New balance: $1050  93: proc 1 (Withdraw:1) race.pml:20 (state 3) [printf('Withdraw: Withdrawing $50. New balance: $%d\\n',balance)]  94: proc 1 (Withdraw:1) race.pml:21 (state 4) [i = (i+1)]  96: proc 1 (Withdraw:1) race.pml:18 (state 1) [((i<10))]  97: proc 1 (Withdraw:1) race.pml:19 (state 2) [balance = (balance-50)]  Withdraw: Withdrawing $50. New balance: $1000  98: proc 1 (Withdraw:1) race.pml:20 (state 3) [printf('Withdraw: Withdrawing $50. New balance: $%d\\n',balance)]  99: proc 1 (Withdraw:1) race.pml:21 (state 4) [i = (i+1)]  timeout  #processes: 2  100: proc 1 (Withdraw:1) race.pml:17 (state 5)  100: proc 0 (Deposit:1) race.pml:7 (state 5)  2 processes created |

Как видно из логов, оба процесса одновременно получают доступ и изменяют 'balance', что приводит к непредсказуемому поведению.

Листинг 3 — Моделирование разделяемого доступа с использованием мьютекса.

|  |
| --- |
| int balance = 500;  bool mutex;  #define N 10  #define lock\_mutex(mutex) \  do \  :: 1 -> atomic { \  if \  :: mutex == false -> \  mutex = true; \  break \  fi \  } \  od  #define unlock\_mutex(mutex) \  mutex = false  active proctype Deposit()  {  byte i;  do  :: i < N ->  lock\_mutex(mutex);  balance = balance + 100;  printf("Deposit: Depositing $100. New balance: $%d\n", balance);  unlock\_mutex(mutex);  i = i + 1;  od;  }  active proctype Withdraw()  {  byte i;  do  :: i < N ->  lock\_mutex(mutex);  balance = balance - 50;  printf("Withdraw: Withdrawing $50. New balance: $%d\n", balance);  unlock\_mutex(mutex);  i = i + 1;  od;  } |

В этом примере мьютекс реализован с использованием глобальной переменной 'mutex' и ключевого слова 'atomic', которое делает операции модификации переменной неделимыми. В случае, когда мьютекс занят (mutex == 1), попытка вызова 'lock\_mutex' приведет к ожиданию его освобождения. Результат выполнения такой программы приведен в Листинге 4. Как видно, когда процесс 'Deposit' пытается добавить сумму к разделяемой переменной, процесс 'Withdraw' блокируется в ожидании освобождения мьютекса.

Листинг 4 — Результат работы программы со lock\_mutex.

|  |
| --- |
| 0: proc - (:root:) creates proc 0 (Deposit)  0: proc - (:root:) creates proc 1 (Withdraw)  1: proc 0 (Deposit:1) mutex.pml:26 (state 1) [((i<10))]  2: proc 1 (Withdraw:1) mutex.pml:40 (state 1) [((i<10))]  4: proc 0 (Deposit:1) mutex.pml:27 (state 2) [(1)]  6: proc 1 (Withdraw:1) mutex.pml:41 (state 2) [(1)]  7: proc 0 (Deposit:1) mutex.pml:27 (state 3) [((mutex==0))]  8: proc 0 (Deposit:1) mutex.pml:27 (state 4) [mutex = 1]  9: proc 0 (Deposit:1) mutex.pml:27 (state 11) [break]  10: proc 0 (Deposit:1) mutex.pml:28 (state 12) [balance = (balance+100)]  Deposit: Depositing $100. New balance: $600  11: proc 0 (Deposit:1) mutex.pml:29 (state 13) [printf('Deposit: Depositing $100. New balance: $%d\\n',balance)]  12: proc 0 (Deposit:1) mutex.pml:30 (state 14) [mutex = 0]  13: proc 1 (Withdraw:1) mutex.pml:41 (state 3) [((mutex==0))]  14: proc 1 (Withdraw:1) mutex.pml:41 (state 4) [mutex = 1]  15: proc 1 (Withdraw:1) mutex.pml:41 (state 11) [break]  16: proc 0 (Deposit:1) mutex.pml:31 (state 15) [i = (i+1)]  18: proc 0 (Deposit:1) mutex.pml:26 (state 1) [((i<10))]  19: proc 1 (Withdraw:1) mutex.pml:42 (state 12) [balance = (balance-50)]  Withdraw: Withdrawing $50. New balance: $550  20: proc 1 (Withdraw:1) mutex.pml:43 (state 13) [printf('Withdraw: Withdrawing $50. New balance: $%d\\n',balance)]  22: proc 0 (Deposit:1) mutex.pml:27 (state 2) [(1)]  23: proc 1 (Withdraw:1) mutex.pml:44 (state 14) [mutex = 0]  24: proc 1 (Withdraw:1) mutex.pml:45 (state 15) [i = (i+1)]  25: proc 0 (Deposit:1) mutex.pml:27 (state 3) [((mutex==0))]  26: proc 0 (Deposit:1) mutex.pml:27 (state 4) [mutex = 1]  28: proc 0 (Deposit:1) mutex.pml:27 (state 11) [break]  29: proc 1 (Withdraw:1) mutex.pml:40 (state 1) [((i<10))]  30: proc 0 (Deposit:1) mutex.pml:28 (state 12) [balance = (balance+100)]  Deposit: Depositing $100. New balance: $650  31: proc 0 (Deposit:1) mutex.pml:29 (state 13) [printf('Deposit: Depositing $100. New balance: $%d\\n',balance)]  33: proc 0 (Deposit:1) mutex.pml:30 (state 14) [mutex = 0]  34: proc 1 (Withdraw:1) mutex.pml:41 (state 2) [(1)]  35: proc 1 (Withdraw:1) mutex.pml:41 (state 3) [((mutex==0))]  36: proc 1 (Withdraw:1) mutex.pml:41 (state 4) [mutex = 1]  37: proc 0 (Deposit:1) mutex.pml:31 (state 15) [i = (i+1)]  39: proc 1 (Withdraw:1) mutex.pml:41 (state 11) [break]  40: proc 1 (Withdraw:1) mutex.pml:42 (state 12) [balance = (balance-50)]  41: proc 0 (Deposit:1) mutex.pml:26 (state 1) [((i<10))]  Withdraw: Withdrawing $50. New balance: $600  43: proc 1 (Withdraw:1) mutex.pml:43 (state 13) [printf('Withdraw: Withdrawing $50. New balance: $%d\\n',balance)]  44: proc 1 (Withdraw:1) mutex.pml:44 (state 14) [mutex = 0]  45: proc 0 (Deposit:1) mutex.pml:27 (state 2) [(1)]  46: proc 1 (Withdraw:1) mutex.pml:45 (state 15) [i = (i+1)]  48: proc 1 (Withdraw:1) mutex.pml:40 (state 1) [((i<10))]  50: proc 0 (Deposit:1) mutex.pml:27 (state 3) [((mutex==0))]  51: proc 0 (Deposit:1) mutex.pml:27 (state 4) [mutex = 1]  52: proc 0 (Deposit:1) mutex.pml:27 (state 11) [break]  53: proc 1 (Withdraw:1) mutex.pml:41 (state 2) [(1)]  54: proc 0 (Deposit:1) mutex.pml:28 (state 12) [balance = (balance+100)]  Deposit: Depositing $100. New balance: $700  55: proc 0 (Deposit:1) mutex.pml:29 (state 13) [printf('Deposit: Depositing $100. New balance: $%d\\n',balance)]  56: proc 0 (Deposit:1) mutex.pml:30 (state 14) [mutex = 0]  57: proc 1 (Withdraw:1) mutex.pml:41 (state 3) [((mutex==0))]  58: proc 1 (Withdraw:1) mutex.pml:41 (state 4) [mutex = 1]  59: proc 0 (Deposit:1) mutex.pml:31 (state 15) [i = (i+1)]  60: proc 1 (Withdraw:1) mutex.pml:41 (state 11) [break]  62: proc 1 (Withdraw:1) mutex.pml:42 (state 12) [balance = (balance-50)]  Withdraw: Withdrawing $50. New balance: $650  63: proc 1 (Withdraw:1) mutex.pml:43 (state 13) [printf('Withdraw: Withdrawing $50. New balance: $%d\\n',balance)]  64: proc 1 (Withdraw:1) mutex.pml:44 (state 14) [mutex = 0]  65: proc 1 (Withdraw:1) mutex.pml:45 (state 15) [i = (i+1)]  66: proc 0 (Deposit:1) mutex.pml:26 (state 1) [((i<10))]  68: proc 0 (Deposit:1) mutex.pml:27 (state 2) [(1)]  70: proc 0 (Deposit:1) mutex.pml:27 (state 3) [((mutex==0))]  71: proc 0 (Deposit:1) mutex.pml:27 (state 4) [mutex = 1]  72: proc 0 (Deposit:1) mutex.pml:27 (state 11) [break]  73: proc 0 (Deposit:1) mutex.pml:28 (state 12) [balance = (balance+100)]  74: proc 1 (Withdraw:1) mutex.pml:40 (state 1) [((i<10))]  Deposit: Depositing $100. New balance: $750  76: proc 0 (Deposit:1) mutex.pml:29 (state 13) [printf('Deposit: Depositing $100. New balance: $%d\\n',balance)]  77: proc 1 (Withdraw:1) mutex.pml:41 (state 2) [(1)]  78: proc 0 (Deposit:1) mutex.pml:30 (state 14) [mutex = 0]  79: proc 0 (Deposit:1) mutex.pml:31 (state 15) [i = (i+1)]  80: proc 1 (Withdraw:1) mutex.pml:41 (state 3) [((mutex==0))]  81: proc 1 (Withdraw:1) mutex.pml:41 (state 4) [mutex = 1]  82: proc 1 (Withdraw:1) mutex.pml:41 (state 11) [break]  84: proc 1 (Withdraw:1) mutex.pml:42 (state 12) [balance = (balance-50)]  85: proc 0 (Deposit:1) mutex.pml:26 (state 1) [((i<10))]  Withdraw: Withdrawing $50. New balance: $700  86: proc 1 (Withdraw:1) mutex.pml:43 (state 13) [printf('Withdraw: Withdrawing $50. New balance: $%d\\n',balance)]  87: proc 1 (Withdraw:1) mutex.pml:44 (state 14) [mutex = 0]  89: proc 0 (Deposit:1) mutex.pml:27 (state 2) [(1)]  90: proc 0 (Deposit:1) mutex.pml:27 (state 3) [((mutex==0))]  91: proc 0 (Deposit:1) mutex.pml:27 (state 4) [mutex = 1]  92: proc 1 (Withdraw:1) mutex.pml:45 (state 15) [i = (i+1)]  94: proc 0 (Deposit:1) mutex.pml:27 (state 11) [break]  95: proc 0 (Deposit:1) mutex.pml:28 (state 12) [balance = (balance+100)]  96: proc 1 (Withdraw:1) mutex.pml:40 (state 1) [((i<10))]  Deposit: Depositing $100. New balance: $800  97: proc 0 (Deposit:1) mutex.pml:29 (state 13) [printf('Deposit: Depositing $100. New balance: $%d\\n',balance)]  98: proc 0 (Deposit:1) mutex.pml:30 (state 14) [mutex = 0]  100: proc 1 (Withdraw:1) mutex.pml:41 (state 2) [(1)]  101: proc 1 (Withdraw:1) mutex.pml:41 (state 3) [((mutex==0))]  102: proc 1 (Withdraw:1) mutex.pml:41 (state 4) [mutex = 1]  103: proc 0 (Deposit:1) mutex.pml:31 (state 15) [i = (i+1)]  104: proc 1 (Withdraw:1) mutex.pml:41 (state 11) [break]  105: proc 1 (Withdraw:1) mutex.pml:42 (state 12) [balance = (balance-50)]  Withdraw: Withdrawing $50. New balance: $750  106: proc 1 (Withdraw:1) mutex.pml:43 (state 13) [printf('Withdraw: Withdrawing $50. New balance: $%d\\n',balance)]  107: proc 1 (Withdraw:1) mutex.pml:44 (state 14) [mutex = 0]  109: proc 0 (Deposit:1) mutex.pml:26 (state 1) [((i<10))]  110: proc 1 (Withdraw:1) mutex.pml:45 (state 15) [i = (i+1)]  112: proc 1 (Withdraw:1) mutex.pml:40 (state 1) [((i<10))]  114: proc 0 (Deposit:1) mutex.pml:27 (state 2) [(1)]  116: proc 0 (Deposit:1) mutex.pml:27 (state 3) [((mutex==0))]  117: proc 0 (Deposit:1) mutex.pml:27 (state 4) [mutex = 1]  118: proc 0 (Deposit:1) mutex.pml:27 (state 11) [break]  119: proc 0 (Deposit:1) mutex.pml:28 (state 12) [balance = (balance+100)]  120: proc 1 (Withdraw:1) mutex.pml:41 (state 2) [(1)]  Deposit: Depositing $100. New balance: $850  121: proc 0 (Deposit:1) mutex.pml:29 (state 13) [printf('Deposit: Depositing $100. New balance: $%d\\n',balance)]  122: proc 0 (Deposit:1) mutex.pml:30 (state 14) [mutex = 0]  123: proc 0 (Deposit:1) mutex.pml:31 (state 15) [i = (i+1)]  125: proc 1 (Withdraw:1) mutex.pml:41 (state 3) [((mutex==0))]  126: proc 1 (Withdraw:1) mutex.pml:41 (state 4) [mutex = 1]  127: proc 1 (Withdraw:1) mutex.pml:41 (state 11) [break]  128: proc 0 (Deposit:1) mutex.pml:26 (state 1) [((i<10))]  130: proc 0 (Deposit:1) mutex.pml:27 (state 2) [(1)]  131: proc 1 (Withdraw:1) mutex.pml:42 (state 12) [balance = (balance-50)]  Withdraw: Withdrawing $50. New balance: $800  132: proc 1 (Withdraw:1) mutex.pml:43 (state 13) [printf('Withdraw: Withdrawing $50. New balance: $%d\\n',balance)]  133: proc 1 (Withdraw:1) mutex.pml:44 (state 14) [mutex = 0]  134: proc 0 (Deposit:1) mutex.pml:27 (state 3) [((mutex==0))]  135: proc 0 (Deposit:1) mutex.pml:27 (state 4) [mutex = 1]  136: proc 1 (Withdraw:1) mutex.pml:45 (state 15) [i = (i+1)]  137: proc 0 (Deposit:1) mutex.pml:27 (state 11) [break]  138: proc 0 (Deposit:1) mutex.pml:28 (state 12) [balance = (balance+100)]  Deposit: Depositing $100. New balance: $900  140: proc 0 (Deposit:1) mutex.pml:29 (state 13) [printf('Deposit: Depositing $100. New balance: $%d\\n',balance)]  141: proc 1 (Withdraw:1) mutex.pml:40 (state 1) [((i<10))]  143: proc 0 (Deposit:1) mutex.pml:30 (state 14) [mutex = 0]  144: proc 1 (Withdraw:1) mutex.pml:41 (state 2) [(1)]  145: proc 0 (Deposit:1) mutex.pml:31 (state 15) [i = (i+1)]  147: proc 1 (Withdraw:1) mutex.pml:41 (state 3) [((mutex==0))]  148: proc 1 (Withdraw:1) mutex.pml:41 (state 4) [mutex = 1]  149: proc 1 (Withdraw:1) mutex.pml:41 (state 11) [break]  150: proc 0 (Deposit:1) mutex.pml:26 (state 1) [((i<10))]  151: proc 1 (Withdraw:1) mutex.pml:42 (state 12) [balance = (balance-50)]  Withdraw: Withdrawing $50. New balance: $850  152: proc 1 (Withdraw:1) mutex.pml:43 (state 13) [printf('Withdraw: Withdrawing $50. New balance: $%d\\n',balance)]  153: proc 1 (Withdraw:1) mutex.pml:44 (state 14) [mutex = 0]  155: proc 0 (Deposit:1) mutex.pml:27 (state 2) [(1)]  156: proc 1 (Withdraw:1) mutex.pml:45 (state 15) [i = (i+1)]  157: proc 0 (Deposit:1) mutex.pml:27 (state 3) [((mutex==0))]  158: proc 0 (Deposit:1) mutex.pml:27 (state 4) [mutex = 1]  160: proc 0 (Deposit:1) mutex.pml:27 (state 11) [break]  161: proc 0 (Deposit:1) mutex.pml:28 (state 12) [balance = (balance+100)]  162: proc 1 (Withdraw:1) mutex.pml:40 (state 1) [((i<10))]  Deposit: Depositing $100. New balance: $950  163: proc 0 (Deposit:1) mutex.pml:29 (state 13) [printf('Deposit: Depositing $100. New balance: $%d\\n',balance)]  164: proc 0 (Deposit:1) mutex.pml:30 (state 14) [mutex = 0]  166: proc 0 (Deposit:1) mutex.pml:31 (state 15) [i = (i+1)]  168: proc 1 (Withdraw:1) mutex.pml:41 (state 2) [(1)]  169: proc 1 (Withdraw:1) mutex.pml:41 (state 3) [((mutex==0))]  170: proc 1 (Withdraw:1) mutex.pml:41 (state 4) [mutex = 1]  171: proc 1 (Withdraw:1) mutex.pml:41 (state 11) [break]  172: proc 1 (Withdraw:1) mutex.pml:42 (state 12) [balance = (balance-50)]  Withdraw: Withdrawing $50. New balance: $900  173: proc 1 (Withdraw:1) mutex.pml:43 (state 13) [printf('Withdraw: Withdrawing $50. New balance: $%d\\n',balance)]  174: proc 0 (Deposit:1) mutex.pml:26 (state 1) [((i<10))]  175: proc 1 (Withdraw:1) mutex.pml:44 (state 14) [mutex = 0]  176: proc 1 (Withdraw:1) mutex.pml:45 (state 15) [i = (i+1)]  178: proc 0 (Deposit:1) mutex.pml:27 (state 2) [(1)]  180: proc 0 (Deposit:1) mutex.pml:27 (state 3) [((mutex==0))]  181: proc 0 (Deposit:1) mutex.pml:27 (state 4) [mutex = 1]  182: proc 0 (Deposit:1) mutex.pml:27 (state 11) [break]  183: proc 0 (Deposit:1) mutex.pml:28 (state 12) [balance = (balance+100)]  184: proc 1 (Withdraw:1) mutex.pml:40 (state 1) [((i<10))]  Deposit: Depositing $100. New balance: $1000  185: proc 0 (Deposit:1) mutex.pml:29 (state 13) [printf('Deposit: Depositing $100. New balance: $%d\\n',balance)]  187: proc 1 (Withdraw:1) mutex.pml:41 (state 2) [(1)]  188: proc 0 (Deposit:1) mutex.pml:30 (state 14) [mutex = 0]  189: proc 1 (Withdraw:1) mutex.pml:41 (state 3) [((mutex==0))]  190: proc 1 (Withdraw:1) mutex.pml:41 (state 4) [mutex = 1]  191: proc 0 (Deposit:1) mutex.pml:31 (state 15) [i = (i+1)]  192: proc 1 (Withdraw:1) mutex.pml:41 (state 11) [break]  194: proc 0 (Deposit:1) mutex.pml:26 (state 1) [((i<10))]  196: proc 1 (Withdraw:1) mutex.pml:42 (state 12) [balance = (balance-50)]  197: proc 0 (Deposit:1) mutex.pml:27 (state 2) [(1)]  Withdraw: Withdrawing $50. New balance: $950  198: proc 1 (Withdraw:1) mutex.pml:43 (state 13) [printf('Withdraw: Withdrawing $50. New balance: $%d\\n',balance)]  199: proc 1 (Withdraw:1) mutex.pml:44 (state 14) [mutex = 0]  200: proc 0 (Deposit:1) mutex.pml:27 (state 3) [((mutex==0))]  201: proc 0 (Deposit:1) mutex.pml:27 (state 4) [mutex = 1]  202: proc 1 (Withdraw:1) mutex.pml:45 (state 15) [i = (i+1)]  203: proc 0 (Deposit:1) mutex.pml:27 (state 11) [break]  204: proc 0 (Deposit:1) mutex.pml:28 (state 12) [balance = (balance+100)]  Deposit: Depositing $100. New balance: $1050  205: proc 0 (Deposit:1) mutex.pml:29 (state 13) [printf('Deposit: Depositing $100. New balance: $%d\\n',balance)]  206: proc 0 (Deposit:1) mutex.pml:30 (state 14) [mutex = 0]  207: proc 0 (Deposit:1) mutex.pml:31 (state 15) [i = (i+1)]  210: proc 1 (Withdraw:1) mutex.pml:40 (state 1) [((i<10))]  212: proc 1 (Withdraw:1) mutex.pml:41 (state 2) [(1)]  213: proc 1 (Withdraw:1) mutex.pml:41 (state 3) [((mutex==0))]  214: proc 1 (Withdraw:1) mutex.pml:41 (state 4) [mutex = 1]  215: proc 1 (Withdraw:1) mutex.pml:41 (state 11) [break]  216: proc 1 (Withdraw:1) mutex.pml:42 (state 12) [balance = (balance-50)]  Withdraw: Withdrawing $50. New balance: $1000  217: proc 1 (Withdraw:1) mutex.pml:43 (state 13) [printf('Withdraw: Withdrawing $50. New balance: $%d\\n',balance)]  218: proc 1 (Withdraw:1) mutex.pml:44 (state 14) [mutex = 0]  219: proc 1 (Withdraw:1) mutex.pml:45 (state 15) [i = (i+1)]  timeout  #processes: 2  220: proc 1 (Withdraw:1) mutex.pml:39 (state 16)  220: proc 0 (Deposit:1) mutex.pml:25 (state 16)  2 processes created |

**Вывод:**

Исходя из представленных решений, очевидно, что эффективное использование мьютексов позволяет успешно решать проблемы гонок при одновременном доступе к общим данным нескольких процессов. Первое решение без мьютексов демонстрировало гонки, приводя к непредсказуемому поведению при модификации баланса. Однако во втором решении, благодаря внедрению мьютексов, была обеспечена исключительная доступность к критической секции, что предотвратило одновременные изменения общей переменной баланса. Следовательно, мьютексное решение подчеркивает важность механизмов синхронизации в параллельном программировании для поддержания целостности данных и предотвращения гонок.