Нижегородский государственный университет им. Н.И.Лобачевского

Факультет Вычислительной математики и кибернетики

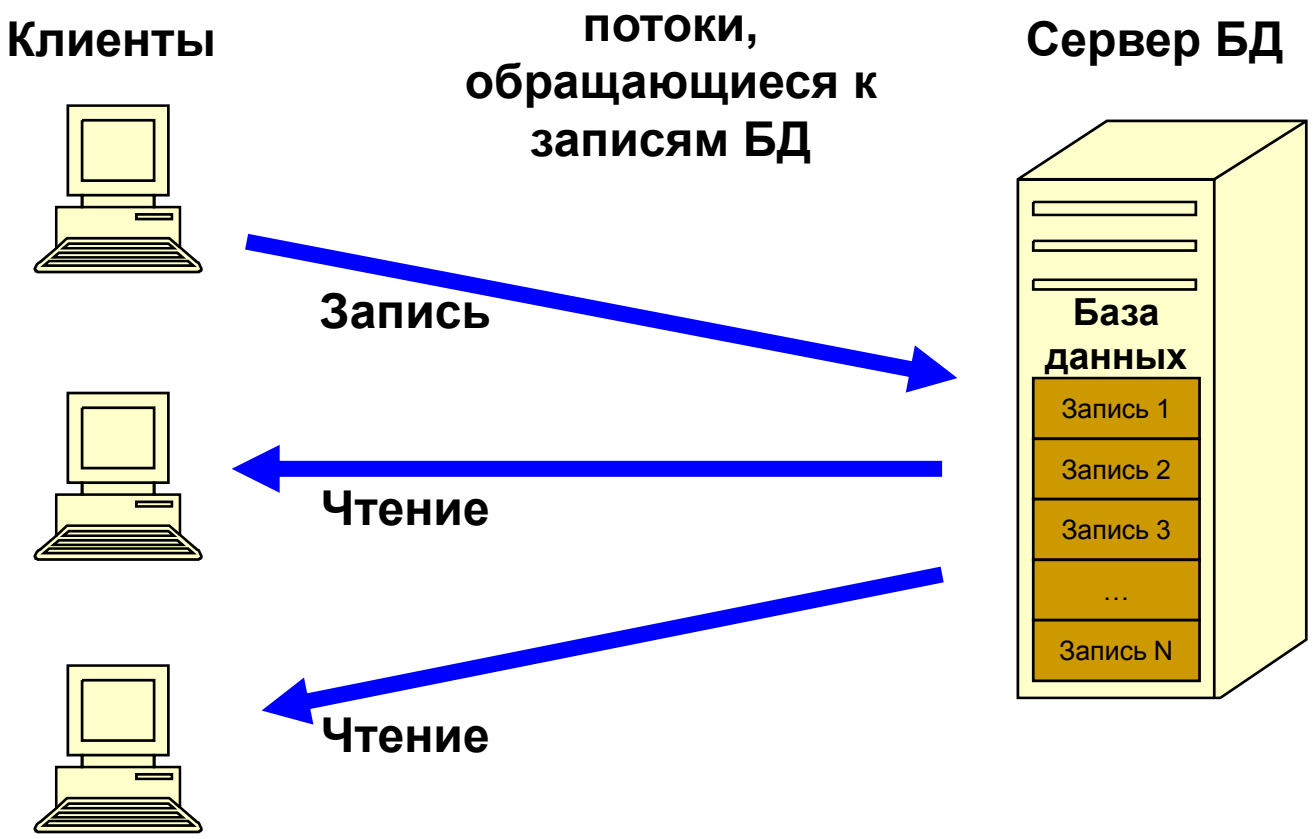
Операционные системы: *аспекты параллелизма*

Задача "Читатели-Писатели"
Лабораторная работа

Линёв А.В.

2007

"Читатели-Писатели" Постановка задачи



"Читатели-Писатели" Постановка задачи

- Имеются критические данные, над которыми определены операции чтения и записи
  - Потоки-писатели изменяют данные; если поток-писатель работает с данными, они не могут использоваться другими потоками
  - Потоки-читатели не изменяют данные; если поток-читатель работает с данными, они могут быть использованы другими потоками-читателями
- Потоки выполняются параллельно

"Читатели-Писатели" Решение № 1

```
Writer(){
1  while(true){
2    PrepareData();
3    Write(Data);
   }
}
```

```
Reader(){
4    while(true){
5      Read(&Data);
6      UseData();
     }
}
```

Предположим, выполняются два потока–писателя

После шага 6 оба потока–писателя одновременно окажутся в критической секции (вызов Write(Data)).

Мьютексы

- Объекты, которые могут находиться в одном из двух состояний: "свободен" или "занят"
  - lock(m): "захват" мьютекса, если это возможно. Если мьютекс уже занят, поток, вызывающий операцию lock(), ждет, пока мьютекс освободится
  - unlock(m): "освобождение" мьютекса
- Операции над мьютексами в Win32
  - WaitForSingleObject(…) – захват
  - ReleaseMutex(…) – освобождение

"Читатели-Писатели" Решение № 2

```
TMutex m=1;

Writer(){
  while(true){
    PrepareData();
    WaitForSingleObject(m);
    Write(Data);
    ReleaseMutex(m);
  }
}
```

```
Reader(){
  while(true){
    WaitForSingleObject(m);
    Read(&Data);
    ReleaseMutex(m);
    UseData();
  }
}
```

Данное решение обеспечивает взаимное исключение, однако не позволяет одновременно получить доступ к данным нескольким потокам-читателям

"Читатели-Писатели" Решение № 3

```
TMutex m=1;
bool ReadCount=0;

Writer(){
  while(true){
5   PrepareData();
6   WaitForSingleObject(m);
7   Write(Data);
    ReleaseMutex(m);
  }
}
```

```
Reader(){
 while(true){
1   ReadCount ++;                2
    if( ReadCount == 1 )         3
      WaitForSingleObject(m);
    Read(&Data);                 4
    ReadCount --;
    if( ReadCount == 0 )
      ReleaseMutex(m);
    UseData();
 }
}
```

Предположим, выполняются два потока–читателя и один поток-писатель

После шага 7 одновременно находятся в критической секции поток-читатель (вызов Read(&Data)) и поток-писатель (вызов Write(Data)).

"Читатели-Писатели" Решение № 4

```
TMutex m=1;
bool ReadCount=0;

Writer(){
  while(true){
    PrepareData();
    WaitForSingleObject(m);
    Write(Data);
    ReleaseMutex(m);
  }
}
```

```
Reader(){
 while(true){
1   if( ReadCount == 0 )        2
5     WaitForSingleObject(m);   3
    ReadCount ++;               4
    Read(&Data);
    ReadCount --;
    if( ReadCount == 0 )
      ReleaseMutex(m);
    UseData();
 }
}
```

Предположим, выполняются два потока–читателя

После шага 5 правый поток-читатель выполняет чтение, но левый поток-читатель не может начать операцию, поскольку заблокирован на мьютексе. Отметим, что в данном решении выполняется условие взаимного исключения.

"Читатели-Писатели" Решение № 5

```
TMutex m1=1,m2=1;
bool ReadCount=0;

Writer(){
  while(true){
    PrepareData();
    WaitForSingleObject(m1);
    Write(Data);
    ReleaseMutex(m1);
  }
}
```

```
Reader(){
  while(true){
    WaitForSingleObject(m2);
    ReadCount ++;
    if( ReadCount == 1 )
      WaitForSingleObject(m1);
    ReleaseMutex(m2);
    Read(&Data);
    WaitForSingleObject(m2);
    ReadCount --;
    if( ReadCount == 0 )
      ReleaseMutex(m1);
    ReleaseMutex(m2);
    UseData();
  }
}
```

Это одно из правильных решений

"Читатели-Писатели" Решение № 6

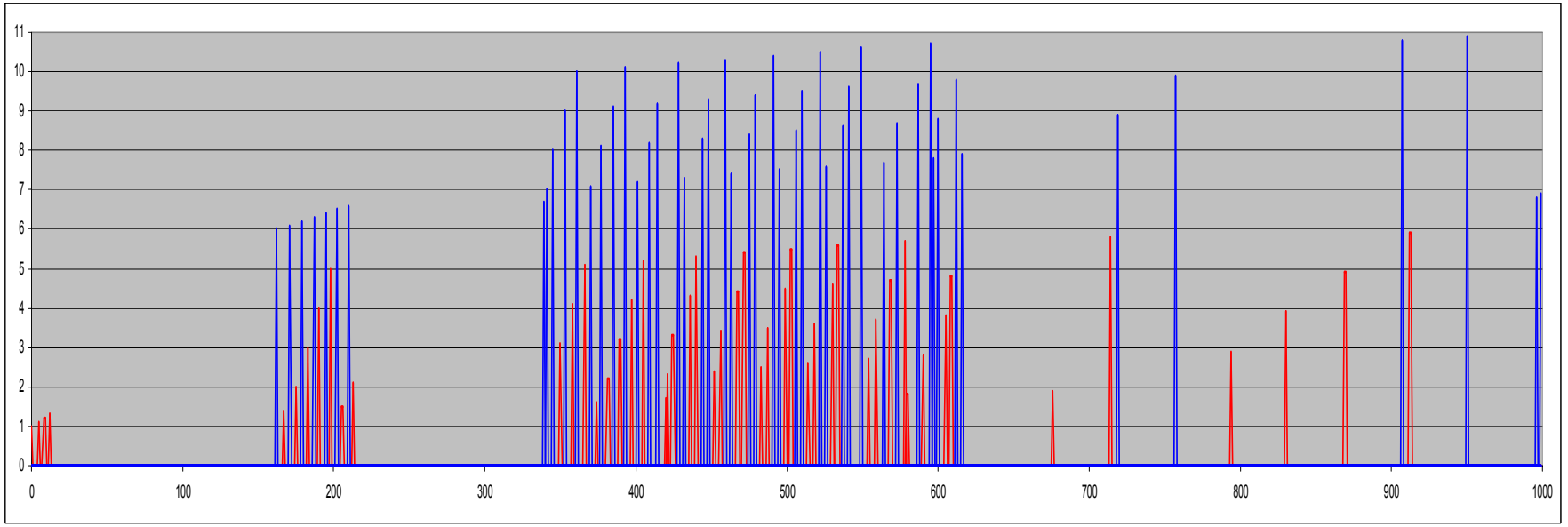
```
TMutex m1=1,m2=1, m3=1;
bool ReadCount=0;

Writer(){
  while(true){
    PrepareData();
    WaitForSingleObject(m3);
    WaitForSingleObject(m1);
    ReleaseMutex(m3);
    Write(Data);
    ReleaseMutex(m1);
  }
}
```

```
Reader(){
  while(true){
    WaitForSingleObject(m3);
    WaitForSingleObject(m2);
    ReleaseMutex(m3);
    ReadCount ++;
    if( ReadCount == 1 )
      WaitForSingleObject(m1);
    ReleaseMutex(m2);
    Read(&Data);
    WaitForSingleObject(m2);
    ReadCount --;
    if( ReadCount == 0 )
      ReleaseMutex(m1);
    ReleaseMutex(m2);
    UseData();
  }
}
```

Это еще одно правильное решение. В данном решении при появлении потока, желающего выполнить операцию записи, блокируется доступ к данным для новых потоков-читателей

"Читатели-Писатели" Диаграмма выполнения потоков

После выполнения работы в файле "Results-ReadersWriters.xls" вы должны получить диаграмму следующего вида



Красный горизонтальный участок на ненулевой высоте означает, что в данное время писатель находился в критической секции (целая часть высоты – номер писателя, дробная пропорциональна номеру операции), синий участок означает, что в это время в критической секции находился читатель

Результаты

- Для каждого решения – объяснение его правильности либо доказательство того, что оно ошибочно
- Диаграммы выполнения потоков для различных решений, иллюстрирующие верное или ошибочное поведение программы