



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»

## Отчёт по лабораторной работе №5

Название: Многопоточная реализация системы очередей

Дисциплина: Анализ алгоритмов

Студент

ИУ7-55Б

(Группа)

(Подпись, дата)

Хетагуров П.К

(И.О. Фамилия)

Преподаватель

(Подпись, дата)

Л.Л. Волкова

(И.О. Фамилия)

Москва, 2020

# Содержание

<b>Введение</b>	<b>3</b>
<b>1 Аналитическая часть</b>	<b>4</b>
1.1 Цель и задачи работы . . . . .	4
1.2 Организация взаимодействия потоков . . . . .	4
1.3 Вывод . . . . .	4
<b>2 Конструкторская часть</b>	<b>5</b>
2.1 Требования к ПО . . . . .	5
2.2 Описание системы . . . . .	5
2.3 Вывод . . . . .	5
<b>3 Технологическая часть</b>	<b>6</b>
3.1 Средства реализации . . . . .	6
3.2 Реализации алгоритмов . . . . .	6
3.3 Вывод . . . . .	10
<b>4 Экспериментальная часть</b>	<b>11</b>
4.1 Пример работы программы . . . . .	11
4.2 Вывод . . . . .	11
<b>Заключение</b>	<b>12</b>
<b>Список литературы</b>	<b>13</b>

## Введение

В данной лабораторной работе будет рассмотрена система конвейерной обработки.

# 1 Аналитическая часть

В данном разделе будут поставлены цели и задачи работы, будут рассмотрены основные теоретические сведения связанные с алгоритмами сортировки.

## 1.1 Цель и задачи работы

**Цель работы:** Научиться работать с потоками и разделяемыми ресурсами.

**Задачи работы:**

1. разработать систему из трех последовательных очередей, выполняющихся в отдельных потоках;
2. реализовать разработанную систему;
3. провести эксперимент, показывающий параллельное наступление событий.

## 1.2 Организация взаимодействия потоков

Так как потоки выполняются в общем адресном пространстве необходимо обеспечить корректное обращение к разделяемой памяти. Одним из способов достигнуть этого является мьютекс.

Мьютекс (от mutual exclusion — «взаимное исключение») — примитив синхронизации, обеспечивающий взаимное исключение исполнения критических участков кода [1].

## 1.3 Вывод

В данной части были поставлены задачи и цель работы, рассмотрено взаимодействие потоков, понятие мьютекса.

## 2 Конструкторская часть

В данном разделе будут рассмотрены схемы алгоритмов, требования к функциональности ПО.

### 2.1 Требования к ПО

ПО должно иметь один режим работы, в котором производится демонстрация работы. На вход системы подаются  $N$  элементов. В результате работы программы выводятся все события, произошедшие в системе, отсортированные в порядке наступления. Каждая очередь должна работать в своем потоке и завершать свою работу после обработки  $N$  элементов.

### 2.2 Описание системы

Система состоит из трех очередей, последовательно соединенных между собой. В первую очередь генератором записываются входные данные, а из последней очереди данные попадают в результирующий массив. Во время входа и выхода из очереди, ставится временная метка.

Между очередями находятся некоторые обработчики, обрабатывающие очередной элемент из предыдущей очереди и записывающие его в следующую. Причем время обработки элемента должно быть больше времени диспетчирезации.

### 2.3 Вывод

В данном разделе были рассмотрены реализуемое ПО и обозначены требования к нему.

## 3 Технологическая часть

Ниже будут представлены средства реализации и листинги реализованной программы.

### 3.1 Средства реализации

Выбранный язык программирования - Golang, так как он предоставляет удобные средства для параллелизма - goroutines [2]. Среда разработки - Visual Studio [3].

Технические характеристики машины, на которой проводились тесты:

- Windows 10 x64;
- 8 ГБ оперативной памяти;
- CPU: AMD FX(tm)-6350 Six-Core Processor 3.90GHz;
- 6 логических ядер.

### 3.2 Реализации алгоритмов

Ниже представлены листинги реализованной программы. На листинге [1] представлен файл main.go.

**Листинг 1** – main.go

```
1 package main
2
3 import (
4     "fmt"
5     "sync"
6     "time"
7
8     queue "aa/lab_05/queue"
9 )
10
11 const MAX_ELEMENT = 100
12
13 var wg sync.WaitGroup
14 var log []string
15 var mutexLog sync.Mutex
16
17 type SetTimestamp func(*queue.Element, int64)
18
19 func pingPlace(id int, message string, time int64) {
20     go func() {
21         mutexLog.Lock()
```

```

22     log = append(log, fmt.Sprintf("%d    %s    %d", id, message, time))
23     mutexLog.Unlock()
24 }()
25 }
26
27 func doStuff(milliseconds int) {
28     time.Sleep(time.Millisecond * time.Duration(milliseconds))
29 }
30
31 func startHandle(queueFirst *queue.Queue, milliseconds int) {
32     defer wg.Done()
33     for i := 0; i < MAX_ELEMENT; {
34         element := new(queue.Element)
35         element.ID = i
36         queueFirst.Mu.Lock()
37         queueFirst.Push(element)
38         queueFirst.Mu.Unlock()
39         timeNow := time.Now().UnixNano()
40         queue.SetFl(element, timeNow)
41         pingPlace(element.ID, fmt.Sprintf("  Insert %d  ", 1), timeNow)
42         doStuff(milliseconds)
43         i++
44     }
45 }
46
47 func handleQueue(firstQueue *queue.Queue, secondQueue *queue.Queue, stLeave
48     SetTimestamp, stIn SetTimestamp, queueNumber int, milliseconds int) {
49     defer wg.Done()
50     for i := 0; i < MAX_ELEMENT; {
51         firstQueue.Mu.Lock()
52         element := firstQueue.Pop()
53         firstQueue.Mu.Unlock()
54         if element != nil {
55             timeNow := time.Now().UnixNano()
56             stLeave(element, timeNow)
57             pingPlace(element.ID, fmt.Sprintf("  Leave %d  ", queueNumber), timeNow)
58             doStuff(milliseconds)
59             secondQueue.Mu.Lock()
60             secondQueue.Push(element)
61             secondQueue.Mu.Unlock()
62             timeNow = time.Now().UnixNano()

```

```

62     stln(element, timeNow)
63     pingPlace(element.ID, fmt.Sprintf("  Insert %d  ", queueNumber+1), timeNow)
64     i++
65 }
66 }
67 }
68
69 func main() {
70     firstQueue := queue.Queue{}
71     secondQueue := queue.Queue{}
72     thirdQueue := queue.Queue{}
73     answerQueue := queue.Queue{}
74
75     wg.Add(1)
76     go handleQueue(&firstQueue, &secondQueue, queue.SetFO, queue.SetSI, 1, 300)
77     wg.Add(1)
78     go handleQueue(&secondQueue, &thirdQueue, queue.SetSO, queue.SetTI, 2, 600)
79     wg.Add(1)
80     go handleQueue(&thirdQueue, &answerQueue, queue.SetTO, queue.SetA, 3, 300)
81     wg.Add(1)
82     go startHandle(&firstQueue, 100)
83
84     wg.Wait()
85
86     for i := 0; i < len(log); i++ {
87         fmt.Println(log[i])
88     }
89 }

```

На листинге [2] представлен файл queue.go.

#### Листинг 2 – queue.go

```

1 package queue
2
3 import (
4     "sync"
5 )
6
7 type Element struct {
8     ID                int
9     TimestampFirstIn  int64
10    TimestampFirstOut  int64

```



```

11     TimestampSecondIn    int64
12     TimestampSecondOut  int64
13     TimestampThirdIn    int64
14     TimestampThirdOut   int64
15     TimestampAnswer      int64
16 }
17
18 func SetFI(this *Element, timestamp int64) {
19     this.TimestampFirstIn = timestamp
20 }
21
22 func SetFO(this *Element, timestamp int64) {
23     this.TimestampFirstOut = timestamp
24 }
25 func SetSI(this *Element, timestamp int64) {
26     this.TimestampSecondIn = timestamp
27 }
28 func SetSO(this *Element, timestamp int64) {
29     this.TimestampSecondOut = timestamp
30 }
31 func SetTI(this *Element, timestamp int64) {
32     this.TimestampThirdIn = timestamp
33 }
34 func SetTO(this *Element, timestamp int64) {
35     this.TimestampThirdOut = timestamp
36 }
37 func SetA(this *Element, timestamp int64) {
38     this.TimestampAnswer = timestamp
39 }
40
41 type Queue struct {
42     Mu      sync.Mutex
43     Array  []*Element
44 }
45
46 func (this *Queue) Push(element *Element) {
47     this.Array = append(this.Array, element)
48 }
49
50 func (this *Queue) Pop() (element *Element) {
51     if len(this.Array) > 0 {

```

```
52     element = this.Array[0]
53     this.Array = this.Array[1:]
54 }
55 return
56 }
```

### 3.3 Вывод

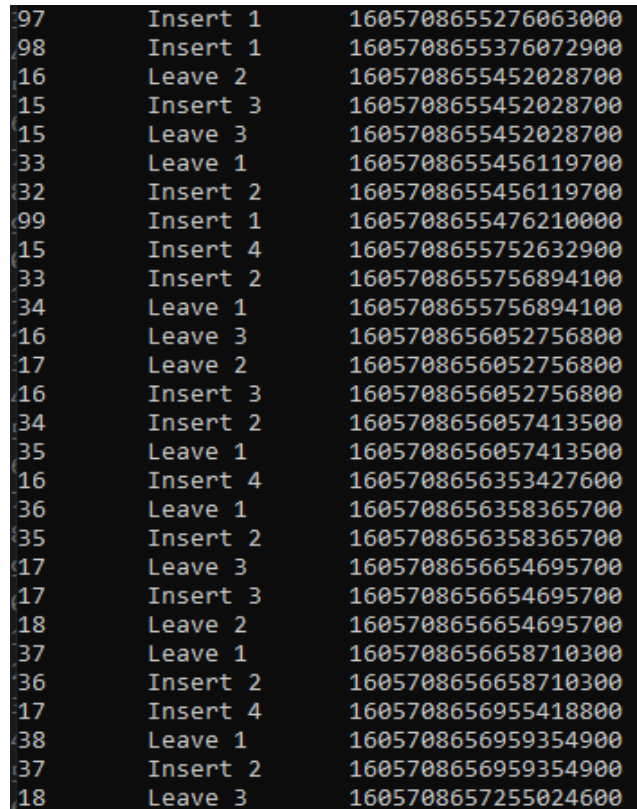
В данном разделе были описаны программные и аппаратные средства реализации, были представлены листинги программы.

## 4 Экспериментальная часть

В данной главе будет представлен пример работы программы и продемонстрировано параллельное наступление событий.

### 4.1 Пример работы программы

Пример работы программы представлен на рисунке [1]



97	Insert 1	1605708655276063000
98	Insert 1	1605708655376072900
16	Leave 2	1605708655452028700
15	Insert 3	1605708655452028700
15	Leave 3	1605708655452028700
33	Leave 1	1605708655456119700
32	Insert 2	1605708655456119700
99	Insert 1	1605708655476210000
15	Insert 4	1605708655752632900
33	Insert 2	1605708655756894100
34	Leave 1	1605708655756894100
16	Leave 3	1605708656052756800
17	Leave 2	1605708656052756800
16	Insert 3	1605708656052756800
34	Insert 2	1605708656057413500
35	Leave 1	1605708656057413500
16	Insert 4	1605708656353427600
36	Leave 1	1605708656358365700
35	Insert 2	1605708656358365700
17	Leave 3	1605708656654695700
17	Insert 3	1605708656654695700
18	Leave 2	1605708656654695700
37	Leave 1	1605708656658710300
36	Insert 2	1605708656658710300
17	Insert 4	1605708656955418800
38	Leave 1	1605708656959354900
37	Insert 2	1605708656959354900
18	Leave 3	1605708657255024600

Рисунок 1 – Пример работы программы

### 4.2 Вывод

Как видно из вывода программы, события выполнялись параллельно.

## Заключение

В данной лабораторной работе была разработана система конвейеров, работающих с разделяемыми ресурсами, она была реализована, а также было показано параллельность выполнения конвейеров. Цель работы достигнута, все задачи выполнены.

## Список литературы

- [1] Mutex. Wikipedia [Электронный ресурс]. Режим доступа: (дата обращения - 18.11.2020) Свободный. URL: [https://en.wikipedia.org/wiki/Lock\\_\(computer\\_science\)](https://en.wikipedia.org/wiki/Lock_(computer_science))
- [2] Golang [Электронный ресурс]. Режим доступа: (дата обращения - 18.11.2020) Свободный. URL: <https://golang.org/>
- [3] Visual Studio [Электронный ресурс]. Режим доступа: (дата обращения - 18.11.2020) Свободный. URL: <https://visualstudio.microsoft.com/ru/>