

# Теория и практика многопоточного программирования

## Семинар 8

Неганов Алексей

Московский физико-технический институт (национальный исследовательский университет)  
Кафедра теоретической и прикладной информатики

Москва 2020

# Задача производителя-потребителя

```
package main
import "fmt"

var done = make(chan bool)
var msgs = make(chan int)

func produce () {
    for i := 0; i < 10; i++ {
        msgs <- i
    }
    done <- true
}

func consume () {
    for {
        msg := <-msgs
        fmt.Println(msg)
    }
}

func main () {
    go produce()
    go consume()
    <- done
}
```

# Задача производителя-потребителя

```
int itemCount = 0;

void producer(void) {
    while (1) {
        item = produceItem();
        if (itemCount == BUFFER_SIZE)
            sleep();

        putItemIntoBuffer(item);
        itemCount = itemCount + 1;

        if (itemCount == 1)
            wakeup(consumer);
    }
}
```

```
void consumer(void) {
    while (1) {
        if (itemCount == 0)
            sleep();

        item = removeItemFromBuffer();
        itemCount = itemCount - 1;

        if (itemCount == BUFFER_SIZE - 1)
            wakeup(producer);

        consumeItem(item);
    }
}
```

Почему такой код не является корректным?

```
# ALGOL up
# POSIX sem_post
function V(semaphore S, integer I):
    [S ← S + I]

# ALGOL down
# POSIX sem_wait
function P(semaphore S, integer I):
    repeat:
        [if S < I:
            S ← S - I
            break]
```

Операции *P* и *V* имеет право делать **любой** поток, в отличие от mutex.

# Задача производителя-потребителя: семафоры

```
sem_t fill, empty;
sem_init(&fill, 0, 0);
sem_init(&empty, 0, BUFFER_SIZE);

void producer(void) {
    while (1) {
        item = produceItem();
        sem_wait(&empty);
        putItemIntoBuffer(item);
        sem_post(&fill);
    }
}
```

```
procedure consumer() {
    while (1) {
        sem_wait(&fill);
        item = removeItemFromBuffer();
        sem_post(&empty);
        consumeItem(item);
    }
}
```

## Задача производителя-потребителя: семафоры

```
pthread_mutex_t lock;  
sem_t fill, empty;  
sem_init(&fill, 0, 0);  
sem_init(&empty, 0, BUFFER_SIZE);  
  
void producer(void) {  
    while (1) {  
        item = produceItem();  
        sem_wait(&empty);  
        pthread_mutex_lock(&lock);  
        putItemIntoBuffer(item);  
        pthread_mutex_unlock(&lock);  
        sem_post(&fill);  
    }  
}
```

```
procedure consumer() {  
    while (1) {  
        sem_wait(&fill);  
        pthread_mutex_lock(&lock);  
        item = removeItemFromBuffer();  
        pthread_mutex_unlock(&lock);  
        sem_post(&empty);  
        consumeItem(item);  
    }  
}
```

## Задача читателя-писателя: приоритет читателя

```
semaphore resource=1;  
semaphore reentry=1;
```

```
int rcount=0;
```

```
writer() {  
    P(resource);  
    // write  
    V(resource)  
}
```

```
reader() {  
    P(reentry);  
    readcount++;  
    if (++rcount == 1)  
        P(resource);  
    v(reentry)  
  
    // read  
  
    P(reentry)  
    if (--rcount == 0)  
        V(resource);  
    V(reentry);  
}
```

См. pthread\_rwlock\_t

## Задача читателя-писателя: приоритет писателя

```
int rcount = 0, wcount = 0;
semaphore reentry = 1, wentry = 1;
semaphore readTry = 1, resource = 1;

writer() {
    P(wentry);
    if (++wcount == 1)
        P(readTry);
    V(wentry);

    P(resource);
    // write
    V(resource);

    P(wentry);
    if (--wcount == 0)
        V(readTry);
    V(wentry);
}

reader() {
    P(readTry);
    P(reentry);
    if (++readcount == 1)
        P(resource);
    V(reentry);
    V(readTry);

    // read

    P(reentry);
    if (--readcount == 0)
        V(resource);
    V(reentry)
}
```



## Задача читателя-писателя: честное решение

```
int rcount = 0;
semaphore reentry = 1, resource = 1;
semaphore queue = 1;

writer() {
    P(queue);
    P(resource);
    V(queue);

    // write

    V(resource);
}
```

```
reader() {
    P(queue);
    P(reentry);
    if (++readcount == 1)
        P(resource);
    V(reentry);
    V(queue);

    // read

    P(reentry);
    if (--readcount == 0)
        V(resource);
    V(reentry);
}
```

- 1 Реализовать RW lock с помощью семафоров POSIX
- 2 Реализовать RW lock с помощью атомарных примитивов
- 3 Реализовать передачу содержимого файла из одного процесса в другой через разделяемый буфер, используя семафоры. Требование: смерть любого процесса должна корректно обработаться.