

# Лекция 9

- Критические области и состязательная ситуация.
- Спин-блокировка и алгоритм Петерсона.
- Синхронизация средствами POSIX Threads.

# Спин-блокировка

```
#include <stdio.h>
#include <pthread.h>
#include <unistd.h>

int sh=0;
//int flag[2]={1,1};
```

```
int turn=0;
```

```
void* my_thread0( ){  
    int i=0;  
    for(;i<100;i++){  
        while(turn); //spinlock  
        sh++; //критическая область  
        turn=1;  
        usleep(1); //некритическая область  
    }  
    //flag[0]=0;  
}
```

```
void* my_thread1(){  
    int i=0;  
    for(;i<100;i++){  
        while(!turn);  
        sh+=2;  
        turn=0;  
        usleep(100);  
    }  
    //flag[1]=0;  
}
```

```
int main() {  
    pthread_t th_id[2];  
  
    pthread_create(&th_id[0], NULL, &my_thread0, NULL);  
    pthread_create(&th_id[1], NULL, &my_thread1, NULL);  
  
    //while(flag[0] || flag[1]);  
    pthread_join(th_id[0], NULL);  
    pthread_join(th_id[1], NULL);  
  
    printf("%i\n", sh);  
  
    return 0;  
}
```

```
~/Lecture9> ./lab9
300
~/Lecture9> ./lab9
298
~/Lecture9> ./lab9
299
~/Lecture9> ./lab9
293
~/Lecture9> ./lab9
300
```

```
~/Lecture9> ./lab9
300
~/Lecture9> ./lab9
300
~/Lecture9> ./lab9
300
~/Lecture9> ./lab9
300
.....
~/Lecture9> ./lab9
300
```

# Реализация спин-блокировки POSIX Threads

```
#include <stdio.h>
#include <pthread.h>
#include <unistd.h>
int sh=0;

//int turn=0;
pthread_spinlock_t lock;
```

```
void*  my_thread0 ( ) {  
    int i=0;  
    for (;i<100;i++) {  
        //while (turn);  
        pthread_spin_lock(&lock);  
        sh++;  
        //turn=1;  
        pthread_spin_unlock(&lock);  
        usleep(1);  
    }  
}
```

```
void*  my_thread1 () {  
    int i=0;  
    for (;i<100;i++) {  
        //while(!turn);  
        pthread_spin_lock(&lock) ;  
        sh+=2;  
        //turn=0;  
        pthread_spin_unlock(&lock) ;  
        usleep(1) ;  
    }  
}
```



```
int main() {  
    pthread_t th_id[2];  
  
    pthread_spin_init(&lock, PTHREAD_PROCESS_PRIVATE);  
  
    pthread_create(&th_id[0], NULL, &my_thread0, NULL);  
    pthread_create(&th_id[1], NULL, &my_thread1, NULL);  
  
    pthread_join(th_id[0], NULL);  
    pthread_join(th_id[1], NULL);  
  
    printf("%i\n", sh);  
    return 0;  
}
```

# Алгоритм Петерсона

```
int turn=0, flagReady[2]={1,1};

void* thread0( ){
    int i=0;
    for(; i < 100; i++){
        flagReady[0]=1;
        turn=1;
        while(turn==1 && flagReady[1]);
        sh++; //критическая область
        flagReady[0]=0;
        usleep(1); //некритическая
    }
}
```

```
void* thread1(){
    int i=0;
    for(;i<100;i++){
        flagReady[1]=1;
        turn=0;
        while(turn==0 && flagReady[0]);
        sh+=2; //критическая область
        flagReady[1]=0;
        //некритическая область:
        usleep(1); //sleep(1);
    }
}
```

```
bool readyFlags[2];
```

```
int turn;
```

```
void EnterCriticalRegion(int threadId){
```

```
    readyFlags [threadId] = true; // Флаг текущего потока
```

```
    turn = 1 - threadId; // Флаг очереди исполнения
```

```
    while (turn == 1-threadId && readyFlags [1-threadId]);
```

```
}
```

```
void LeaveCriticalRegion(int threadId)
```

```
{
```

```
    readyFlags[threadId] = false; // Сброс флага текущего потока
```

```
}
```

```
void* thread0( ){  
    int i=0;  
    for(; i < 100; i++){  
        EnterCriticalRegion(0);  
        sh++; //критическая область  
        LeaveCriticalRegion(0);  
        usleep(1); //некритическая область  
    }  
}
```

```
void* thread1(){  
    int i=0;  
    for(;i<100;i++){  
        EnterCriticalRegion(1);  
        sh+=2; //критическая область  
        LeaveCriticalRegion(1);  
        usleep(1); //sleep(1); //некритическая область  
    }  
}
```

## Фрагменты кода к лабораторной №9

```
char sh[6];  
void* Thread( void* pParams );  
  
int main( void ){  
    pthread_t thread_id;  
    pthread_create(&thread_id, NULL, &Thread, NULL);  
    .....  
    while( 1 ) printf(“%s\n”, sh);  
    .....  
}
```

```
void* Thread( void* pParams ){  
    int counter = 0;  
    while ( 1 ){  
        if(counter%2){  
            sh[0]='H';sh[1]='e';sh[2]='l';sh[3]='l';sh[4]='o';sh[5]='\0';  
        }  
        else{  
            sh[0]='B';sh[1]='y';sh[2]='e';sh[3]='_';sh[4]='u';sh[5]='\0';  
        }  
        counter++;  
    }  
    return NULL;  
}
```