## Федеральное агентство связи

## Сибирский государственный университет телекоммуникаций и информатики

Лабораторная работа №9

Выполнил: студент группы ИП-211

Оганесян Альберт

Лацук Андрей

Проверил:

Профессор кафедры ПМиК

Малков Е. А.

Новосибирск 2024

**Задание:** протестируйте спин-блокировку используя фрагменты кода лекции 9. **Дополнительное задание:** протестировать мьютекс блокировку используя фрагменты кода лекции 9. Сравнить время с спин-блокировкой. **Цель:** знакомство с синхронизацией потоков.

1. Используем фрагменты кода из лекции 9, чтобы написать программу:

```
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>
#include <signal.h>
#include <string.h>
#include <unistd.h>
volatile int running = 1;
char sh[6];
void *Thread(void *pParams);
void handle sigint(int sig)
    running = 0;
int main(void)
    pthread t thread id;
    signal(SIGINT, handle sigint);
    pthread create (&thread id, NULL, &Thread, NULL);
    while (running)
        printf("%s", sh);
        fflush(stdout);
    pthread cancel(thread id);
    pthread join(thread id, NULL);
    return 0;
void *Thread(void *pParams)
```

```
{
    int counter = 0;
    while (running)
    {
        if (counter % 2)
        {
            strcpy(sh, "Hello\n");
        }
        else
        {
            strcpy(sh, "Bye_u\n");
        }
        counter++;
    }
    return NULL;
}
```

**2.** Запустим программу (Рис. 2.1). Основная проблема в том, что потоки работают одновременно, из-за чего основной поток по несколько раз выводит то же значение **sh**.

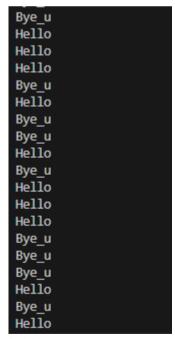


Рис. 2.1 Часть вывода программы

**3.** Теперь добавим в код спин-блокировку и алгоритм Питерсона, чтобы программа выводила обновленное значение только после измерения и добавим измерения времени:

```
#include <stdio.h>
#include <stdlib.h>
```

```
#define MAX COUNT 100000
volatile int running = 1;
char sh[6];
pthread spinlock t spinlock;
void *Thread(void *pParams);
void handle sigint(int sig)
  running = 0;
int main(void)
  pthread t thread id;
  pthread spin init(&spinlock, PTHREAD PROCESS PRIVATE);
  signal(SIGINT, handle_sigint);
  pthread create(&thread id, NULL, &Thread, NULL);
  while (running)
    pthread spin lock(&spinlock);
    if (turn == 0)
      printf("%s", sh);
      fflush(stdout);
      turn = 1;
```

```
pthread spin unlock(&spinlock);
       pthread cancel(thread id);
       pthread join(thread id, NULL);
       pthread spin destroy(&spinlock);
       real end time = times(&end time);
       double user time = (double) (end time.tms utime -
start time.tms utime) / sysconf( SC CLK TCK);
       double system time = (double) (end time.tms stime -
start time.tms stime) / sysconf( SC CLK TCK);
       printf("User CPU time: %.6f seconds\n", user time);
       printf("System CPU time: %.6f seconds\n", system time);
       return 0;
     void *Thread(void *pParams)
       int counter = 0;
       while (counter < MAX COUNT)</pre>
         pthread spin lock(&spinlock);
           if (counter % 2)
             strcpy(sh, "Hello\n");
             strcpy(sh, "Bye_u\n");
           counter++;
         pthread spin unlock(&spinlock);
```

```
}
running = 0;
return NULL;
}
```

## **4.**Сделаем то же самое с Mutex блокировкой:

```
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>
#include <signal.h>
#include <string.h>
#include <unistd.h>
#include <sys/times.h>
#define MAX COUNT 100000
volatile int running = 1;
int turn = 1;
char sh[6];
pthread mutex t mutex;
void *Thread(void *pParams);
void handle sigint(int sig)
  running = 0;
int main(void)
  pthread mutex init(&mutex, NULL);
  signal(SIGINT, handle sigint);
  pthread create(&thread id, NULL, &Thread, NULL);
```

```
while (running)
    pthread mutex lock(&mutex);
    if (turn == 0)
     printf("%s", sh);
     fflush(stdout);
      turn = 1;
    pthread mutex unlock(&mutex);
  pthread cancel(thread id);
  pthread join(thread id, NULL);
  pthread mutex destroy(&mutex);
  real end time = times(&end time);
  double user time = (double) (end time.tms utime -
start time.tms utime) / sysconf( SC CLK TCK);
 double system time = (double) (end time.tms stime -
start time.tms stime) / sysconf( SC CLK TCK);
  printf("User CPU time: %.6f seconds\n", user time);
  printf("System CPU time: %.6f seconds\n", system time);
  return 0;
void *Thread(void *pParams)
  int counter = 0;
  while (counter < MAX COUNT)</pre>
    pthread mutex lock(&mutex);
        strcpy(sh, "Hello\n");
```

```
    else
    {
        strcpy(sh, "Bye_u\n");
    }
    counter++;
    turn = 0;
}
    pthread_mutex_unlock(&mutex);
}
running = 0;
return NULL;
}
```

**5.** Запустим обе программы на значение counter до 100000 сравним время Mutex:

```
Hello
Bye_u
Hello
Bye_u
Hello
Bye_u
Hello
Bye_u
Hello
User CPU time: 0.950000 seconds
System CPU time: 2.420000 seconds
```

Spin-lock:

```
Bye_u
Hello
User CPU time: 2.240000 seconds
System CPU time: 0.380000 seconds
```

**Вывод:** Мы научились работать с синхронизацией потоков и сравнили время выполнения mutex и spin-lock. Mutex затрачивает больше системного времени, а Spin-lock пользовательского.