Министерство цифрового развития

Сибирский Государственный Университет Телекоммуникаций и

Информатики

СибГУТИ

Кафедра прикладной математики и кибернетики

Лабораторная работа №3

по дисциплине: Операционные системы

Выполнил: Гринченко А. В.

Группа: ЗП-022

Номер зачетки: 73210117

Проверил: проф. Малков Е. А.

Новосибирск, 2023

# Оглавление

[Оглавление 2](#_Toc1)

[Постановка задачи 3](#_Toc2)

[Инструментарий 4](#_Toc3)

[Разработка 5](#_Toc4)

[Вычисление суммы на основе последовательного кода 5](#_Toc5)

[Вычисление суммы на основе интерфейса Pthreads 7](#_Toc6)

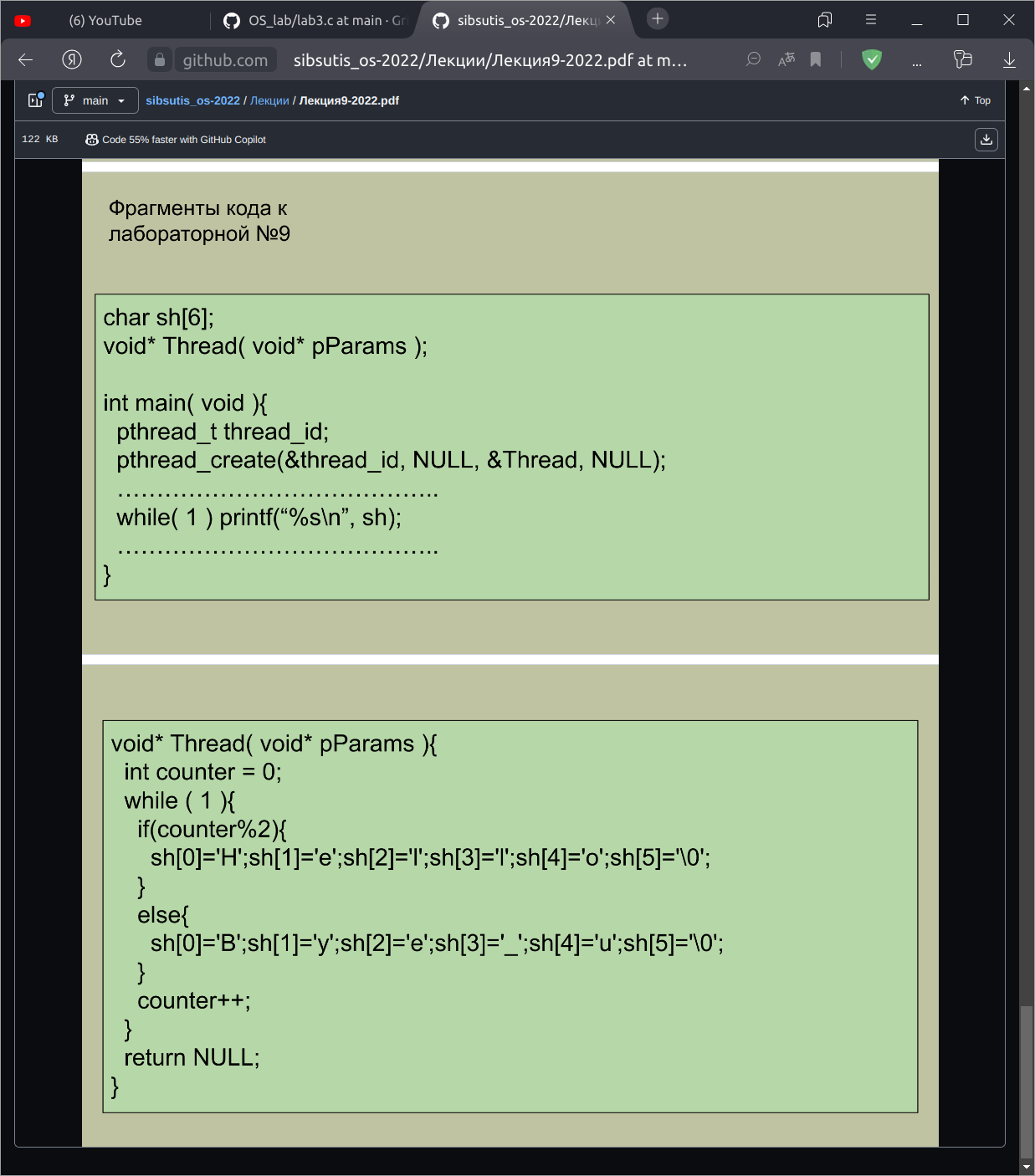
[Вычисление суммы на основе интерфейса C++ <thread> 10](#_Toc7)

[Заключение 12](#_Toc8)

# **Постановка задачи**

**Цель работы:** знакомство с синхронизацией потоков.

**Задание:** Протестировать спин-блокировку, используя фрагменты когда лекции 9:



# Инструментарий

В качестве инструментов для разработки приложения использованы текстовый редактор, компиляторы gcc , терминал для команд сборки и запуска исполняемого файла.

# Разработка

Листинг программы до того, как в нее была добавлена спин-блокировка:

#include <stdio.h>

#include <unistd.h>

#include <pthread.h>

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';

}

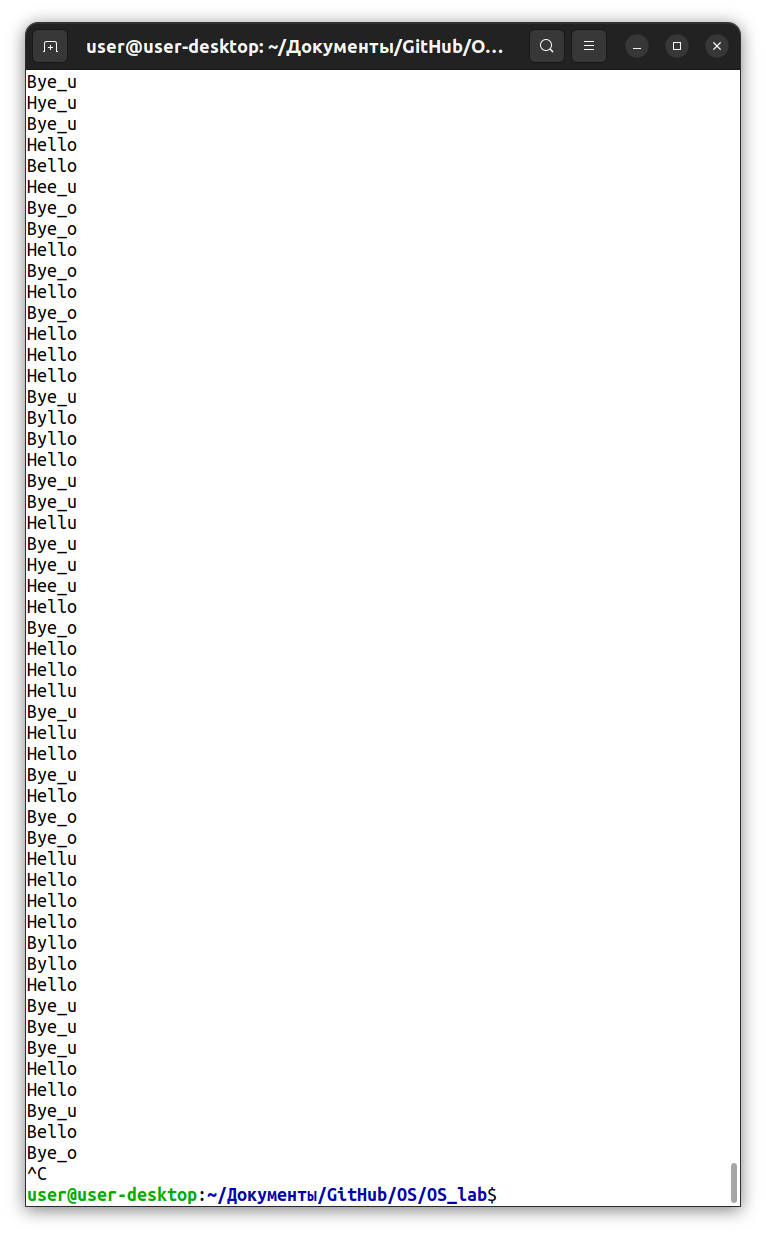
counter++;

}

return NULL;

}

Результат запуска программы:



Как видно из скриншота, вывод массива местами осуществлялся до полного его изменения, что приводило к тому, что две разные строки - «Hello» и «Bye\_u» - смешивались между собой.

Для того, чтобы поток не захватывал ресурс, который находится в процессе изменения, была введена спин-блокировка:

Листинг программы:

#include <stdio.h>

#include <unistd.h>

#include <pthread.h>

char sh[6];

**pthread\_spinlock\_t lock;**

void\* Thread(void\* pParams);

int main(void)

{

pthread\_t thread\_id;

**pthread\_spin\_init(&lock, PTHREAD\_PROCESS\_PRIVATE);**

pthread\_create(&thread\_id, NULL, &Thread, NULL);

while(1) { printf("%s\n", sh); }

}

void\* Thread(void\* pParams)

{

int counter = 0;

while(1)

{

**pthread\_spin\_lock(&lock);**

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';

}

**pthread\_spin\_unlock(&lock);**

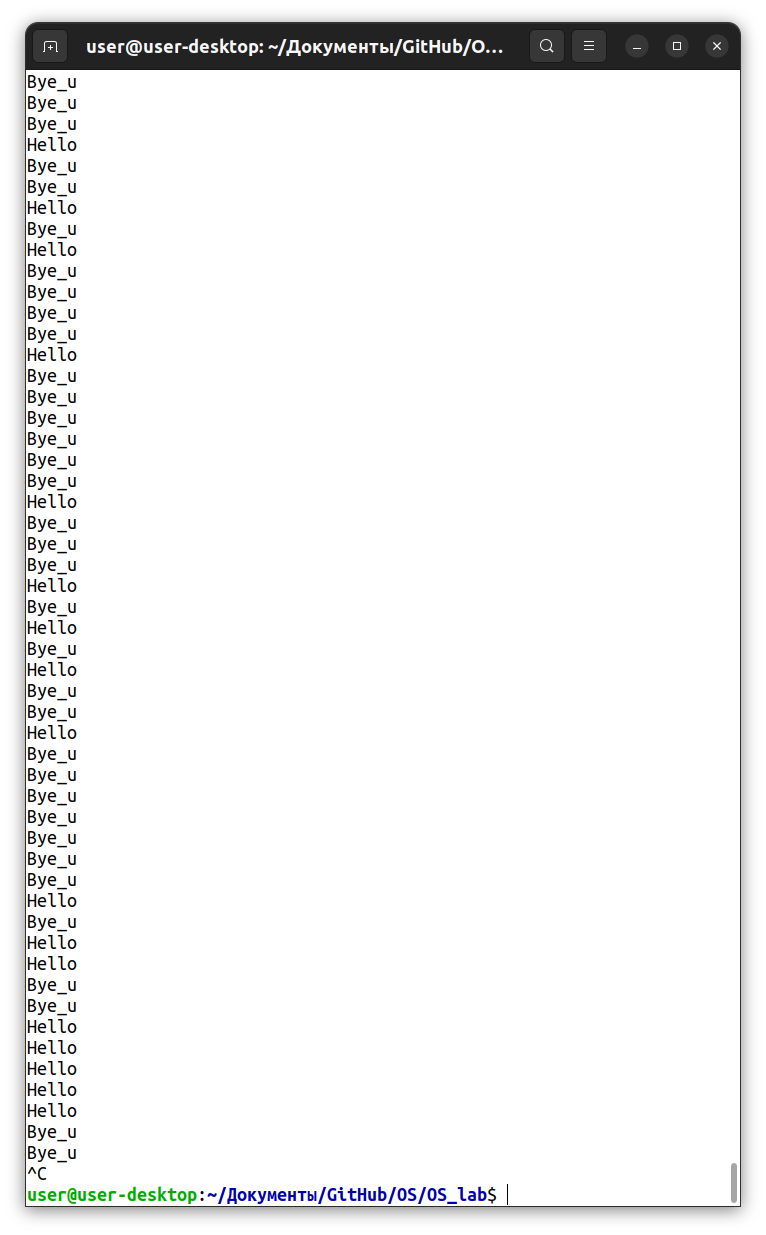
counter++;

}

return NULL;

}

Теперь ресурс заблокирован для потока в момент перестройки массива, в результате чего программа работает по-другому:



В результате выполнения программы на дисплей выводится полностью измененные строчки без присутствия в них элементов другой строки. Поток при этом не блокируется, поэтому строчки чередуются не через одну, а через некоторое количество дубликатов.

# Заключение

В процессе выполнения лабораторной работы было осуществлено знакомство с синхронизацией потоков в Linux.

В результате выполнения лабораторной работы была протестирована спин-блокировка с использованием фрагментов кода из лекции 9. Был сделан вывод, что спин-блокировка позволяет ограничить доступ потока к изменяемому ресурсу на время критически важных для этого ресурса изменений, не останавливая при этом сам поток.