Министерство цифрового развития

Сибирский Государственный Университет Телекоммуникаций и

Информатики

СибГУТИ

Кафедра прикладной математики и кибернетики

Лабораторная работа №2

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 программными интерфейсами управления потоками Linux.

**Задание:** программно реализовать вычисление суммы последовательности чисел на основе последовательного кода, интерфейсов Pthreads и C++ <thread>. Сравнить время вычислений.

# Инструментарий

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

# Разработка

## Вычисление суммы на основе последовательного кода

Листинг программы:

#include <stdio.h>

#include <time.h>

int main()

{

double time\_spent = 0.0;

const unsigned int first = 100000000;

const unsigned int last = 399999999;

long long unsigned int result = 0;

clock\_t begin = clock();

for(int i = first; i <= last; i++)

{

result = result + i;

}

clock\_t end = clock();

time\_spent += (double)(end - begin)/CLOCKS\_PER\_SEC;

printf("Result: %llu\nTime spent: %f sec\n\n", result, time\_spent);

return 0;

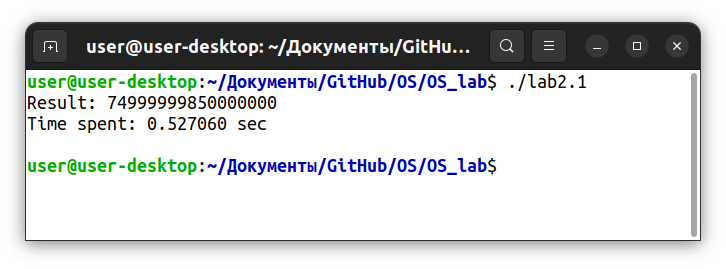
}

Переменная **double time\_spent** хранит время, затраченное для расчета суммы. В качестве начального и конечного значений взяты числа 100 000 000 и 399 999 999 соответственно. Рассчитанная сумма хранится в переменной **long long unsigned int result**.

Перед началом вычислений с помощью функции **clock()**, которая возвращает количество временных тактов, прошедших с начала запуска программы. Переменные типа **clock\_t begin** и **end** хранят значения, возвращаемые clock() непосредственно перед началом расчетов и сразу после них. Их разность, разделенная на константу **CLOCKS\_PER\_SEC**, даст время, прошедшее с момента начала расчетов и до их окончания.

Вычисление суммы реализовано последовательным суммированием чисел.

Результат работы программы:



В результате выполнения программы на дисплей выводится сумма и время, потраченное на ее вычисление.

## Вычисление суммы на основе интерфейса Pthreads

Листинг программы:

#include <stdio.h>

#include <stdlib.h>

#include <pthread.h>

#include <unistd.h>

#include <time.h>

double time\_spent = 0.0;

const unsigned int first = 100000000;

const unsigned int last = 399999999;

long long unsigned int result = 0;

unsigned int parts[4] = {0, 1, 2, 3};

pthread\_t threads[4]; //массив потоков

void\* sum(void\* me)

{

int part = \*((int\*)me);

int offset = 75000000;

long long unsigned int res = 0;

for (int i = first+part\*75000000; i <= last-(3-part)\*offset; i++)

{

res = res + i;

}

result = result + res;

}

int main()

{

pthread\_attr\_t attrs; //атрибуты потока

if (pthread\_attr\_init(&attrs) != 0)

{

perror("Cannot initialize attributes");

abort();

}

pthread\_attr\_setdetachstate(&attrs, PTHREAD\_CREATE\_JOINABLE);

clock\_t begin = clock();

for (int i = 0; i < 4; i++)

{

if (pthread\_create(&threads[i], &attrs, sum, &parts[i]) != 0)

{

perror("Cannot create a thread");

abort();

}

}

pthread\_attr\_destroy(&attrs);

for (int i = 0; i < 4; i++)

{

if (pthread\_join(threads[i], NULL) != 0)

{

perror("Cannot join a thread");

abort();

}

}

clock\_t end = clock();

time\_spent += (double)(end - begin)/CLOCKS\_PER\_SEC;

printf("Result: %llu\nTime spent: %f sec\n\n", result, time\_spent);

return 0;

}

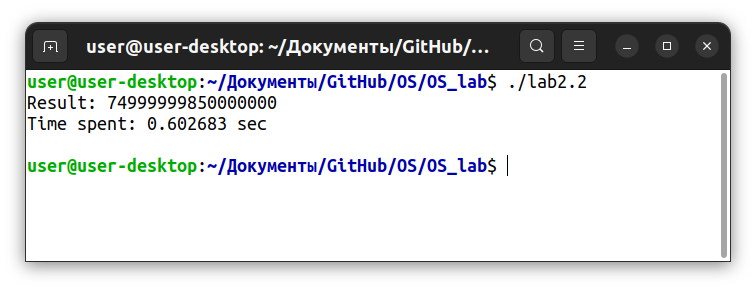
Функция **int main()** создает 4 потока, которые параллельно выполняют суммирование, разделяя диапазон суммируемых значений на 4 части. Для этого: вводятся и инициализируются атрибуты потока – поток создается как ожидаемый, т.е. создающий его поток будет ожидать его завершения (вычисления суммы всего выделенного потоку диапазона).

Потоковая функция **void\* sum(void\*)** выполняет суммирование чисел для каждого выделенного потока в соответствующем диапазоне. Она принимает через pthread\_create аргумент, который переводит в целое число **part**, используемое для расчетов диапазона вычислений. Переменная **int offset** также используется для вычисления этого диапазона. **int res** – промежуточная переменная, используется для вычисления частичной суммы (итоговой суммы отдельного потока).

Поскольку здесь не нужно защищать итоговую переменную от потоков, которые по очереди изменяют ее (прибавляют вычисленное значение из своего диапазона), не были введены мьютексы.

Время, потраченное на вычисления, рассчитывается так же, как и в программе на основе последовательного кода. Общий диапазон вычислений также не был изменен.

Результат работы программы:



В результате выполнения программы на дисплей выводится сумма и время, потраченное на ее вычисление. В случае использования интерфейса Pthreads время, затраченное на вычисление суммы, выше, чем в случае использования последовательного кода.

## Вычисление суммы на основе интерфейса C++ <thread>

Листинг программы:

#include <iostream>

#include <thread>

#include <ctime>

double time\_spent = 0.0;

const unsigned int first = 100000000; //100 000 000

const unsigned int last = 399999999; //399 999 999

long long unsigned int result = 0;

void sum(int me)

{

int part = me;

int offset = 75000000;

long long unsigned int res = 0;

for (unsigned int i = first+part\*75000000; i <= last-(3-part)\*offset; i++)

{

res = res + i;

}

result = result + res;

}

int main()

{

std:: thread threads[4];

clock\_t begin = clock();

for (int i = 0; i < 4; i++)

{

threads[i] = std::thread(sum, i);

}

for (int i = 0; i < 4; i++)

{

threads[i].join();

}

clock\_t end = clock();

time\_spent += (double)(end - begin)/CLOCKS\_PER\_SEC;

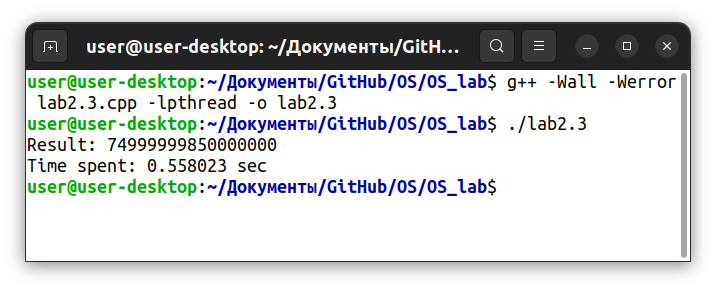
std::cout << "Result: " << result << "\nTime spent: " << time\_spent << " sec\n";

return 0;

}

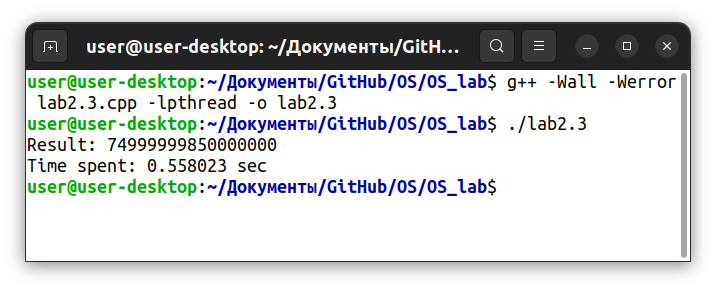
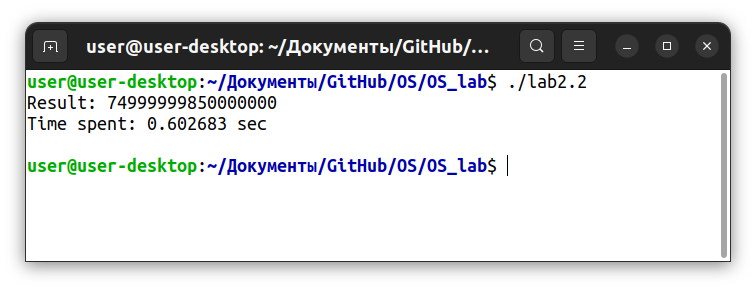
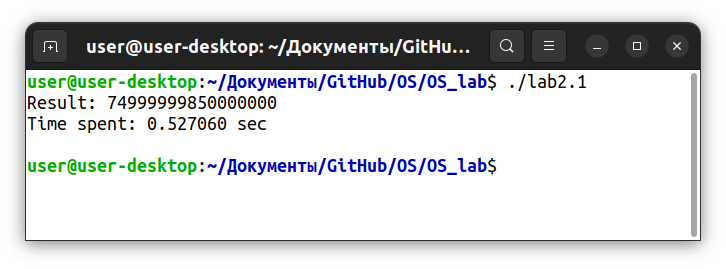
Программа представляет собой аналог предыдущей программы на языке C++.

Результат работы программы:



В результате выполнения программы на дисплей выводится сумма и время, потраченное на ее вычисление. В случае использования интерфейса C++ <thread> время, затраченное на вычисление суммы, выше, чем в случае использования последовательного кода, но ниже, чем в случае использования интерфейса Pthread.

Результаты работы всех трех программ:



# Заключение

В процессе выполнения лабораторной работы было осуществлено знакомство с программными интерфейсами управления потоками в Linux.

В результате выполнения лабораторной работы были разработаны приложения, вычисляющие суммы последовательности чисел на основе последовательного кода, интерфейсов Pthreads и C++ <thread>. Были сделаны выводы о том, что время вычислений, потраченное во всех трех случаях на суммирование последовательности чисел, разное. В данном случае быстрее всего вычисление суммы произошло в случае последовательного кода, среди многопоточных реализаций быстрее всего работает программа на C++.