Министерство цифрового развития, связи и массовых коммуникаций Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Сибирский государственный университет телекоммуникаций и информатики»

(СибГУТИ)

Кафедра ПМиК

Расчетно-графическое задание

по дисциплине

«Операционные системы реального времени»

Выполнил: студент 4 курса

ИВТ, гр. ИП-813

Бурдуковский И.А.

Проверил: Преподователь кафедры ПМиК

Белевцова Екатерина Андреевна

.

Новосибирск 2021

Оглавление

[Задание 3](#_Toc59733303)

[Выполнение 4](#_Toc59733304)

[Вывод 14](#_Toc59733305)

# Задание

1. Сравните время запуска (создания) нити и время активизации с помощью семафора заранее созданной нити.

2. Функции типа fwrite(), работающие через структуру FILE, используют внутреннюю буферизацию данных. Анализируя время выполнения функций, определите размер буфера

# Выполнение

1. Для выполнения первого задания была реализована программа с использованием функций pthread\_create, sem\_init, sem\_wait, sem\_post, для отслеживания времени был выбран ClockCycles.

int pthread\_create( pthread\_t\* thread, const pthread\_attr\_t\* attr, void\* (\*start\_routine)(void\* ), void\* arg) –создаёт нить на которой выполняется указанная при создании функция ;

int sem\_init( sem\_t \* sem, int pshared, unsigned value ) – инициализирует семафор для возможности синхронизации нитей с указанным значением value;

int sem\_post (sem\_t\* sem) – увеличивает семафор на 1 и выводит одну нить из ожидания, если находится в очереди;

int sem\_wait (sem\_t\* sem) – уменьшает семафор на 1. Однако: если семафор = 0, то нить блокируется до тех пор, пока кто-то не увеличит его. Если несколько нитей вызывают sem\_wait на нулевом семафоре, то они ставятся в очередь, упорядоченную по приоритетам.

int ClockCycles() – функция из библиотеки sys/neutrino.h, возвращает текущее значение 64-битного счетчика циклов процессора. Чтобы вычислить количество прошедших секунд необходимо сначала вычислить количество циклов в секунду у системы с помощью SYSPAGE\_ENTRY(qtime)->cycles\_per\_sec и уже поделить наши циклы на это количество.

Замер был произведен на создание нити с пустой функцией void \*testThread(void \*args). Также были созданы нити с функцией void \*semaphoreThread(void \*args), которая сразу при создании вставала в очередь ожидания с помощью sem\_wait().

Второй замер был произведён при отработке функции sem\_post(), которая выводила по очереди все нити из ожидания.

Как показали результаты замеров и сравнения времени: активизация с помощью семафора заранее созданной нити оказалась быстрее чем создание новой нити.

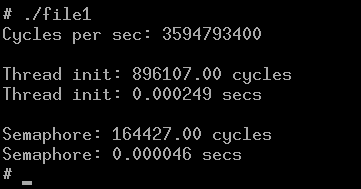
В зависимости от количества повторений создания и активизаций нитей увеличивалась точность и стабильность замеров между повторными запусками программ.

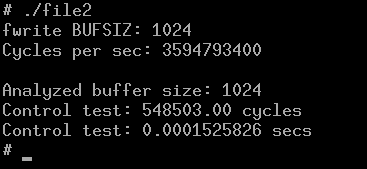
1. Для выполнения второго задания в программе также использовалась функция ClockCycles() для замера времени работы и функция fwrite.

size\_t fwrite(const void \*buf, size\_t size, size\_t count, FILE \*stream) – функция записывает count объектов (каждый объект по size символов в длину) в поток - в нашем случае это указатель на открытый файл. В функции имеется внутренний буфер, в который первоначально передаются записываемые символы. Это позволяет сократить количество обращений к файлу и соответственно затрачиваемое на это время. Как только буфер заполняется – срабатывает прерывание, во время которого всё содержимое буфера сохраняется в файл.

В программе циклом производится замер времени записи одного символа char в файл. В тот момент, когда буфер полностью заполнится – сработает прерывание с записью, количество замеренного времени будет гораздо больше того, которое затрачивается для простой записи символа в буфер. В таком случае нам станет известен размер буфера, равный номеру итерации в цикле.

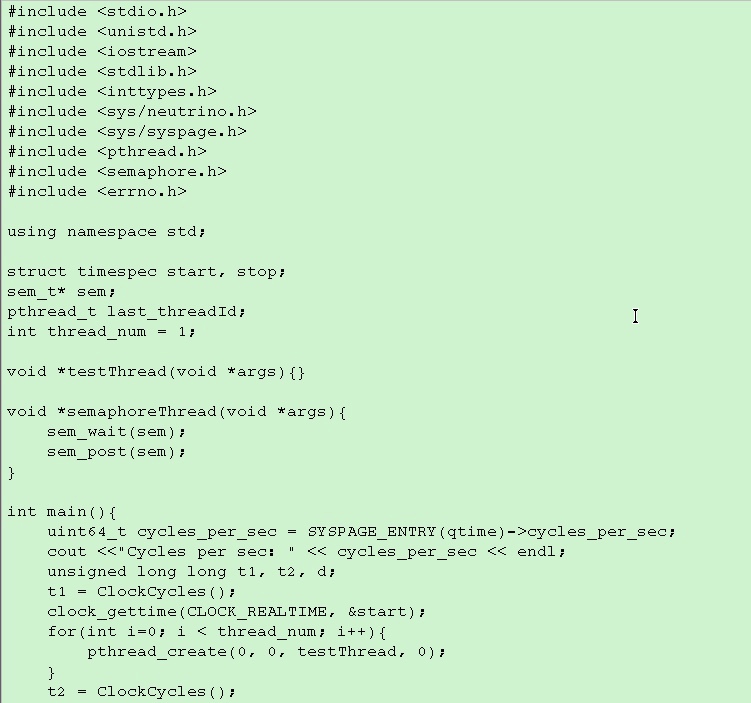
# Результаты работы программ

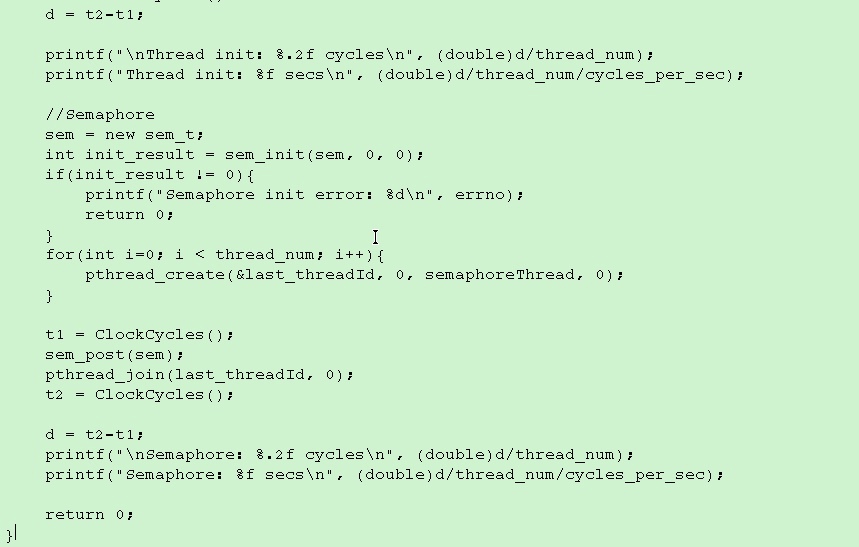




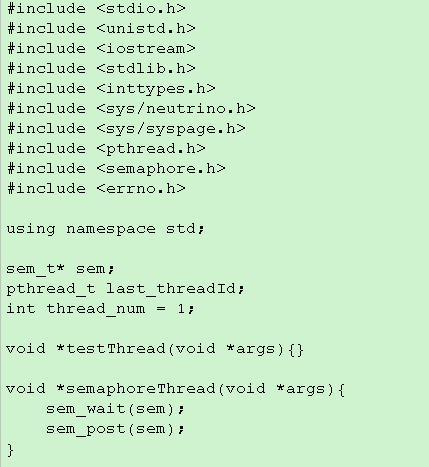
# Листинг

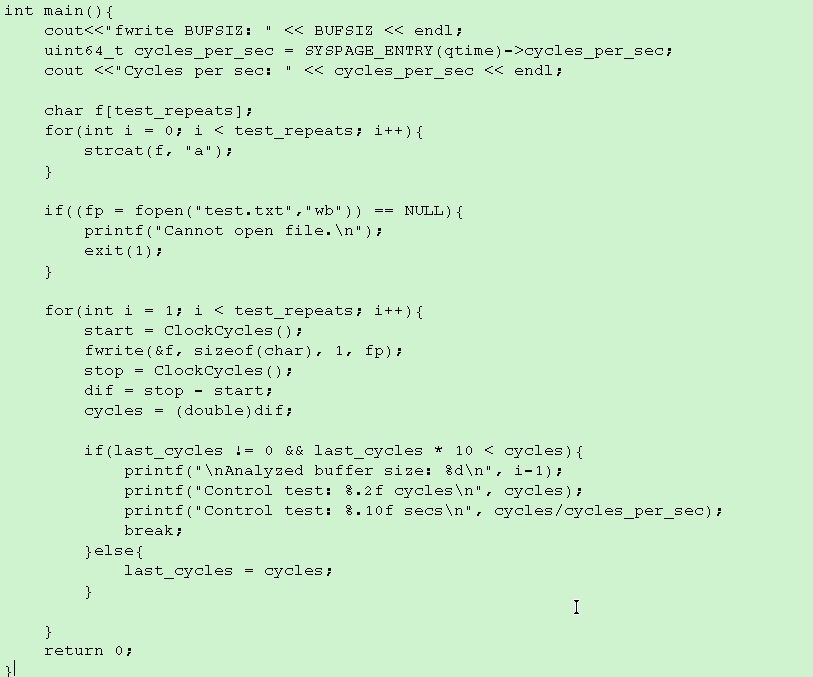
**File1.cpp**





**File2.cpp**





# Вывод

В рамках данного курса я узнал о операционных системах реального времени и научился работать в одной из них, QNX, узнал о принципе системы, чем она отличается от других систем.

В процессе выполнения расчетно-графического задания я углубил свои теоретические знания об ОСРВ, в частности о QNX. Больше узнал об алгоритме планирования и распределения ресурсов системы. Узнал о том, как измерить время работы части программы в QNX.

А также благодаря практической реализации подтвердил известную мне теорию, касающуюся предложенных 2ух расчетно-графических заданий.