Федеральное государственное бюджетное образовательное учреждение высшего образования
«Сибирский государственный университет телекоммуникаций и информатики»
(СибГУТИ)

Институт информатики и вычислительной техники

09.03.01 "Информатика и вычислительная техника"
профиль "Программное обеспечение средств
вычислительной техники и автоматизированных систем"

Кафедра прикладной математики и кибернетики

Операционные системы реального времени

Расчётно-графическая работа

Выполнил:

студент гр.ИП-213 Дмитриев Антон Александрович
ФИО студента

«__» _____ 2025 г.

Проверил:

Преподаватель Шевелькова Валерия Юрьевна
ФИО преподавателя

«__» _____ 2025 г. Оценка_____

Новосибирск 2025 г.

1. Задание

1. Продумайте методику и измерьте время

контекстного переключения между нитями в одном процессе.

2. Нить может динамически менять свой приоритет. Будет ли выполняться

перепланирование нитей, если нить просто подтверждает свой приоритет?

2. Объяснение идеи выполнения задания

Задание 1: Измерение времени контекстного переключения

Основная идея

Программа измеряет время, необходимое для передачи управления между двумя нитями одного процесса. Это время включает:

- Пробуждение планировщика
- Сохранение контекста текущей нити
- Восстановление контекста другой нити
- Передачу управления между нитями

Механизм работы

Синхронизация через флаг volatile int flag;

volatile предотвращает оптимизацию компилятора

Обе нити используют флаг для координации работы

Алгоритм одной итерации:

1. Главная нить сбрасывает флаг: flag = 0
2. Главная нить начинает измерение: start = ClockCycles()
3. Главная нить устанавливает флаг: flag = 1 (сигнал второй нити)
4. Главная нить ждет сброса флага: while (flag == 1)
5. Вторая нить обнаруживает установку флага: while (flag == 0)
6. Вторая нить сбрасывает флаг: flag = 0
7. Главная нить завершает измерение: end = ClockCycles()

Что именно измеряется

Главная нить: flag=1 → (контекстное переключение) → Вторая нить: flag=0

Измеряется время от момента установки флага до момента его сброса, которое включает два контекстных переключения:

- Главная → Вторая нить
- Вторая → Главная нить

Задание 2: Проверка перепланирования при установке приоритета

Основная идея

Программа проверяет, вызывает ли вызов pthread_setschedparam() с теми же параметрами приоритета перепланирование нитей.

Механизм работы

Главная нить

↓

Мониторинговая нить (monitoring_thread)

↓ создает

Тестовая нить (high_priority_thread)

Последовательность выполнения:

1. Мониторинговая нить ждет установки ready = 1
2. Мониторинговая нить создает тестовую нить и начинает измерение
3. Тестовая нить:
    - Получает текущие параметры планирования
    - Устанавливает ready = 1 (сигнал мониторинговой нити)
    - Вызывает pthread_setschedparam() с теми же параметрами
    - Устанавливает detection_flag = 1
    - Мониторинговая нить завершает измерение при detection_flag = 1

4. В тестовой нити:
    - pthread_getschedparam(pthread_self(), &policy, &param) Получаем параметры
    - pthread_setschedparam(pthread_self(), policy, &param) Устанавливаем ТЕ ЖЕ параметры

Что проверяется

- Если вызов pthread_setschedparam() вызывает перепланирование, то время измерения будет больше
- Если перепланирования не происходит, время будет минимальным

3. Программный код

context_switch.cpp:

```
#include <stdio.h>
#include <pthread.h>
#include <sys/neutrino.h>
#include <unistd.h>
#include <stdint.h>
#include <semaphore.h>

#define ITERATIONS 100000

sem_t sem1, sem2;

void* thread_func(void* arg) {
    int i;
    for (i = 0; i < ITERATIONS; i++) {
        sem_wait(&sem2);
        sem_post(&sem1);
    }
    return NULL;
}

int main() {
    pthread_t thread;
    uint64_t start, end;
    int i;
    uint64_t total_cycles = 0;

    // Инициализируем семафоры
    sem_init(&sem1, 0, 0);
```

```
    sem_init(&sem2, 0, 0);

    pthread_create(&thread, NULL, thread_func, NULL);

    for (i = 0; i < ITERATIONS; i++) {
        start = ClockCycles();
        sem_post(&sem2);
        sem_wait(&sem1);
        end = ClockCycles();
        total_cycles += (end - start);
    }

    pthread_join(thread, NULL);

    sem_destroy(&sem1);
    sem_destroy(&sem2);

    // Мы измерили время туда и обратно, поэтому делим на 2
    double avg_cycles = (double)total_cycles / (2 * ITERATIONS);
    printf("Среднее время контекстного переключения: %.2f циклов\n", avg_cycles);
    printf("За %d итераций\n", ITERATIONS);

    return 0;
}
```

priority_test.cpp:

```
#include <stdio.h>
#include <pthread.h>
#include <sched.h>
#include <sys/neutrino.h>
#include <unistd.h>

volatile int counter1 = 0;
volatile int counter2 = 0;
volatile int running = 1;

void* worker_thread(void* arg) {
    while (running) {
        counter1++;
    }
    return NULL;
}

void* priority_test_thread(void* arg) {
    struct sched_param param;
    int policy;

    pthread_getschedparam(pthread_self(), &policy, &param);

    while (running) {
        // Устанавливаем тот же приоритет
        pthread_setschedparam(pthread_self(), policy, &param);
        counter2++;
    }
    return NULL;
```

```
}

int main() {
    pthread_t worker, tester;

    printf("Test of redeployment when setting the same priority\n");

    pthread_create(&worker, NULL, worker_thread, NULL);
    pthread_create(&tester, NULL, priority_test_thread, NULL);

    sleep(1); // Работаем 1 секунду

    running = 0;

    pthread_join(worker, NULL);
    pthread_join(tester, NULL);

    printf("Working thread: %d iterations\n", counter1);
    printf("Test thread: %d iterations\n", counter2);

    if (counter2 > counter1 * 0.1) {
        printf("OUTPUT: Rescheduling occurs when the same priority is set\n");
    } else {
        printf("OUTPUT: Rescheduling does NOT occur when the same priority is set\n");
    }

    return 0;
}
```

4. Результаты

```
# on -C 0 ./context_switch
Average switch time: 1006.10 cycles
On 100000 iterations
```

```
# on -C 0 ./priority_test
Test of redeployment when setting the same priority
Working thread: 1419702887 iterations
Test thread: 253 iterations
OUTPUT: Rescheduling does NOT occur when the same priority is set
#
```