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

Расчётно-графическая работа

по дисциплине «Операционные системы реального времени»

Выполнил: студент группы ИП-111

Маландий И.И.

Проверил: профессор кафедры ПМиК

Фионов А.Н

Новосибирск 2024

Содержание

1. Постановка задачи 3
2. Постановка эксперимента 4
3. Листинг 6
4. Результат работы 9

1. Постановка задачи

1. Определите время пересылки пустого сообщения между нитями в рамках одного процесса и между нитями, принадлежащими разными процессам.

2. Определите среднюю неточность задержки и диапазон изменения неточности при использовании функции delay().

.

2. Постановка эксперимента

1. **Определение времени передачи пустого ответа между нитями в рамках одного процесса и между нитями, принадлежащими разным процессам.**

Для выполнения замера времени передачи пустого ответа между нитями использовалась функция ClockCycles(), которая позволяет получить текущее значение счетчика тактов процессора. Этот метод обеспечивает высокую точность измерения времени.

Для передачи сообщений между нитями были использованы функции MsgReceive() и MsgSend(), которые предоставляют механизм межпоточного взаимодействия в QNX. Для обмена информацией между процессами, принадлежащими разным процессам, была использована функция mmap(), которая выделяет общую память для двух процессов.

В рамках одного процесса:

- Создаются две нити: серверная и клиентская.
- Серверная нить создает канал с помощью ChannelCreate() и ожидает сообщения от клиентской нити с помощью MsgReceive().
- Клиентская нить подключается к каналу с помощью ConnectAttach() и отправляет пустое сообщение с помощью MsgSend().
- После получения сообщения серверная нить отправляет ответ с помощью MsgReply() и засекает время с помощью ClockCycles().
- Клиентская нить также засекает время перед отправкой сообщения.
- Разница между временем на серверной и клиентской нитях используется для вычисления времени передачи сообщения.

Между нитями, принадлежащими разным процессам:

- Создается дочерний процесс с помощью fork().
- В дочернем процессе создается серверная нить, которая создает канал и ожидает сообщения.
- В родительском процессе создается клиентская нить, которая подключается к каналу дочернего процесса и отправляет пустое сообщение.
- Для обмена данными между процессами используется общая память, выделенная с помощью mmap().
- После получения ответа серверная нить в дочернем процессе засекает время, а клиентская нить в родительском процессе также засекает время.
- Разница между временем на серверной и клиентской нитях используется для вычисления времени передачи сообщения.

В процессе выполнения сначала выделяется память, затем при помощи функции fork() создается дочерний процесс, который далее создает нить и ждет сообщения от родительского процесса. В свою очередь родительский процесс дает время дочернему процессу время и запускает нить, где отправляет пустое сообщение дочернему процессу. В ответ на сообщение, дочерний отправляет ответ и засекает время. Для максимально быстрой передачи сообщения снижается приоритет дочерней нити, что позволяет игнорировать процессы завершения нити. После получения ответа родительский процесс также засекает время и нить завершает работу. После этого вычисляется время, и дочерний процесс завершает свою работу. Для замера времени внутри одного процесса, родительский процесс продолжает работу и выполняет вышеперечисленные действия на внутренних нитях.

2. Определение средней неточности задержки и диапазона изменения неточности при использовании функции delay().

Для выполнения замера неточности задержки использовалась функция delay(), которая обеспечивает задержку на заданное количество миллисекунд. Для измерения фактического времени задержки использовалась функция ClockCycles().

- В цикле выполняется 1000 итераций, в каждой из которых засекается время перед вызовом delay(1) и после него.
- Разница между этими временами дает фактическую задержку в тактах процессора.
- После завершения цикла вычисляется средняя задержка, минимальная и максимальная задержка.
- Результаты переводятся в секунды для удобства интерпретации.

3. Листинг

Задание 1.

Определите время пересылки пустого сообщения между нитями в рамках одного процесса и между нитями, принадлежащими разными процессам.
В рамках одного процесса:

```
#include <stdio.h>
#include <unistd.h>
#include <pthread.h>
#include <iostream>
#include <sys/mman.h>
#include <sys/neutrino.h>
#include <time.h>
#include <stdlib.h>
#include <sys/syspage.h>
#include <limits>

using namespace std;

int idServer, idClient;
unsigned int time1, time2;

void *Server(void *arg) {
    idServer = ChannelCreate(0);
    int rcvid = MsgReceive(idServer, NULL, NULL, NULL);
    time2 = ClockCycles();
    cout << "Clock2\n";
    MsgReply(rcvid, 0, 0, 0);
}

void *Client(void *arg) {
    idClient = ConnectAttach(0, 0, idServer, 0, 0);
    cout << "Clock1\n";
    time1 = ClockCycles();
    MsgSend(idClient, NULL, NULL, NULL, NULL);
}

typedef numeric_limits<double> dbl;

int main() {
```

```
    pthread_t id, id2;
    pthread_create(&id, NULL, Server, NULL);
    sleep(1);  // добавляем задержку, чтобы сервер успел начать
    pthread_create(&id2, NULL, Client, NULL);
    pthread_join(id, NULL);
    pthread_join(id2, NULL);

    cout << "time1: " << time1 << "\n";
    cout << "time2: " << time2 << "\n";
    cout << "Result (nsec): " << ((time2 - time1) /
    (double)SYSPAGE_ENTRY(qtime)->cycles_per_sec) * 1000000000 << "\n";
    cout << "Result (sec): " << ((time2 - time1) /
    (double)SYSPAGE_ENTRY(qtime)->cycles_per_sec) << "\n";
    return 0;
}
```

Между нитями, принадлежащими разными процессам:

```
#include <stdio.h>
#include <unistd.h>
#include <pthread.h>
#include <iostream>
#include <sys/mman.h>
#include <sys/neutrino.h>
#include <time.h>
#include <stdlib.h>
#include <sys/syspage.h>
#include <limits>

using namespace std;

int *idClient, *idServer;
unsigned int *time1, *time2;
int *pid;

void *Server(void *arg) {
    *pid = getpid();
    *idServer = ChannelCreate(0);
    int rcvid = MsgReceive(*idServer, NULL, NULL, NULL);
    *time2 = ClockCycles();
    cout << "Clock2\n";
    MsgReply(rcvid, 0, 0, 0);
}

void *Client(void *arg) {
    *idClient = ConnectAttach(0, *pid, *idServer, 0, 0);
    cout << "Clock1\n";
```

```
    *time1 = ClockCycles();
    MsgSend(*idClient, NULL, NULL, NULL, NULL);
}

typedef numeric_limits<double> dbl;

int main() {
    time1 = (unsigned int*)mmap(NULL, sizeof *time1, PROT_READ |
    PROT_WRITE, MAP_SHARED | MAP_ANONYMOUS, -1, 0);
    time2 = (unsigned int*)mmap(NULL, sizeof *time2, PROT_READ |
    PROT_WRITE, MAP_SHARED | MAP_ANONYMOUS, -1, 0);
    idServer = (int*)mmap(NULL, sizeof *idServer, PROT_READ | PROT_WRITE,
    MAP_SHARED | MAP_ANONYMOUS, -1, 0);
    idClient = (int*)mmap(NULL, sizeof *idClient, PROT_READ | PROT_WRITE,
    MAP_SHARED | MAP_ANONYMOUS, -1, 0);
    pid = (int*)mmap(NULL, sizeof *pid, PROT_READ | PROT_WRITE, MAP_SHARED
    | MAP_ANONYMOUS, -1, 0);

    int pid2 = fork();
    if (!pid2) {  // В серверном процессе
        pthread_t id;
        pthread_create(&id, NULL, Server, NULL);
        pthread_join(id, NULL);
    } else {  // В клиентском процессе
        pthread_t id2;
        sleep(4);  // небольшая задержка для старта сервера
        pthread_create(&id2, NULL, Client, NULL);
        pthread_join(id2, NULL);
        sleep(1);  // задержка перед выводом результата
        cout << "time1: " << *time1 << "\n";
        cout << "time2: " << *time2 << "\n";
        cout << "Result (nsec): " << ((*time2 - *time1) /
       (double)SYSPAGE_ENTRY(qtime)->cycles_per_sec) * 1000000000 << "\n";
        cout << "Result (sec): " << ((*time2 - *time1) /
       (double)SYSPAGE_ENTRY(qtime)->cycles_per_sec) << "\n";
    }

    return 0;
}
```

Задание 2

```
#include <stdio.h>
#include <time.h>
#include <unistd.h>
#include <iostream>
#include <sys/syspage.h>
#include <sys/neutrino.h>
#include <limits>

using namespace std;

#define NUM_ITERATIONS 1000

inline long long getClockCycles() {
    return ClockCycles();
}

void test_delay_accuracy() {
    long long total_delay = 0;
    long long min_delay = LLONG_MAX;
    long long max_delay = LLONG_MIN;

    for (int i = 0; i < NUM_ITERATIONS; ++i) {
        long long start = getClockCycles();
        delay(1);  // Задержка на 1 миллисекунду
        long long end = getClockCycles();

        long long delay_cycles = end - start;
        total_delay += delay_cycles;

        if (delay_cycles < min_delay) {
            min_delay = delay_cycles;
        }
        if (delay_cycles > max_delay) {
            max_delay = delay_cycles;
        }
    }

    double average_delay = (double)total_delay / NUM_ITERATIONS;
    cout << "Average delay (cycles): " << average_delay << endl;
    cout << "Min delay (cycles): " << min_delay << endl;
    cout << "Max delay (cycles): " << max_delay << endl;

    // Перевод в секунды
    double average_delay_sec = average_delay /
    (double)SYSPAGE_ENTRY(qtime)->cycles_per_sec;
```

```
    cout << "Average delay (sec): " << average_delay_sec << endl;
}

int main() {
    test_delay_accuracy();
    return 0;
}
```

4. Результат работы

```
# ./rgz1_1
Clock1
Clock2
time1: 2111464130
time2: 2111468614
Result (nsec): 1181.72
Result (sec): 1.18172e-06
#
```

```
# ./rgz1_2
Clock2
Clock1
time1: 2117432158
time2: 2117438238
Result (nsec): 1602.34
Result (sec): 1.60234e-06
#
```

Рисунок 1-2. Время пересылки пустого сообщения между нитями внутри одного и двух процессов.

```
./rgz2
Average delay (cycles): 1.1591e+07
Min delay (cycles): 3015451
Max delay (cycles): 16930368
Average delay (sec): 0.00305473
#
```

Рисунок 3. Средняя неточность задержки и диапазон изменения неточности при использовании функции delay().