Национальный исследовательский ядерный университет «МИФИ»
(Московский Инженерно–Физический Институт)
Кафедра №42 «Криптология и кибербезопасность»

Отчёт
по результатам выполнения
Лабораторной работы №2
«Компьютер и время»

Дисциплина:	Практические Аспекты Разработки Высокопроизводительного Программного Обеспечения (ПАРВПО)
Студент:	Гареев Рустам Рашитович
Группа:	Б22-505
Преподаватель:	Куприяшин Михаил Андреевич
Дата:	01.04.2025

Москва - 2025

Оглавление

Технологический стек

memory	8GiB Системная память
processor	11th Gen Intel(R) Core(TM) i5-1135G7 @ 2.40GHz
siblings	8
cpu cores	4
bridge	11th Gen Core Processor Host Bridge/DRAM Registers
display	TigerLake-LP GT2 [Iris Xe Graphics]
gcc	version 13.3.0
OC	Ubuntu 24.04.2 LTS
IDE	Visual Studio Code 1.98.2

Ответы на вопросы

1. Определите, какие структуры данных применяются для хранения меток времени. Рассмотрите только современные структуры. Для Си это struct timespec, для Си++ придётся поискать;

В современных ЯП для хранения меток времени используются структуры struct timespec(C, time.h), std::chrono::time_point(C++) и std::chrono::duration(C++). Также в C++20 появилась дополнительная структура данных std::chrono::hh_mm_ss.

2. Определите, каким образом часы реального времени (RTC) представлены в системной файловой системе Linux и как с ними можно работать;

В файловой системе Linux часы реального времени (RTC) представлены через несколько интерфейсов, которые позволяют читать и записывать время, а также настраивать будильники. Для взаимодействия с RTC используется файл /dev/rtc0 /dev/rtc (который часто является ссылкой на /dev/rtc0). Эти файлы можно открыть в режиме чтения для получения информации о времени или для настройки будильников. Для чтения и записи времени RTC можно использовать команду hwclock:

```
sudo hwclock  # чтение текущего времени RTC

sudo hwclock --set --date="YYYY-MM-DD HH:MM:SS"  # установка времени RTC
```

3. Определите, какие типы часов предоставляет операционная система. Для Си это CLOCK_REALTIME и пр. Для Си++ придётся поискать;

Типы часов в Linux для C:

1. CLOCK_REALTIME;
2. CLOCK_MONOTONIC;
3. CLOCK_MONOTONIC_RAW (Linux-specific);
4. CLOCK_PROCESS_CPUTIME_ID;
5. CLOCK_THREAD_CPUTIME_ID.

В C++ эти типы часов доступны через системные вызовы POSIX, но также есть собственные типы часов, определенные в библиотеке <chrono>:

1. std::chrono::system_clock(аналогичен CLOCK_REALTIME);

2. std::chrono::steady_clock(CLOCK_MONOTONIC);

3. std::chrono::high_resolution_clock(предоставляет наивысшую доступную точность времени)

4. Выясните, чем отличаются эти типы часов;

REALTIME может идти назад при коррекции времени;

MONOTONIC всегда увеличивается, но может корректироваться по скорости;

MONOTONIC_RAW - чистое аппаратное время без коррекций.

5. Когда будет следующая високосная секунда?

Следующая високосная секунда пока не назначена. Последняя была 31 декабря 2016 года.

6. Выясните, как добавить выполнение инструкции `RDTSC` в код на Си или Си++ (техника называется "[CPU] intrinsics");

Чтобы добавить выполнение инструкции RDTSC в код на Си или Си++, можно использовать встроенные функции или встроенный ассемблер.

Встроенные функции(intrinsics):

```
#include <x86intrin.h>
uint64_t rdtsc() {
    return __rdtsc();
}
```

Встроенный ASM:

```
#include <stdio.h>
int main() {
   unsigned long long ticks;
   asm volatile("rdtsc" : "=A" (ticks));
   printf("%llu ticks\n", ticks);
   return 0;
}
```

7. Выясните, как определить коэффициент пересчёта тактов `RDTSC` в секунды.

Коэффициент можно определить через замер количества тактов за определенный интервал времени, используя системный таймер. Частота процессора равна количеству тактов, прошедших за единицу времени. Пример кода:

```
struct timespec start, end;
uint64_t tsc1 = __rdtsc();
clock_gettime(CLOCK_MONOTONIC, &start);
sleep(1);
clock_gettime(CLOCK_MONOTONIC, &end);
uint64_t tsc2 = __rdtsc();
double ns = (end.tv_sec - start.tv_sec)*1e9 +
                    (end.tv_nsec - start.tv_nsec);
double tsc_per_ns = (tsc2 - tsc1) / ns;
```

Эксперимент

Таблица 1 — Результаты замера времени

Имя таймера	Точность(по системному вызову), нс	Разрешение (экспериментально), нс	Погрешность разрешения(экспериментально), нс	Время инициализации(A1), нс	Время возврата(A2), нс
CLOCK_REALTIME	1.0	36.1	50.4	19.5	18.6
CLICK_MOMOTONIC	1.0	37.3	35.2	18.8	17.8
CLOCK_MOMOTONIC_RAW	1.0	37.3	23.7	17.9	16.9
CLOCK_PROCESS_CPUTIME_ID	1.0	389.5	122.6	192.3	193.0
system_clock	1.0	21.5	24.4	21.5	21.5
steady_clock	1.0	21.1	28.5	21.1	20.7
high_resolution_clock	1.0	20.7	29.3	20.7	20.3
RTC	1000000.0	996592861.8	21564967.7	509488.1	502107.4
RDTSC	0.4	7.4	13.8	7.4	7.4

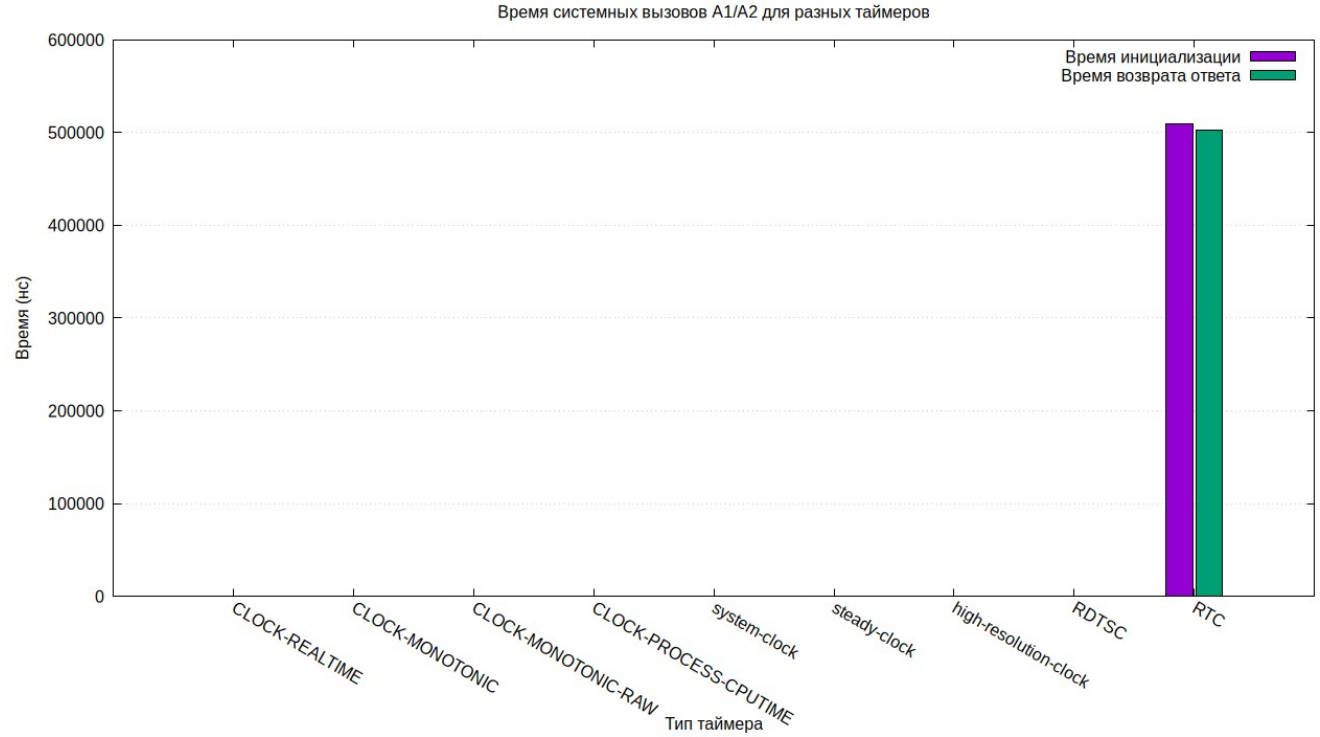

Рисунок 1 — Сравнительная столбчатая диаграмма накладных расходов для каждого вида таймера(с RTC)

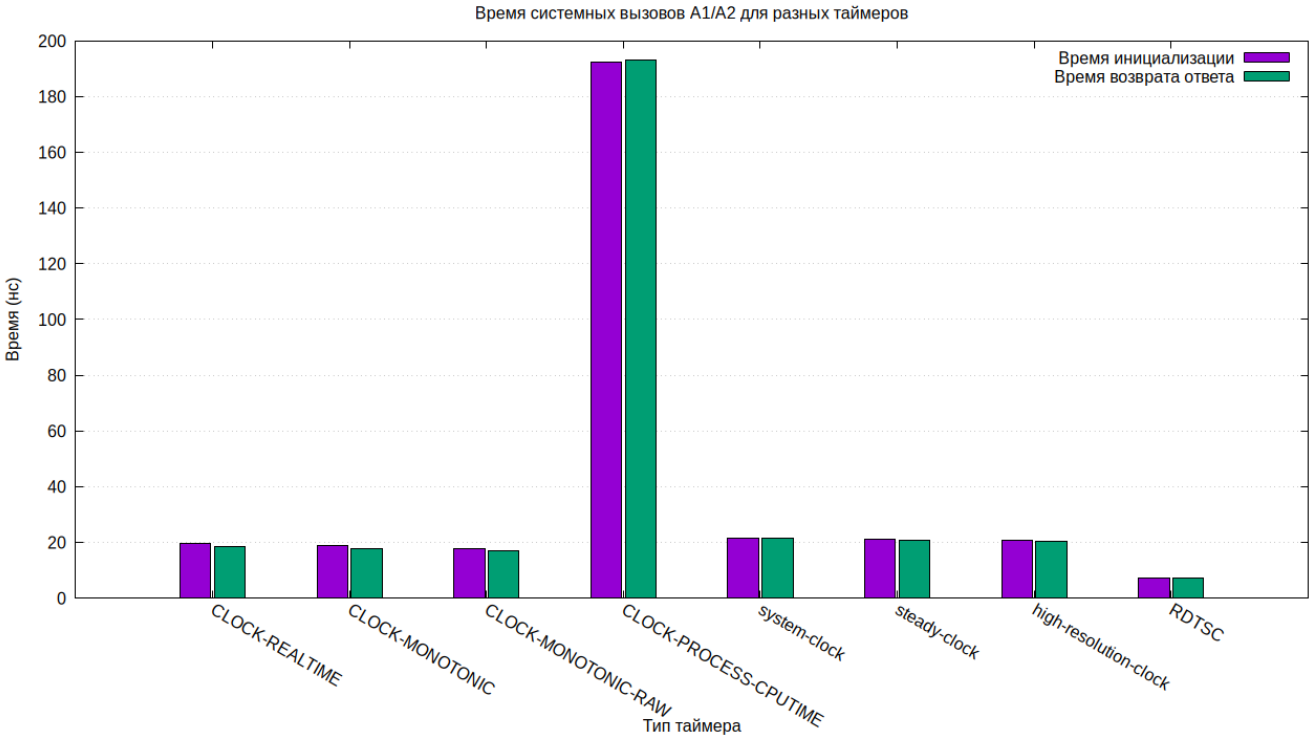

Рисунок 2 — Сравнительная столбчатая диаграмма накладных расходов для каждого вида таймера(без RTC)

Заключение

В результате проделанной работы в моей системе доступны такие типы часов, как system_clock, steady_clock, high_resolution_clock, CLOCK_REALTIME, CLOCK_MONOTONIC, CLOCK_MONOTONIC_RAW, а также CLOCK_PROCESS_CPUTIME_ID и RDTSC.

CLOCK_PROCESS_CPUTIME_ID, измеряющий время работы конкретного процесса, оказался менее точным и показал большие накладные расходы по сравнению с монотонными и высокоточными таймерами, что делает его не лучшим выбором для измерения интервалов, но полезным для оценки процессорного времени конкретного процесса. RTC продемонстрировал огромные задержки и низкое разрешение, что оправдано его аппаратной природой, но исключает его из списка для точных временных измерений. С другой стороны, steady_clock и high_resolution_clock продемонстрировали наилучшие показатели для измерения интервалов времени благодаря низким накладным расходам, а RDTSC показал наивысшую скорость, однако требует дополнительной калибровки и аккуратности при использовании.

Для задач, требующих высокой точности замеров интервалов, оптимальным выбором являются steady_clock и high_resolution_clock, в то время как CLOCK_PROCESS_CPUTIME_ID можно применять для анализа процессорного времени, а RTC оставлять для получения текущего реального времени.

Ссылка на гит-репоизторий : **https://github.com/sagilyp/PAPHSD-2/tree/main/lab2**