Всем доброго времени суток!  
Стало мне любопытно, какие минимальные изменения времени могут измерить функции, доступные программисту на C++. В документации эта информация как правило отсутствует, поскольку зависит от конфигурации компьютера, но сравнить разные функции на одной машине было бы интересно. Функций много, компьютер тоже не один, потому слепил я программку и прогнал её в Windows (на процессорах Intel & AMD) и в Linux (на тех же процессорах плюс на парочке ARMов). Заодно померил и ещё несколько параметров.

Список функций (некоторые с разными ключами):

1. Стандартные функции языка C: timespec\_get(), time() и clock();
2. Стандартные функции языка C++: std::system\_clock::now(), std:: steady\_clock::now() и std:: high\_resolution\_clock::now();
3. Функции Linux: times(), gettimeofday(), getrusage(), clock\_gettime();
4. Функции WinAPI: timeGetTime(), timeGetSystemTime(), QueryUnbiasedInterruptTimePrecise(), QueryUnbiasedInterruptTime(), QueryThreadCycleTime(), QueryProcessCycleTime(), QueryPerformanceCounter(), QueryInterruptTimePrecise(), QueryInterruptTime(), NtQuerySystemTime(), GetTickCount64(), GetTickCount(), GetThreadTimes(), GetSystemTimes(), GetSystemTimePreciseAsFileTime(), GetSystemTimeAsFileTime(), GetSystemTime(), GetProcessTimes(), GetLocalTime(), \_time64(), \_time32();
5. Несколько вариантов использования процессорных команд: RDTSC, RDTSCP, MRS.

Тестовые компьютеры:

1. Windows на процессорах: i7-6700K@4.00GHz, Ryzen 5 2400G@3.60GHz, i7-4770T@2.50GHz, i3-2105@3.10GHz, i7-10510U@1.80GHz, [i3-4005U@1.70GHz](mailto:i3-4005U@1.70GHz);
2. Linux: [i7-6700K@4.00GHz](mailto:i7-6700K@4.00GHz), Ryzen 5 [2400G@3.60GHz](mailto:2400G@3.60GHz), [Cortex-A72@1.80GHz](mailto:Cortex-A72@1.80GHz), ARM1176@0.70GHz.

Всего получилось сравнить 44 функции на 10 компьютерах, потому все данные не выкладываю. Подробности ниже.

Таб.

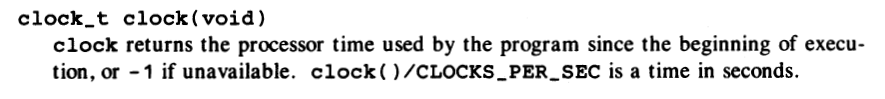
Пояснения. “Discreteness” – этот столбец содержит минимальное значение, на которое прирастает возврат последовательных вызовов функции. Это значение не обязательно равно единице измерения (столбец “One tick”), например на старой Raspberry Pi, функция clock имеет разрешающую способность лишь 7мкс хотя единица измерения равна 1мкс. (1 / CLOCKS\_PER\_SEC в POSIX равно 0,000’001 секунды).

На С++ доступны десятки функций позволяющих измерять интервалы времени. Но какова их реальная точность и сколько ресурсов они отнимают? В документации и стандартах ответов на эти вопросы как правило нет. Потому я попытался замерить основные параметры нескольких функций на разных машинах под Windows и Linux. Результаты в таблицах.

Таб 1,2,3

Что было измерено на примере древней функции clock().

В язык C функция была добавлена во второй редакции 1988-го года (так называемой ANSI C), и описывалась так:



Привилегированный режим не рассматриваем.

Для программного измерения интервалов времени обычно рекомендуют:  
timespec\_get в C;

std:: high\_resolution\_clock::now() в C++;

QueryPerformanceCounter() для WinAPI;

clock\_gettime() в LINUX;

А также ассемблерные команды доступа к TSC.

“One tick” - время, соответствующее единичке возвращённой исследуемой функцией. Например, в Windows ф-я clock(void) возвращает

Для примера исследуем всем известную древнейшую функцию clock(void) из первого стандарта ANSI C [1988]. Стандарт гласит, что функция возвращает приблизительное процессорное время, используемое процессом с начала определённого реализацией момента времени, относящегося к выполнению программы. И для преобразования возвращаемого значения его необходимо поделить на CLOCKS\_PER\_SEC (простоты ради будем считать, что это целочисленная константа).

В реализации CRT от Microsoft значение этой константы равно 1’000, В POSIX 1’000’000.