**Прикладная программа, процессор, архитектура, конфигурация**

**sys – время, затраченное на выполнение системных вызовов программы.**

**МЕТОДИКА ИЗМЕРЕНИЯ ВРЕМЕНИ ВЫПОЛНЕНИЯ ПРИКЛАДНОЙ ПРОГРАММЫ**

Измерение времени выполнения прикладной программы или ее частей является одним из основных способов контроля характеристик аппаратного и программного обеспечения с точки зрения быстродействия. Такой контроль, с одной стороны, полезен для определения «узких мест» в алгоритме или программе, которые нуждаются в оптимизации. С другой стороны, позволяет судить о реальной производительности компьютера. Известно, что на время выполнения программы влияют разные факторы:

• характеристики самой программы,

• архитектура и конфигурация компьютера,

• операционная система,

• совместно работающие процессы,

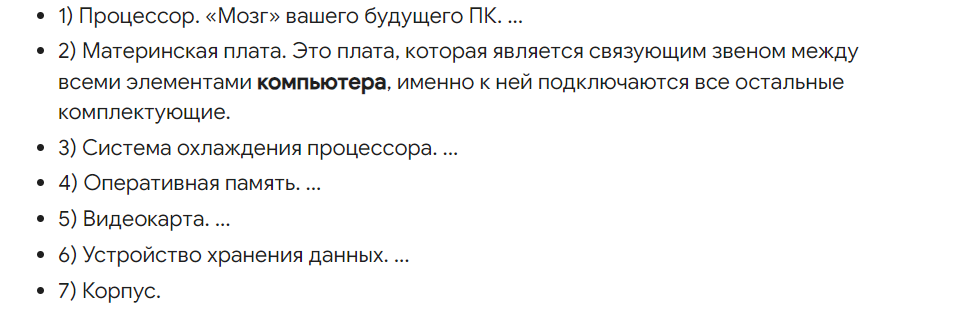
• состояние компьютера на момент старта программы,

• влияние измерителя времени и т.п.

*Операционная система (ОС) – это комплекс программ, предназначенных для управления ресурсами компьютера и организации взаимодействия с пользователем.*

*Центра́льный проце́ссор (ЦП; также центра́льное проце́ссорное устро́йство — ЦПУ;*[*англ.*](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA)*central processing unit, CPU, дословно — центральное обрабатывающее устройство, часто просто процессор) —*[*электронный блок*](https://ru.wikipedia.org/wiki/%D0%AD%D0%BB%D0%B5%D0%BA%D1%82%D1%80%D0%BE%D0%BD%D0%BD%D0%BE%D0%B5_%D1%83%D1%81%D1%82%D1%80%D0%BE%D0%B9%D1%81%D1%82%D0%B2%D0%BE)*либо*[*интегральная схема*](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D0%B3%D1%80%D0%B0%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F_%D1%81%D1%85%D0%B5%D0%BC%D0%B0)*, исполняющая*[*машинные инструкции*](https://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D1%88%D0%B8%D0%BD%D0%BD%D1%8B%D0%B9_%D0%BA%D0%BE%D0%B4)*(код программ), главная часть*[*аппаратного обеспечения*](https://ru.wikipedia.org/w/index.php?title=%D0%90%D0%BF%D0%BF%D0%B0%D1%80%D0%B0%D1%82%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5&action=edit&redlink=1)[*компьютера*](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80)*или*[*программируемого логического контроллера*](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D1%83%D0%B5%D0%BC%D1%8B%D0%B9_%D0%BB%D0%BE%D0%B3%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B9_%D0%BA%D0%BE%D0%BD%D1%82%D1%80%D0%BE%D0%BB%D0%BB%D0%B5%D1%80)*. Иногда этот компонент называют просто процессором.*

*В области информационных и компьютерных систем под конфигурацией персонального компьютера понимают определенный набор комплектующих, исходя из их предназначения, номера и основных характеристик.*

**

*Архитектура ПК - это логическая организация и структура вычислительных систем, определяющая взаимодействие аппаратных и программных компонентов. Включает:*

*процессор*

*память*

*ввод-вывод*

*Пример: архитектура x86, основанная на магистрально-модульном принципе.*

*Архитектура компьютера представляет собой сложную систему взаимодействующих компонентов, при этом основную вычислительную нагрузку берет на себя центральный процессор. Именно он выполняет основную работу, обрабатывая данные и управляя задачами, которые выполняет компьютер.*

*Программное обеспечение (ПО) — это совокупность всех программ и служебных данных для них, предназначенных для управления компьютером*

Отделить влияние интересующих характеристик от прочих далеко не всегда просто. Для этого существуют приемы, позволяющие снизить влияние нежелательных факторов, а также интерпретировать полученные временные характеристики запусков программ.

Время выполнения программы или её частей (далее – программы), как правило, определяется разницей показаний таймера, которые снимаются перед началом исполнения программы и после её завершения.

*Независимо от используемого таймера, измерение времени работы программы всегда выполняется с некоторой погрешностью (абсолютной или относительной). Погрешность измерения используемого таймера не должна выходить за рамки допустимой точности измерения. Величина допустимого значения погрешности определяет тип таймера, которым следует пользоваться при определении времени работы программы.*

*Абсолютная погрешность определяется разницей, например, в секундах между временем таймера и точным временем выполнения программы. Поскольку точное время выполнения программы никогда не известно, то абсолютная погрешность измерения оценивается точностью измерительного прибора. Например, если точность таймера 1 мс, то время было измерено с погрешностью не более 1 мс.*

*Относительная погрешность – это отношение абсолютной погрешности к величине временного интервала. Например, абсолютная погрешность в 1 мс даст относительную погрешность в 50%, если весь интервал был 2 мс и 0,1%, если интервал был 1 с. Таким образом, относительная погрешность показывает величину погрешности относительно всего интервала времени.*

В современном компьютере имеется несколько таймеров с разной точностью. Для измерения времени работы программы можно использовать и внешние измерительные приборы, например, секундомер. Если точность выбранного таймера заведомо выше, чем требуется для заданного измерения, это ещё не гарантирует, что полученный результат измерения выполнен с требуемой точностью, т.к., возможно, измерение было искажено влиянием посторонних факторов. Для уменьшения этого возможного влияния существует ряд приёмов.

**Процедура измерения времени выполнения программы**

1. Пусть задана величина допустимого значения погрешности и её тип (относительная и/или абсолютная точность). Если требуется удовлетворить только абсолютной точности измерения, то п.2. данной процедуры пропускается.

2. Относительная точность преобразуется в абсолютную. Для этого необходимо любым способом оценить время выполнения программы. Далее выполняется п.3.

3. На компьютере, где будет выполняться измерение времени, оценивается степень загрузки процессора другими процессами. Если степень загрузки процессора высока, то выбирается таймер времени процесса, в противном случае выбирается произвольный таймер, например, из списка таймеров в п. «Способы получения показаний некоторых таймеров». Выбранный таймер должен обеспечивать требуемую точность измерения времени, иначе см. п.4.

4. Если доступные таймеры не могут обеспечить необходимую точность измерения из-за малой величины измеряемого интервала времени, увеличение точности измерения можно достигнуть следующим образом. Программа многократно запускается в цикле, и измеряется общее время выполнения этого цикла. Полученное значение времени выполнения цикла делится на число итераций в цикле. Абсолютная погрешность измерения уменьшится пропорционально числу итераций цикла.

5. Измеряется время выполнения программы с использованием приемов из пункта «Приемы уменьшения влияния посторонних факторов на время выполнения прикладной программы», затем проверяется качество измерения в п.6.

6. Если по условию измерения времени программы требовалось обеспечить абсолютную точность, то полученное время является результатом измерения, в противном случае полученное измерение времени считается новым оценочным временем выполнения программы, для которого ещё раз вычисляется требуемая абсолютная точность из заданной относительной. Если новая абсолютная точность выше, чем точность выбранного таймера, то выполняется возврат к п.3 с новым оценочным временем выполнения программы.

**ПРИЕМЫ УМЕНЬШЕНИЯ ВЛИЯНИЯ ПОСТОРОННИХ ФАКТОРОВ НА ВРЕМЯ ВЫПОЛНЕНИЯ ПРИКЛАДНОЙ ПРОГРАММЫ**

**Многократное измерение.** Время работы программы измеряется несколько раз. Измерения, как правило, будут отличаться. Это связано с тем, что сторонние факторы вносят различный вклад в измеряемый интервал при каждом запуске. Время же работы самой программы остаётся неизменным. Поэтому, из всех полученных измерений наиболее точным будет минимальное.

**Исключение из измерения стадий инициализации и завершения**. Если требуется измерить время работы некоторого фрагмента кода, например, некоторой процедуры, то целесообразно вынести весь код программы, предшествующий вызову процедуры, за первое измерение времени работы программы, а второе измерение осуществлять сразу после завершения её работы. Особенно это касается команд работы с устройствами ввода-вывода (чтение с клавиатуры, вывод на экран, работа с файлами) и, в меньшей степени, команд работы с оперативной памятью.

**Уменьшение влияния кода измерения времени.** Сам код снятия показаний того или иного таймера выполняется не мгновенно. Это означает, что в измеряемый интервал времени частично попадает и время его исполнения. Это влияние следует, по возможности, уменьшать. Во-первых, не следует снимать показания времени часто, особенно в циклах. В идеале, код замера времени должен быть вызван лишь дважды – в начале и в конце работы измеряемого фрагмента кода. Во-вторых, необходимо следить за тем, чтобы измеряемый интервал был существенно больше, чем время работы функции замера времени.

**Уменьшение влияния посторонних процессов.** В случаях, когда процессор загружен другими процессами (например, системными процессами или задачами других пользователей), для получения более точного результата целесообразно использовать таймер времени выполнения процесса (см. примеры соответствующих таймеров в разделе 5.3). Такие таймеры, как правило, обладают более низкой точностью и плохо подходят для измерения небольших интервалов времени.

**Сброс буфера отложенной записи на диск.** В современных операционных системах, как правило, используется механизм кэширования при работе с внешней памятью, например, жесткими дисками. Этот механизм заключается в том, что при записи данных на диск данные сначала попадают в буфер отложенной записи, который располагается в оперативной памяти, а затем уже будут записаны позже. Этот механизм служит для ускорения доступа к файлам и уменьшению износа аппаратного обеспечения. В таких ОС возможна ситуация, когда во время работы вашей программы ОС решит сгрузить накопленные данные на диск, что наверняка повлияет на чистоту эксперимента. Поэтому перед проведением замеров времени следует освобождать буфер отложенной записи на диск. В ОС Linux/UNIX это делается с помощью утилиты sync. Эту команду можно набрать перед запуском своей программы из командной строки, либо вызвать её прямо из кода своей программы с помощью системного вызова system (см. system (3) man page).

*ОЗУ — это кратковременное хранилище компьютера, обеспечивающее быстрый доступ к данным, которое выступает в качестве буфера между процессором и твердотельным накопителем/жестким диском.*

*Жесткий диск — это запоминающее устройство для хранения ваших файлов и данных в течение длительного времени. Сохраняя файл на свой компьютер, вы сохраняете его на жестком диске устройства. Жесткий диск — это картотечный шкаф для ваших цифровых файлов.*

*Кэш — это память программы или устройства, которая сохраняет временные или часто используемые файлы для быстрого доступа к ним. Это увеличивает скорость работы приложений и операционной системы. Процесс сохранения таких файлов в специальном месте называется кэшированием.*

Примечание: Тут и далее под man page подразумевается ссылка на стандартную линукс-документацию (Linux man pages). Чтобы посмотреть документацию по той или иной команде можно в командной строке набрать следующую команду: user@host:~ man 3 system Либо найти соответствующую страницу в Интернете.

**ТАЙМЕРЫ**

Вычислительная система имеет несколько программных и аппаратных таймеров, отражающих течение времени с различных точек зрения. Необходимо различать следующие таймеры:

• **Таймер системного времени** (system time, wall-clock time) – аппаратный счетчик, который отражает течение времени с точки зрения вычислительной системы и, как правило, соответствует реальному течению времени. Значение системного времени в каждый момент одинаково для всех программ, работающих на данном компьютере. Величина временного интервала, измеренного с помощью таймера системного времени, включает в себя время работы не только замеряющего процесса, но и других. Функции для получения системного времени:

Windows: GetSystemTime(), GetTickCount(), time(),

Linux: gettimeofday(), times(), time(), clock\_gettime().

Кроме того, в Windows используется функция clock(), которая позволяет получить величину системного времени, прошедшего с момента запуска данного процесса.

**Таймер времени процесса** (process time, CPU time) – программный счетчик, который отражает использование процессорного времени только конкретным процессом. Шаг изменения этого счетчика относительно велик, поэтому его не следует использовать для измерения малых промежутков времени. Функции для получения времени процесса:

Windows: GetThreadTimes(),GetProcessTimes(),

Linux: times(), clock().

**Счетчик тактов процессора** (CPU time stamp counter) – аппаратный счетчик, значение которого увеличивается на каждом такте процессора. Такт процессора – самый малый интервал времени в вычислительной системе, который теоретически может быть замерен. Поэтому счетчик тактов позволяет с большой точностью измерять малые промежутки времени (вплоть до нескольких команд процессора). Счетчик тактов процессора имеет смысл использовать только для измерения интервалов времени меньших кванта времени, выделяемого процессу операционной системой. Для получения значения счетчика тактов используются специальные команды процессора, свои для каждой архитектуры:

x86/x86-64: rdtsc,

Alpha: rpcc,

Itanium: ar.itc,

PowerPC: mftb, mftbu.

**СПОСОБЫ ПОЛУЧЕНИЯ ПОКАЗАНИЙ НЕКОТОРЫХ ТАЙМЕРОВ**

**1.Утилита time**

Утилита time выдаёт следующие временные характеристики работы программы:

real – общее время работы программы согласно системному таймеру,

user – время, которое работал пользовательский процесс (кроме времени работы других процессов)

sys – время, затраченное на выполнение системных вызовов программы.

Точность: определяется точностью системного таймера и точностью измерения времени работы процесса (см. описание соответствующих таймеров ниже).

Достоинство: готовая утилита, не требуется вносить изменения в программу.

Недостаток: измеряется только время работы всей программы, нет возможности измерить время работы отдельных её частей.

**2.** **Библиотечная функция clock\_gettime ПОЧИТАТЬ ДОПОЛНИТЕЛЬНО**

Библиотечная функция clock\_gettime получает значения системного таймера в ОС Linux/UNIX

Функция clock\_gettime с параметром CLOCK\_MONOTONIC\_RAW сохраняет значение системного таймера в структуру struct timespec. Структура состоит из двух полей: tv\_sec и tv\_nsec (можно считать их тип long int), задающих количество секунд и наносекунд (10−9 cек.), прошедших с некоторого неспецифицированного момента времени в прошлом. В приведённом примере сохраняется значение таймера перед выполнением некоторого кода и после него. Разница показаний преобразуется в секунды и выводится на экран. Кроме системного таймера, функция позволяет получать значения и других таймеров, например, времени процесса или потока. Реализация функции clock\_gettime находится в библиотеке rt, поэтому при компиляции программы необходимо добавить ключ компиляции ‘-lrt’. Пример команды компиляции (в некоторых системах требуется ключ ‘-lrt’ писать в конце команды): user@host:~$ gcc prog.c –o prog -lrt

Точность: зависит от точности системного таймера. Обычно в ОС Windows: 55 мс (55∙10−3 с), в ОС GNU Linux/UNIX: 1 нс (1∙10−9 с).

Достоинство: переносимость – вне зависимости от аппаратного обеспечения функция доступна пользователю, т.к. реализуется ОС.

Недостатки: относительно низкая точность (обычно ниже, чем у счётчика тактов, но выше, чем у функции times), и измеренный интервал включает время работы других процессов, которые работали на процессоре в измеряемый период.

*Аппаратное обеспечение персонального компьютера включает электронные компоненты и различные внешние устройства для обработки, хранения данных, обмена информацией с пользователем и связи с другими электронными устройствами.*

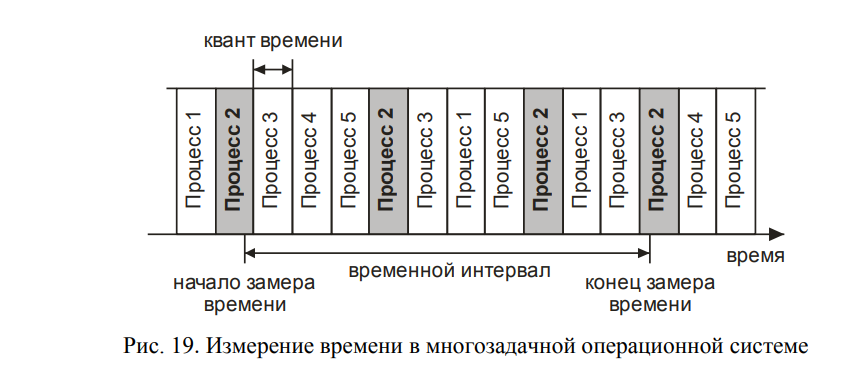
3. **Библиотечная функция times**

Функция измерения времени работы процесса times позволяет определить время работы данного процесса в многозадачной операционной системе, где каждый процессор (или ядро) выполняет несколько процессов (программ) в режиме разделения времени (рис.19). Этот показатель особенно важен, когда процессор сильно загружен другими процессами. В этом случае показания, например, системного таймера могут быть очень далеки от действительного времени работы программы. Процессор переключается между процессами с некоторой периодичностью. В обычных Linux/UNIX системах это около 10 мс (10−2 с). Функция times (листинг 3) позволяет определить, сколько таких квантов времени проработал наш процесс. Если перевести количество этих квантов во время, то можно определить, какое время работал процесс.

*Квант времени - это минимальный интервал времени, который может быть гарантированно выделен процессором для выполнения задачи.*

*Ядро ОС распределяет время процессора между несколькими процессами или потоками, планируя их выполнение. Обычно каждый процесс или поток получает квант времени, после чего он прерывается, и планировщик передает управление другому процессу или потоку.*

*Размер кванта времени обычно устанавливается ОС и может варьироваться от нескольких миллисекунд до сотен миллисекунд. Оптимальный размер кванта зависит от типа системы, характера выполняемых задач и количества процессов или потоков.*

****

В целом, её использование аналогично функции clock\_gettime, отличие состоит в преобразовании единиц измерения из квантов времени в секунды. Количество квантов в секунду позволяет узнать системный вызов sysconf.

Дополнительная информация: times, sysconf.

Точность: зависит от кванта планировщика процессов, обычно 10 мс (10−2 с).

Достоинство: Из измеряемого времени исключается время, которое работали другие процессы, это обеспечивает более точное измерение времени работы программы на сильно загруженных процессорах по сравнению с другими способами.

Недостаток: относительно низкая точность, определяемая квантом времени переключения процессов.

4. **Машинная команда rdtsc**

Машинная команда rdtsc (Read Time Stamp Counter) снимает показания счётчика тактов в виде 64-разрядного беззнакового целого числа, равного количеству тактов, прошедших с момента запуска процессора (листинг 4). Время в секундах получается делением количества тактов на тактовую частоту процессора. В этом примере используется ассемблерная вставка команды процессора rdtsc, результат выполнения который записывается в объединение (union) ticks (старшая и младшая части). Разница показаний счётчика тактов преобразовывается в секунды в зависимости от тактовой частоты. Узнать тактовую частоту процессора в ОС Linux/UNIX можно, например, распечатав содержимое системного файла /proc/cpuinfo.

Точность: один такт (величина в секундах зависит от тактовой частоты процессора).

Достоинство: максимально возможная точность измерения времени.

Недостатки: привязка к архитектуре x86, в других архитектурах могут существовать аналогичные машинные команды. Затруднительно преобразование в секунды в процессорах с динамическим изменением частоты.

*Тактовая частота процессора - это мера того, сколько тактовых импульсов в секунду может выполнять процессор. Тактовые импульсы используются для синхронизации операций внутри процессора и других компонентов компьютера.*

*Таймер типа "счетчик тактов" (RDTSC) - это инструкция в наборе команд x86, которая позволяет считывать количество тактовых импульсов, прошедших с момента загрузки процессора. Эту инструкцию можно использовать для измерения времени выполнения кода или других операций.*

*Минусы использования таймера типа "счетчик тактов" RDTSC:*

*• Непостоянная тактовая частота: Тактовая частота процессора может изменяться динамически в зависимости от нагрузки и настроек системы, что может привести к неточным измерениям времени.*

*• Отсутствие портативности: Инструкция RDTSC не поддерживается всеми процессорами и операционными системами, что ограничивает ее переносимость.*

*1. Проблемы с многоядерностью*

*На многоядерных системах каждый ядро может иметь собственный счетчик тактов. Это означает, что выполнение RDTSC на разных ядрах может дать разные значения. Если процессоры имеют разные частоты, результаты могут быть еще менее сопоставимыми.*

*2. Изменение частоты тактирования*

*Многие современные процессоры используют технологии динамической смены частоты (например, Intel SpeedStep, AMD Cool'n'Quiet). Частота может значительно меняться в зависимости от нагрузки на систему, что приводит к непредсказуемому поведению RDTSC. Следовательно, программа, использующая RDTSC, может получить совершенно разные результаты в зависимости от текущей частоты.*

**Дополнительные вопросы:**

**Функция clock\_gettime() принимает два аргумента:**

**\* clock\_id: Тип часов для получения времени. Это константа, определяющая источник времени.**

**\* tp: Указатель на структуру timespec, в которую будет записано время.**

**Параметры функции clock\_gettime():**

**clock\_id:**

**Возможные значения clock\_id зависят от системы, но обычно включают следующие:**

**\* CLOCK\_REALTIME: Время реального времени, включая время, проведенное в режиме ожидания и сна.**

**\* CLOCK\_MONOTONIC: Монотонно возрастающее время, не зависящее от изменений системного времени.**

**CLOCK\_REALTIME**

Часы системы, отсчитывающие реальное (т. е., бытовое) время. Для настройки этих часов требуются соответствующие права. Данные часы подвержены скачкам системного времени (например, если системный администратор вручную изменяет время) и постепенной подгонке, выполняемой **[adjtime](https://ru.manpages.org/adjtime/3)**(3) и NTP.

**CLOCK\_REALTIME\_COARSE** (начиная с Linux 2.6.32; есть только в Linux)

Более быстрая, но менее точная версия **CLOCK\_REALTIME**. Используйте, если нужны не очень точные метки времени, но быстро.

**CLOCK\_MONOTONIC**

Часы, которые не могут быть настроены и показывают монотонный ход времени отсчитываемой с некой неопределённой начальной точки. Эти часы не подвержены скачкам системного времени (например, системный администратор вручную изменил время), но на них влияет постепенная подгонка, выполняемая **[adjtime](https://ru.manpages.org/adjtime/3)**(3) и NTP.

**CLOCK\_MONOTONIC\_COARSE** (начиная с Linux 2.6.32; есть только в Linux)

Более быстрая, но менее точная версия **CLOCK\_MONOTONIC**. Используйте, если нужны не очень точные метки времени, но быстро.

**CLOCK\_MONOTONIC\_RAW** (начиная с Linux 2.6.28; есть только в Linux)

Похожи на **CLOCK\_MONOTONIC**, но предоставляют прямой доступ к аппаратным часам, которые не подводятся NTP или постепенной подгонкой, выполняемой **[adjtime](https://ru.manpages.org/adjtime/3)**(3).

**CLOCK\_BOOTTIME** (начиная с Linux 2.6.39; есть только в Linux)

Идентичны **CLOCK\_MONOTONIC**, но также содержат любое время, на которое система была приостановлена (suspended). Это позволяет приложениям получить учитывающие приостановку монотонные часы без обращения к сложностям **CLOCK\_REALTIME**, которые могут быть неоднородны, если время изменили с помощью **[settimeofday](https://ru.manpages.org/settimeofday/2)**(2).

**CLOCK\_PROCESS\_CPUTIME\_ID** (начиная с Linux 2.6.12)

Настраиваемые для каждого процесса часы ЦП (измеряют время ЦП, затраченное всеми нитями процесса).

**CLOCK\_THREAD\_CPUTIME\_ID** (начиная с Linux 2.6.12)

Часы, работающие на ЦП, для каждой нити.