ТЕМА 2.5. КОМПЬЮТЕРНОЕ ВРЕМЯ

В данной теме рассматриваются следующие вопросы:

- Эталоны времени.
- Нелинейность времени.
- Календари.
- Часовые пояса.
- Локальное время и UTC.
- Эпоха UNIX.
- Единицы измерения компьютерного времени.
- Форматы времени.
- Функции POSIX для работы с временем.
- Ожидаемые таймеры.
- Протокол NTP.

Лекции – 2 часа, лабораторные занятия – 2 часа, самостоятельная работа – 2 часа.

Экзаменационные вопросы по теме:

- Компьютерное время. Таймеры ожидания.

2.5.1. Эталоны времени

Первые часы или были связаны с солнцем, или основывались на физических процессах известной длительности (горение свечи, вытекание воды). По принципу водяных часов работают и песочные часы. Механические часы подсчитывали количество колебаний маятника или балансира. В электрических часах маятник раскачивался электромагнитом. Электронные часы отображают время посредством светодиодных или ЖКИ-индикаторов. В настоящее время электронные часы встроены почти в любую технику.

Рис. 2.5.1. Различные часы

В метрологии время — физическая величина, одна из семи основных величин Международной системы величин (ISQ), а единица измерения времени «секунда» — одна из семи основных единиц в Международной системе единиц (СИ) (SI).

Время характеризуется своей однонаправленностью, одномерностью, наличием ряда свойств симметрии.

Также время как физическая величина определяется периодическими процессами в некой системе отсчёта, шкала времени которой может быть как неравномерной (процесс вращения Земли вокруг Солнца или человеческий пульс), так и равномерной.

С древности точное время определялось путём астрономических наблюдений. Но в XX веке развитие науки привело к тому, что техническими средствами стало возможно обеспечить измерение времени с большей точностью, чем из астрономических наблюдений. В 1964 году Международный комитет мер и весов в качестве эталона времени принял атомные цезиевые часы. Теперь сигналы точного времени, передаваемые по радио, соответствуют «атомному времени». Ежедневное вращение Земли нерегулярно и постоянно замедляется, поэтому атомные часы представляют собой гораздо более стабильную временную базу. Основанный на них стандарт UTC почти в миллион раз точнее астрономического среднего времени по Гринвичу.

Эталон секунды — 9 192 631 770 периодов излучения, соответствующего переходу между двумя сверхтонкими уровнями основного состояния атома цезия-133 при отсутствии возмущения внешними полями. Это определение — не произвольное, а связанное с наиболее точными периодическими процессами, доступными человечеству на данном этапе развития экспериментальной физики.

Эфемеридное время, ephemeris time (ET) — равномерная шкала времени, основанная на определении секунды, введённом в 1952 году на 8-м съезде Международного астрономического союза, которое не зависит от изменяющейся скорости вращения Земли. В 1956 году Генеральной конференцией по мерам и весам (CGPM) это определение было рекомендовано к использованию, а в 1960 году эфемеридная секунда была принята за основную единицу времени в Международной системе единиц СИ.

С 1984 года шкала эфемеридного времени ET заменена двумя шкалами динамического времени DT:

Земное динамическое время TDT, равное по масштабу ET, отнесено к центру масс Земли и служит независимым аргументом видимых геоцентрических эфемерид, в том числе при определении эфемерид ИСЗ

Барицентрическое динамическое время TDB, которое учитывает движение центра масс Солнца вокруг центра масс всей Солнечной системы (барицентра Солнечной системы). Отнесено к барицентру Солнечной системы и является аргументом дифференциальных уравнений всех гравитационных теорий движения тел Солнечной системы.

За период 1991—2006 гг. из-за сложностей и несогласованностей базовых определений шкалы TDB и TDT были переопределены и заменены, соответственно, на Геоцентрическое координатное время (TCG) и Барицентрическое координатное время (TCB), которые и применяются в настоящее время (2011 год). Эти шкалы отражают темп времени, скорректированного в соответствии с общей теорией относительности к часам, покоящимся относительно соответствующей нуль-точки вне гравитационного колодца Земли и Солнечной системы, соответственно. Из-за этого темп хода времени в данных шкалах несколько выше, чем скорость хода атомных часов на поверхности Земли, и соответственно они расходятся с локально определённым Международным атомным временем (TAI) линейно с некоторым колебаниями. Для учёта этого обстоятельства определена также шкала Земного времени (Terrestrial Time, TT), заменяющая TDT и выводимая как такое линейное решкалирование TCG, чтобы TT давала среднюю длительность секунды, совпадающую с секундой атомного времени TAI, а также переопределена шкала Барицентрического динамического времени (TDB) как линейное решкалирование TCB с аналогичными свойствами.

Международное атомное время (TAI) — время, в основу измерения которого положены электромагнитные излучения, излучаемые атомами или молекулами при переходе из одного энергетического состояния в другое. Масштаб системы TAI принят равным масштабу эфемеридного времени (ET), то есть атомные часы есть физическое воспроизведение шкалы ET. Точность воспроизведения до $2 \cdot 10^{-12}$. Неудобно для астрономических вычислений.

Земное время, Terrestrial Time (TT) — современный астрономический стандарт, разработанный Международным астрономическим союзом для определения времени астрономических наблюдений, сделанных с поверхности Земли. Земное время является наследником динамического времени и эфемеридного времени. Земное время отличается от шкалы времени, используемого для повседневного применения (UTC). Единица времени TT — секунды в системе СИ, основанные на стандарте цезиевых атомных часов.

Всемирное время или UT (Universal Time) — шкала времени, основанная на вращении Земли. Всемирное время является современной заменой среднего времени по Гринвичу (GMT), которое сейчас иногда некорректно используется в качестве синонима для всемирного координированного времени (UTC). Всемирное время введено 1 января 1925 года. Фактически термин «всемирное время» является многозначным, так как существует несколько версий всемирного времени, главными из которых является UT1 и UTC. Все версии всемирного времени основаны на вращении Земли относительно далёких небесных объектов (звёзд и квазаров), используя коэффициент масштабирования и другие подстройки для того, чтобы быть ближе к солнечному времени.

Существует несколько версий всемирного времени. Приведем здесь только две.

UT1, или универсальное время — основная версия всемирного времени. Хотя концептуально это местное среднее солнечное время на долготе 0°, но измерения среднего Солнца трудноосуществимы, поэтому UT1 вычисляется пропорционально углу вращения Земли относительно квазаров, а точнее, относительно международной небесной системы координат (ICRS).

UTC (всемирное координированное время) — атомная шкала времени, аппроксимирующая UT1. Это международный стандарт, на котором базируется время часовых поясов. В UTC в качестве единицы времени используется секунда СИ, поэтому

UTC идёт синхронно с международным атомным временем (TAI). Обычно в сутках UTC 86 400 секунд СИ, но для поддержания расхождения UTC и UT1 не более чем 0,9 с, при необходимости, 30 июня или 31 декабря добавляется (или, теоретически, вычитается) секунда координации.

Международное атомное время

Международное атомное время (**TAI**, temps atomique International) — это высокоточный стандарт атомного координатного времени, основанный на воображаемом прохождении собственного времени на геоиде Земли. TAI — это средневзвешенное значение времени, которое показывают более 450 атомных часов в более чем 80 национальных лабораториях по всему миру. Это непрерывная шкала времени, **без дополнительных секунд**, и это основная реализация земного времени (с фиксированным смещением эпохи). Это основа всемирного координированного времени (UTC), которое используется для гражданского времени на всей поверхности Земли и имеет дополнительные секунды.

UTC отклоняется от TAI на несколько целых секунд. По состоянию на 1 января 2017 года, когда была введена в силу еще одна дополнительная секунда, UTC в настоящее время отстает от TAI ровно на 37 секунд. 37 секунд являются результатом первоначальной разницы в 10 секунд в начале 1972 года плюс 27 дополнительных секунд по UTC с 1972 года.

TAI можно получить с использованием традиционных средств указания дней, перенесенных из неоднородных стандартов времени, основанных на вращении Земли. В частности, используются как юлианские дни, так и григорианский календарь. TAI в этой форме был синхронизирован со Всемирным временем в начале 1958 года, и с тех пор они разошлись, главным образом из-за замедления вращения Земли.

TAI — это средневзвешенное значение времени, которое показывают более 450 атомных часов в более чем 80 национальных лабораториях по всему миру. Большинство задействованных часов являются цезиевыми часами. Определение секунды в Международной системе единиц (СИ) основано на цезии. Часы сравниваются с использованием сигналов GPS и двусторонней спутниковой передачи времени и частоты. Благодаря усреднению сигнала TAI на порядок стабильнее своих лучших составляющих тактовых импульсов.

В отличие от TAI, UTC представляет собой прерывистую шкалу времени. Иногда его корректируют на дополнительные секунды. Между этими корректировками он состоит из сегментов, которые сопоставлены с атомным временем постоянным смещением. С момента своего создания в 1961 году по декабрь 1971 года корректировки производились регулярно с долями дополнительных секунд, так что UTC приближалось к UT2. Впоследствии эти корректировки производились только за целые секунды, чтобы приблизиться к UT1. Это было компромиссное соглашение, позволяющее обеспечить публичную трансляцию временной шкалы. Менее частые корректировки с точностью до целой секунды означали, что шкала времени будет более стабильной и ее будет легче синхронизировать на международном уровне. Тот факт, что он продолжает приближаться к UT1, означает, что такие задачи, как навигация, требующие источника всемирного времени, по-прежнему хорошо выполняются общественным вещанием UTC.

Разница между земным и эфемеридным временем

ΔT — это разница между идеальным равномерно текущим временем и «временем», определённым по вращению Земли (которое замедляется, причём неравномерно).

Причины замедления вращения Земли: приливное трение, последствия таяния материкового ледникового щита (распрямление и поднятие коры в приполярных областях) и другие.

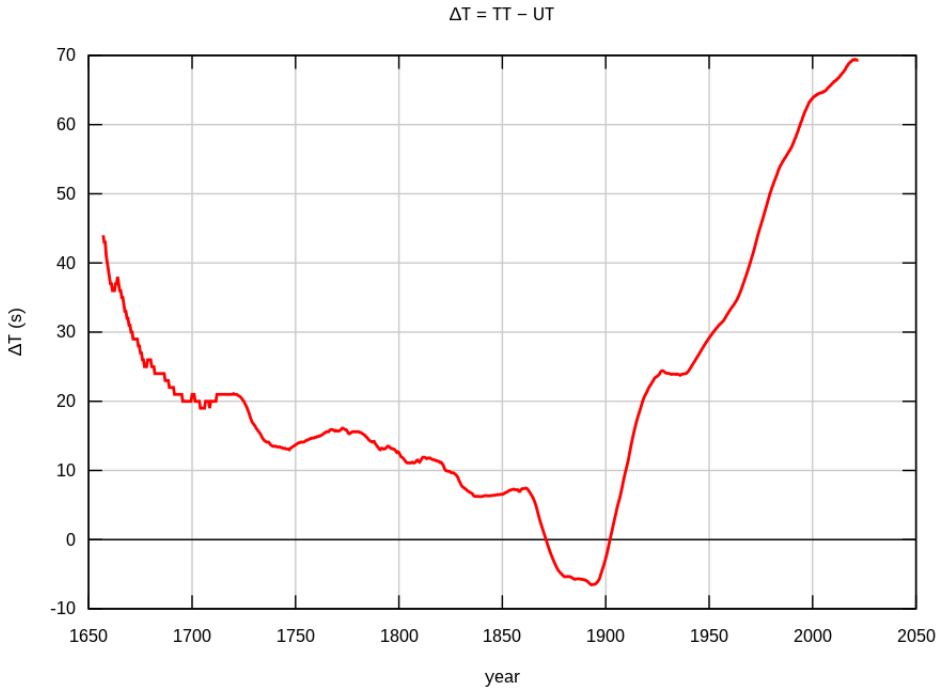

Рис. 2.5.2. Разница между земным и эфемеридным временем

Социальное время

Социальное время — это концепция понимания времени в социальных науках и философии. Этот термин в 1937 году предложили эмигрировавший в США из России социолог Питирим Сорокин и профессор Колумбийского университета Роберт Мертон. Сегодня их исследования стали классикой социологии [1].

Социальное время отличается от астрономического. В его основе лежат не циклы движения планет и звёзд, а изменения в обществе, происходящие по воле человека. То есть измеряется оно не единицами продолжительности (минута, час, год), а такими абстрактными мерами, как эпоха, поколение, жизнь.

Социальное время отражает не то, сколько длится событие, а то, как ощущается его продолжительность. Например, полуторачасовая лекция может казаться нам невыносимо долгой, а вся прожитая до этого момента жизнь — мгновением. Из-за этого социальное время часто соотносят с психологическим — индивидуальным восприятием длительности. Но социальное время, как считают исследователи, — это ещё и «время общества» — реакция на поток происходящих событий в границах какой-то страны, сообщества или семьи.

Левис и Вайгерт выделяют три цикла социального времени: дневной цикл, недельный и годовой цикл [2]. Физический день с его сменой дня и ночи и градациями рассвета, утра, полдня, сумерек и ночи является основой для ежедневного цикла деятельности человека. Дневной цикл начинается с подъема утром и заканчивается отходом ко сну ночью. Недельный цикл. Уик-энд, конец недели, стал временем, свободным от повседневной деятельности. В зависимости от того или иного дня недели меняется соотношение свободного и рабочего времени, времени, которое человек тратит на свою семью и на себя. Годовой цикл. Времена года также отличаются по своему социальному значению. Во-первых, в зависимости от времени года меняются способы проведения свободного времени. Во-вторых, каждый сезон отличается от других своим праздником: Рождество-Новый год, национальные праздники. Символически трансформируясь, смена времен года влияет на наши мысли, чувства, поведение. В каждой региональной и национальной структуре существует своя временная структура этих природных циклов.

Прогресс приводит к тому, что плотность событий на один промежуток астрономического времени значительно возрастает. Однако само время при этом течёт точно так же. Чем

больше событий происходит, тем больше информации поступает в мозг человека, в результате растёт нагрузка на него.

Человек постоянно находится в режиме многозадачности и необходимости соблюдать дедлайны. Перерывы в восприятии информации сокращаются или вообще сходят на нет. Мы вынуждены отказываться от времязатратных мероприятий в пользу тех, что экономят минуты и часы.

Ускорение социального времени рождает один из парадоксов современности: развитие общества и технологий по идее должно было освободить для нас некоторое количество времени, но при этом ощущение его нехватки растёт.

Современный человек вынужден всё делать на бегу и под непрерывной бомбардировкой огромных объёмов данных. Особая роль здесь принадлежит информационному шуму — большинство сообщений из внешнего мира для нас не важны или имеют малую значимость, поэтому мозгу приходится их фильтровать. Нам нужно принимать решения, совершать действия и делать это максимально быстро.

Что делать

1. Успокойтесь: успеть всё на свете нереально, и в чём-то опоздать — это нормально. В выходные обойдитесь без дедлайнов, отдохните от нехватки времени. Постарайтесь меньше заглядывать в интернет. Сходите погулять — именно погулять, а не пофотографироваться для Instagram. Займитесь хобби: например, поиграйте на гитаре или поучитесь этому, если давно хотели.
2. Найдите способ эффективно распределять время в будни. Например, можно научиться концентрироваться на главном. Да и в целом попробуйте отказываться от всего лишнего.
3. Поменьше читайте новости и ленты социальных сетей. У вас своя жизнь, сосредоточьтесь на ней.
4. Применяйте техники тайм-менеджмента и повышения продуктивности.

Гипотезы о времени

Является ли время непрерывным, неизвестно. Например, есть гипотеза о квантах времени.

Хронон — гипотетический квант времени, неделимая единица времени, используемая в теории, которая принимает его разрыв.

В теории хронона время называется полем хронона. Скалярное поле называют полем хронона.

В стандартной квантовой механике время — непрерывная величина. Но уже в первые годы её формулировки было сделано предположение, что время может подобно энергии состоять из минимальных неделимых интервалов. В этом смысле в 1927 году Роберт Леви предлагает понятие и термин хронон. Первые разработки появляются два десятилетия спустя.

Апории Зенона о движении

Апории Зенона (от др.-греч. ἀπορία «трудность») — внешне парадоксальные рассуждения на тему о движении и множестве древнегреческого философа Зенона Элейского (V век до н. э.). Апории Зенона связаны с противоречием между данными опыта и их мысленным анализом. Современники упоминали более 40 апорий Зенона, до нас дошли девять.

Наиболее известны парадокс «Ахиллес и черепаха» и другие апории Зенона о движении, которые обсуждаются более двух тысячелетий, им посвящены сотни исследований.

Ошибочно воспринимать эти рассуждения как софизмы или полагать, что с появлением высшей математики все апории разрешены. Бертран Рассел писал, что апории Зенона «в

той или иной форме затрагивают основания почти всех теорий пространства, времени и бесконечности, предлагавшихся с его времени до наших дней».

В двух апориях (**Ахиллес** и **Дихотомия**) предполагается, что время и пространство непрерывны и неограниченно делимы. Третья апория («**Стрела**»), напротив, рассматривает время как дискретное, составленное из точек-моментов. Неправильно утверждать, будто Зенон считал движение несуществующим. Цель аргументации Зенона была более узкой: выявить противоречия в позиции оппонента.

Ахиллес и черепаха — одна из апорий древнегреческого философа Зенона Элейского. Споры вокруг этой и других апорий Зенона насчитывают более двух тысячелетий и продолжаются в наши дни.

> ***Быстроногий Ахиллес никогда не догонит неторопливую черепаху, если в начале движения черепаха находится впереди Ахиллеса.***

Допустим, Ахиллес бежит в десять раз быстрее, чем черепаха, и находится позади неё на расстоянии в тысячу шагов. За то время, за которое Ахиллес пробежит это расстояние, черепаха в ту же сторону проползёт сто шагов. Когда Ахиллес пробежит сто шагов, черепаха проползёт ещё десять шагов, и так далее. Процесс будет продолжаться до бесконечности, Ахиллес так никогда и не догонит черепаху.

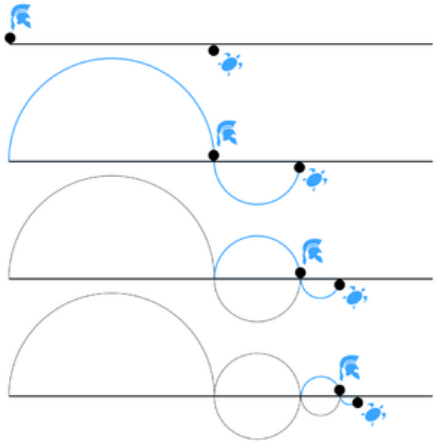

Рис. 2.5.3. Схема движения Ахиллеса и черепахи

Дихотомия — одна из апорий Зенона Элейского, утверждающая логическую противоречивость математической модели движения.

Самая ранняя формулировка данной апории приведена в «Физике» Аристотеля. Название «Дихотомия» (по-гречески: деление пополам) дано Аристотелем там же.

> ***[Идея Зенона] состоит в том, что нет никакого движения, потому что то, что сдвинулось, должно дойти до половины прежде, чем дойти до конца***

Современная формулировка:

> ***Чтобы преодолеть путь, нужно сначала преодолеть половину пути, а чтобы преодолеть половину пути, нужно сначала преодолеть половину половины, и так до бесконечности. Поэтому движение никогда не начнётся.***

Апория «Дихотомия» — парная по отношению к апории «Ахиллес и черепаха», которая, наоборот, доказывает, что движение никогда не закончится.

2.5.2. Нелинейность времени

Некоторые исторические факты

24 февраля 1582 года Папа Римский Григорий XIII издал буллу о переходе на новый, астрономически более верный календарь, который стали именовать «григорианским» в отличие от прежнего, «юлианского», введенного еще в 45 году до нашей эры Юлием

Цезарем. Новый календарь вступил в действие 4 октября 1582 года и сразу на момент принятия сдвинул на 10 дней текущую дату из-за накопившихся ошибок.

Указание перейти на новый календарь сразу приняли к исполнению только католические страны: Италия, Испания, Португалия, Франция. Германские княжества перешли на новый стиль в 1700 году, Англия — в 1752-м, Россия — в 1918-м, Югославия и Греция — в 1925-м, а Вьетнам — только в 1967-м году. Поэтому 1918 год был самым коротким в истории России. Он составлял всего 352 дня.

В 1712 году в Швеции в целях корректировки календаря, в феврале, помимо 29-го, также назначили и 30-е число.

В 1929 году в СССР предлагалось ввести советский революционный календарь, где каждая неделя имела бы пять дней (пятидневки) и каждый месяц длился бы 30 дней, или ровно шесть недель. Оставшиеся 5 или 6 дней становились так называемыми «безмесячными каникулами».

Високосная секунда

Всемирное время (UT) является шкалой времени, основанной на суточном вращении Земли, которое не вполне равномерно на относительно коротких интервалах времени (от дней до столетий), и поэтому любые измерения времени, основанные на такой шкале, не могут иметь точность лучше, чем 10^{-8}.

Для согласования всемирного координированного времени (UTC) со средним солнечным временем UT1 в его шкалу иногда добавляется дополнительная (високосная, скачущая) секунда, или секунда координации.

Объявление о дополнительной секунде осуществляется Международной службой вращения Земли. Теоретически возможно также и вычитание одной секунды, если средние солнечные сутки окажутся короче календарных, при этом после времени 23:59:58 будет следовать 00:00:00 (23:59:59 опускается). На практике отрицательные дополнительные секунды никогда не объявлялись.

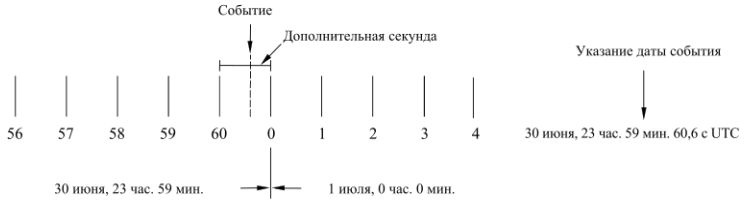

Рис. 2.5.4. Обозначение даты и времени вблизи положительной дополнительной секунды

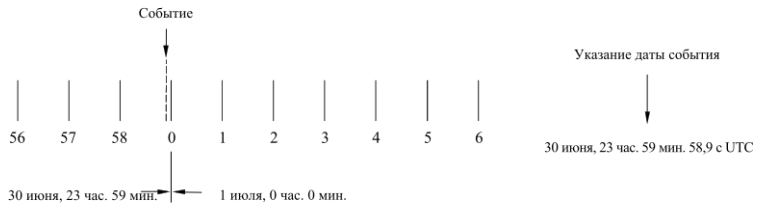

Рис. 2.5.5. Обозначение даты и времени вблизи отрицательной дополнительной секунды

Поскольку вращение Земли постепенно замедляется, разница между средними солнечными сутками и сутками в СИ (составляющими ровно 24 часа) в среднем растёт. Как следствие, в будущем дополнительные секунды надо будет вводить всё чаще и чаще, в каждом следующем веке надо будет вводить примерно на 64 секунды больше, чем в

предыдущем. Так, в XXII веке надо будет вводить в среднем по две секунды в год, а через 2000 лет — примерно раз в месяц.

Год	30 июня 23:59:60	31 декабря 23:59:60
2011		
2012[1]	+1 секунда	
2013		
2014		
2015[2]	+1 секунда	
2016[3]		+1 секунда
2017		
2018		
2019		
2020		

Рис. 2.5.6. Високосные секунды за последние 12 лет

Автономные часы, в том числе часы-календари, игнорируют дополнительную секунду, так как она вводится не по календарному признаку, и заранее запрограммировать расписание невозможно. Иначе обстоят дела с устройствами, имеющими внешнюю синхронизацию. Распространённый протокол синхронизации времени NTP содержит средства информирования о наступающей дополнительной секунде. Спутники системы GPS передают уведомление о наступающей секундной коррекции, но большинство GPS-приёмников работают по протоколу NMEA, который передаёт информацию только об уже наступившей дополнительной секунде. Таким образом, устройство, имеющее внешнюю синхронизацию, может либо учитывать наступление дополнительной секунды, либо синхронизировать собственные часы уже после корректировки.

2.5.3. Календари

С древних времен для измерения длительных промежутков времени использовались Солнце и Луна, так как они влияют на жизнь на земле и периодичность их движения по небу очевидна и предсказуема.

> **Календарь** — система счисления больших промежутков времени, основанная на периодичности движения небесных тел: Солнца — в солнечных календарях, Луны — в лунных календарях и одновременно Солнца и Луны в лунно-солнечных календарях.

Название происходит от латинского слова calendārium — «долговая книга». В Древнем Риме должники платили проценты в дни календ (первых чисел месяца).

Универсальный и идеальный календарь построить невозможно, так как:

- Земля движется вокруг Солнца по эллиптической орбите;
- Земля имеет крупный спутник;
- скорость вращения земли непостоянна;
- период обращения вокруг Солнца не кратен периоду вращения вокруг собственной оси;
- пока Луна совершает полный оборот вокруг Земли, Земля смещается по своей орбите вокруг Солнца, возникает неоднозначность, что считать полным оборотом Луны вокруг Земли.

Каждый народ использовал свои способы датировки исторических событий. Некоторые пытались вести отсчёт лет от сотворения мира: евреи датировали его 3761 годом до н. э., александрийская хронология считала этой датой 25 мая 5493 года до н. э. Римляне

начинали отсчёт от легендарного основания Рима (753 год до н. э.). Парфяне, вифиняне и селевкиды вели отсчёт лет от вступления на трон первого царя, египтяне — с начала правления каждой следующей династии. Свой календарь основывала каждая мировая религия: согласно византийскому календарю, с 14 сентября 2023 года по григорианскому календарю идёт 7532 год «от сотворения мира», в исламском календаре — 1444 год хиджры (с 30 июля 2022 года), по буддийскому календарю идёт 2566 год эры Нирвана, по календарю бахаи — 179 год.

Перевод из одного летосчисления в другое представляет определённые трудности из-за различной продолжительности года и из-за различной даты начала года в разных системах.

Счёт года с 1 января был введён в Риме Юлием Цезарем в 45 году до н. э. (юлианский календарь). На Руси с 1492 года началом года стало считаться не 1 марта, а 1 сентября.

Юлианский календарь установил среднюю продолжительность года в 365,25 суток: обычные годы длились 365 дней, один раз в четыре года (високосный год) — 366 дней.

Развитием юлианского календаря является **григорианский календарь** (новый стиль). Он введён при папе римском Григории XIII 15 октября 1582 года взамен юлианского календаря (старого стиля). Реформа, которую провёл Григорий XIII и признали в большинстве католических стран, состояла из двух частей:

1) Была устранена ошибка в 10 дней, накопившаяся со времён I Вселенского собора (325 год), на котором были установлены правила вычисления христианской Пасхи.

2) На будущее же была введена поправка, обеспечивающая более точное соответствие с солнечным исчислением, которая заключается в том, что из каждых 400 лет должны были быть исключены три високосных года. Таким образом, ошибка в один день накапливается лишь через 3333 года.

Правила определения високосности:

Если номер года заканчивается не на два нуля, то он считается високосным тогда, когда номер года кратен четырём (например, 1996, 2004, 2008 годы).

Если год заканчивается на два нуля, то он високосный только тогда, когда число сотен в нём также кратно четырём (например, 1600, 2000, 2400 годы).

Во всех остальных случаях год считается невисокосным (например, 1900 и 2100 годы).

Единицы, используемые в календаре

Сутки — единица измерения времени, приблизительно равная периоду обращения Земли вокруг своей оси. Обычно под сутками подразумевают астрономическое понятие солнечные сутки. В обиходе сутки часто называют днём. Они длятся 24 часа.

Неделя — единица времени, большая, чем день, и меньшая месяца. Семидневная неделя не является константой, но в большинстве современных календарей, включая Григорианский календарь, неделя включает 7 дней, что делает её самой большой общепринятой единицей времени, содержащей точное количество дней. Неделя не имеет прямой астрономической основы, она широко используется как единица времени.

Месяц — единица измерения времени, связанная с обращением Луны вокруг Земли. Лунные месяцы являются основой многих календарей. Поскольку удобство счёта требует целого числа дней в месяце, а различные периоды обращения луны составляют 27,2—29,6 суток и в целых сутках исчислены быть не могут, календари издавна стремились компенсировать неточность переменной продолжительностью месяцев и/или введением дополнительных дней. В григорианском и юлианском календаре используется фиксированная длина месяца, не связанная со сменой фаз Луны.

Год — условная единица измерения времени, которая исторически означала однократный цикл сезонов (весна, лето, осень, зима). В большинстве стран календарная продолжительность года равна 365 или 366 дням. В настоящее время год употребляется также в качестве временной характеристики обращения планет вокруг звёзд в планетарных системах, в частности Земли вокруг Солнца.

Календарный год в григорианском и юлианском календарях равен 365 суток в невисокосные годы, и 366 суток в високосные годы. Средняя же продолжительность года составляет 365,2425 суток для григорианского и 365,25 суток для юлианского календарей.

Век — единица измерения времени, равная 100 годам. Десять веков составляют **тысячелетие**. Согласно григорианскому календарю, I век н. э. начался 1 января 1 года и закончился 31 декабря 100 года. II век начался в 101 году, III век — в 201 году и т. д. Последний год века начинается с номера этого века (например, 2000 год — последний год XX века). В григорианском календаре нет «нулевого века»: после I века до н. э. начался I век н. э.

200	199	...	102	101	100	99	...	2	1	1	2	...	99	100	101	102	...	199	200
II век до н. э.					I век до н. э.					I век н. э.					II век н. э.				

Рис. 2.5.7. Разбиение годов на века в григорианском календаре

2.5.4. Часовые пояса

Понятие часовой пояс имеет два основных значения:

Географический часовой пояс — условная полоса на земной поверхности шириной ровно 15° (±7,5° относительно среднего меридиана). Средним меридианом нулевого часового пояса считается гринвичский меридиан.

Административный часовой пояс — участок земной поверхности, на котором в соответствии с некоторым законом установлено определённое официальное время. Как правило, в понятие административного часового пояса включается ещё и совпадение даты — в этом случае, например, пояса UTC−10:00 и UTC+14:00 будут считаться различными, хотя в них действует одинаковое время суток.

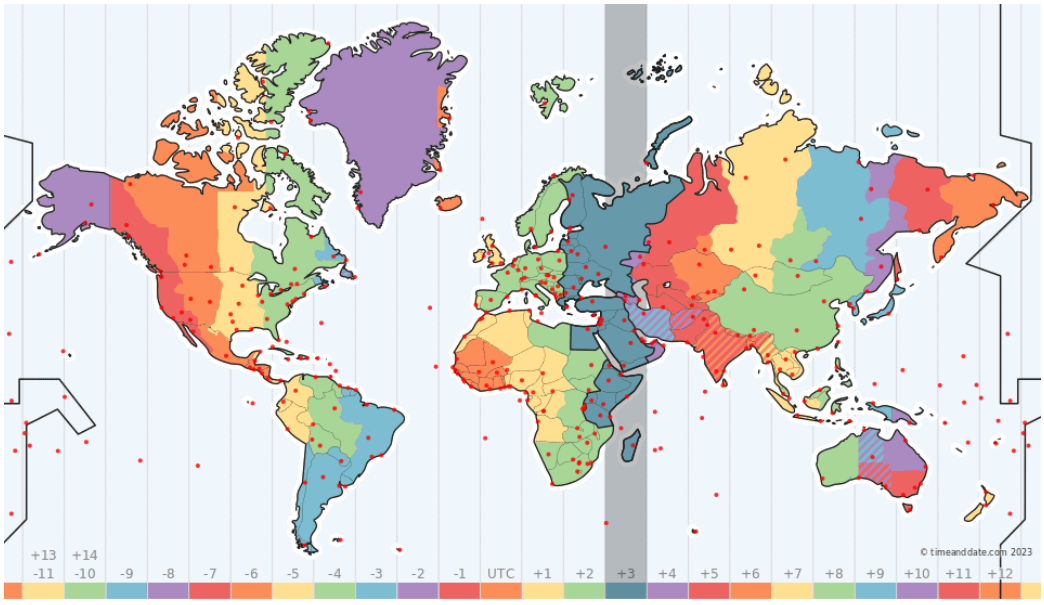

Рис. 2.5.8. Карта часовых поясов (вертикальные линии отделяют географические часовые пояса)

Формирование часовых поясов (часовых зон — time zones) связано со стремлением, с одной стороны, учитывать вращение Земли вокруг своей оси, а с другой стороны,

определить территории (временные зоны) с примерно одинаковым местным солнечным временем таким образом, чтобы различия по времени между ними были кратны одному часу. В результате было достигнуто решение, что должно быть 24 административных часовых пояса и каждый из них должен более или менее совпадать с географическим часовым поясом. За точку отсчёта был принят гринвичский меридиан — нулевой меридиан.

Сейчас поясное время устанавливается относительно всемирного координированного времени (UTC), введённого взамен времени по Гринвичу (GMT). Шкала UTC базируется на равномерной шкале атомного времени (TAI) и более удобна для гражданского использования. Часовые пояса относительно нулевого меридиана выражаются как положительное (к востоку) и отрицательное (к западу) смещение от UTC. Для территорий часового пояса, где используется перевод часов на летнее время, смещение относительно UTC на летний период меняется.

2.5.5. Локальное время и UTC

UTC (Всемирное Координированное Время) - одно из общеизвестных названий для UTC+0 часового пояса, который на 0ч. впереди UTC (Всемирного координированного времени).

Используется как стандартное время. Нет отклонения от Гринвича (UTC+00).

Локальное (местное) время служит для того, чтобы солнечный день приходился приблизительно на один и тот же промежуток времени.

Переход на Летнее время (DST – Daylight Saving (Summer) Time) – сезонный перевод стрелки часов на один час вперёд, который ежегодно производили в последнее воскресенье марта, чтобы получить дополнительный час в светлое время суток, для экономии электроэнергии (на освещение и прочее). Возврат к зимнему времени проводился в последнее воскресение октября. В разных странах даты перехода могут отличаться.

Локальное время в Беларуси на 3 часа больше UTC. Перехода на летнее время нет.

08:30 UTC+03 = 05:30 UTC+00

Время для часовых поясов имеет буквенное обозначение.

В Европе:		
Greenwich Mean Time	GMT +00:00	BST +01:00
Western European Time	WET +00:00	WEST +01:00
Central European Time	CET +01:00	CEST +02:00
Eastern European Time	EET +02:00	EEST +03:00
Moscow Time	MSK +03:00	
В США:		
Eastern Standard Time	EST -05:00	EDT -04:00
Central Standard Time	CST -06:00	CDT -05:00
Mountain Standard Time	MST -07:00	MDT -06:00
Pacific Standard Time	PST -08:00	PDT -07:00
Alaska Standard Time	AKST -09:00	AKDT -08:00
Hawaii-Aleutian Standard Time	HST -10:00	

2.5.6. Эпоха UNIX

Unix-время (также POSIX-время) — система описания моментов во времени, принятая в Unix и других POSIX-совместимых операционных системах [3]. Определяется как количество секунд, прошедших с полуночи (00:00:00 UTC) 1 января 1970 года; этот момент называют **эпохой Unix**.

Unix-время представлено целым числом, которое увеличивается с каждой прошедшей секундой без необходимости вычислений для определения года, месяца, дня, часа или минуты для удобства восприятия человеком. Современное Unix-время согласуется с UTC — отсчет происходит в секундах СИ. Временной промежуток одного дня почти всегда разбит на 86 400 секунд, но при объявлении дополнительных секунд составляет 86 401 секунду. Такие секунды, согласно Всемирному времени, сохраняют длительность дней синхронизированной со временем оборота планеты. В Unix-времени соответствующие номера секунд повторяются, то есть високосные секунды не учитываются.

В программах для хранения Unix-времени используется целочисленный знаковый тип. 32-битные числа со знаком могут ссылаться на моменты времени от пятницы 13 декабря 1901 года 20:45:52 до вторника 19 января 2038 года 03:14:07 включительно.

Чтобы узнать текущее Unix-время в большинстве Unix-подобных систем, можно использовать команду date +%s.

2.5.7. Единицы измерения компьютерного времени

Персональные компьютеры для измерения времени использует или системный таймер (меньшая точность) или часы реального времени (большая точность). Начиная с процессора Pentium появился 64-разрядный регистр TSC (Time Stamp Counter), который считает процессорные циклы с момента его запуска или последнего сброса. Хотя он и позволяет измерять время с максимальным разрешением, на него нельзя полагаться на мультипроцессорных системах или в режиме энергосбережения.

Системный таймер играет более значительную роль для отслеживания хода времени ядром [4]. Независимо от аппаратной платформы, идея, которая лежит в основе системного таймера, одна и та же — это обеспечение механизма управления прерываниями, которые возникают периодически с постоянной частотой. Для некоторых аппаратных платформ это реализуется с помощью электронных часов, которые генерируют колебания с программируемой частотой. В других аппаратных платформах используется декрементный счетчик (decrementer), куда можно записать некоторое начальное значение, которое будет периодически, с фиксированной частотой, уменьшаться на единицу, пока значение счетчика не станет равным нулю. Когда значение счетчика становится равным нулю, генерируется прерывание. В любом случае эффект получается один и тот же.

Для аппаратной платформы x86 главный системный таймер — это программируемый интервальный таймер (programmable interval timer, PIT). Таймер PIT существует на всех машинах платформы PC. Со времен операционной системы DOS он используется для управления прерываниями. Ядро программирует таймер PIT при загрузке, для того чтобы периодически генерировать прерывание номер нуль с частотой HZ. Этот таймер— простое устройство с ограниченными возможностями, но, тем не менее, хорошо выполняющее свою работу. Другие эталоны времени для аппаратной платформы x86 включают таймер APIC (Advanced Programmable Interrupt Controller, расширенный программируемый контроллер прерываний) и счетчик отметок времени (TSC, Time Stamp Counter).

Часы реального времени (ЧРВ, RTC — Real Time Clock) — электронная схема, предназначенная для учёта хронометрических данных (текущее время, дата, день недели и др.), представляет собой систему из автономного источника питания и учитывающего устройства. Чаще всего часы реального времени встречаются в вычислительных машинах, хотя на самом деле ЧРВ присутствуют практически во всех электронных устройствах, которые должны хранить время.

Название «real-time clock» используется в английском языке для различения от тактовых генераторов (которые в английском языке называются «clock signals»). Тактовые генераторы не ведут счёт в «человеческих» единицах исчисления времени.

Применение специализированной схемы для ЧРВ позволяет добиться более низкого энергопотребления, освобождения центрального процессора для критичных по времени задач, обеспечить более высокую точность.

Большинство ЧРВ использует кварцевый резонатор, но некоторые используют частоту питающей сети. В большинстве случаев используется кварцевый резонатор на частоте 32768 Гц. Та же частота используется в кварцевых часах. Такая частота обеспечивает 215 циклов в секунду, что очень удобно для простых двоичных счётчиков.

HPET (High Precision Event Timer, таймер событий высокой точности) — тип таймера, используемый в PC-совместимых компьютерах. Совместно разработан Intel и Microsoft, и стал внедряться в чипсеты с 2005 года. Ранее назывался Intel как мультимедийный таймер. Название HPET было выбрано для исключения путаницы с программными таймерами, появившимися в Windows 3.0, которые также назывались Multimedia Timers.

Очень старые операционные системы не поддерживают HPET и могут использовать только старые таймеры (наподобие RTC).

Блок HPET состоит из независимых возрастающих счётчиков (от 3 до 32 в блоке) с фиксированной частотой счёта. Каждый счётчик состоит из компаратора и регистра с пороговым значением. Всего может быть до 8 блоков HPET. Каждый компаратор может вызывать прерывание в тот момент, когда счётчик достигает заранее заданного значения в регистре.

Каждый таймер может быть настроен в режим единичного срабатывания или в периодический режим. В режиме единичного срабатывания (one-shot mode) таймер вызывает прерывание единожды (в момент достижения значения в регистре), в периодическом режиме после прерывания таймер начинает отсчёт заново, генерируя прерывания через заданные интервалы времени.

Счетчик меток времени (TSC) — это 64-битный регистр, присутствующий во всех процессорах x86, начиная с Pentium. Он подсчитывает количество циклов процессора с момента его сброса. Инструкция RDTSC возвращает TSC в EDX:EAX. В режиме x86-64 RDTSC также очищает старшие 32 бита RAX и RDX.

Счетчик отметок времени когда-то был отличным способом получения программами информации о синхронизации процессора с высоким разрешением и низкими издержками. С появлением многоядерных/гиперпотоковых процессоров, систем с несколькими процессорами и спящих операционных систем нельзя полагаться на TSC в предоставлении точных результатов — если не уделять особого внимания исправлению возможных недостатков: скорости такта и все ядра (процессоры) имеют одинаковые значения в своих регистрах хранения времени. Нет никаких гарантий, что счетчики временных меток нескольких процессоров на одной материнской плате будут синхронизированы. Следовательно, программа может получить надежные результаты, только ограничившись выполнением на одном конкретном процессоре. Даже в этом случае скорость ЦП может измениться из-за мер по энергосбережению, принятых ОС или BIOS, или система может перейти в режим гибернации, а затем возобновить работу со сбросом TSC. В последних случаях, чтобы оставаться актуальной, программа должна периодически перекалибровывать счетчик. Счетчик меток времени (TSC) — это 64-битный регистр, присутствующий во всех процессорах x86, начиная с Pentium. Он подсчитывает количество циклов процессора с момента его сброса. Инструкция RDTSC возвращает TSC в EDX:EAX. В режиме x86-64 RDTSC также очищает старшие 32 бита RAX и RDX.

Счетчик отметок времени когда-то был отличным способом получения программами информации о синхронизации процессора с высоким разрешением и низкими издержками. С появлением многоядерных/гиперпотоковых процессоров, систем с несколькими процессорами и спящих операционных систем нельзя полагаться на TSC в предоставлении точных результатов — если не уделять особого внимания исправлению возможных недостатков: скорости такта и все ядра (процессоры) имеют одинаковые значения в своих регистрах хранения времени. Нет никаких гарантий, что счетчики временных меток нескольких процессоров на одной материнской плате будут синхронизированы. Следовательно, программа может получить надежные результаты, только ограничившись выполнением на одном конкретном процессоре. Даже в этом случае скорость ЦП может измениться из-за мер по энергосбережению, принятых ОС или BIOS, или система может перейти в режим гибернации, а затем возобновить работу со сбросом TSC. В последних случаях, чтобы оставаться актуальной, программа должна периодически перекалибровывать счетчик.

Время Windows

Время Windows — это количество времени в миллисекундах, прошедшее с момента последнего запуска системы [5]. Этот формат в первую очередь предназначен для обеспечения обратной совместимости с 16-разрядной версией Windows. Чтобы обеспечить успешное выполнение приложений, предназначенных для 16-разрядной версии Windows, функция **GetTickCount** возвращает текущее время Windows.

Обычно функция **GetTickCount** или **GetTickCount64** используется для сравнения текущего времени Windows с временем, возвращаемым функцией **GetMessageTime**. **GetMessageTime** возвращает время создания указанного сообщения в Windows. **GetTickCount** и **GetTickCount64** ограничены разрешением системного таймера, которое составляет от 10 до 16 миллисекунд. Затраченное время, полученное **GetTickCount** или **GetTickCount64**, включает время, которое система проводит в спящем режиме или гибернации.

Если вам нужен таймер с более высоким разрешением, используйте функцию **QueryUnbiasedInterruptTime**, таймер мультимедиа или таймер с высоким разрешением. Время, затраченное функцией **QueryUnbiasedInterruptTime**, включает только время, которое система проводит в рабочем состоянии.

С помощью счетчика производительности Время работы системы можно получить количество секунд, прошедших с момента запуска компьютера. Этот счетчик производительности можно получить из данных о производительности в разделе реестра HKEY_PERFORMANCE_DATA. Возвращаемое значение представляет собой 8-байтовое значение.

Функция GetTickCount (sysinfoapi.h)

Извлекает количество миллисекундах, прошедших с момента запуска системы, до 49,7 дней.

Синтаксис

```
DWORD GetTickCount();
```

Разрешение функции **GetTickCount** ограничено разрешением системного таймера, которое обычно находится в диапазоне от 10 до 16 миллисекундах. На разрешение функции **GetTickCount** не влияют корректировки, внесенные функцией **GetSystemTimeAdjustment**.

Затраченное время сохраняется в виде значения DWORD. Таким образом, время будет равно нулю, если система непрерывно работает в течение 49,7 дней. Чтобы избежать

этой проблемы, используйте функцию **GetTickCount64**. В противном случае проверка для условия переполнения при сравнении времени.

Время прерывания

Время прерывания — это время с момента последнего запуска системы в интервалах по 100 наносекунд. Счетчик времени прерываний начинается с нуля при запуске системы и увеличивается при каждом прерывании часов на длину такта часов. Точная длина тактов часов зависит от базового оборудования и может отличаться в разных системах [6].

В отличие от системного времени, количество прерываний не подлежит корректировке пользователями или службой времени Windows, что делает его лучшим выбором для измерения коротких длительностей. Приложения, которым требуется более высокая точность, чем количество прерываний, должны использовать таймер с высоким разрешением. Используйте функцию **QueryPerformanceFrequency** для получения частоты таймера высокого разрешения и функцию **QueryPerformanceCounter** для получения значения счетчика.

Для получения количества прерываний можно использовать функции **QueryInterruptTime**, **QueryInterruptTimePrecise**, **QueryUnbiasedInterruptTime** и **QueryUnbiasedInterruptTimePrecise**. Беспристрастное время прерывания означает, что учитывается только время, когда система находится в рабочем состоянии, поэтому счетчик времени прерывания не является "смещенным" по времени, которое система проводит в спящем режиме или гибернации.

Функция GetSystemTimeAsFileTime (sysinfoapi.h)

Извлекает текущую системную дату и время. Сведения отображаются в формате UTC.

Синтаксис

```
void GetSystemTimeAsFileTime(
  [out] LPFILETIME lpSystemTimeAsFileTime );
```

Параметры

[out] **lpSystemTimeAsFileTime**

Указатель на структуру FILETIME для получения текущей системной даты и времени в формате UTC.

Теоретическая точность такого времени 100 нс, но в действительности это значение получается на основе системного таймера, то есть, практическая точность не превышает одной мс.

Для справки:

1 мс = 0,001 с

100 нс = 0,0000001 с

Функция QueryPerformanceCounter (profileapi.h)

Извлекает текущее значение счетчика производительности, представляющее собой метку времени с высоким разрешением (<1 мкс), которую можно использовать для измерения интервалов времени.

Синтаксис

```
BOOL QueryPerformanceCounter(
  [out] LARGE_INTEGER *lpPerformanceCount );
```

Параметры

[out] **lpPerformanceCount**

Указатель на переменную, которая получает текущее значение счетчика производительности в счетчиках («тактах» процессора).

QPC не зависит от внешних ссылок на время и не синхронизируется с ней. Чтобы получить метки времени, которые можно синхронизировать с внешней ссылкой на время, например, в формате UTC, для использования в измерениях времени с высоким разрешением, используйте **GetSystemTimePreciseAsFileTime**.

Счетчики производительности

Майкрософт настоятельно не рекомендует использовать инструкцию процессора **RDTSC** или **RDTSCP** для прямого запроса TSC (Time Stamp Counter), так как вы не получите надежные результаты в некоторых версиях Windows, при динамической миграции виртуальных машин и в аппаратных системах без инвариантных или тесно синхронизированных TSC. Вместо этого рекомендуется использовать QPC для использования абстракции, согласованности и переносимости, которые он предлагает.

RDTSC (англ. Read Time Stamp Counter) — ассемблерная инструкция для платформ x86 и x86_64, читающая счётчик TSC (Time Stamp Counter) и возвращающая в регистрах EDX:EAX 64-битное количество тактов с момента последнего сброса процессора. **RDTSC** поддерживается в процессорах Pentium (и совместимых с ними) и более новых.

Функция GetSystemTimePreciseAsFileTime (sysinfoapi.h)

Функция **GetSystemTimePreciseAsFileTime** извлекает текущую системную дату и время с максимально возможной точностью (<1 мкс). Полученные сведения отображаются в формате UTC.

Синтаксис

```
void GetSystemTimePreciseAsFileTime(
  [out] LPFILETIME lpSystemTimeAsFileTime );
```

Параметры

[out] **lpSystemTimeAsFileTime**

Указатель на структуру FILETIME, содержащую текущую системную дату и время в формате UTC.

Локальное время в Windows

Хотя система внутренне использует время в формате UTC, в приложениях обычно отображается местное время, которое является датой и временем суток для вашего часового пояса. Поэтому, чтобы обеспечить правильные результаты, необходимо знать, ожидает ли функция получать время в формате UTC или местное время, а также возвращает ли функция время в формате UTC или местное время.

Текущие параметры часового пояса управляют преобразованием системы между временем в формате UTC и местным временем. Текущие параметры часового пояса можно получить с помощью функции **GetTimeZoneInformation**. Функция копирует результат в структуру TIME_ZONE_INFORMATION и возвращает значение, указывающее, является ли местное время в настоящее время стандартным или переходом на летнее время (DST). Параметры часового пояса можно задать с помощью функции **SetTimeZoneInformation**. Для поддержки границ летнего времени, которые меняются из года в год, используйте функции **GetTimeZoneInformationForYear**, **GetDynamicTimeZoneInformation** и **SetDynamicTimeZoneInformation**.

Чтобы получить местное время, используйте функцию **GetLocalTime**. Чтобы преобразовать время в формате UTC в местное время, используйте функцию **SystemTimeToTzSpecificLocalTime**.

Функция GetLocalTime (sysinfoapi.h)

Извлекает текущую локальную дату и время.

Синтаксис

```
void GetLocalTime(
  [out] LPSYSTEMTIME lpSystemTime );
```

Параметры

[out] lpSystemTime

Указатель на структуру SYSTEMTIME для получения текущей локальной даты и времени.

Структура SYSTEMTIME (minwinbase.h)

Указывает дату и время, используя отдельные элементы для месяца, дня, года, дня недели, часа, минуты, секунды и миллисекунд. Время определяется в формате UTC или местном времени в зависимости от вызываемой функции.

```
typedef struct _SYSTEMTIME {
  WORD wYear;          // Допустимые значения — от 1601 до 30827.
  WORD wMonth;         // 1 – январь, 12 - декабрь
  WORD wDayOfWeek;     // 0 – воскресенье, 1 - понедельник
  WORD wDay;
  WORD wHour;
  WORD wMinute;
  WORD wSecond;
  WORD wMilliseconds;
} SYSTEMTIME, *PSYSTEMTIME,   *LPSYSTEMTIME;
```

Структура FILETIME (minwinbase.h)

Содержит 64-разрядное значение, представляющее число 100-наносекундных интервалов с 1 января 1601 г. (UTC).

```
typedef struct _FILETIME {
  DWORD dwLowDateTime;
  DWORD dwHighDateTime;
} FILETIME, *PFILETIME, *LPFILETIME;
```

Чтобы преобразовать структуру FILETIME в время, которое легко отобразить пользователю, используйте функцию **FileTimeToSystemTime**.

Для получения относительного времени не рекомендуется добавлять и вычитать значения из структуры FILETIME. Вместо этого следует скопировать части файлового времени низкого и высокого порядка в ULARGE_INTEGER структуру, выполнить 64-разрядную арифметику для элемента QuadPart и скопировать элементы LowPart и HighPart в структуру FILETIME.

Не приводите указатель на структуру FILETIME к значению ULARGE_INTEGER* или __int64*, так как это может привести к сбоям выравнивания в 64-разрядной версии Windows.

2.5.8. Функции POSIX для работы с временем

Календарное время включает в себя год, месяц, день месяца и текущее время суток. Для хранения календарного времени используется тип **time_t**. В POSIX-системах календарное время хранится как количество секунд, прошедшее от «начала эпохи», причём для простоты считается, что каждый 4-й год всегда високосный. Эпоха началась (для UNIX) в полночь 1 января 1970 г. по Гринвичу. Время отсчитывается по часовому поясу Гринвичского меридиана без перехода на летнее время (так называемое «Универсальное координированное время» UTC — Coordinated Universal Time) [7].

В настоящее время в POSIX-системах тип **time_t** эквивалентен типу **long**. Диапазона значений типа **long** (32 бита) достаточно для представления дат, лежащих в отрезке длиной примерно 136 лет. Поскольку отрицательные числа используются для представления дат до 1970 года, переполнение текущего представления типа **time_t** может произойти в 2038 году, если, конечно, до этого момента представление не будет изменено.

Тип struct tm

Для более удобного представления календарного времени используется тип struct tm, определённый следующим образом.

```
struct tm
{
int tm_sec;   /* секунды [0-60] */
int tm_min;   /* минуты [0-59] */
int tm_hour;  /* час [0-23] */
int tm_mday;  /* день месяца [1-31] */
int tm_mon;   /* месяц [0-11], 0 = январь */
int tm_year;  /* число лет от 1900 г */
int tm_wday;  /* день недели [0-6], 0 = воскр. */
int tm_yday;  /* день года [0-365] */
int tm_isdst; /* летнее время */ };
```

Поле tm_sec обычно хранит значения в интервале от 0 до 59, но может хранить и 60, когда для коррекции времени в календарь вставляется дополнительная секунда.

Функции

В стандарте ANSI C также определяется набор функций для манипулирования с календарным временем. Прототипы этих функций определены в заголовочном файле <time.h>.

Функция **time** позволяет получить текущее системное календарное время. Функция имеет следующий прототип:

```
time_t time(time_t *pval);
```

Функция возвращает текущее календарное время. Кроме того, если указатель pval ненулевой, функция записывает текущее время по указанному адресу.

Функции **localtime**, **gmtime** преобразовывают дату из секундного представления **time_t** в структурное представление **struct tm**. Они имеют следующие прототипы:

```
struct tm *gmtime(const time_t *timep);
struct tm *localtime(const time_t *timep);

extern char *tzname[2];
long int timezone;
```

Функция **gmtime** конвертирует календарное время **timep** в развёрнутое структурное представление, выражающее универсальное координированное время (UTC), т. е. время Гринвичского меридиана без перехода на летнее время. Функция возвращает указатель на статическую область памяти, которая переписывается при каждом вызове функций работы с календарным временем. Поэтому при необходимости эту структуру нужно скопировать в другое место.

Функция **localtime** конвертирует календарное время timep в развёрнутое структурное представление в определённом пользователем или системой часовом поясе. Функция устанавливает переменную tzname в имя текущего часового пояса (tzname[0] — имя часового пояса зимнего времени, tzname[1] — имя часового пояса летнего времени), переменную timezone в разницу в секундах между поясным временем и универсальным

координированным временем (UTC). Функция возвращает указатель на статическую область памяти, которая переписывается при каждом вызове функций работы с календарным временем. Поэтому при необходимости эту структуру нужно скопировать в другое место.

Функция **mktime** конвертирует календарное время местного часового пояса в развёрнутом структурном формате в секундное представление. Функция имеет следующий прототип:

```
time_t mktime(struct tm *timeptr);
```

Функция игнорирует содержимое полей tm_wday, tm_yday, tm_isdst и перевычисляет их, используя оставшиеся поля структуры. Если значение какого-либо поля структуры находится за пределами допустимых значений, оно нормализуется, например, 40 октября становится 9 ноября.

Вызов **mktime** заполняет переменную tzname информацией о местном часовом поясе. Если заданное структурное представление не может быть преобразовано в посекундное представление, функция возвращает значение `(time_t)(-1)` и не изменяет значения полей tm_wday и tm_yday.

Функции **ctime** и **asctime** конвертируют календарное время в символьную строку. Они имеют следующие прототипы:

```
char *asctime(const struct tm *timeptr);
char *ctime(const time_t *timep);
```

Функция **ctime** преобразует календарное время **timep** в строку вида

"Tue Nov 7 11:32:11 2000\n"

Дни недели сокращаются до ‘Sun’, ‘Mon’, ‘Tue’, ‘Wed’, ‘Thu’, ‘Fri’, ‘Sat’. Имя месяца сокращается до ‘Jan’, ‘Feb’, ‘Mar’, ‘Apr’, ‘May’, ‘Jun’, ‘Jul’, ‘Aug’, ‘Sep’, ‘Oct’, ‘Nov’, ‘Dec’. Возвращаемое значение указывает на статическую область памяти, которая переписывается при вызовах функций **ctime** и **asctime**. Функция заполняет переменную tzname информацией о местном часовом поясе.

Функция **asctime** преобразовывает развёрнутое структурное представление **timeptr** в строку того же самого формата, что функция **ctime**. Возвращаемое значение указывает на статическую область памяти, которая переписывается при вызовах функций **ctime** и **asctime**. Функция заполняет переменную tzname информацией о местном часовом поясе.

Функция **strftime** позволяет получать строковое представление времени согласно заданной форматной строке. Прототип следующий:

```
size_t strftime(char *s, size_t max, const char *format,
const struct tm *tm);
```

Функция форматирует время в развёрнутом структурном представлении **tm** в соответствии со спецификацией формата **format** и помещает результат в символьный массив s размера max.

Обычные символы копируются из форматной строки в строку s без преобразования.

Спецификаторы формата начинаются со знака ’%’ и приведены в таблице.

Функция возвращает количество символов, записанных в массив s, не считая последнего символа ’\0’ (то есть, длину строки s), при условии, что длина получившейся строки s меньше max. В противном случае функция возвращает 0, а содержимое массива s не определено.

Спецификаторы формата функции strftime

%a сокращённое имя дня недели (например, Tue)

%A полное имя дня недели (например, Tuesday)

%b сокращённое имя месяца (например, Nov)

%B полное имя месяца (например, November)

%c дата и время (например, Nov 7 11:32:11 2000)

%d день месяца (например, 07)

%H час в 24-часовой шкале (от 00 до 23)

%I час в 12-часовой шкале (от 01 до 12)

%j день года (от 001)

%m месяц года (от 01)

%M минуты в часе (00–59)

%p индикатор AM/PM

%S секунды в минуте (00–59)

%U неделя (считая от воскресенья) в году (от 00)

%w день недели (0 — воскресенье)

%W неделя (считая от понедельника) в году (от 00)

%x дата (например, Nov 7 2000)

%X время (например, 11:32:11)

%y год в веке (00-99)

%Y год (например, 2000)

%Z имя часового пояса (если определено) (например, MSK)

%% символ %

Время и таймеры POSIX.1b

Традиционные обычные таймеры UNIX в Linux, функции **setitimer** и **getitimer**, не отвечают требованиям большинства приложений реального времени. Таймеры POSIX.1b имеют следующие преимущества по сравнению с обычными таймерами UNIX [8].

- Процесс может иметь несколько таймеров.
- Лучшая точность таймеров. Таймеры могут использовать значения с точностью до наносекунд.
- Уведомление о завершении работы таймера может быть сделано либо с помощью любого произвольного сигнала (реального времени) или с помощью потоков. Для уведомления о завершении работы таймера в обычных таймерах UNIX есть лишь ограниченный набор сигналов.
- Таймеры POSIX.1b предоставляют поддержку различных часов, таких как CLOCK_REALTIME, CLOCK_MONOTONIC, и так далее, которые могут иметь различные источники с различным разрешением. Обычный таймер UNIX, с другой стороны, связан с системными часами.

Ядро таймеров POSIX.1b представляет собой набор часов, которые используются как привязка ко времени. Linux обеспечивает поддержку следующих часов:

- **CLOCK_REALTIME**: общесистемные часы реального времени, видимые для всех процессов, работающих в системе. Часы измеряют количество времени в секундах и наносекундах с начала эпохи. Разрешение часов равно 1/**HZ** секунд. Таким образом, если **HZ** равно 100, то разрешение часов составляет 10 мс. Если **HZ** равно 1000, то разрешение часов составляет 1 мс. Так как это время базируются на времени настенных часов, оно может быть изменено.
- **CLOCK_MONOTONIC**: время непрерывной работы системы, видимое всем процессам в системе. В Linux оно измеряется как количество времени в секундах

и наносекундах после загрузки системы. Его разрешение равно 1/**HZ** с. Это время не может быть изменено каким-либо процессом.

- **CLOCK_PROCESS_CPUTIME_ID**: часы, измеряющие время работы процесса. Время текущего процесса, потраченное на выполнение в системе, измеряется в секундах и наносекундах. Разрешение равно 1/**HZ**. Это время может быть изменено.
- **CLOCK_THREAD_CPUTIME_ID**: То же, что и **CLOCK_PROCESS_CPUTIME_ID**, но для текущего потока.

Обычно CLOCK_REALTIME используется для указания абсолютного времени ожидания. CLOCK_MONOTONIC используется для относительного времени ожидания и периодических задач. Поскольку время этих часов не может быть изменено, периодическим задачам не нужно беспокоиться о преждевременном или задержанном пробуждении, которое могло бы произойти с CLOCK_REALTIME. Двое других часов могут использоваться для целей учёта.

HZ – частота системного таймера (обычно, 100, 250, 1000), параметр ядра. Чтобы узнать значение HZ в вашей системе, посмотрите файл <kernel-source>/include/asm/param.h.

Интерфейсы времени и таймеров POSIX.1b перечислены в следующей таблице.

Метод	*Описание*
clock_settime	Установка значения указанным часам.
clock_gettime	Получение времени.
clock_getres	Получение разрешения (точности) часов.
clock_nanosleep	Приостановка выполнения вызывающего приложения на указанное время.
timer_create	Создание таймера на основе указанного времени.
timer_delete	Удаление таймера.
timer_settime	Установка времени работы таймера.
timer_gettime	Получение текущего значения таймера.
timer_getoverrun	Возвращает число, показывающее сколько раз таймер закончил работу между моментом генерации сигнала и его доставкой

Рис. 2.5.9. Интерфейсы времени и таймеров POSIX.1b

2.5.9. Таймеры ожидания

Объект таймера ожидания — это объект синхронизации, состояние которого по достижении указанного срока устанавливается в значение Signaled. Существует два типа таймеров ожидания, которые можно создать: сброс вручную и синхронизация. Таймер любого типа также может быть периодическим [9].

Таймер сброса вручную

Таймер, состояние которого остается сигнальным до вызова `SetWaitableTimer`, чтобы установить новое время выполнения.

Таймер синхронизации

Таймер, состояние которого остается сигнальным до тех пор, пока поток не завершит операцию ожидания в объекте таймера.

Периодический таймер

Таймер, который повторно активируется каждый раз, когда истечет указанный период, пока таймер не будет сброшен или отменен. Периодический таймер — это либо периодический таймер сброса вручную, либо периодический таймер синхронизации.

Поток использует функцию **CreateWaitableTimer** или **CreateWaitableTimerEx** для создания объекта таймера. Поток создания указывает, является ли таймер таймером сброса вручную или таймером синхронизации. Создающий поток может указать имя

объекта таймера. Потоки в других процессах могут открывать дескриптор для существующего таймера, указывая его имя в вызове функции **OpenWaitableTimer**. Любой поток с дескриптором объекта таймера может использовать одну из функций ожидания для ожидания, пока состояние таймера будет задано как сигнальное.

Поток вызывает функцию **SetWaitableTimer** для активации таймера.

Поток может использовать функцию **CancelWaitableTimer** для установки таймера в неактивное состояние. Чтобы сбросить таймер, вызовите **SetWaitableTimer**. Завершив работу с объектом таймера, вызовите **CloseHandle**, чтобы закрыть дескриптор для объекта таймера.

Диаграмма состояний таймера ожидания приведена на рис. 2.5.10.

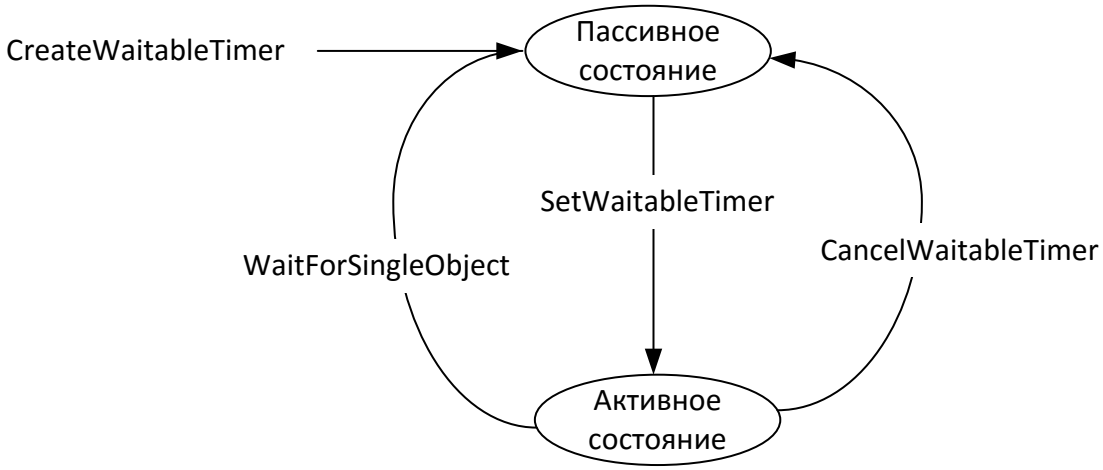

Рис. 2.5.10. Диаграмма состояний таймера ожидания

2.5.10. Протокол NTP

Атомные часы — это достаточно сложное и дорогостоящее устройство, требующее квалифицированного обслуживания. По этой причине пользователи вынуждены обращаться к услугам удаленных эталонов. Всеобщее распространение Интернета потребовало синхронизации работы различных процессов в серверах и программах клиента. Для этого используется сетевой протокол задания времени NTP. Он предусматривает возможности работы с иерархически распределенными первичными эталонами (такими как синхронизуемые радиочасы).

NTP в основном используется в следующих сценариях, когда часы всех сетевых устройств должны быть согласованы:

- **Управление сетью**: Синхронизированное время используется в качестве эталона, когда система управления сетью (NMS) анализирует журналы и отладочную информацию, собранную с разных устройств.
- **Система начисления платы**: для обеспечения точности и достоверности информации о начислении платы требуется синхронизированное время.
- **Несколько систем взаимодействуют по одному и тому же сложному событию**: системы должны использовать одни и те же часы в качестве эталона, чтобы обеспечить правильную последовательность операций.
- **Инкрементное резервное копирование между сервером резервного копирования и клиентом**: синхронизированное время обеспечивает целостность данных резервного копирования, которые можно использовать для восстановления производственной системы.

Принцип работы протокола NTP

Клиент и сервер добавляют к сообщению метки времени при каждом отправлении и получении сообщения. В результате при получении ответа от сервера клиент имеет четыре метки времени: T1, T2, T3 и T4.

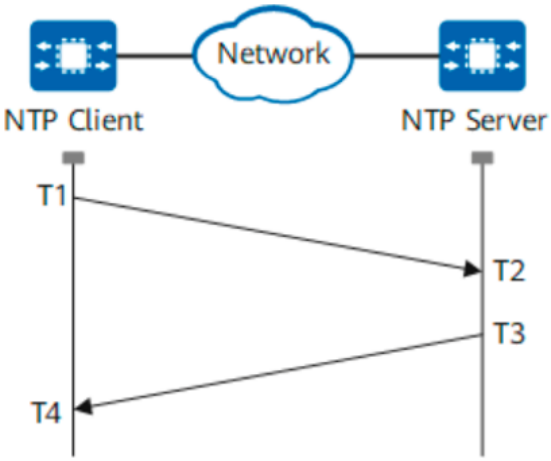

Рис.2.5.11. Метки времени

В примере предполагается, что односторонняя задержка канала равна **Delay**, а разница во времени между клиентом NTP и сервером NTP равна **Offset** (текущее время сервера NTP минус время клиента NTP). Формулы расчета следующие:

T2 - T1 = Delay + Offset

T4 - T3 = Delay - Offset

Следовательно, смещение рассчитывается следующим образом: **Offset = [(T2 - T1) - (T4 - T3)]/2**. Клиент NTP настраивает свои часы на основе значения смещения для достижения синхронизации с сервером NTP.

Сетевая архитектура NTP

Подсеть синхронизации NTP представляет собой сеть первичных и вторичных серверов времени, клиентов и взаимосвязанных путей передачи.

Основной сервер времени напрямую синхронизируется с основным эталонным источником, обычно радиочасами или глобальной системой позиционирования (GPS). Дополнительный сервер времени получает синхронизацию через первичный сервер времени или другие вторичные серверы времени и использует NTP для передачи информации о времени другим хостам в локальной сети (LAN).

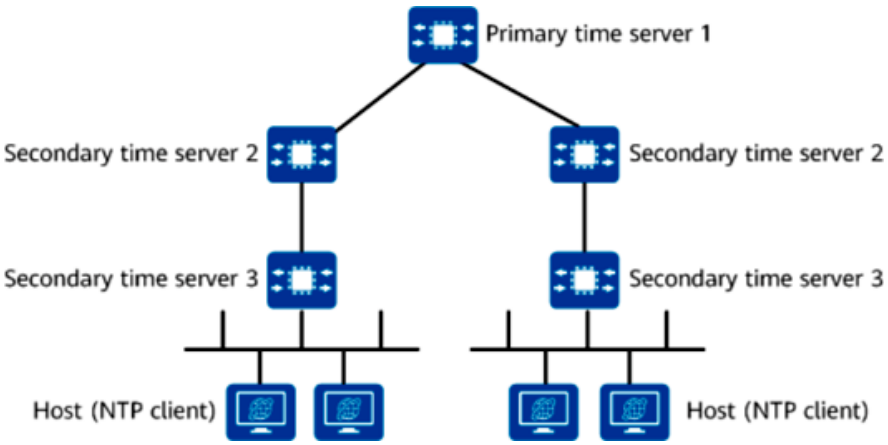

Рис.2.5.12. Сетевая архитектура NTP

Список использованных источников

1. Что такое социальное время и почему нам стало сложнее всё успевать
https://lifehacker.ru/socialnoe-vremya/

2. Лекция 4. Структура социального времени.
http://socio.isu.ru/ru/chairs/ksf/courses/SSV/kurs_lekzij/l_4.html

3. Unix-время
https://ru.wikipedia.org/wiki/Unix-время

4. Системный таймер
https://it.wikireading.ru/1855

5. Служба времени Windows
https://learn.microsoft.com/ru-ru/windows/win32/sysinfo/windows-time

6. Время прерывания
https://learn.microsoft.com/ru-ru/windows/win32/sysinfo/interrupt-time

7. Работа с календарным временем и интервалами времени
https://ejudge.ru/study/3sem/time.pdf

8. 7.3.7 Время и таймеры POSIX.1b
https://dmilvdv.narod.ru/Translate/ELSDD/elsdd_posix_1b_clock_and_timers.html

9. Объекты таймера, доступные для ожидания
https://learn.microsoft.com/ru-ru/windows/win32/sync/waitable-timer-objects