(Preprint) AAS 11-681

# TIMEKEEPING SYSTEM IMPLEMENTATIONS: OPTIONS FOR THE *PONTIFEX MAXIMUS*

**Steven L. Allen** *

A representation for the meaning of time and the rules for handling it is built into many operational systems – civil, legal, hardware, software, etc. Many of these systems have avoided implementing the complexity required to handle leap seconds, yet some demand their existence. A plausible change to the scheme of UTC must be compatible with existing systems and should be easy to implement. I propose a small change to the representation of leap seconds which allows the `tz` code to describe them in a way that alleviates the underlying problems with information processing systems. It preserves the traditional meaning of civil time as earth rotation. It allows for trivial testing of the effects of leap seconds on software and hardware systems. It is a compromise that gives easy access to all forms of time information. It is not without consequences that will have to be handled.

## WHO IS THE *PONTIFEX MAXIMUS*?

The calendar of the Roman Republic had months which had already abandoned any basis in astronomical observations of the moon. Months, and thus the Roman calendar dates and year, were decided by the *Pontifex Maximus.* Julius Caesar employed astronomer Sosigenes of Alexandria to re-conform the calendar with the sun. Augustus remedied a problem with the leap year implementation and produced a predictable progression of civil dates which closely tracked the sun for over 1000 years. Pope Gregory XIII employed astronomer Christopher Clavius to bring the calendar even closer to the sun, but the costs were discontinuity of 10 days and lack of international consensus for centuries after.

**Figure 1. Civil Authorities: Augustus Caesar, Pope Gregory XII, and the 1884 International Meridian Conference changed the rules of the calendar**

*UCO/Lick Observatory, 1156 High Street, Santa Cruz, CA 95064

**Figure 2. Astronomers: Sosigenes (Hume Cronyn), Christopher Clavius, and Simon Newcomb provided the mathematics for the calendar**

The Lords Commissioners of the Admiralty consulted the Royal Astronomical Society and decreed that the Nautical Almanac would tabulate mean solar time instead of apparent.[1] The 1884 International Meridian Conference resolved the basis by which the day is related to the sun,[2] and Simon Newcomb produced the mathematical details for subdividing the day.[3] Astronomers, horologists, national metrology institutes and broadcast engineers provided time signals, the Bureau International de l'Heure (BIH) analyzed those, and the Consultative Committee on International Radio (CCIR) documented their best efforts in Recommendations.* Physicists produced atomic chronometers,[4] broadcast engineers immediately employed them, astronomers raced to calibrate them,[5] and the CCIR struggled to find an acceptable balance between technologies.[6,7,8]

All of these actors have played the role of *Pontifex Maximus*. In the thespian slang, most of these plays have been *two-handers* – one actor with political power, and one with technical ability. The role of the rest of humanity has been audience. The details of precision timekeeping, and thus the decree that a new day had started, were only available to a few.

## TIME IMPLEMENTATIONS IN COMPUTING

At the time the CCIR recommended that radio broadcast time signals should have leap seconds there were few devices which could keep a continuous count of seconds. Most time keeping devices continued the tradition of approximate subdivision of days. Today most humans have routine encounters with a device that counts seconds; many of us wear them throughout the day. This does not mean that the control over time has been democratized, but the count of seconds, and failure to handle that count, is more immediately apparent than the count of days.

Among the many computer implementations of time, Unix prevails. By the 1980s the "open" systems had converged on a system clock counting seconds since 1970-01-01T00:00:00, and this was incorporated into the first Portable Operating System Interface for Unix (POSIX).[9] Unfortunately, CCIR Recommendation 460 was not openly available† and the committees who produced both the

*Early CCIR Recs. on broadcast time were 70 (1951), 122 (1953), 179 (1956), and 319 (1959).

†Recent versions of ITU-R TF recommendations became openly available online in 2010-12 `http://www.itu.`

ANSI C[10] and POSIX[11] standards incorporated a nonexistent concept. The "double leap second" in `<time.h>` allowed 62 seconds in a minute numbered 0 to 61.

The double leap second error was corrected in C99[12] and POSIX 2004,[13] but programmers remain confused about the implementation of leap seconds. Section 4.15 of the current POSIX[14]* specifies that "Seconds Since the Epoch" approximates elapsed seconds of Coordinated Universal Time and requires that "each and every day shall be accounted for by exactly 86400 seconds." The rationale section A.4.15 admits that UTC has leap seconds and says "POSIX time is therefore not necessarily UTC". It describes the lack of consensus, the impossibility for POSIX to mandate that a system clock matches any official clock, and allows that POSIX seconds may not all have the same length.

Independent of POSIX efforts, Dr. David Mills produced a method for synchronizing computer clocks.[15] Network Time Protocol (NTP)[16]† is routinely distributed along with computer operating systems. Many machines on the Internet, including some at national metrology institutes, provide accurate time via NTP. The RFC explains "The Coordinated Universal Time (UTC) timescale represents mean solar time as disseminated by national standards laboratories." However, the relation between seconds counted by the NTP protocol and seconds of UTC is specified not to count leap seconds. With both POSIX and NTP not counting them, every leap second produces observable time deviations as different operating systems and implementations adjust.

In contrast to the Unix model of time, IBM S/390 contained the concept of Leap Second Offset (LSO) that could optionally be set to the number of leap seconds which have occurred since 1972-01-01.[17] The document has a lengthy description of the finicky manual details required to implement the LSO. The document now applies to the newer IBM System z, or z/OS.‡ Along with the system specifics there is also a rationale which attempts to explain GMT, UT1, UTC and leap seconds. The text seems to originate from the CCIR/ITU-R documents and it includes 'UTC is the official replacement for (and generally equivalent to) the better-known "Greenwich Mean Time" (GMT).' Online discussions among IBM sysadmins indicate that setting the LSO is rarely practiced.

Microsoft operating systems have run on machines from many different vendors with wide variations in the capabilities of the hardware clock. This means that Microsoft Windows has not needed to consider support for leap seconds because of limitations in the hardware. A Microsoft Support boundary describes their version of NTP: "The W32Time service cannot reliably maintain sync time to the range of 1 to 2 seconds. Such tolerances are outside the design specification of the W32Time service."§

Insufficient information, unavailable or unclear standards, and lack of consensus mean that various vendors and researchers continue to try different schemes for handling leap seconds. Michel Hack and a team from IBM tried handling the 2008 leap second by hacking Linux kernels in a fashion similar to z/OS.[18] The results were not POSIX compliant. Site reliability engineer Christopher Pascoe described how Google handled the 2008 leap second.¶ Their result was POSIX-conformant, but they changed the length of seconds in way unsuitable for real-time control processes.

---

int/rec/R-REC-TF/en

*http://www.unix.org/

†http://www.ntp.org/

‡http://www.ibm.com/systems/z/os/zos/

§http://support.microsoft.com/kb/939322

¶http://googleblog.blogspot.com/2011/09/time-technology-and-leaping-seconds.html

## TIME ZONES AND DAYLIGHT TIME

There is another form of *Pontifex Maximus* well known to about half the population of earth. Civil authorities routinely exercise control of time by moving zone boundaries and changing the dates of transitions between standard time and daylight (or summer) time.

In the 1960s and 1970s some counties in Indiana ignored federal and state regulations specifying their time zone rules. In 1999 several Australian states changed their daylight rules for the 2000 Olympics with less than a year of advance notice. In 2006 the rules were changed again for the Commonwealth Games. In 2005 the governor of Indiana enacted a law requiring all counties to observe daylight time starting less than a year hence. In 2007 president Hugo Chávez announced that Venezuela would shift from GMT-04:00 to GMT-04:30 the next week (other authorities persuaded him to wait until year's end).

**Figure 3. Indiana governor Mitch Daniels and Venezuela president Hugo Chávez insisted that their constituencies adopt new rules for the start time of each day.**

All of these changes in time zones and rules, and many others, are documented in the `tz` database.* Arthur David Olson of NIH instituted the `tz` database and has coordinated a community which performs ongoing maintenance. Although the `tz` database is not authoritative, most operating systems use some form of it to convert between system time and civil time. The vendors of most systems provide updates to the `tz` database as part of routine patches.

The `tz` database consists of source code and data. For POSIX systems the time conversion is performed by the `tzcode` using the `tzdata`; the kernel has no role. The kernel code ostensibly keeps UTC, and civil time conversions happen in user code. This removes any need for the kernel to be updated or know about changes.

*`http://www.twinsun.com/tz/tz-link.htm`

The `tzdata` distribution contains a file `leapseconds`. This file is intended to be used by the `tzcode` when one of the "`right`" timezones is selected by the sysadmin or a user. As provided the `leapseconds` file contains a list of all leap seconds which have been inserted into the broadcast time scale. Use of the "`right`" timezones, however, is not conformant with POSIX because it produces days with 86401 seconds. The "`right`" timezones are also not compatible with NTP, for they presume that the system clock value of `time_t` is a count including all leap seconds.

The recent revision of the iCalendar[19] data format highlighted the connection between calendar and clock for scheduling events. To facilitate the worldwide updating of iCalendar schedules the Calendaring and Scheduling Consortium (CALCONNECT) has tasked its TIMEZONE Technical Committee to create a timezone registry and API for a timezone service.* This also produced an Internet Draft proposing that the IANA should maintain the `tz` database.† The committee is considering how to describe and serve the `leapseconds` file along with the rest of `tzdata`. This might provide a robust and machine-readable means of distributing leap second announcements.

## REINTERPRETING POSIX

What does POSIX really want for a kernel? The standard and rationale make enough apologies that the answer is not immediately clear.

POSIX does not want to know about astrometry or geophysics. As far as POSIX is concerned these are equivalent to the whims of politicians changing timezones and daylight rules.

POSIX wants to be conformant with the needs of "real-time" systems. The self-inconsistent words of the POSIX standard when it mentions leap seconds and UTC make this goal unreachable with the status quo.

POSIX does demand 86400 seconds in a day. This fact is built into far too much code. Changing this would be prohibitively expensive. Knowing nothing of astrometry or geophysics, however, POSIX also is oblivious to the kind of "day" that has 86400 seconds.

POSIX standard is mistaken when it says it wants UTC. The use of the term UTC was merely an update to the original notion of GMT.‡ The practical result of the evolution of systems and hardware means that POSIX really wants the time scale which is internationally approved for use in radio broadcasts. POSIX systems do not care what name humans use for that time scale, and a change in the name of that time scale cannot affect the operation of the kernel.

POSIX requires that the zoneinfo mechanism be able to handle offsets expressed in hours, minutes, and *seconds* between `time_t` and the local timezone.§ This requirement is the key to a possible compromise.

## A POSIX CONFORMANT WAY TO RETAIN LEAP SECONDS IN UTC

The time scale used by the GPS satellites was equal to UTC on the inception date of the system, 1980-01-06. Several vendors supply NTP time servers which rely on signals from the GPS satellites to maintain correct time. The normal configuration of an NTP time server uses the information in the GPS signals to convert GPS time into UTC.

---

*`http://calconnect.org/tc-timezone.shtml`

†`http://tools.ietf.org/html/draft-lear-iana-timezone-database-04`

‡see the Definition of "Epoch"

§see System Interfaces for `tzset()` and Environment Variables for `TZ`

Owners of time servers from Meinberg and Symmetricom can configure them to provide NTP service as GPS time instead of UTC. Combining that with some small changes to the `tz` database produces a scenario I call “right+GPS”. Figure 4 shows the `leapseconds` file where leaps before the GPS epoch are deleted and one test leap is added at the end. Figure 5 shows the shell script which produces the output in Figure 6.

```
# Leap  YEAR    MONTH   DAY     HH:MM:SS        CORR    R/S
Leap    1981    Jun     30      23:59:60        +       S
Leap    1982    Jun     30      23:59:60        +       S
Leap    1983    Jun     30      23:59:60        +       S
Leap    1985    Jun     30      23:59:60        +       S
Leap    1987    Dec     31      23:59:60        +       S
Leap    1989    Dec     31      23:59:60        +       S
Leap    1990    Dec     31      23:59:60        +       S
Leap    1992    Jun     30      23:59:60        +       S
Leap    1993    Jun     30      23:59:60        +       S
Leap    1994    Jun     30      23:59:60        +       S
Leap    1995    Dec     31      23:59:60        +       S
Leap    1997    Jun     30      23:59:60        +       S
Leap    1998    Dec     31      23:59:60        +       S
Leap    2005    Dec     31      23:59:60        +       S
Leap    2008    Dec     31      23:59:60        +       S
Leap    2011    Oct      3      20:51:60        +       S
```

**Figure 4. the `leapseconds` file from `tzdata` hacked to demonstrate right+GPS zoneinfo**

```
#! /bin/sh

then='empty'
isofmt='+%Y-%m-%dT%H:%M:%S'
MYTZ=$HOME/tzdir2011k+gps/etc/zoneinfo-leaps/US/Pacific
while true; do
  now=`date "$isofmt"`
  if [ x"$now" != x"$then" ]; then
    right=`TZ=:$MYTZ date "$isofmt"`
    time_t=`date +%s`
    echo "$time_t POSIX $now right+GPS $right"
    then=$now
  fi
  usleep 500000 2>/dev/null || sleep 0.5
done
```

**Figure 5. shell script demonstrates right+GPS zoneinfo**

The output shows that the `time_t` of the system clock incremented uniformly, and the interpretation of the clock was POSIX-conformant, but the time presented to the users included the leap second. This strategy produces a POSIX-conformant system by redefining the notion of “POSIX day” as 86400 seconds of atomic time while allowing ongoing leap seconds in the civil day. This strategy can only work if the ITU-R changes the name of the broadcast time scale along with omitting leap seconds from the broadcasts (which was the advice given to the ITU-R WP7A SRG at the 2003 colloquium they held in Torino).

This strategy requires a non-conformant NTP server and maintenance of the hacked `tzdata` distribution. With the current form of UTC only a few sites can afford the manpower requirements for

```
1317675120 POSIX 2011-10-03T13:52:00 right+GPS 2011-10-03T13:51:45
1317675121 POSIX 2011-10-03T13:52:01 right+GPS 2011-10-03T13:51:46
1317675122 POSIX 2011-10-03T13:52:02 right+GPS 2011-10-03T13:51:47
1317675123 POSIX 2011-10-03T13:52:03 right+GPS 2011-10-03T13:51:48
1317675124 POSIX 2011-10-03T13:52:04 right+GPS 2011-10-03T13:51:49
1317675125 POSIX 2011-10-03T13:52:05 right+GPS 2011-10-03T13:51:50
1317675126 POSIX 2011-10-03T13:52:06 right+GPS 2011-10-03T13:51:51
1317675127 POSIX 2011-10-03T13:52:07 right+GPS 2011-10-03T13:51:52
1317675128 POSIX 2011-10-03T13:52:08 right+GPS 2011-10-03T13:51:53
1317675129 POSIX 2011-10-03T13:52:09 right+GPS 2011-10-03T13:51:54
1317675130 POSIX 2011-10-03T13:52:10 right+GPS 2011-10-03T13:51:55
1317675131 POSIX 2011-10-03T13:52:11 right+GPS 2011-10-03T13:51:56
1317675132 POSIX 2011-10-03T13:52:12 right+GPS 2011-10-03T13:51:57
1317675133 POSIX 2011-10-03T13:52:13 right+GPS 2011-10-03T13:51:58
1317675134 POSIX 2011-10-03T13:52:14 right+GPS 2011-10-03T13:51:59
1317675135 POSIX 2011-10-03T13:52:15 right+GPS 2011-10-03T13:51:60 < leap
1317675136 POSIX 2011-10-03T13:52:16 right+GPS 2011-10-03T13:52:00
1317675137 POSIX 2011-10-03T13:52:17 right+GPS 2011-10-03T13:52:01
1317675138 POSIX 2011-10-03T13:52:18 right+GPS 2011-10-03T13:52:02
1317675139 POSIX 2011-10-03T13:52:19 right+GPS 2011-10-03T13:52:03
1317675140 POSIX 2011-10-03T13:52:20 right+GPS 2011-10-03T13:52:04
1317675141 POSIX 2011-10-03T13:52:21 right+GPS 2011-10-03T13:52:05
```

**Figure 6. A POSIX-conformant leap second by using right+GPS. Note the shift of GPS − UTC from 15 to 16 s.**

maintaining such an aberrant configuration. If the ITU-R were to change the name of the broadcast time scale then this strategy could be adapted as the default used by all systems.

The code here demonstrates that leap seconds can be handled by code that is already tested, widely-distributed, and in use by POSIX-conformant systems. A shift in the representation of time, and the nomenclature used by POSIX systems, allows a compromise which preserves the traditional meaning of civil time while enabling technologies that require a new meaning for broadcast time signals.

Furthermore, as demonstrated here, on a POSIX system this strategy allows any user to test any software at any time. No special hardware is needed to simulate the effects of a leap second. In this scheme leap seconds and UTC could become a timezone. This is a form of “localization”, the term used to describe computer outputs which can be formatted differently according to cultural preferences.

## CONCLUSION

Discussions on the details of UTC and leap seconds often turn into flame wars. There has been little consensus. Notions of the rules for keeping time are built into many systems. The subject is broad and esoteric. Many people have preconceptions based on outdated information resources, and misconceptions based on wrong resources. Good pedagogy is lacking.

If the ITU-R abandons leap seconds but retains the name UTC the `tz` database still allows jurisdictions to declare mean solar time as their legal civil time. Any authority who decides to continue inserting leap seconds can use this code and data to insert them. If such a time scale were named “Global Mean Time” or “Greenwich Meridian Time” then the currently synonymous terms UTC and GMT would gain notably distinct meanings.

Explaining the subject of UTC, GMT, and leap seconds is difficult. When talking with a journalist

a good metaphor is the story of the blind men and the elephant. A journalist is likely to receive a different description from each different pundit, and there may be more blind men who never contribute their knowledge of the elephant. Getting a comprehensible description of the entire elephant takes a lot of work.

**Figure 7. "Blind monks examining an elephant" by Hanabusa Itchō**

## REFERENCES


[1] Royal Astronomical Society, "Report of the Council to the Eleventh Annual General Meeting of the Society," *Monthly Notices of the Royal Astronomical Society*, Vol. 2, Jan. 1831, pp. 11–12.

[2] US Department of State, ed., *International Meridian Conference*. Washington, D.C.: Gibson Bros., 1884. Protocols of the Proceedings.

[3] S. Newcomb, "Tables of the motion of the earth on its axis and around the sun," *Astronomical papers prepared for the use of the American ephemeris and nautical almanac, v. 6, pt. 1, [Washington, Bureau of Equipment, Navy Dept., 1895], 169 p. 29 cm.*, Vol. 6, 1895, pp. 2–+.

[4] L. Essen and J. V. L. Parry, "An Atomic Standard of Frequency and Time Interval: A Cæsium Resonator," *Nature*, Vol. 176, Aug. 1955, pp. 280–282, 10.1038/176280a0.

[5] W. Markowitz, R. G. Hall, L. Essen, and J. V. Parry, "Frequency of Cesium in Terms of Ephemeris Time," *Physical Review Letters*, Vol. 1, Aug. 1958, pp. 105–107, 10.1103/PhysRevLett.1.105.

[6] CCIR, "Recommendation 374, Standard-Frequency and Time-Signal Emissions," *Documents of the Xth Plenary Assembly*, Vol. III, 1963, p. 193.

[7] CCIR, "Recommendation 374-1, Standard-Frequency and Time-Signal Emissions," *Documents of the XIth Plenary Assembly*, Vol. III, 1966, p. 281.

[8] CCIR, "Avis 460, Émissions de Fréquences Étalon et de Signaux Horaires," *XII$^{e}$ Assemblée Plénière*, Vol. III, 1970, p. 227.

[9] Institute of Electrical and Electronics Engineers, *IEEE Standard Portable Operating System Interface for Computer Environments, Std 1003.1-1988*. 1988.

[10] American National Standards Institute, *American National Standard for Information Systems, Programming language C, ANSI X3.159-1989*. 1989.

[11] The Austin Common Standards Revision Group, *Single UNIX Specification, Version 2*. The Open Group, 1997.

[12] International Organization for Standardization, *Programming languages – C, ISO/IEC 9899:1999*. 1999.

[13] The Austin Common Standards Revision Group, *Single UNIX Specification, Version 3, 2004 Edition*. The Open Group, 2004.

[14] The Austin Common Standards Revision Group, *Single UNIX Specification, Version 4, 2010 Edition*. The Open Group, 2010.

[15] D. Mills, *Computer network time synchronization : the Network Time Protocol*. Boca Raton, FL: CRC/Taylor & Francis, 2006.

[16] D. Mills, J. Martin (ed.), J. Burbank, and W. Kasch, *RFC 5905: Network Time Protocol Version 4: Protocol and Algorithms Specification*. Internet Engineering Task Force, 2010.

[17] Ken Trowell and Marg Beal and Noshir Dhondy and Helen Howard and Greg Hutchison, *S/390 Time Management and IBM 9037 Sysplex Timer*. IBM Corporation, 1999.

[18] M. Hack, X. Meng, S. Froehlich, and L. Zhang, "Leap Second support in computers," *2010 International IEEE Symposium on Precision Clock Synchronization for Measurement Control and Communication (ISPCS)*, 2010, pp. 91–96.

[19] B. Desruisseaux, ed., *RFC 5545: Internet Calendaring and Scheduling Core Object Specification (iCalendar)*. Internet Engineering Task Force, Sept. 2009.