

A POSIX® Status Update

April 2001

Joseph M. Gwinn, IEEE PASC SSWG-RT Chair
Andrew Josey, IEEE P1003.1 Chair



THE *Open* GROUP

Agenda

- What is POSIX?
- POSIX Realtime standards
- SSWG Realtime Working Group Status
 - POSIX 1003.13 Profiles
 - Ongoing Work
- POSIX 1003.1 Revision Status



What is POSIX ?

- POSIX is the codification and de-jure standardization of the common core of UNIX® practice, and now of Realtime OS practice
 - The basic objective is portability of both programmers and application source code
 - Portability of the OS kernel itself and/or application binary code are not objectives
- POSIX is a set of books specifying APIs
 - It is neither a piece of code nor an operating system



THE *Open* GROUP

What is an API?

- An API (Application Program Interface) is a written contract between kernel writers (employed by the platform vendor) and application writers (employed by the vendor's customers)
 - An API is not a piece of code, it's a piece of paper defining what the two sets of writers are guaranteed to receive and are in turn responsible for providing



Hard and Soft Realtime; Non-Realtime Definitions

- The two realtime cases (hard and soft) are marked respectively by a specification of either a hard maximum response time or an allowed statistical distribution of response times for each and every action, always at some required heavy average level of traffic (typically specified in actions per second or per minute).
- By contrast, the non-realtime cases are marked by a specification of a required average throughput in actions per hour; the latency typically being unspecified.



Hard and Soft Realtime; Non-Realtime Definitions (Cont'd)

- The average performance of non-realtime systems is usually much better than that of soft realtime systems, which in turn have better average performance than hard realtime systems.
- If the problem is really realtime (where latency or even worst-case latency is the key issue) but one instead measures only average throughput performance, one will be led to exactly the wrong choice.
- Realtime is thus not a number, it's a behavior resulting from correct selection of the architecture and algorithms in all of the following: application code, middleware, communications systems, and operating system.



Embedded Realtime

- A computer is considered *embedded* if it is a component of a larger system whose primary purpose is not computation. Embedded computers need not be small, or even all that realtime.
- The terms *embedded* and *realtime* are independent of one another
 - Realtime computers are generally embedded as well. However, many embedded computers are not at all realtime. A microwave oven is the classic example of an embedded but non-realtime application.
 - Embedded computers (often called *target* computers) are not necessarily suited to software development, and may be able to run only the application code. In short, host target in such systems.
- For example, many industrial control computers are both embedded and stringently realtime. There may also be some simultaneous soft-realtime and non-realtime activities as well.



Realtime DII-COE and POSIX.13

- Simply put, POSIX.13 (IEEE Std 1003.13-1998) is the core of RT DII-COE
- POSIX.13 is a subset profile standard allowing non UNIX realtime operating systems “clothed” with a runtime library to comply
 - This standardizes the application-to-RTOS API, allowing considerable application code portability between different RTOS offerings, which portability had not been possible in the past
 - RTOS+wrapper offerings can be compared and competed directly
 - There are currently four profiles, with a fifth being developed for DISA and RT DII-COE as POSIX draft P1003.13a



THE *Open* GROUP

SSWG-RT Road Ahead

- Support The Austin Group in incorporating the existing realtime POSIX standards (1003.1d, .1j, .1q, and .13) into 1003.1-200x
- Develop P1003.13a, in conjunction with the SAE, to add a fifth profile
- Develop P1003.13b, to incorporate 1003.1d, 1003.1j, 1003.1q, and anything else relevant in the 1003.1-200x to come
- Revive Interrupt Control and Device Control, but as freestanding POSIX standards. Device Control is now P1003.26. Interrupt Control is a Study Group
- Add user-defined kernel-level process and/or thread scheduling as an option. This may be a tough sell, as it isn't immediately obvious how to do such an apparently kernel-dependent thing in a portable and standardizable manner, so a small group will go off and prove out the details first.



THE *Open* GROUP

Perspectives on OS Sizes and Maturities

- RTOS kernels
 - Contain up to a few hundred thousand of lines of code; many are much smaller
 - First emerged in the 1960s and 1970s
- UNIX System kernels
 - Contain one or two million lines of code
 - First emerged in the late 1960s
- Windows NT (now called Windows 2000)
 - Contains 40 million lines of code (as of Jan 1999); still growing rapidly
 - First emerged in 1993
 - Not known for great stability, even in benign office environments



THE *Open* GROUP

Perspectives on OS Sizes and Maturities

- Summary:
 - The UNIX system is one twentieth the size and complexity and twenty years more mature than Windows NT
 - RTOS kernels are about the same age as UNIX, but are a fraction the size and complexity, and are specifically designed for embedded realtime control applications, unlike UNIX and Windows NT
- Recommendation: Don't use desktop operating systems (or office productivity products) for mission- or safety-critical applications. Fit the tool to the application.



THE *Open* GROUP

The Yorktown Incident

- For almost three hours in September 1997, the Yorktown, a billion-dollar Aegis guided-missile cruiser, drifted helplessly off Cape Charles, VA, without propulsion, steering, weapons, or sensors, all due to a crash in a Windows-NT client, which took the entire shipboard LAN down with it (Ref: “Was it Windows or Human Error”, WSJ 27 Aug 1998, page B6)
 - If this had happened in a storm or a battle, or far from home, ship and crew could have been lost. Many naval battles have been lost and/or ships sunk simply because a ship's rudder became stuck. A dead propulsion system is worse.
- The Yorktown was the prototype IT-21 “Smart Ship”
 - The Smart Ship Program finally died of its injuries in January 2001



THE *Open* GROUP

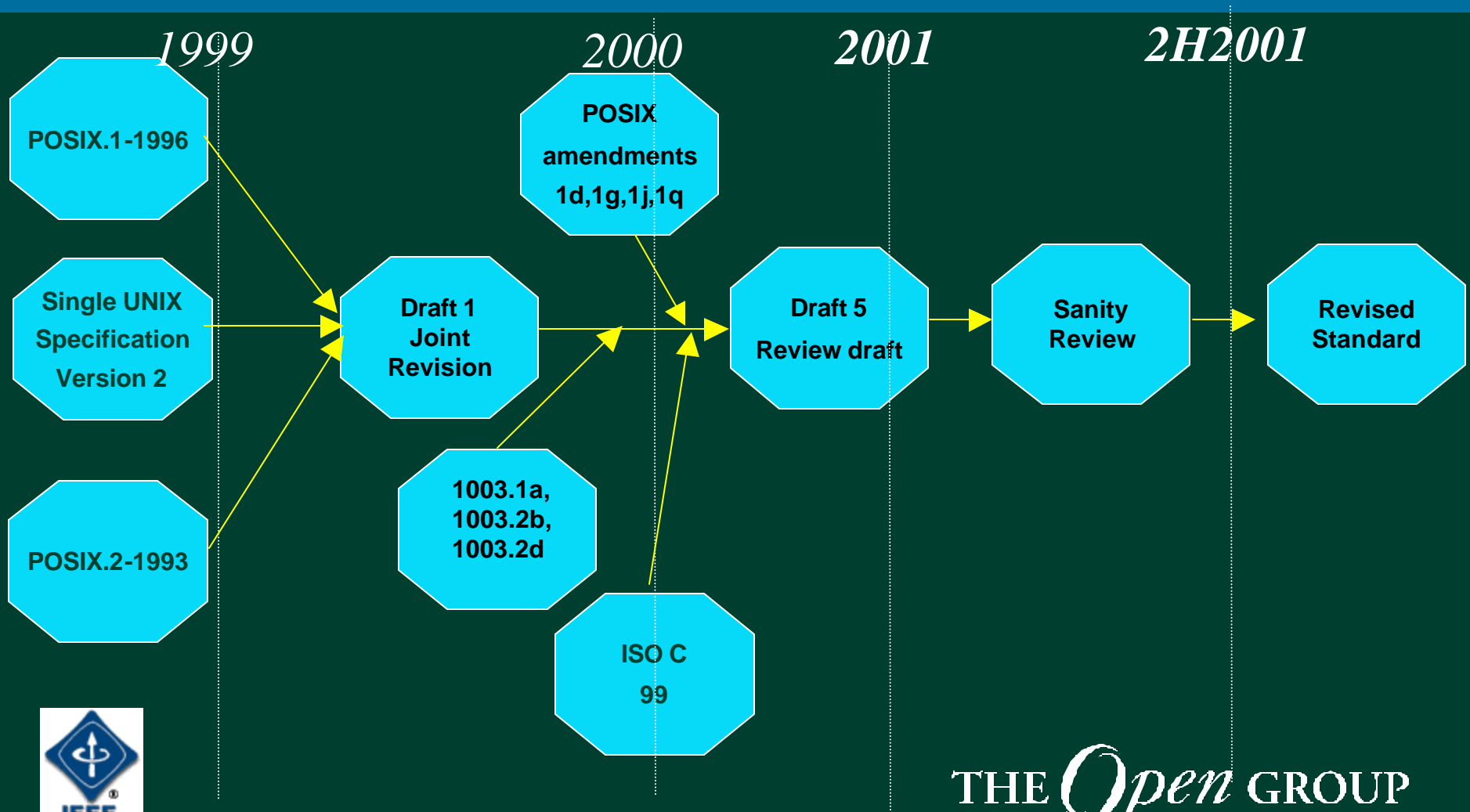
The Yorktown Incident (Cont'd)

- The fundamental problem was the inappropriate use of a desktop operating system for an application that is both mission-critical and safety-critical
 - Civilian Air Traffic Control systems have since the 1970s or 1980s had full fault tolerance, with tenth-second recovery times. Warships should be able to do at least this well.
 - A traditional two-box solution, with a standard RTOS controlling propulsion et al in one set of boxes and NT in a different set of boxes, communicating via standard Internet protocols, would have allowed the Yorktown to sail on unaffected while Windows was being brought back up.
 - For safety-critical applications, Windows NT may never suffice, or be accepted by the relevant safety review boards



THE *Open* GROUP

The POSIX.1 Revision Roadmap



THE *Open* GROUP

The POSIX.1 Revision and The *Austin Group*

- The Austin Common Standards Revision Group
- An open industry initiative to revise the core *POSIX standard* and the *Single UNIX Specification*; standards that lie at the heart of today's open systems
- Chair and editors from The Open Group



THE *Open* GROUP

The *Austin Group* (Cont'd)

- The *Austin Group* combines the formal standards process of the IEEE and ISO, with the industry standards of The Open Group and the community at large.
- *Electronic participation*
- *Participation in the group is free.*
- *Draft specifications are available for download to registered members of the group from the world wide web.*



THE *Open* GROUP

About the *Austin Group*

- 396 Participants (as of April 2001)
- Wide industry support
 - *AT&T, Compaq, Fujitsu, HP, IBM, Lucent, Microsoft, Red Hat, SGI, Siemens, Sun*
 - *DoD, USENIX, Canada Revenue*
- Participation in the *Austin Group* from the Open Source community includes
 - *The Linux Standard Base, NetBSD, FreeBSD, and many others*



THE *Open* GROUP

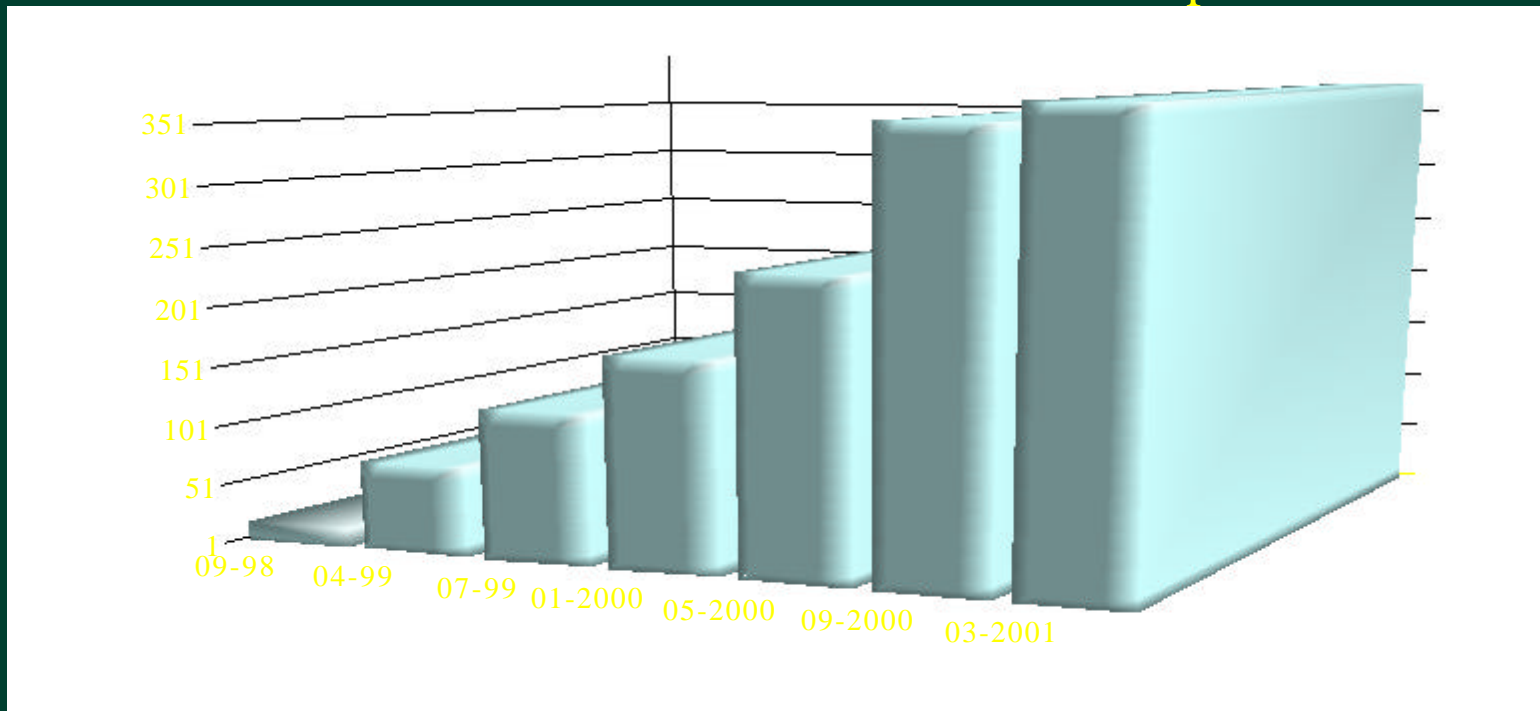
JDOCS Procedures

- The Austin Group operates under the JDOCS procedures
- Procedures *approved* by the three organizations
- Officers
 - Chair
 - Three organizational Reps (ORs)



Membership Growth

of Members of the Austin Group



THE *Open* GROUP

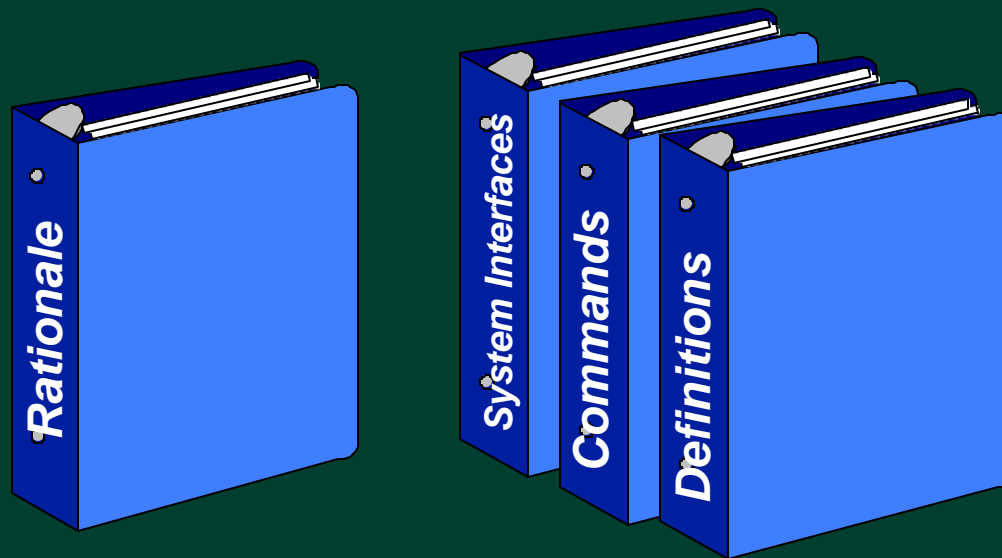
Objectives

- To target the joint specification at the programmer / user rather than the system implementor
- Organization based on the Core volumes of the Single UNIX Specification, organized alphabetically, and including Rationale
- To produce a new standard in year 2001



THE *Open* GROUP

The New Common Specification



THE *Open* GROUP

Further Information

- The Austin Group
 - <http://www.opengroup.org/austin>
- *The IEEE PASC Web Site*
 - <http://www.pasc.org>
- *The Single UNIX Specification*
 - <http://www.UNIX-systems.org>



THE *Open* GROUP

How You Can Help?

- To participate in the *Austin Group*, send email to
 - *Austin-group-request@opengroup.org* with the word *subscribe* in the subject line
- Visit the web site
 - <http://www.opengroup.org/austin/>

