CERN LIBRARIES, GENEVA

CM-P00065712

CERN-CW 91-9
0W9131

CERN-COMPUTING & NETWORKS DIVISION

CN/91/9 C1
July 1991

**Standardisation of Realtime Software
POSIX 1003.4**

Christoph ECK

(Presented at the Real-Time 91 Conference, Jülich, 24-28 June 1991)

Standardisation of Realtime Software POSIX 1003.4

Christoph Eck

Computing and Networks Division, CERN, 1211 Geneva 23, Switzerland

Abstract

The acronym POSIX is known as the label for the most widely accepted standard, or more precisely group of standards, in the traditional UNIX environment. The POSIX 1003.4 Working Group has been preparing the document "Realtime Extension for Portable Operating Systems" for several years. This paper describes the results of the effort to create a standard operating system interface for portable realtime application software as they present themselves after the last two POSIX meetings of January and April 1991. Realtime POSIX addresses the full extent of realtime systems, from full scale UNIX down to small embedded kernels with the highest demands on hard realtime performance. After an introduction into realtime software standardisation and the POSIX.4 group a short report is given on all draft standards produced by this group so far.

I. INTRODUCTION

The necessity of standards for all computer related products is by now obvious to their users. Most manufacturers recognise the importance of standards by being involved in standardisation projects. Nevertheless, looking at the history, even recent, of standards in the computer industry one learns quickly that in many cases manufacturers were mainly paying lip service to the idea of standardisation. They were happy to support "de facto" standards, either as original manufacturer of the product or in order to profit from an established market by copying it. Besides that, why should a manufacturer support a move which could only lead to bigger independence of the user community?

The first "real" standards came therefore in areas where standardisation was practically unavoidable, where systems from different manufacturers had to exchange data, e.g. ASCII or IEEE floating point format, or had to be networked together, e.g. all the standards for local and wide area networks.

The first manufacturers who actually understood standardisation as a tool to improve their time-to-market figures were the people building modular computers based on standard buses. The so-called "open bus systems" are by now firmly established.

Yet, hardware is only part, and more and more apparently the easier part, of a computer system. The "black art" of software was for a long time covered in a very limited way by the standardisation process. There have been standard high level languages but in the operating system area we were still left with de facto standards at the best. Microsoft's DOS can claim this status and so can UNIX. Still, DOS serves only one specific architecture and UNIX suffers from the fact that it comes in too many, partially incompatible flavours.

II. THE UNIX "STANDARD"

By now one can find UNIX everywhere. It runs on all sizes of systems from micros to supercomputers. Every computer manufacturer supports UNIX. Most of them as their only choice, some in parallel to their proprietary operating system (notably DEC and IBM). It seems very likely that proprietary systems have only a small chance of survival.

Against this background it is not surprising to see how much effort is invested in the creation of a UNIX standard. Even if part of this effort is masked by fights for market share, well expressed by the formation of groups like UNIX International, the Open Software Foundation (OSF), and recently the Advanced Computing Environment (ACE), quite a number of basic standard documents in this area are finding general agreement: SVID (the System V Interface Definition), the X/Open guide, and most importantly the POSIX standards written by IEEE, the well known US standards organisation.

III. POSIX

The acronym POSIX stands for Portable Operating System Interface with the X added to indicate its origin. POSIX is written by IEEE working groups as a US standard. Yet, it acquires international status via its acceptance by ISO/IEC (International Standards Organisation / International Electrotechnical Commission) as International Standard 9945.

POSIX is a software interface standard which guarantees portability of application source code and should not be understood as an operating system implementation standard. This point is well exemplified by the fact that such different system implementations as VMS and OS/2 will both become POSIX compliant.

Work on the POSIX standard is split into many projects, roughly 20 at this time, labelled IEEE P1003.n. Until now only one of these projects has its document agreed as an official standard: the System Application Program Interface, IEEE Std 1003.1-1990, ISO/IEC 9945-1:1990(E).

IV. REALTIME SYSTEM STANDARDISATION

Before discussing realtime system standardisation it will be necessary to define the "realtime" aspect of an operating system. Most people will automatically associate execution speed with realtime behaviour. Whilst speed is important for most realtime applications it does not define the essential difference between a fast timesharing and a realtime system. Quoting the definition from a POSIX document, "realtime" is "the ability of the operating system to provide a required level of service in a bounded response time". In other words, this implies a guarantee for worst case response. Basic UNIX is

not designed to be a realtime system. Its goal is to give a fair share of system resources to every user, not to have, for instance, a high priority process take the CPU and keep it as long as necessary.

A standard timesharing UNIX was therefore not much help for realtime applications. Even worse, there was no hope for a de facto standard in the realtime systems or kernel area. A (too) large number of kernels and systems are on the market, a lot of them with nearly identical functionality but nevertheless incompatible with each other. None of these systems has a sufficient share of the realtime market to be a candidate for a de facto standard.

Recognising this, VITA, the VFEA International Trade Association, started the ORKID (Open Realtime Kernel Interface Definition) project with the hope to create a standard kernel definition via a consistent amalgamation of existing products. The resulting document is a solid piece of technical work. Yet, the project was more or less killed by political battles, not only between the group of companies participating in the ORKID work, and outsiders, but even between the former companies themselves. These battles provided one more lesson on the difficulties to convince manufacturers about the intrinsic value of software standardisation.

Luckily, the POSIX work had, in the mean time, gained so much general support that its realtime extension project was protected. It became rather clear that IEEE P1003.4 had grown to be the last and only chance for arriving at a standard realtime system. In consequence of this, the ORKID group joined the P1003.4 project at the end of 1990.

V. POSIX P1003.4 AND RELATED DOCUMENTS

The official title of the P1003.4 standard is: "Realtime Extension for Portable Operating Systems". In order to understand the scope of this document and how it is written the answers to the following questions will be useful.

Who participates in the POSIX.4 Group?

The latest address list of POSIX.4 contains 129 names. Typically 40-50 of these people show up at meetings. Around 70 of the 129 come from computer companies or software houses. The others, and indeed a surprisingly strong subgroup, are either users of realtime software or come from a university or other organisation doing research related to realtime systems.

Officially, members of IEEE working groups participate as private individuals and cannot represent a company. In reality, participants from industry have the interests of their companies in mind. Yet, practically all discussions in the group are based solely on technical arguments. Working group etiquette which forbids the use of "we" in all discussions has probably something to do with this success.

How do they work?

The POSIX.4 group meets four times per year. These meetings are scheduled together with the meetings of all other POSIX projects, bringing together some 400 people. Until now all meetings were held in some big US hotel, moving as

justly as possible from the South to the North to the West and to the East.

Meetings go on for a whole week from 8 a.m. to 6 p.m. and a lot of the actual creative work is done at the meetings themselves. Apparently, only few participants have the time to work for more than a few days on the standard in between meetings.

What are their aims?

Quoting the P1003.4 draft: "To make POSIX.1 minimally usable to realtime applications on single processor systems. The scope is to take existing realtime operating system practice and add it to the base standard."

This statement needs some interpretation. Multiprocessor support is under discussion in the working group. It is clear by now that an exclusively monoprocessor standard would not cover a fast growing part of the applications, and "adding" realtime system practice does not mean that every compliant realtime system will have to carry the overhead of full background POSIX. Actually, POSIX.4 compliance will cover a wide range of systems from small embedded kernels in ROM to large full-function POSIX systems, the smaller systems being always compatible subsets of the larger ones.

What did they achieve so far?

The POSIX.4 group has so far created drafts of four documents: P1003.4, P1003.4a, P1003.4b, and P1003.13.

The base document is P1003.4: "Realtime Extension for Portable Operating Systems". At the time of writing this paper Draft 10 from February 6, 1991, is the most recent version. P1003.4 contains the specification of interfaces to handle:

- binary semaphores
- memory locking
Whole realtime processes or parts of them can be declared unswappable in order to guarantee worst case response times.
- shared memory and memory mapped files
- priority scheduling
Fully preemptive, priority based scheduling is supported following two main scheduling policies. Fifo scheduling establishes process lists on each priority level ordered by the time its processes have been on the list without being executed. Round-robin scheduling establishes separate time-slicing for processes on each priority level.
- realtime signals
These signals are a minimal extension to the POSIX.1 signals avoiding possible loss of asynchronous event notification, enabling their prioritised handling, and differentiation between multiple occurrences of events of the same type.
- clocks and timers
High resolution sleep down to nanoseconds and absolute, relative, and periodic timers are supported.
- IPC message passing
Inter Process Communication is supported via message

queues for passing arbitrary amounts of data between processes in a deterministic way.

- asynchronous I/O
The possibility to start an I/O operation without waiting for the operation to finish.
- synchronised I/O
The possibility to do I/O with the guarantee that data are recorded on the output medium when the operation returns.
- realtime files
Support for fast I/O onto contiguous files or equivalent structures.

Every chapter of P1003.4 ends with a "Performance Metrics" section. This section specifies how to measure the execution time of the different operations in a given implementation. Manufacturers claiming compliance of their system with P1003.4 have to document these times in the user manual for the system.

Draft 9 of P1003.4 was the first draft to undergo the IEEE balloting process. Each IEEE standard has a balloting body, a representative group of IEEE members who vote on the acceptability of the standard. For acceptance more than 75% of this body has to vote and more than 75% of the voters have to accept the standard as it is.

Draft 9 was balloted at the end of last year and failed with around 3000 objections (a lot of them directed nevertheless at the same points). The latest document, Draft 10, was balloted last spring and failed with 1500 objections. These figures document well the level of scrutiny the POSIX standards are submitted to and give also an indication how the standard is progressing towards acceptability.

The next document prepared by the POSIX.4 group is P1003.4a: "Threads Extension for Portable Operating Systems", the most recent version being Draft 5, of December 7, 1990. POSIX processes have quite a heavy context and process switching may well be slow. Yet, one process can have multiple threads in a common memory map with a fairly light context. Conceptually, such a thread can be seen as much the same thing as a typical realtime kernel task.

In addition, P1003.4a introduces further synchronisation primitives: mutexes and condition variables. They are based on shared memory and allow very fast implementations. Mutexes also support priority inheritance, a feature specifically designed to avoid the typical problem of priority based schedulers known as priority inversion. Thread A of low priority has a resource, thread C of high priority needs it, but thread B of medium priority preempts A disallowing it to liberate the resource. In this situation C can wait an unbounded time for lower priority threads.

The mutex has the concept of ownership. That means that the priority of the owner of the mutex can be temporarily raised to the highest priority of a thread blocked on this mutex. In this way priority inheritance allows a speedy release of the requested resource.

P1003.4a failed its first ballot this spring. Draft 6 of P1003.4a is expected for July 1991.

A further document under preparation by the POSIX.4 group is P1003.4b. This standard, a first viable draft of which is also expected for July 1991, will contain additional extensions. Work has started on specifying timeouts for blocking operations, fast interrupt handling by user code, and the support for memory allocation from different regions representing the different physical areas of memory available in realtime systems: fast or slower (on/off-board) memory, ROM, battery backed-up, shareable between processors etc. More topics are planned to be handled, most importantly multiprocessing.

Yet another document prepared by POSIX.4 is P1003.13: Realtime Profiles. Draft 2 is expected for July 1991. A profile document calls out a set of options from POSIX and related standards. This handles the problem of compatibility between different implementations based on standards with a large number of options. Logically, conformance of a product can only be claimed by referencing a specific profile (or profiles).

P1003.13 profiles will cover the full range of system sizes, from small embedded kernels in ROM to full-fledged systems, each profile being a subset of a higher one.

Where are the problems?

Obviously, the creation of a standard by consensus among a large group of individuals with different backgrounds is not an easy matter. The main risk is that everybody's pet idea gets included into a draft, making it too complex and heavy for the advocates of "hard" realtime systems. Opposition to this approach manifests itself clearly in the balloting results. It is interesting to note that during the pruning process demanded by the ballot, the ORKID document is more and more taken as a concise description of "industrial practice", defining the minimal level of required functionality.

Another difficulty stems from the fact that the realtime extensions are handled as a set of several documents being in different stages of completion. Whilst this approach is justified by the wish to get incremental voting on the parts of the standard coming close to their final shape, it also carries the risk of losing consistency between P1003.4, .4a, and .4b.

Finally, there remain jobs to be done which will not change the basic content of the standard but are requested, mainly by ISO/IEC, to obtain formal approval. One is a language independent form of the standard which is until now written in the form of C-language function specifications. Another is the need to add test assertions for conformance testing.

VI. CONCLUSIONS

The realtime extensions to POSIX are not only a necessary step towards realtime system standardisation but also the most promising one. It is expected to have P1003.4 accepted as official standard in May 1992 with good hopes to have the

whole process finished sometime in 1993.

It looks as if most companies will make their products POSIX compliant. Some companies are already claiming conformance of their systems. Officially, this is possible for POSIX.1. For the realtime extensions this is more of a problem. The standard is not frozen yet and, specifically, the profiles defining the different sets of options are in a very early state.

But one thing is certain. Every professional working with realtime operating systems and realtime programs should start to learn about POSIX as soon as possible.

VII. REFERENCES

POSIX Standards and Drafts can be obtained from:
TCOS Standards Subscriptions
c/o Lisa Granoien
IEEE Computer Society
1730 Massachusetts Ave. NW
Washington DC 20036-1903

The ORKID Document can be obtained from:
VITA
10229 N. Scottsdale Rd., Suite B
Scottsdale AZ 85253
or:
VITA
P.O.Box 192
NL-5300 AD Zaltbommel
The Netherlands