

# A proposal for the POSIX.13 revision

**Developed by: the POSIX Realtime System Services Working Group**

**Presented by: Michael González ([mgh@unican.es](mailto:mgh@unican.es))**

**Anaheim, CA, USA, January 2002.**

# Table of Contents

---

1. Introduction. What is POSIX?
2. Overview of current real-time profiles
3. New proposed POSIX.13 requirements
4. Future work

# 1. Introduction: Real-Time Systems

---

A Real-time system is a combination of a computer, hardware I/O devices, and special-purpose software, in which:

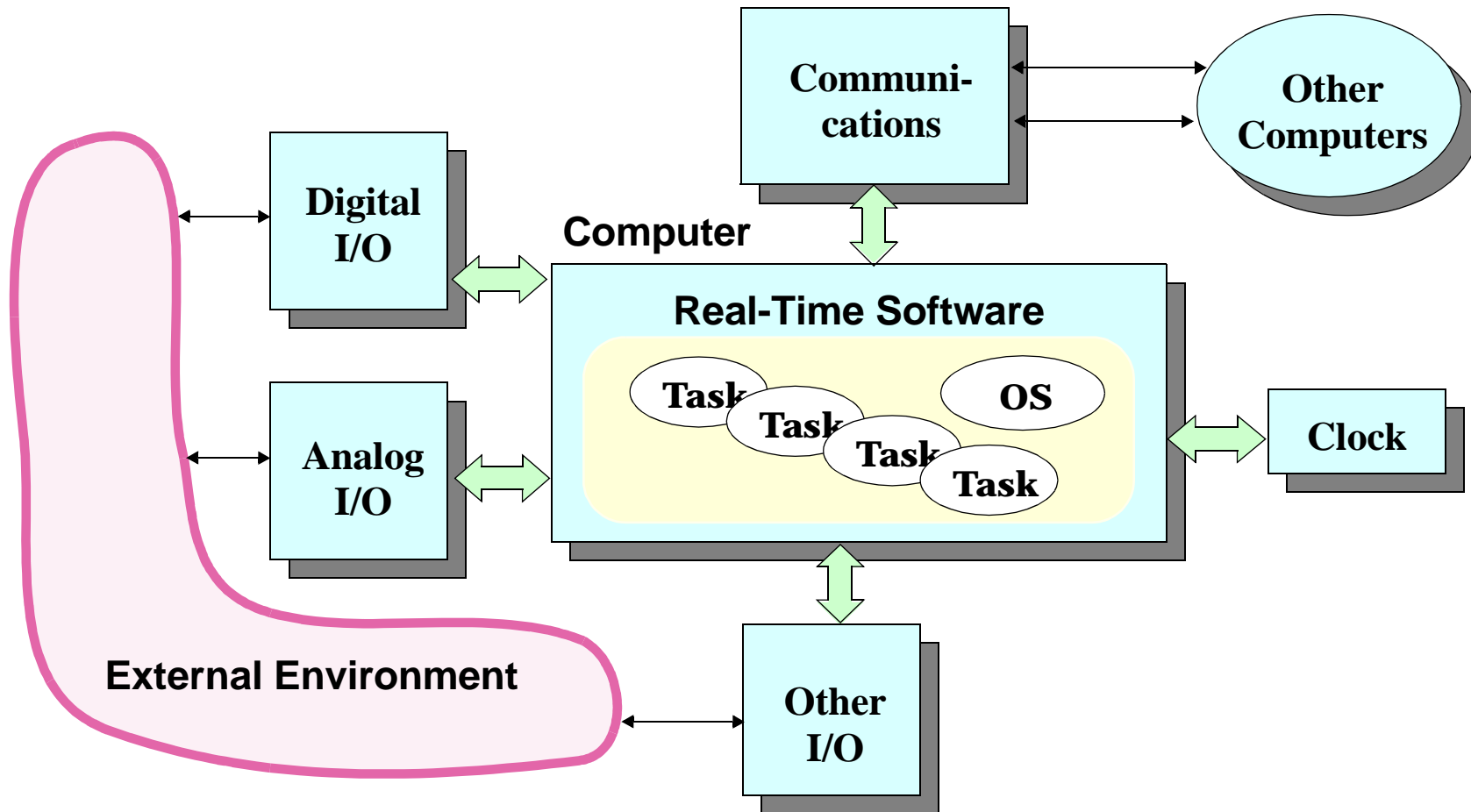
- there is a strong interaction with the environment
- the environment changes with time
- the system simultaneously controls and/or reacts to different aspects of the environment

As a result:

- timing requirements are imposed on software
- software is conceptually concurrent

Many real-time systems are *embedded*: components of a larger system

# Elements of a Real-Time System



# Real-Time Operating Systems

---

In the past, many real-time systems did not need an operating system.

Today, many applications require operating system services such as:

- concurrent programming, device I/O, communication networks, file system, etc.

The timing behavior of a program depends strongly on the operating system's behavior.

POSIX.1b definition of real-time in operating systems:

“The ability of the operating system to provide a required level of service in a bounded response time.”

# What is POSIX?

---

## Portable Operating System Interface

Based on UNIX operating systems

The goal is portability of

- applications (at the source code level)
- programmers

Sponsored by the IEEE Computer Society (PASC).

Three categories of standards:

- Base standards (in C language)
- Profiles for different application environments
- Bindings in different programming languages (Ada, F77)

# Real-Time POSIX: Motivations

---

**Large variety of real-time systems:**

- **Real-time kernels (VRTX, VxWorks,..)**
- **Ada Run-time Executives**
- **OS's for larger systems: VMS, OS9, proprietary systems**
- **Some real-time UNIX implementations**

**A standard was necessary to achieve portability**

**Subsets of the operating system services were needed: for embedded systems**

# Real-Time Extensions

---

The goal was to add to POSIX.1 those OS services that were necessary to:

- **Achieve predictable timing behavior:**
  - **Process scheduling**
  - **Virtual memory management**
  - **Real-time signals**
  - **Clocks and timers**
- **Facilitate concurrent programming:**
  - **Process synchronization**
  - **Shared memory**
  - **Message queues**
  - **Asynchronous I/O and synchronized I/O**

# Threads Extension

---

The process model provided protection, but was inadequate for systems requiring high efficiency:

- High context switch times
- High creation and destruction times
- Need for special hardware (MMU)
- Not adequate for fine-grain multiprocessors

In real-time kernels, tasks usually share the same address space, and have a small associated context

POSIX.1c defined C language interfaces to support multiple threads of control inside each POSIX process, sharing the same address space

# Additional Real-Time Extensions

---

POSIX.1d and POSIX.1j defined additional interfaces for services that are useful to many real-time applications:

- **Timeouts for blocking services**
- **Execution time and monotonic clocks**
- **Sporadic server scheduling policy**
- **Spawn a process**
- **Advisory information for real-time files**
- **Typed memory**
- **Multiprocessor synchronization primitives**

# Tracing

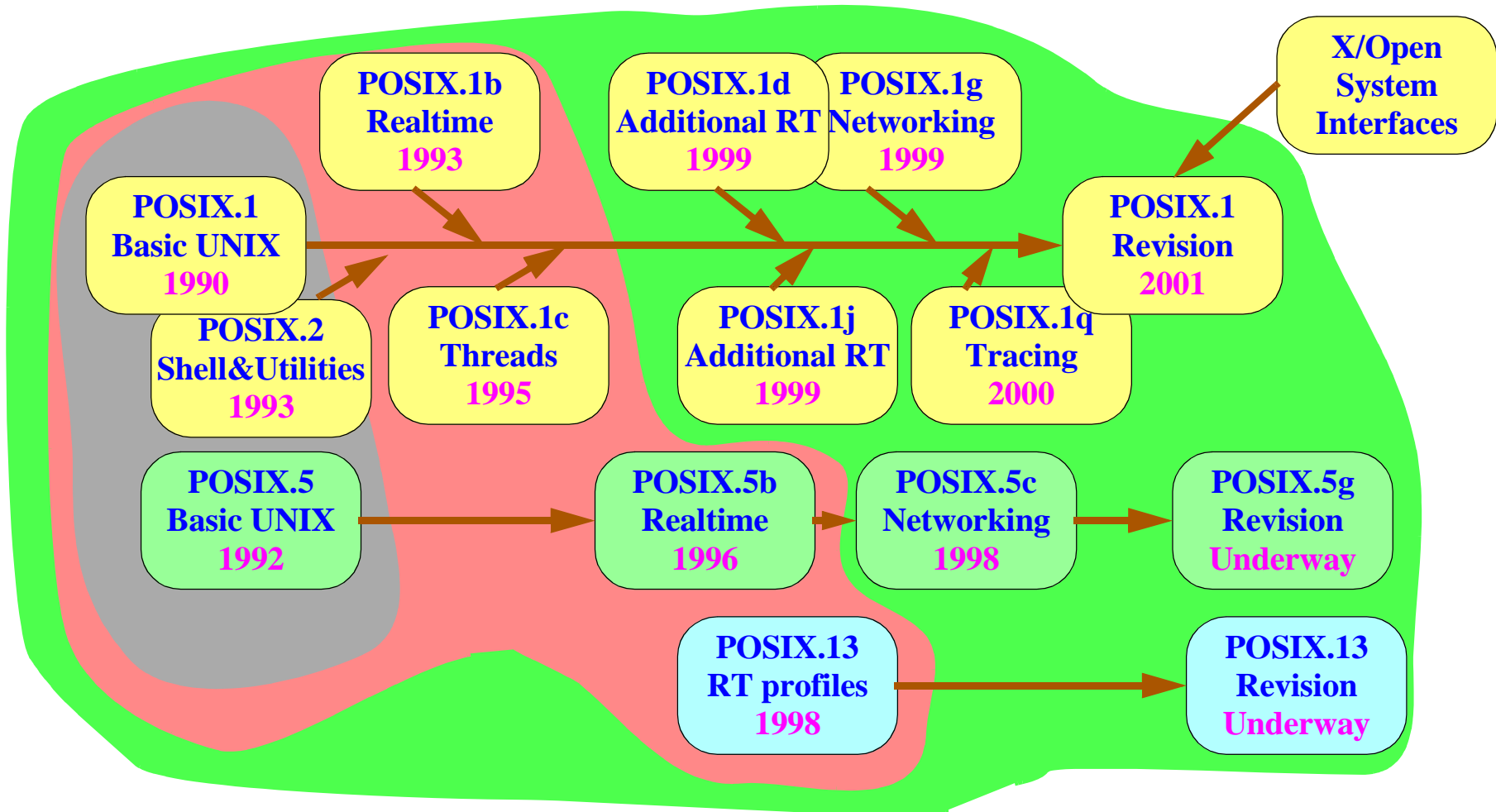
---

**POSIX.1q defines interfaces for:**

- **Tracing actions such as**
  - **system calls**
  - **context switches**
  - **user-defined events**
- **Trace events are stored in a buffer in memory**
- **The buffer may be:**
  - **read from an analyzer process**
  - **or stored in a file or remote device for off-line analysis**

**A process may control one or more traces, each tracing that process or a different one.**

# History of (some of) the POSIX standards



# Motivation for POSIX.13

---

## The POSIX Standard:

- Allows writing portable real-time applications
- Very large: inappropriate for embedded real-time systems

## POSIX.13:

- Defines four real-time system subsets (profiles)
  - **Minimal**: Small embedded systems
  - **Controller**: Industrial controllers
  - **Dedicated**: Large embedded systems
  - **Multi-Purpose**: Large general-purpose systems with realtime requirements
- C and Ada language options

# 2. Overview of Current Profiles

---

## *Minimal* Real-Time System

- Platform: Small embedded system, with no MMU, no disk, no terminal
- Model: controller of a “Toaster”

## Real-Time *Controller*

- Platform: Special purpose controller, with no MMU, but with a disk containing a simplified file system
- Model: industrial robot controller

# Overview (cont'd)

---

## *Dedicated* Real-Time System

- **Platform:** Large embedded system with no disk, but with an MMU; software is complex and requires memory protection and network communications
- **Models:** avionics controller, cellular phone cell node

## *Multi-Purpose* Real-Time System

- Large real-time system with all the features, including a development environment, network communications, file system on disk, terminal and graphical user interfaces, etc.
- **Model:** workstation with realtime requirements:
  - air traffic control systems
  - telemetry systems for Formula One racing cars

# Characteristics of the Profiles

Profile	File System	Multiple Processes	Threads
Minimal System	NO	NO	YES
Controller	YES	NO	YES
Dedicated System	NO	YES	YES
Multipurpose System	YES	YES	YES

**Note that “file system” refers to a full file system with directories and the capability of creating new files**

**Device I/O using `open()`, `read()`, `write()`, and `close()` on device files is supported in all profiles**

**Multiprocessor support is included in all profiles**

# Structure of a Profile

---

POSIX.1 & .5 have optional groups of functions

- The profile specifies which *options* are required
- Example: The *Timers* option for real-time

POSIX.1 & .5 have a large set of mandatory functions

- The profile creates *Units of Functionality*
  - a separately implementable element of the standard
  - composed of a set of interfaces
- Example: *POSIX\_DEVICE\_SPECIFIC* (for terminal control)

Some functions need further *restrictions*

- Example: `open()` fails when creating a file in an implementation with no file system

# 3. New Proposed POSIX.13 Requirements

---

## Contents:

- Options required in all profiles (thus, defining *Minimal*)
- File system-related options (*Controller, Multi*)
- Multiple processes-related options (*Dedicated, Multi*)
- Options required only in the *Multi-Purpose* RTS
- Miscellaneous options
- Options not required in any profile

## 4. Future Work

---

### Add a new profile

- between Dedicated and Multi
- perhaps adding support for a file system

### Develop Ada bindings for new POSIX.1 revision, including:

- additional real-time extensions (POSIX.1d & .1j)
- tracing (POSIX.1q)

### Develop new interfaces

- device control (the old `ioctl()`)
- interrupt control
- application-defined scheduling

# Options required in all profiles (*Minimal*)

- Language support (C or Ada libraries)
  - except math library (not in *Minimal*)
  - and Wide Characters library (only in *Multi*)
- Device I/O and synchronized I/O
- File locking (for thread-safe I/O)
- Signals and realtime signals
- Single process (configuration and process environment)
- Threads, mutexes, and condition variables
  - and X/Open thread extensions

# Options required in all profiles (*Minimal*) Cont'd

- **Priority scheduling**
  - Including sporadic server scheduling
  - and priority inheritance and protection
- **Time services**
  - Clocks and high resolution timers
  - Timeouts
  - Monotonic clock
  - Clock selection
  - CPU-time
- **Memory management**
  - Memory locking
  - Shared memory objects

# File system-related options (*Controller*, *Multi*)

---

Required in both (*Controller* and *Multi*):

- Basic file system
- Memory-mapped files

Required only in the *Multi-purpose* RTS:

- File attributes
- FIFO special files
- File system extensions
- Symbolic links
- Advisory information (for predictable timing behavior)

# Multiple processes-related options (*Dedicated, Multi*)

---

- Multiple processes (`fork`, `exec`, ...)
- Pipes
- Spawn (fast process creation)
- Memory Protection
- Process shared attribute for mutexes and condition variables
- Signal jump

# Options required only in the *Multi-Purpose* RTS

---

## Shell-related:

- Shell and utilities
- Regular expressions
- Shell functions

## Security-related

- System database
- Users and groups

## Others:

- Wide character and wide character I/O
- Device-specific functions (terminal control)

# Miscellaneous options

---

## *Dedicated* and *Multi*:

- Networking
  - including IP V6, raw sockets, and event mgmt. (**select**)
- Asynchronous I/O and prioritized I/O
- Semaphores

## All but *Minimal*:

- Math library
- File descriptor management
- Message Passing
- Tracing, including event filtering
  - trace logging only for *Controller* and *Multi*.

# Options not required in any profile

- X/Open System Interface (XSI) options
  - except thread and mutex extensions
  - many are alternatives to existing POSIX interfaces (e.g., `XSI_TIMERS`)
  - others are optional features, not mandatory in POSIX.1
- Non realtime synchronization primitives:
  - barriers
  - reader/writer locks
  - spin locks
- Trace inheritance
- Typed memory objects
- POSIX.2 options (except software development options)

## 4. Future Work

---

### Add a new profile

- between *Dedicated* and *Multi*
- perhaps adding support for a file system

### Develop Ada bindings for new POSIX.1 revision, including:

- additional real-time extensions (POSIX.1d & .1j)
- tracing (POSIX.1q)

### Develop new interfaces

- device control (the old `ioctl()`)
- interrupt control
- application-defined scheduling