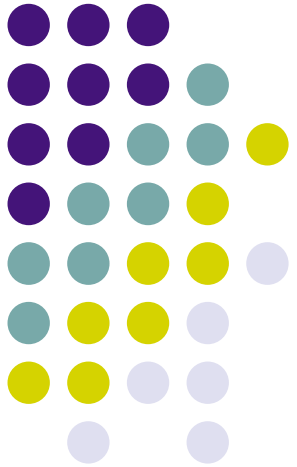
POSIX in Real-Time

By Kevin M. Obenland

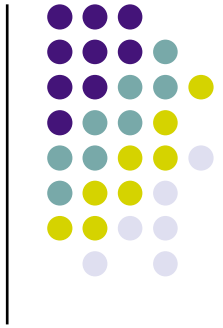
03/15/2001



Daniel Correia nºMec 18713

Carlos Guisado nºMec 49099

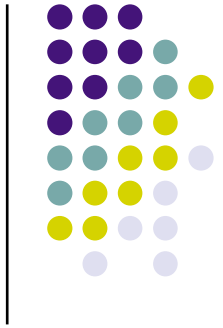
Posix. What is and why?



- The original Portable Operating System Interface for Computing Environments (POSIX) standard was first published in 1990.
- POSIX is based on UNIX.
- The original POSIX standard defines interfaces to core functions such as file operations, process management, signals, and devices.

- The need to develop open systems is driven by three major factors:
  - First, gone are the days when a single developer could implement the entire system from scratch.
  - Second, software does not operate in isolation; it must co-exist with the vast amount of commercially available software.
  - Last, the lifecycle of a software application is typically long, requiring numerous modifications and updates as new features are added.

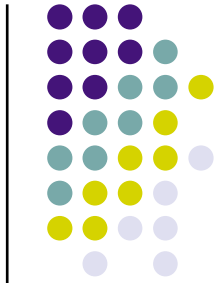
POSIX real-time extensions.



- POSIX 1003.1b, as well as 1003.1d and 1003.1j, define extensions useful for development of real-time systems. Functions defined in the original real-time extension standard 1003.1b are supported across a wider number of operating systems than the other two specifications. For this reason this article focuses on POSIX 1003.1b. The following items constitute the bulk of the features defined in POSIX 1003.1b:
  - Timers, Priority scheduling, Real-time signals, Semaphores, Memory queues, Shared memory and Memory locking

- POSIX 1003.1b provides support for fixed priority preemptive scheduling. To be compliant with POSIX, an operating system must implement at least 32 priorities.

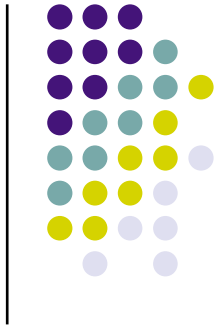
Commercial support for POSIX



- Because POSIX 1003.1a is based on UNIX, any UNIX-based operating system will naturally be very close to the standard. To be conformant to the POSIX standard, the operating system and hardware platform have to be certified using a suite of tests.

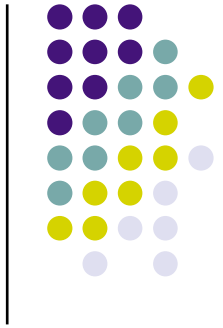
- The design of an operating system can have a significant impact on its ability to be used in a real-time system. This includes the internal design of the operating system as well as the features it provides to the application programmer.

Solaris Implementation



- Multithreaded preemptable kernel
- Global priority model - Threads = LWP
- Configurable clock tick - Configurable Scheduler
- High resolution POSIX timers - (ns/ms)
- Priority I/O streams
- POSIX real-time APIs - POSIX 1003.1b
- Symmetrical multiprocessing support - Reservation for RT

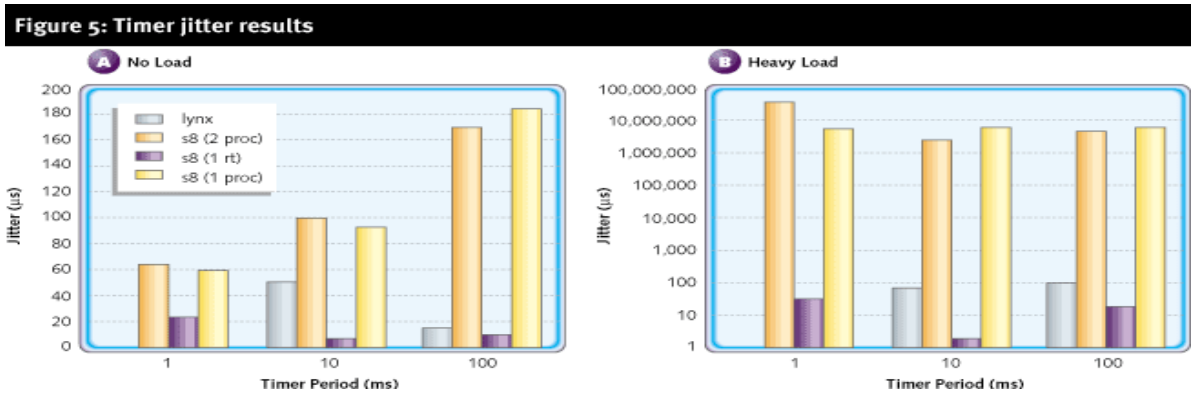
Lynx Implementation



- preemptable, reentrant kernel, small(97KB)
- Single scheduling policy (256 levels)
- Clock tick is fixed 100Hz(10ms)
- Scheduler also called on asynchronous events and change on system state
- 256 levels for interrupts(application even/interrupts odd)
- Interrupts = device drivers - different levels according to parents

Benchmarks (1/4)

- Systems
  - Lynx Dell Pentium 2 266MHz
  - Solaris 2xSPARC 360Mhz
  - Solaris 1xSPARC 360Mhz
  - Solaris 2xSPARC 360Mhz (1 RT)
- Time jitter
  - Create a periodic thread and measure the deviation between desired and actual expiration
    - No Load - every system has a response under 200ms.
    - High Load - Lynx and 1RT keep a good response but other take as long as 10 secs to reply



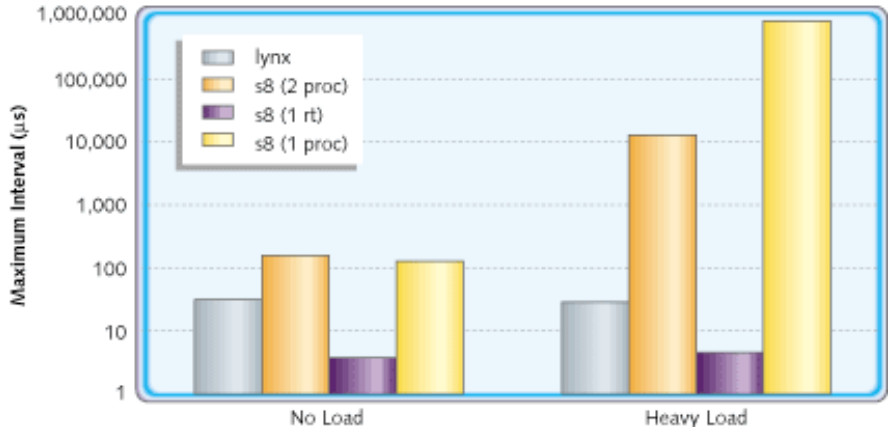
Benchmarks (2/4)

- Application response
  - Execute a fixed processing load and measure its execution time over a number of runs
    - No Load - Results near the 10 ms calibrated.
    - High Load - Lynx and 1RT come close to 10 ms, other as worst as 1000 times more-
- Bintime
  - Call a time of day clock and measure interval between calls
    - No Load - 1RT below 10ms rest under 100ms
    - High Load - Solaris without 1RT non deterministic.

Table 8: Worst case response results (in milliseconds)

	add		copy		Whet	
Configuration	**No load**	**Heavy load**	**No load**	**Heavy load**	**No load**	**Heavy load**
Lynx	9.9	9.9	10	10.1	10.1	10.2
Solaris (2 procs)	10.1	11236.5	10.7	12061.7	10.6	12162.8
Solaris (1 proc)	10.2	7310.7	10.2	4599.3	10.7	6328.2
Solaris (1 rt)	10	10	10	10	10.5	10.5

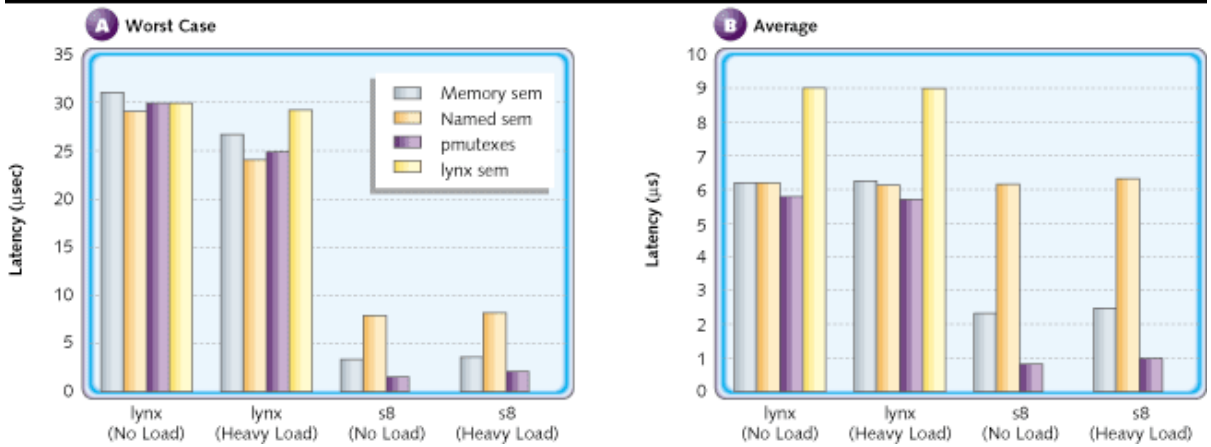
Figure 6: Bintime results



Benchmarks (3/4)

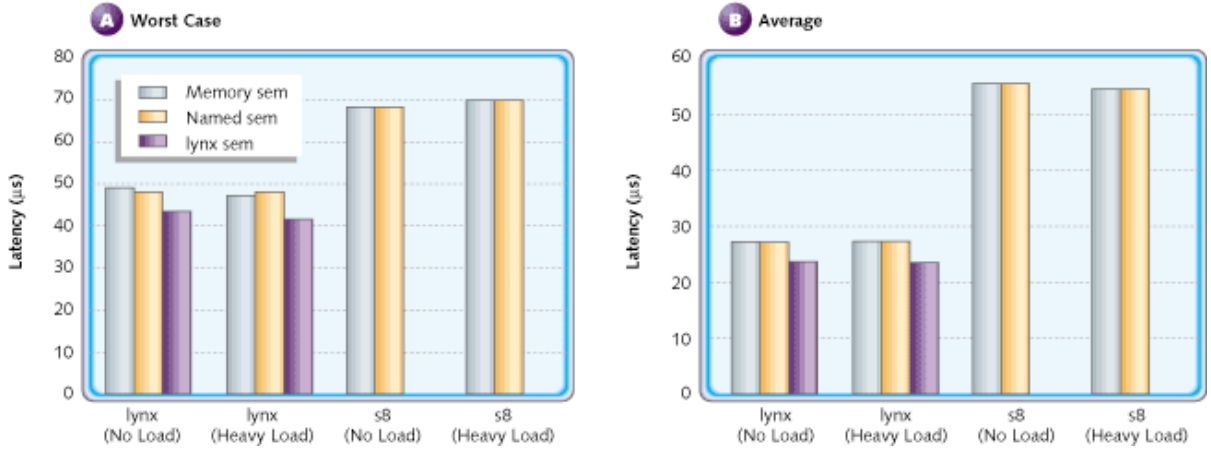
Signaling within a thread

Figure 7: Sync Test 1: Lynx and Solaris (1 rt)

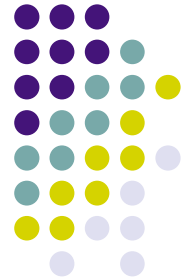


Inter-thread signaling

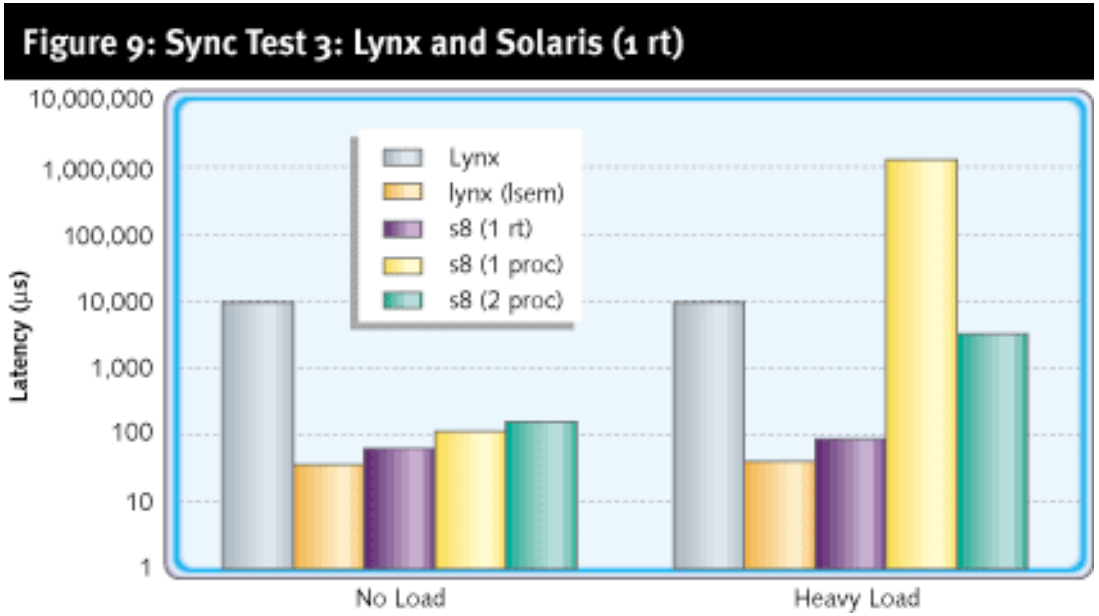
Figure 8: Sync Test 2: Lynx and Solaris (1 rt)



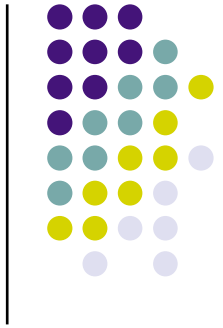
Benchmarks (4/4)



Priority inversion



Conclusion



- Both are suitable for use in real-time systems.
- LynxOS exhibited a low overhead for all operations and was deterministic even under heavy loading conditions.
- Solaris 8 contains a number of features that are important in real-time development, including high-resolution timers, processor partitioning, and SMP support. These last two features are key in Solaris's use as a real-time operating system.