COMPARISON BETWEEN QNX RTOS V6.2 AND RED HAT EMBEDDED LINUX (ELDS) V1.1

Disclaimer

Although all care has been taken to obtain correct information and accurate test results, Dedicated Systems Experts and Dedicated Systems Magazine cannot be liable for any incidental or consequential damages (including damages for loss of business, profits or the like) arising out of the use of the information provided in this report, even if Dedicated Systems Experts and Dedicated Systems Magazine have been advised of the possibility of such damages.

http://www.dedicated-systems.com

E-mail: info@dedicated-systems.com

EVALUATION REPORT LICENSE

This is a legal agreement between you (the downloader of this document) and/or your company and the company DEDICATED SYSTEMS EXPERTS NV, Gerechtsstraat 21, 1070 Brussels, Belgium.
It is not possible to download this document without registering and accepting this agreement on-line.

1. **GRANT**. Subject to the provisions contained herein, Dedicated Systems Experts hereby grants you a non-exclusive license to use its accompanying proprietary evaluation report for projects where you or your company are involved as major contractor or subcontractor. You are not entitled to support or telephone assistance in connection with this license.
2. **PRODUCT**. Dedicated Systems Experts shall furnish the evaluation report to you electronically via Internet. This license does not grant you any right to any enhancement or update to the document.
3. **TITLE**. Title, ownership rights, and intellectual property rights in and to the document shall remain in Dedicated Systems Experts and/or its suppliers or evaluated product manufacturers. The copyright laws of Belgium and all international copyright treaties protect the documents.
4. **CONTENT**. Title, ownership rights, and an intellectual property right in and to the content accessed through the document is the property of the applicable content owner and may be protected by applicable copyright or other law. This License gives you no rights to such content.
5. **YOU CAN NOT**:
   - You can not, make (or allow anyone else make) copies, whether digital, printed, photographic or others, except for backup reasons. The number of copies should be limited to 2. The copies should be exact replicates of the original (in paper or electronic format) with all copyright notices and logos.
   - You can not, place (or allow anyone else place) the evaluation report on an electronic board or other form of on line service without authorisation.
6. **INDEMNIFICATION**. You agree to indemnify and hold harmless Dedicated Systems Experts against any damages or liability of any kind arising from any use of this product other than the permitted uses specified in this agreement.
7. **DISCLAIMER OF WARRANTY**. All documents published by Dedicated Systems Experts on the World Wide Web Server or by any other means are provided "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT. This disclaimer of warranty constitutes an essential part of the agreement.
8. **LIMITATION OF LIABILITY**. Neither Dedicated Systems Experts nor any of its directors, employees, partners or agents shall, under any circumstances, be liable to any person for any special, incidental, indirect or consequential damages, including, without limitation, damages resulting from use of OR RELIANCE ON the INFORMATION presented, loss of profits or revenues or costs of replacement goods, even if informed in advance of the possibility of such damages.
9. **ACCURACY OF INFORMATION**. Every effort has been made to ensure the accuracy of the information presented herein. However Dedicated Systems Experts assumes no responsibility for the accuracy of the information. Product information is subject to change without notice. Changes, if any, will be incorporated in new editions of these publications. Dedicated Systems Experts may make improvements and/or changes in the products and/or the programs described in these publications at any time without notice. Mention of non-Dedicated Systems Experts products or services is for information purposes only and constitutes neither an endorsement nor a recommendation.
10. **JURISDICTION**. In case of any problems, the court of BRUSSELS-BELGIUM will have exclusive jurisdiction.

Agreed by downloading the document via the internet.

1 Introduction

Recently, Dedicated Systems Experts evaluated the following real-time operating systems (RTOS):

- The QNX NEUTRINO RTOS v6.2 from QNX Software Systems Ltd.
- The Red Hat Embedded Linux Developer’s Suite v1.1 (ELDS) from Red Hat, Inc. This uses the Red Hat Linux kernel 7.2 (derived from the Linux kernel 2.4.5).

This report summarizes and compares the key elements of the full evaluation reports of both RTOSs.

2 Installation and Configuration

QNX NEUTRINO v6.2	0	8	10
ELDS v1.1	0	3	10

2.1 QNX NEUTRINO RTOS v6.2

The QNX NEUTRINO RTOS v6.2 is quick and easy to install. After only a few minutes the basic modules are installed i.e., the kernel and a user interface (Photon Windowing System). Additional packages like compilers etc, can be installed by means of the package manager.

Configuring the QNX NEUTRINO RTOS v6.2 is reasonably straightforward. When installing the full environment, the most critical components like storage devices and network cards are detected automatically. If further configuration is necessary, it can be done through the graphical user interface.

Creating and configuring a custom QNX image is done in the IDE or by text-based build files. Modules can be added, removed and configured by the System Builder Tool or manually editing text-based files. The documentation contains plenty of examples of such build files.

New in the QNX NEUTRTINO RTOS v6.2 is the graphical System Builder Tool which enables the management of QNX images. It replaces the build files with a graphical tool to create images (both boot images and flash images) and allows the importing of existing build files. The system builder tool features dependency analysis (tells you which libraries might be missing), as well as a “dietician”, which creates smaller versions of the shared libraries you’re using that only contain the functions you need.

2.2 ELDS v1.1

The Red Hat Embedded Linux Developer Suite version 1.1 is a tool-set from which you can configure and build a target kernel, define the packages that are to be included on the target, and finally build the target file-system. On the host, the standard Red Hat 7.2 Linux distribution is used. Installation of the host (on a x86) is straightforward and worked on our hardware without a glitch. The only manual intervention that was required was the addition of the I/O address and the IRQ level of our NE2000 compatible ISA NIC to /etc/modules.conf.

The installation of the Embedded toolset, performed by an automatic install script, essentially involves unpacking a large number of rpm packages from the CDROM to the hard disk of the host. The script furthermore installs specific versions of Perl and Python, required by ELDS, when these should not be present on your host.

The purpose of ELDS is to have a GUI for generating a custom (embedded) target kernel. However, the use of a GUI for creating a project in ELDS, which only allows you to enter a single string, clearly presents no added value whatsoever. In combination with the command-line instructions the developer has to issue immediately thereafter, it gives the impression of being a first, incomplete attempt at what may one day become a useful GUI-based project creation tool.

The Red Hat ELDS provides no documentation whatsoever regarding the configuration of the kernel, nor does it try to verify the validity of your combination of configuration choices. Yet this is precisely a fundamental and very complex step in the development of the target embedded system.

The ELDS provides no assistance in checking the necessary dependencies to ensure that the embedded system you are configuring contains all the required libraries in order to be able to boot.

Suppose you finally have, notwithstanding the exotic combination of command-line and poor graphic tools of the Red Hat Embedded Linux Developer Suite, managed to configure and build a kernel, as well as configure and build a root file-system. You are then confronted with the problem of transferring the target kernel and target file-system to the target host.

On this topic, the ELDS "Getting Started Guide", which is the very thin and only piece of documentation you receive, only gives information for Intel's StrongARM Assabet board. Developers for other platforms are simply on their own. This information is not usable as such for other platforms, and to get it running on our x86 platform we finally had to use the good old boot/root disk combination to transfer the kernel and the file-system to our target.

From all the products that passed through our evaluation process, the Red Hat Embedded Linux Developer Suite is clearly the most difficult one to configure and install on our basic x86 target.

3 RTOS Architecture

For a description of the ratings, the reader is referred to appendix D in the document “report definition and test plan”, which can be downloaded from our website (http://www.dedicated-systems.com/encyc)

QNX NEUTRINO v6.2	0	9	10
ELDS v1.1	0	3	10

3.1 System Architecture

3.1.1 QNX NEUTRINO RTOS v6.2

The QNX NEUTRINO RTOS v6.2 has a true client-server architecture, consisting of a microkernel and optional cooperating processes. The microkernel implements only the core services, like threads, signals, message passing, synchronization, scheduling and timer services. Additional functionality is implemented in cooperative processes, which act as server processes and respond to the request of client processes (e.g. an application process). Examples of such server processes are the file system manager, process manager, device manager, network manager, etc. While the kernel runs at privilege level 0 of the Intel processor, the managers and device drivers run at levels 1 and 2 (to perform IO operations). Application processes on the other hand run at privilege level 3, and can therefore only execute general instructions of the processor.

☺ Every process runs in its own virtual memory space. The QNX NEUTRINO RTOS is a message based OS, and can seamlessly be distributed over multiple nodes. The RTOS supports SMP, and implements several HA (High Availability) features.

3.1.2 ELDS v1.1

ELDS v1.1 uses the Red Hat Linux 7.2 kernel which is based on the general linux kernel v2.4.5.

Linux has its roots in the Unix General Purpose Operating System (GPOS), it has a traditional monolithic kernel and is clearly not built for real-time purposes. Fortunately, Red Hat does not claim real-time behavior for the Linux kernel used by the ELDS.

But let’s start with the good things: Linux is process based and has virtual memory protection between the different user processes and between a user and kernel process. The kernel can be extended by so called “modules” (device drivers). The kernel is well protected against faults in such modules. Swapping (virtual memory swapping to disk) can be avoided by not mounting a swap file-system.

On the other side, it is clear that Linux is not built for real-time purposes. So what’s wrong in the Linux system architecture to make it unsuitable for real-time?

- The kernel is not pre-emptive (there is now a patch available to make it pre-emptive)
- Monolithic OS: Linux is still a monolithic kernel, using large and complex shared data structures. As a result the critical sections are large (and mostly used more than needed) and can lock the processor for a considerable amount of time.

- Primitive kernel (threading/process support): threading and process supported by the rather primitive “spawn()” and “clone()” call. Other APIs are build on these calls.
- No priority inversion avoidance mechanism: This is a problem in all GPOS. As these OS are not built for real-time systems, there is no need for the rather complex protection against priority inversion.

3.2 Basic System Facilities

3.2.1 Task Handling Method

Both operating systems are multi-threaded.

	QNX NEUTRINO v6.2	ELDS v1.1
Model	Threads and processes	Threads and processes
Priority levels	64	99
Maximum number of tasks	4095 processes Every process can have 32767 threads	Limited by the amount of memory available and configuration settings
Scheduling policy	Prioritized FIFO Round-robin scheduling Adaptive Sporadic	Prioritized FIFO round-robin scheduling. Regular non real-time.
Number of documented states	14	From sched.h : 6 (not in documentation)

3.2.2 Memory Management Method

Both operating systems use full virtual memory protection, significantly increasing the robustness and reliability of the system.

	QNX NEUTRINO v6.2	ELDS v1.1
MMU support	Yes	Yes
Physical page size	Depends on architecture	Depends on architecture
Swapping/Demand Paging	Yes/No	Yes/No
Virtual memory	Yes	Yes
Memory protection models	Full virtual memory protection	Full virtual memory protection.

3.2.3 Interrupt Handling Method

For both operating systems the interrupts are prioritized handled even when nested.

For the linux kernel, there is no documentation available (with ELDS) to understand the interrupt handling internals. The only way is to look into the source code…

	QNX NEUTRINO v6.2	ELDS v1.1
Handling	Nested, prioritized	Nested, prioritized
Context	The ISR runs in the context of the thread that attached it	Not documented
Stack	The ISR has its own stack	Not documented
Interrupt to task communication	Signals and pulses	Not documented

4 API Richness

For a description of the ratings, the reader is referred to appendix D in the document “report definition and test plan”, which can be downloaded from our website (http://www.dedicated-systems.com/encyc)

QNX NEUTRINO v6.2	0	7	10
ELDS v1.1	0	5	10

While interpreting these results, the reader should keep in mind that these tables cover a strictly defined set of the most commonly used system calls. All OS have system calls that are not covered by the table below. For more details on what features the different categories in the table below encompass, the reader is referred to the evaluation reports.

	QNX NEUTRINO v6.2	ELDS v1.1
Task management	82%	47%
Clock	100%	43%
Timer	100%	0%
Memory management	27%	0%
Interrupt handling	88%	63%
Semaphore	35%	31%
Mutex	67%	31%
Conditional Variables	60%	0%
Event flags	0%	0%
POSIX signals	100%	78%
Message queue and Mailbox	41%	12%

The API of the QNX NEUTRINO RTOS is a sufficiently rich API. On the other hand the Linux API is not elaborated enough to use in real-time systems. For instance, there are no timer nor memory support functions (except the traditional alloc() call).

5 Internet support

For a description of the ratings, the reader is referred to appendix D in the document “report definition and test plan”, which can be downloaded from our website (http://www.dedicated-systems.com/encyc)

QNX NEUTRINO v6.2	0	8	10
ELDS v1.1	0	8	10

5.1 QNX NEUTRINO RTOS v6.2

QNX Neutrino’s Internet Technology Suite contains the following products and tools:

- Voyager Web Server: an HTTP server to serve web pages. It has support for dynamic HTML pages via SSI (Server Side Includes).
- A Voyager Web Browser for viewing information. It has full HTML 3.2 support, frames support, javascript, cookies, etc. The Mozilla and Opera browsers have recently also been ported to run on the QNX NEUTRINO RTOS v6.2.

A Voyager SDK (Software Development Kit) to build internet-enabled applications into an embedded system.

5.2 ELDS v1.1

There are a lot of servers CGI script engines, browsers and so one available for the Linux Platform.

6 Tools

For a description of the ratings, the reader is referred to appendix D in the document “report definition and test plan”, which can be downloaded from our website (http://www.dedicated-systems.com/encyc)

QNX NEUTRINO v6.2	0	8	10
ELDS v1.1	0	8	10

6.1 QNX NEUTRINO RTOS v6.2

QNX has its own Momentics IDE toolsuite. There are also two other sets of tools available for the QNX NEUTRINO RTOS v6.2: the Metrowerks Codewarrior IDE and the GCC toolkit. Tools for both self-hosted and cross development are available. These toolkits contain the most commonly used tools.

Although the Momentics IDE has all features needed and is pretty intuitive to use, you will need a lot of processing power and ram to get it running at a comfortable speed!

6.2 ELDS v1.1

Embedded Linux developers have a wide range of development tools at their disposal. There is the wide range of GNU tools (editors, compilers for different languages such as gcc and g++, the gdb debugger, the gas assembler, de ld linker, ...) as well as all the other tools that have been developed for the Linux/GNU system. For a number of GNU tools, graphical front-ends have been developed (Source Navigator or KDevelop C/C++ IDE, DDD or Insight front-end for gdb, ...). Most of these tools are integrated in the Red Hat distribution and the developer has plenty of choice. Installation of the tools is easy thanks to the rpm package management system.

7 Documentation and Support

For a description of the ratings, the reader is referred to appendix D in the document “report definition and test plan”, which can be downloaded from our website (http://www.dedicated-systems.com/encyc)

QNX NEUTRINO v6.2	0	7	10
ELDS v1.1	0	2	10

7.1 QNX NEUTRINO RTOS v6.2

The documentation does a decent job giving a general overview of the system and its architecture. The documentation has improved compared with the QNX v6.1 documentation: the meaning of API parameters are now well explained.

7.2 ELDS v1.1

Linux is a free open source operating system, built by a virtual community of enthusiast programmers. Unfortunately, programmers tend to prefer writing code over writing documentation.

In the Linux development model, there are no resources that are dedicated to writing the documentation that goes along with the kernel code and the tools that are developed by the volunteer programmers. Apparently, there are no - or at least very few - technical writers that feel the urge to participate in the Linux development effort.

There is of course the Linux Documentation Project. There is the documentation that is available in the "/usr/src/linux/Documentation" directory of the kernel source, together with (sometimes) comments in the source files. There are the numerous web pages, created by enthusiasts all over the world. There are the archives of newsgroups and mailing lists. Is all this not enough then?

Actually, no it isn't. The Internet is a possible source of information when you use your Linux system for one of the things a lot of other Linux users use their system for as well, for instance a Linux web server or firewall. However, when developing an embedded Linux system, the community of information providers via web pages and participants via newsgroups is a lot smaller, so the probability that the information you need is available somewhere on the Internet becomes relatively low.

At present, the Red Hat Embedded Linux Developer Suite provides very little or no documentation, neither in digital form nor on paper. As a result, the learning curve when using Linux in an embedded project is a steep one.

8 Development methodology

8.1 QNX NEUTRINO RTOS v6.2

The QNX NEUTRINO RTOS originally used the host = target approach only i.e., host and target are the same machine. As was mentioned earlier, the QNX NEUTRINO RTOS v6.2 can be configured with only a microkernel, as well as with many other modules turning it into a fully fledged multi-user operating system capable of serving as a development environment. The advantage of this approach is that the user has the option to do it all on one machine: the application can be tested on the same machine as it was developed on, debugging can be done locally, etc. There are no problems with communication between host and target.

Developers that prefer a standard MS-Windows desktop to the QNX desktop can use cross-development tools. The Metrowerks IDE for MS-Windows can be used to allow the user to do the compiling and debugging from the Windows based host machine.

There's often a lot of discussion about which development method is the better one: self-hosted or cross development. It all really depends on the quality of the tools. If the quality of the cross-development tools is poor, it is better to opt for self-hosted development, and vice versa. Unfortunately, evaluating the quality of the development tools is not within the scope of this report. Nonetheless, it is important advantage that an OS support both methods.

8.2 ELDS v1.1

Red Hat uses the host ≠ target approach. The development environment (ELDS) runs on a standard x86 Red Hat 7.1 or 7.2 host system. The target can be any of the major targets: ARM, MIPS, PowerPC, SuperH or x86.

On the host, the developer can use the wide range of well-known and proven GNU development tools that are available, in combination with a large number of graphical tools and graphical front-ends that are packaged with the Red Hat distribution.

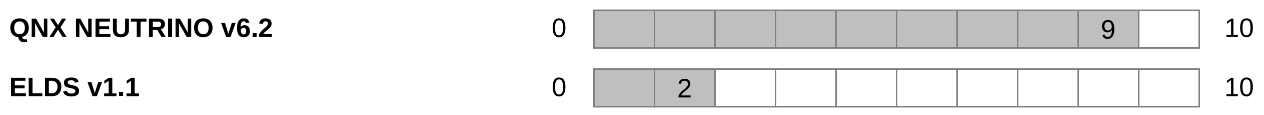
Red Hat provides RedBoot as an embedded debug and bootstrap solution on all the supported targets. RedBoot supports downloading and debugging of applications, flash and network booting of the Linux kernel, and downloading and updating of flash images remotely via serial or Ethernet connections.

RedBoot furthermore provides a target-side hook (a "stub") that enables gdb, the GNU debugger, to communicate with applications built with gcc or g++ running on the target board.

The Linux kernel can be compiled with support for dynamically loadable on demand kernel modules. In this way, during development, different development versions of a driver can be loaded and unloaded.

9 Test results

For a description of the ratings, the reader is referred to appendix D in the document “report definition and test plan”, which can be downloaded from our website (http://www.dedicated-systems.com/encyc).



9.1 QNX NEUTRINO RTOS v6.2

The QNX NEUTRINO RTOS exhibited fast and predictable behaviour during all phases of our testing. None of the tests revealed any problems of any kind. Hence the QNX NEUTRINO RTOS was given a score of 9 for this category.

9.2 ELDS v1.1

As expected, our tests revealed that the Red Hat Linux kernel is not foreseen for real-time purposes. This is normal, as it is a GPOS optimised for General Purpose applications. It has to be said that Red Hat Inc. does not claim any real-time behaviour.

If you would compare the tested Red Hat Linux kernel (7.2) with another GPOS we tested some years ago (Windows NT 4.0) then the results are similar (some are better on the linux platform others are better on Windows NT). Both OSs are not capable of keeping time constraints.

For Windows NT we therefore gave a score of 2 on the real time behaviour found in the test results. For the Red Hat Linux kernel, the score had to be less as we found major flaws in the behaviour of some POSIX API calls. Although Windows NT 4.0 is not real-time we didn’t found API flaws in it… Finally we decided to give it the same score as Windows NT 4.0 due to the better interrupt handling.

The main problems detected in the Red Hat Linux kernel are:

- A Semaphore release does not imply a rescheduling of ready threads (even if a higher priority thread was blocked on the semaphore).

  It has to be said that this behaviour is stated as such in the linux manpages! But this makes this API call not compliant with the behaviour as defined in the POSIX standards! The behaviour is surely not what a real-time programmer expects.

- Both thread yield calls (as defined in POSIX 1003.1b and 1003.1c: sched_yield() and pthread_yield()) do not work. They do something, but the behaviour is not like it is defined in the POSIX standards (put the thread at the back of the ready queue on it’s priority level)

  This bug is known and documented already since the late 1990’s, however it is still not solved in the Red Hat Linux 7.2 kernel.

9.3 Comparison test results

The same test suite was applied to both operating systems.

Table 1 presents most of the performance test results for both RTOS. It does not include the results of the stress tests or the TCP/IP tests. When a test for a particular RTOS is marked "TNE", it means this test was not executed. For a summary of the test identification codes, please refer to Appendix A : Summary test identification codes.

Remarks for the ELDS v1.1:

– The semaphore implementation is not correct: therefore the creation/deletion is fast, but when releasing a semaphore the kernel does not reschedule. We had to make a workaround to have comparable results with QNX.

– As most GPOS, the average time is good. However, for real-time performance the worst case and dependence of the worst case with system load is far more important.

	QNX NEUTRINO v6.2		ELDS v1.1	
Test identification	Average (µs)	Maximum (µs)	Average (µs)	Maximum (µs)
IL-a-1_ISR	1.7	4.3	3.2	4.0
IDL-a-1_ISR	1.9	2.7	1.2	2.0
IL-a-1_IST	2.3	7.7	TNE	TNE
SI-a-1_ISR_HI	1.6	2.5	3.0	5.3
SI-a-ISR_LO	4.1	4.9	7.0	10.9
TF-a-1	175	2880	231	296
TF-b-1	78	102	225	884
TSL-a-2	2.6	8.3	1.7	101
TSL-a-10	2.8	7.9	2.8	123
TSL-a-128	3.6	9.5	25	536
TSL-b-128	8.8	21.8	TNE	TNE
SEO-a-1	3.4	8.5	0.18	12.3
SEO-b-1	3.2	10.2	0.18	8.5
SEO-d-1	2.5	9.1	1.2	9.7
SEO-e-1	2.6	6.4	1.0	27.7

	QNX NEUTRINO v6.2		ELDS v1.1	
Test identification	Average (µs)	Maximum (µs)	Average (µs)	Maximum (µs)
SEO-f-max	6.6	12.7	22	61
SEO-g-3 (mutex)	7.4	13.7	0.6	33.4
FS-a-1	6575	11596	74	8902
FS-b-1	578	759	79	127
FS-c-1 (1 byte)	93	116	3.6	33
FS-c-1 (1 block)	123	440	9.7	41.3
FS-c-1 (10 blocks)	326	710	100	224
FS-d-1 (1 byte)	2340	29998	2380	21180
FS-d-1 (1 block)	20936	54369	6880	46531
FS-d-1 (10 blocks)	24519	58778	26310	60738

Table 1: Performance results for QNX NEUTRINO RTOS v6.2 and ELDS v1.1

Aside from the performance tests in Table 1, we also executed stress tests. These stress tests try to detect memory leaks or performance degradation when the system is loaded. No such problems were detected in the QNX NEUTRINO RTOS 6.2. Also Red Hat Linux 7.2 did not have any memory leaks, however it did exhibit performance degradation when the system is loaded.

Another stress test is the “billion interrupt” test. It generates a billion (10^9) interrupts (at the same IRQ level) at a programmable frequency and we count how many interrupts were serviced, and how many of them were lost. The test is considered successful if not one single interrupt is lost. An interrupt is “lost” when by the time the next interrupt needs to be generated, the previous one is still being serviced or masked out by a higher priority interrupt (e.g. the clock interrupt). Therefore, this test gives a pretty good idea of the worst-case interrupt latency of the RTOS.

Table 2 presents the results. The QNX NEUTRINO RTOS v6.2 served every single interrupt when they were generated every 9µs. The Red Hat Linux kernel could only handle them all when they were generated with a 60µs interval. Here clearly the difference between a RTOS and a GPOS is illustrated!

	QNX Neutrino v6.2	ELDS v1.1
Maximum Sustainable Interrupt Frequency	9µs	60µs

Table 2: Maximum sustainable interrupt frequency – Endurance Test

10 Conclusion

The QNX NEUTRINO RTOS v6.2 and Red Hat Linux 7.2 Operating Systems were evaluated against the same criteria and test suite.

The QNX NEUTRINO RTOS v6.2 performed very well during this evaluation. None of the performance or stress tests revealed any problems and the RTOS was fast, predictable and reliable at all times. The QNX NEUTRINO RTOS is also the only RTOS that has a true message-based client-server architecture well equipped to handle today's requirements concerning distributed processing, high availability, etc.

The Red Hat Embedded Linux Developer's Suite v1.1 (Red Hat Linux kernel 7.2 based on the Linux kernel 2.4.5) is clearly not foreseen to be used in a real-time environment. Linux is made as a GPOS and the test results illustrate this. The added value of the Embedded Linux Developer's Suite is questionable: it does not make it easier to generate a custom target platform.

Although the Linux kernel is royalty free, it comes with a price: documentation is poor and the API is not compatible with (POSIX) standards. The learning curve to get the kernel up and running on your custom target platform is steep.

! Finally, the reader should bear in mind that we tested the QNX NEUTRINO RTOS v6.2 and ELDS v1.1 on an Intel x86 platform only. The results in this report apply to the x86 platform only, not to any of the other platforms these products support.

11 References

[1] The QNX NEUTRINO RTOS v6.2 evaluation report, Dedicated Systems Experts, 2002. (http://www.dedicated-systems.com)

[2] The Red Hat ELDS v1.1 evaluation report, Dedicated Systems Experts, 2002. (http://www.dedicated-systems.com)

12 Appendix A: Product ratings

This appendix contains the evaluation report summary pages for QNX NEUTRINO RTOS v6.2 and Red Hat ELDS v1.1

12.1 QNX NEUTRINO RTOS v6.2

Product

The QNX NEUTRINO RTOS v6.2.0, from QNX Software Systems Ltd.

Positive points

- Fast performance
- Excellent architecture for a distributed and robust system
- Good platform support

Negative points

- Slow Integrated Development Environment

Ratings

For a description of the ratings, the reader is referred to appendix D in the document “report definition and test plan”, which can be downloaded from our website (http://www.dedicated-systems.com/encyc)

Installation and Configuration	0	8	10
RTOS Architecture	0	9	10
API Richness	0	7	10
Internet support	0	8	10
Tools	0	8	10
Documentation and Support	0	7	10
Test Results	0	9	10

Pricing

Please contact the vendor for detailed and up-to-date pricing information.

12.2 Red Hat ELDS v1.1

Why did we test Red Hat Embedded Linux 7.2 in our RTOS evaluation program, although Red Hat and the Linux community clearly states that Linux is not meant to be used in real-time environments?

A lot of our readers explicitly asked for it. However Linux is not an RTOS! The reader should keep this in mind when going through the results published in this report.

Product

Red Hat Embedded Linux Developer Suite (ELDS) version 1.1 from Red Hat Inc. This is based on the Red Hat 7.2 kernel (derived from the linux 2.4.5 kernel).

Positive points

- No license fee
- Used at many universities: chance of finding graduates with a (basic) knowledge of Linux

Negative points

- ELDS provides little or no assistance to configure the embedded target's kernel.
- There is no documentation provided with the ELDS, apart from a very thin booklet.
- No real-time behaviour
- API not compliant with POSIX standards

Ratings

For a description of the ratings, the reader is referred to appendix D in the document "report definition and test plan", which can be downloaded from our website (http://www.dedicated-systems.com/encyc)

Installation and Configuration	0	3	10
RTOS Architecture	0	3	10
API Richness	0	5	10
Internet support	0	8	10
Tools	0	8	10
Documentation and Support	0	2	10
Test Results	0	2	10

Pricing

Although (Red Hat) Linux itself is a free operating system, the embedded toolset is not. Contact the vendor for pricing information.

However, you do not need the embedded toolset to build a custom (tiny) kernel and build an embedded Linux system.

13 Appendix A : Summary test identification codes

Test identification	Description
IL-a-1(_IST or _ISR)	Interrupt Latency (task to interrupt handler) – no rescheduling – 1 thread. Measured on IST or ISR level.
IDL-a-1(_IST or _ISR)	Interrupt Dispatch Latency (interrupt handler to task) – no rescheduling – 1 thread. Measured on IST or ISR level.
IDL-b-1	Interrupt Dispatch Latency (interrupt handler to task) – with rescheduling – 1 thread.
IDL-b-10	Interrupt Dispatch Latency (interrupt handler to task) – with rescheduling – 10 threads.
IDL-b-128	Interrupt Dispatch Latency (interrupt handler to task) – with rescheduling – 128 threads.
SI-a-1_ISR_HI	Simultaneous interrupts. Interrupt latency of the high priority ISR
SI-a-1_ISR_LO	Simultaneous interrupts. Interrupt latency of the low priority ISR
TF-a-1	Thread Creation time
TF-b-1	Thread Deletion time, thread did not execute
TF-c-2	Thread Deletion time, thread has executed.
TSL-a-2	Thread Switch Latency – 2 threads in the same process
TSL-a-10	Thread Switch Latency – 10 threads in the same process
TSL-a-128	Thread Switch Latency – 128 threads in the same process
TSL-b-128	Thread Switch Latency – 128 threads in different processes
SEO-a-1	Synchronization & Exclusion Object (semaphore) Creation time
SEO-b-1	Synchronization & Exclusion Object (semaphore) Deletion time – semaphore was not used.
SEO-c-1	Synchronization & Exclusion Object (semaphore) Deletion time – semaphore was used.
SEO-d-1	Synchronization & Exclusion Object (semaphore) Acquisition time – no contention.
SEO-e-1	Synchronization & Exclusion Object (semaphore) Release time – no contention.
SEO-f-max	Synchronization test executed with a number of threads equal to the number of task priority levels.

Test identification	Description
SEO-g-3	Synchronization & Exclusion object (Mutex) – Priority inversion prevention time.
FS-a-1	File creation.
FS-b-1	File deletion.
FS-c-1	File synchronous read.
FS-d-1	File synchronous write.

14 Appendix B: Document revision history

14.1 Issue 2.49 Preliminary

Preliminary issue.

This version is based on the full evaluation report on QNX NEUTRINO RTOS v6.2 and ELDS v1.1

However, the full evaluation report on ELDS is not available yet. The vendor (Red Hat) has not yet reviewed this report. If we feel that some comments and remarks of the vendor are justified, they are incorporated in the report (which can change the score).

Initial release will only released after vendor comments.