



RoweBots  
Research Inc.

White Paper – 13 Oct 2009

## Unison, Linux, and $\mu$ CLinux

Author: Kim Rowe,  
CTO, RoweBots Research Inc.

*Unison is a trademark of Rowebots Research Inc. All other trademarks are the property of their respective owners*

[www.rowebots.com](http://www.rowebots.com)

### Overview

Unison™ is an ultra tiny Linux™ and POSIX compatible RTOS that allows users to run Linux applications on a much smaller target system. This white paper clarifies the differences between Linux,  $\mu$ CLinux and Unison from many perspectives.

The issues are discussed in the following categories:

- licensing,
- indemnification,
- size,
- modularity,
- real-time performance,
- ease of use,
- boot time,
- API compatibility,
- porting applications,
- and porting Unison.

Each area is discussed in turn.

### Licensing

Linux has a GPL license which makes the use of a memory management unit (MMU) a requirement unless you are prepared to give away your application. The “you see” Linux variant ( $\mu$ CLinux) does not require an MMU and has the LGPL license which offers less restrictive licensing but still requires any modifications to be provided back to the community.

## Licensing

continued

Unison has a modified open source license which is FREE for commercial and non commercial applications. Your applications are not at risk with this license agreement. Note that Unison is focused on 100% adherence to both Linux and POSIX, therefore; redistribution is limited so there is central testing and quality assurance much like Linux.

Unison also has a modified open source license which offers extended commercial quality enhancements for commercial developers. The commercial elements of Unison are licensed separately. Unison is focused on adherence to both Linux and POSIX, redistribution is limited so there is central testing and quality assurance.

## Indemnification

While BSD and GPL licensed components offer no protection for copyright and patent infringement, the Unison license offers indemnification for the commercial version of Unison. This eliminates any concerns OEMs may have about copyright, patent and trademark issues.

## Size

Minimum Linux configurations are 8M of flash and 32M of ram. Typical  $\mu$ CLinux implementations are of the same size. Some specialized implementations reduce the flash and ram requirements to 2M and 2-8M respectively; however, this is still much too large for a microcontroller environment.

Unison is intended to work in a microcontroller environment with Flash size as little as 8K and Ram size as small as 2K bytes. Large Unison systems run in 512K or 1M of Flash and 64K or 96K of RAM today. This is a completely different footprint and market than Linux and  $\mu$ CLinux.

## Modularity

Linux and  $\mu$ CLinux tend to be a large, interdependent, monolithic structure as seen in *figures 1a-1e*. In comparison, Unison has a very compartmentalized and modular architecture which is layered rather than interdependent. The Unison nano-kernel approach allows higher level features and modules to be easily removed and makes the architecture easily and quickly understood. The Unison architecture is shown in *figure 2*.

## Real-Time Performance

The real time performance of Linux is typically 100 times worse than  $\mu$ CLinux in terms of context switching and communication due mainly to cache flushing associated with the MMU operation.  $\mu$ CLinux, on high performance processors, offers interrupt latencies and context switching times of approximately 20usec each. This response is non deterministic and may be much longer.

Unison is a very tiny implementation with an ultra lite thread control block, minimal interrupt latency and little overhead. For this reason, interrupt latency is much lower and context switching is much faster than  $\mu$ CLinux. The specific numbers are architecture dependent. The time response for Unison is completely deterministic.

## Ease of Use

Ease of use is about several things:

- easily understood architecture,
- easily understood use,
- known APIs,
- simple system configuration,
- and flexible adaptation of existing components.

Unison is ultra easy to use. It comes with 30+ demos out of the box that run immediately (commercial version), extensive  
*text continued on pg.5*

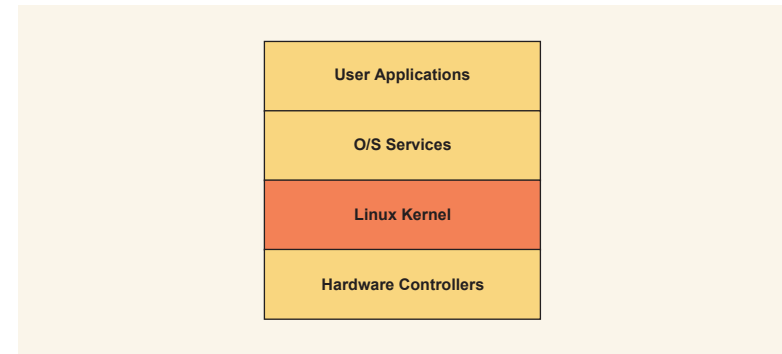


Figure 2a: Kernel Overview

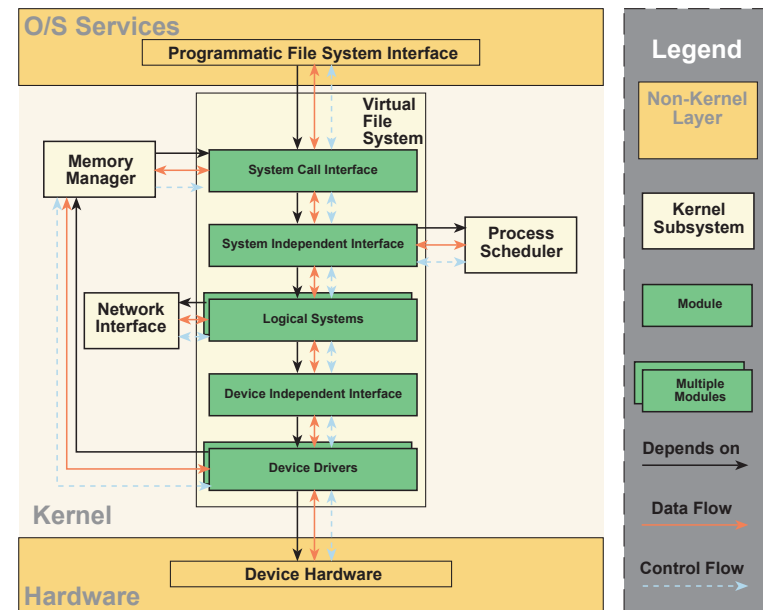


Figure 2b: Virtual File System

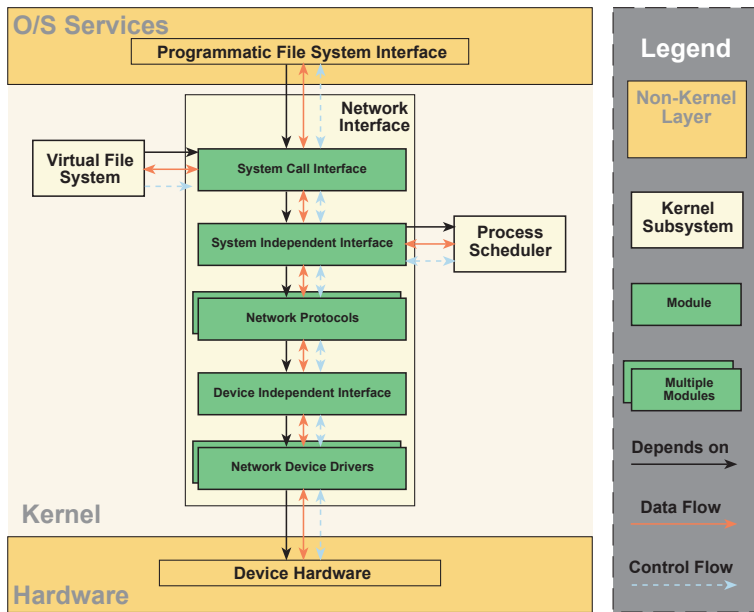


Figure 1c: Network Interface Subsystem

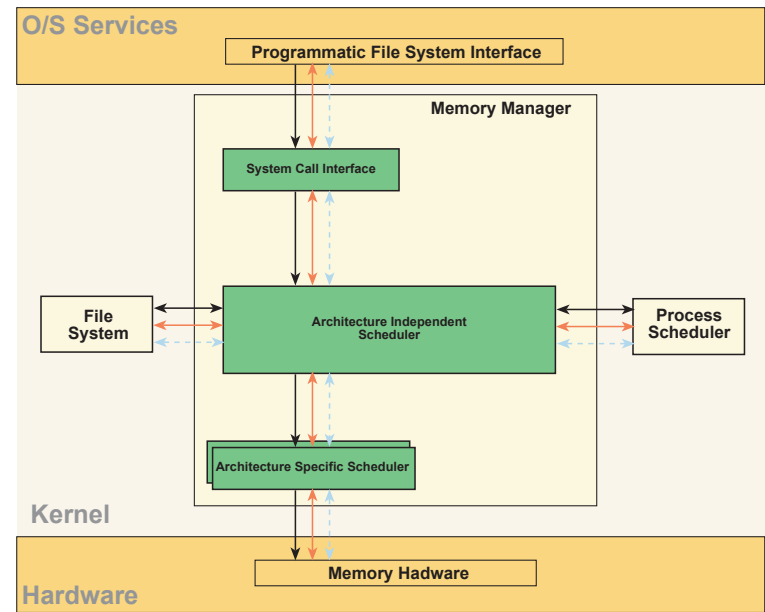


Figure 1e: Memory Manager Details (see figure 2c for Legend key)

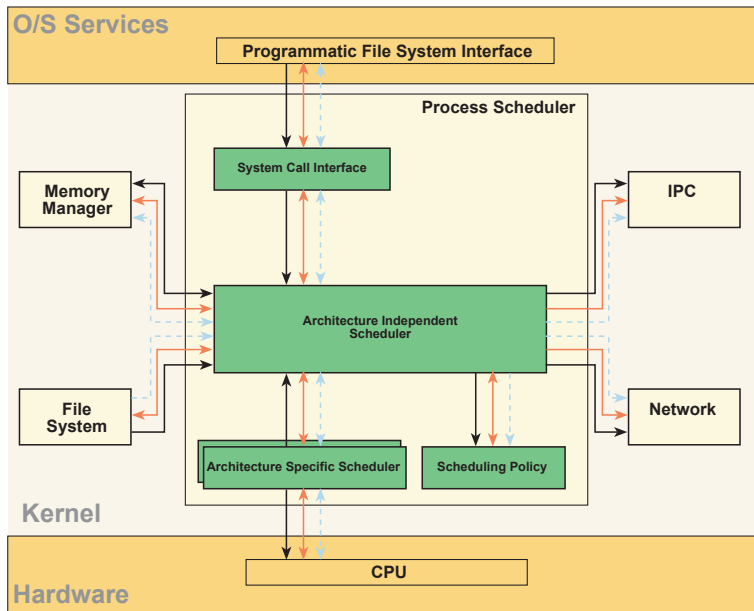


Figure 1d: Process Scheduler Details (see figure 2c for Legend key)

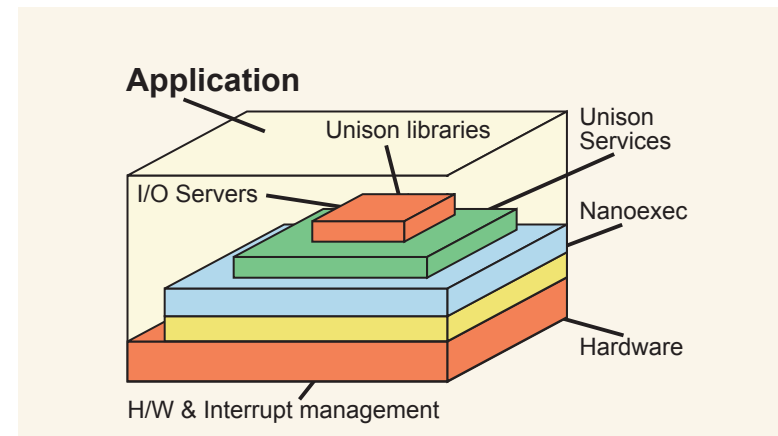


Figure 2: Unison Architecture

## Ease of Use

continued

documentation, Linux and POSIX APIs and a programming by example approach. The very modular Unison architecture is shown in *figure 2*.

For comparison figures 1a-e show the complexity and structure of the various components of Linux and  $\mu$ CLinux. They are both very complex to configure, monolithic and not that flexible. The APIs are well known for Linux but few would call the architecture easily understood.  $\mu$ CLinux offers non standard calls. As long as you are totally at the application level, Linux is relatively easy to use;  $\mu$ CLinux less so.

## Boot Time

Linux takes many seconds to boot.  $\mu$ CLinux can boot in a few seconds although many OEM products take up to 30 seconds to boot. In comparison, Unison typically boots in tens of microseconds.

## API Compatibility

Unison offers a subset of Linux APIs. It has an extensive set of primitives for pthread management, mutexes, semaphores, message queues, memory management, file I/O, BSD sockets, select, conditional variables and interrupt management. All calls are 100% POSIX and Linux compatible including all error codes.

[www.rowebots.com](http://www.rowebots.com)

In comparison, while Linux has a more extensive set of calls, the core Unison calls are ideal for MCU based development of multithreaded applications.  $\mu$ CLinux uses non standard calls in several places.

## Porting Applications

To port Linux or  $\mu$ CLinux applications to Unison you need to know the following things:

- a) Unison offers a complete multiple thread, single process model with complete POSIX and Linux APIs including standard error responses.
- b) Without multiple processes, the main features that are not required are fork and exec. Unison does not support these calls.
- c) Unison offers a special thread specific signal mechanism that does not support all process signals.
- d) Unison offers an environment that scales from a few Kilobytes to a few Mbytes and offers great modularity to do this. For this reason, it does not have a file system requirement. It is optional. File names are absolute paths. Pipes are not supported.

Given these restrictions, applications can be easily ported to Unison from Linux or  $\mu$ CLinux. Generally, signals are converted to thread specific signals, or replaced with semaphores, mutexes, conditional variables or messages. Fork and exec are replaced with pthread\_create. All other calls are supported directly along with error handling. It is necessary to use the Unison include files which share the same names and very similar definitions.

## Porting Unison

Unison can be easily ported to a new MCU.

White Paper – 13 Oct 2009

## Unison, Linux, and $\mu$ CLinux

Author: Kim Rowe,  
CTO, RoweBots Research Inc.

*Contact Information:*

**Kim Rowe**, Founder

[sales@rowebots.com](mailto:sales@rowebots.com)

+1 519 208 0189

+1 519 498 6917



**RoweBots**  
Research Inc.