

# Training

Embedded-systems training at LynuxWorks



## A knowledgeable engineering team is a productive engineering team

Developers of tough embedded applications have known for over 20 years that LynuxWorks™ operating systems provide the unmatched performance, stability, and safety that their projects demand.

Of course, there's more to LynuxWorks than just solid operating systems.

Enroll in a LynuxWorks programming workshop and learn to get the most out of your chosen LynuxWorks operating system.

A week of hands-on labs with LynuxWorks' expert instructors provides the

solid foundation needed in four key areas of embedded-system development:

- System integration and BSP (board-support package) configuration
- Application development
- Device-driver development
- Development tools

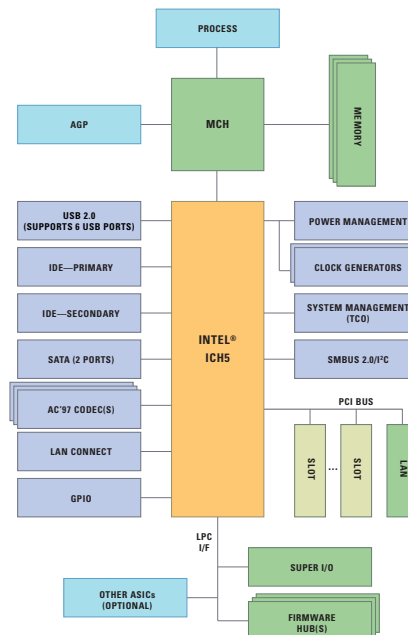
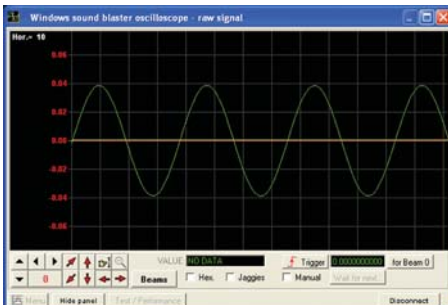
LynuxWorks programming workshops are ideal for software engineers and project managers with 2+ years of experience in C applications, especially for Linux or LynxOS operating systems.

### Key features

- Week-long training workshops with hands-on labs in four key areas
- Develop portable and efficient device drivers supporting PCI, DMA, interrupts and kernel threads
- Write code using POSIX calls for maximum code portability
- Practice exercises on a preconfigured x86 COTS (commercial off-the-shelf) platform
- Learn to design and fine-tune state-of-the-art multi-partition embedded systems
- Learn how to save time with debuggers and analysis tools

### Sample lab:

*Developing an ICH5 AC'97 CODEC output driver*



Experienced programmers who are new to embedded-systems development will find our courses especially valuable.

Even those who are not yet using LynuxWorks operating systems will benefit from the skills learned in our training courses.

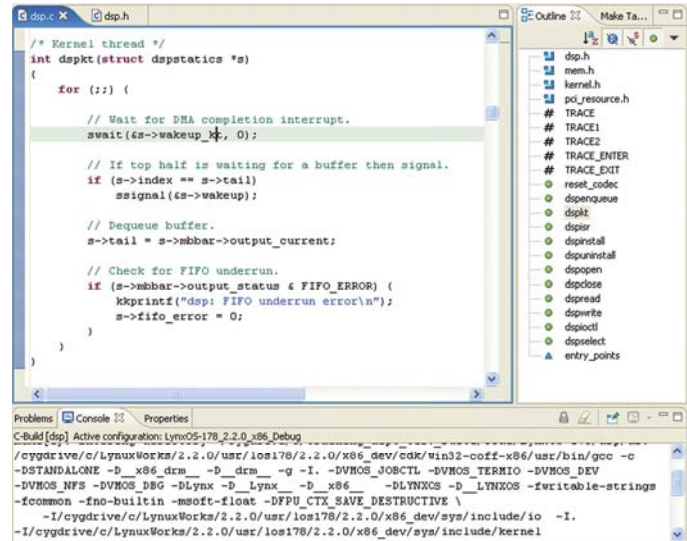
## Teams need engineers who can write device drivers

A crucial component in many embedded and real-time systems is the device driver. Device drivers manage the connections between devices and the operating system or application.

Writing device drivers is such an important skill that we spend significant time on the subject, allowing you to write a device driver that maximizes use of your embedded platform.

### Sample device-driver syllabus

- Interaction between operating system, drivers, devices and applications
- Driver components
- Driver synchronization methods
- Interrupt handlers
- Physical address translation and DMA
- Programming PCI devices
- Kernel threads and priority tracking
- Methodologies for optimizing drivers for real-time



```
/* Kernel thread */
int dspkt(struct dspstatics *s)
{
    for (;;) {
        // Wait for DMA completion interrupt.
        swait(&s->wakeup_key, 0);

        // If top half is waiting for a buffer then signal.
        if (s->index == s->tail)
            ssignal(&s->wakeup);

        // Dequeue buffer.
        s->tail = s->mbbar->output_current;

        // Check for FIFO underrun.
        if (s->mbbar->output_status & FIFO_ERROR) {
            printk("dsp: FIFO underrun error\n");
            s->fifo_error = 0;
        }
    }
}
```

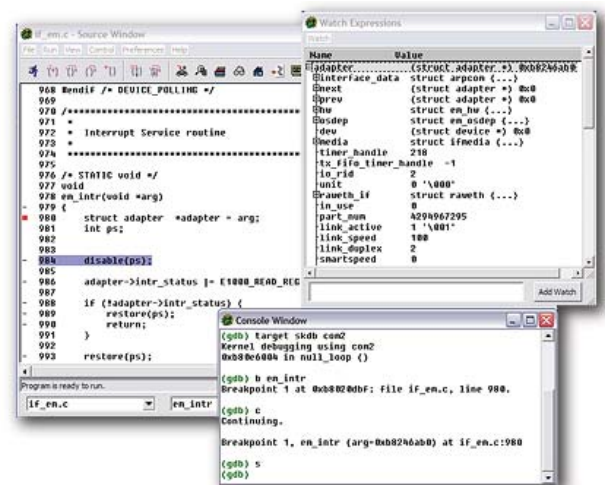
Editing kernel threads in Luminosity Eclipse-based IDE

## Hands-on labs—a focal point of our workshops

We schedule more than a dozen hands-on labs into our week-long courses to demonstrate real-world applicability. To save time and maximize your learning opportunities, preconfigured x86 COTS (commercial off-the-shelf) platforms are provided for most courses offered at our San José training center.

### Sample hands-on labs

- Cross Development Kit (CDK)
- Open Development Environment (ODE)
- POSIX applications
- Character device driver template
- Multithreaded debugging
- Kernel debugging
- System event tracing (SpyKer)
- Develop an ICH5 AC'97 CODEC output driver
- Create a PCI device driver using scatter/gather DMA



Kernel debugging (hands-on lab)

# Learn to write portable POSIX code

The POSIX® standard assures code portability between systems. POSIX is increasingly mandated for commercial applications and government contracts.

LynxWorks operating systems have embraced interoperability and the POSIX standard for years. Our LynxOS real-time operating system is POSIX-conformant and features Linux® ABI-compatibility to allow Linux application to run unchanged inside a LynxOS environment.

Our RTOS courses delve into the all-important POSIX model of parent-child processes, schedulable threads, and virtual-memory management.

As a result, you will have a complete understanding of our operating systems,

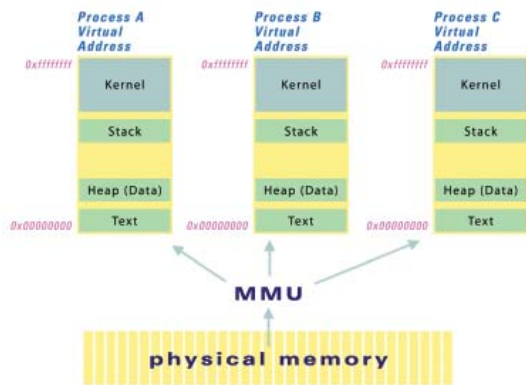
allowing you to use them to your best advantage, whatever the nature of your current project.

## Sample POSIX syllabus

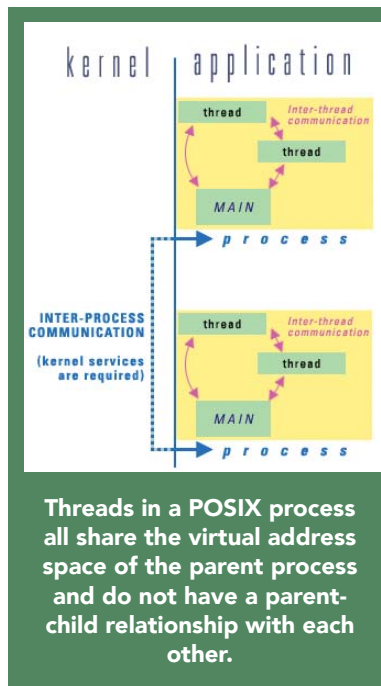
- Processes
- Virtual address space
- `fork()` and `exec()`
- Threads
  - Thread model
  - Thread-safe functions
  - Creating a thread
  - Thread and process attributes
  - Scheduler
- Pipes
- Signals
- Date/time
- Timers
- Shared memory
- Semaphores and mutexes

- Condition variables
- Barriers
- Reader/writer locks
- BSD socket programming

# Discover the power of programming in POSIX

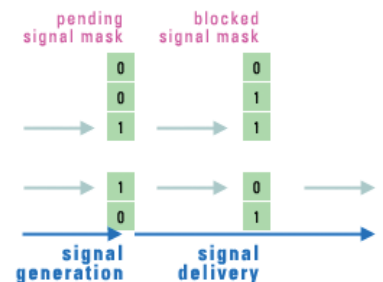


*In POSIX, each executing instance of a program is called a process, and is created when a parent process invokes the `fork()` call, or when a process is spawned. Each process has its own protected address space.*



Threads in a POSIX process all share the virtual address space of the parent process and do not have a parent-child relationship with each other.

*POSIX signals may be caught, ignored, blocked, unblocked, or used to trigger events like termination of a child process or informing a process that it has issued a memory violation.*



## Learn to use state-of-the-art debuggers and analysis tools

Learn approaches to debugging embedded systems at both application level and system level, as well as core dump analysis.

Simply put, all bugs in embedded systems are not alike. Choose your debugging tools according to the task at hand.

### SpyKer—the ultimate system trace tool

SpyKer™, the first dynamically instrumented system trace tool, provides easy visibility into program execution so you can track down elusive embedded application bugs and fine-tune system performance.

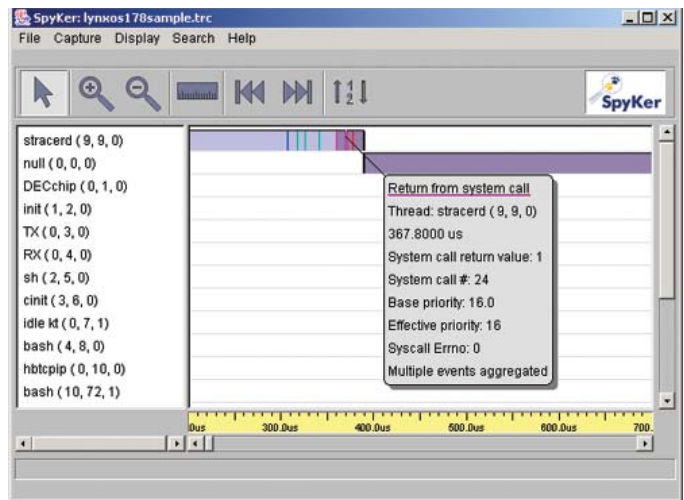
The progression of events and pauses in an embedded system should always be studied to ensure that the interaction, timing and priority of system tasks are as the developer intended.

SpyKer captures operating-system tasks and application events on a running system and then displays a map of what it sees.

### Advanced debugging with Luminosity—Eclipse-based IDE

Luminosity's powerful debugger allows access to the running target system and displays debugging views such as breakpoints, memory, register and variable views, code disassembly, target connection views, source file navigation and console output windows.

Luminosity's cross process viewer monitors running targets and their processes so that system bottlenecks can be identified. Multithreaded debugging is supported and partition awareness is provided for time- and space-partitioned operating systems such as LynxOS-178.



SpyKer maps application events and system tasks

## Schedules and registration

Our instructors travel the world to deliver customized training on LinuxWorks operating systems at regional and customer locations around the world.

Additionally, regularly scheduled training courses are offered at LinuxWorks corporate headquarters in San José, California.

To register or arrange hands-on training on LinuxWorks operating systems at a facility near you, please contact one of our Training Coordinators at (408) 979-4353 or at [training-usa@lnxw.com](mailto:training-usa@lnxw.com).

Visit our web site at [www.linuxworks.com/training/](http://www.linuxworks.com/training/) for a roster of course offerings at worldwide locations and course descriptions.



1.800.255.5969



**LynuxWorks, Inc.**  
855 Embedded Way  
San José, CA 95138-1018  
408.979.3900  
408.979.3920 fax  
[www.linuxworks.com](http://www.linuxworks.com)

**LynuxWorks Europe**  
50 Broadway  
London SW1H 0RG  
United Kingdom  
+44 208 906 9506  
+44 208 906 2338 fax

©2011 LynuxWorks, Inc. LynuxWorks and the LynuxWorks logo are trademarks, and LynxOS and BlueCat are registered trademarks of LynuxWorks, Inc. Linux is a registered trademark of Linus Torvalds. All other trademarks are the trademarks and registered trademarks of their respective owners.

All rights reserved. Printed in the USA.