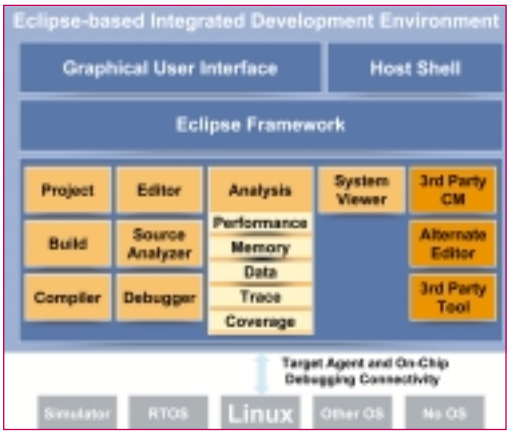
Open-source is changing the way developers work

by Pauline Shulman and Lori Fraleigh, Real-Time Innovations

OPEN-SOURCE IS CHANGING THE WAY EMBEDDED DEVELOPERS DO THEIR JOBS. WHILE LINUX USE IS TAKING OFF IN EMBEDDED APPLICATIONS, ECLIPSE IS ALSO EMERGING AS A POWERFUL OPEN-SOURCE FRAMEWORK FOR DEVELOPER TOOLS.



n The trend toward faster and cheaper silicon architectures, lower-cost and higher-density memories, and pervasive connectivity make Linux a viable and compelling platform for embedded developers. This is especially true for consumer and telecom applications where new and increasingly sophisticated device requirements continue to drive the movement to a formal operating system. The growth of Linux is further supported by organizations such as the Open Source Development Lab (OSDL) and the Consumer Electronics Linux Forum (CELF) who are providing specifications that are helping to standardize the Linux operating system.

The emergence of an open-source Linux embedded platform has energized a tools-centric focus among leading embedded solutions providers. Many of them, including LynuxWorks, MontaVista Software, TimeSys, and Wind River, are creating open-source Eclipse-based IDEs. The Eclipse platform has emerged as a widely supported open foundation for IDE-based products and an extensible framework for interoperability with other third-party and in-house plug-in tools. Software tool makers like Real-Time Innovations (RTI) are adopting a strategy of offering their tools as Eclipse plug-ins and view Eclipse as the best and most cost-effective way to integrate their offerings with other software development tools suppliers. Leading software providers see Eclipse as an opportunity to leverage the ecosystem of strong technology partners to empower developers by providing them with a compelling development environment and the best choice of tools. With its new Eclipse-based Workbench, Wind River can offer its customers the same rich development environment whether they use Linux or VxWorks targets, or a mix of the two. For developers, Eclipse offers the promise of a better automated work flow by providing a means for plug-in tools to share data and services at each step in the development process.

Most of the standard tools for software development are geared towards building and bringing up a new system. Typically, these tools help engineers write code, compile it, and debug the system in a stop-and-start fashion. None of these tools, however, fulfil the developer's need to understand exactly what happens when the application is actually running. Dynamic visualization and analysis tools, including the RTI ScopeTools, are emerging as a special class of tools that do fit this need. They gather large, statistically significant amounts of data about a running system, analyze it, and present useful information graphically in a way that makes it easy for a developer to understand and improve an application. They help answer basic questions every developer will have about their application. Am I getting the best performance possible from the CPU? Is my memory usage effective and reliable? Is the integration between my code and third-party code functioning properly? Is my system behaving as expected? Is my testing efficient and complete?

Without the right tools, addressing any of these questions could take days or weeks. In many cases, without analysis tools, these questions cannot really be answered at all. Dynamic visualization tools can provide insight into a running system within minutes. In the past, such tools were very expensive and available for only a limited selection of operating systems and microprocessors. Recently, however, they have become more widespread, have dropped in price, and are available for the most popular operating systems, including Linux. Here we'll focus on the tools that can help address the first two questions: system profilers and memory leak detection tools.

Let's look at the first question, "Am I getting the best performance possible from the CPU?" There are many different types of profilers. If you want a quick picture of what is going on in the entire system, then a dynamic system profiler is what you need. It will help answer "Where are the processor hotspots?" "Is a particular thread using more CPU than expected?" "Does a particular system call or library call seem to use a reasonable amount of CPU when called in one thread, but then usage across

all threads seems too high?" A system profiler allows you to visualize how the CPU is used across all threads and even interrupt handlers in the system. For example, with it you can watch the progress of a network packet as it goes through the application layer, the socket layer, the TCP layer, the IP layer, and out the Ethernet driver. If you are using a profiler that updates its information graphically in real-time, you can even see how the performance scales with increased numbers of packets, with different packet sizes, or using different network stacks. Dynamic system profilers also allow you to analyze the trade-offs between different system calls (such as bcopy and memmove) or the trade-offs of performing certain operations in hardware or software (such as datagram checksums).

There are also a variety of tools that help engineers track down memory leaks. Does your code make a system call or other API call that allocates memory? Did you read ALL the documentation, including the fine print that says you, the caller, need to free the memory when you are done with it? We've all heard horror stories of companies unable to ship systems because they crash after three days of system tests, or worse yet, they crash after one month in the field. In reality, most memory leaks occur in code you did not write. (As we all know, the bug could not possibly be OUR fault.) If you are using tools that require scouring the source code to find the leaks, you're going to be recompiling the entire application and operating system for hours every time you want to debug. Since most projects incorporate a variety of third-party applications or libraries, often without the full source code, you are likely to be puzzling for days trying to figure out where the leak is occurring. Dynamic analysis tools, which analyze any binary on the system, are going to be the key to finding these "well-hidden" memory leaks. If you have a tool that plugs directly into your development environment (namely Eclipse), you can instantly analyze your already running system and save yourself days of frustration. These tools will help you answer the second question above, "Is my memory usage effective and reliable?"

On their own, dynamic visualization tools provide a wealth of information. If you run them within an Eclipse-based IDE, you can accomplish even more. Do you want to see the source code that leads to a particular memory leak? Just select the information in the memory tool and the source code is displayed in your editor. Wouldn't problem solving be much easier if you could see a graph of variables as they change in real time? Just bring up your debugger, highlight the variables you want to plot, and watch how your variables change both when you hit breakpoints and in between. Want to see the results of tests run on multiple distributed systems? Create a project that compiles the test code, downloads it to the remote target, runs the test, uploads the results, and automatically generates a report of the results. This is just the tip of the iceberg. As more and more vendors adopt Eclipse as a strategic part of their Linux-based platform offerings, you can expect to see greater numbers of features integrated into everyday use. Developers can leave the dark ages of printf behind them and evolve to a higher standard of system analysis. n