Linux Eases Development of Advanced Capabilities for Smart Wireless Devices

From a designer's point of view, managing the integration of smart wireless devices while maintaining time to market schedules and project budgets can be a challenge.

Unlike the low-cost cell phone market, where products boast few bells and whistles beyond the ability to make and receive calls, smart wireless devices are all about features. From the user's point of view, carrying a single device that does everything on the road is much better than having to carry multiple devices and manage synchronizations. A designer of a smart wireless device has a much different perspective. Managing the integration of today's complex devices while maintaining time-to-market schedules and project budgets can be quite challenging for even the most seasoned engineering teams.

The use of Linux as an operating system (OS) in the server market has proliferated because of its modular extensibility, open source code community, vast libraries of existing applications, robustness and, not surprisingly, favorable licensing. The obvious benefits of these same features suggest that Linux will also become the dominant OS for wireless devices.

A typical Linux based framework has a layered design with several elements. First, the Linux kernel provides the operating system services. Hardware is isolated from the Linux kernel with drivers. Depending on the application, designers can decide to run various system services on top of the kernel such as X-windows, Gstreamer and Graphical Tool Kit (GTK). The applications framework and user interface is then built on top of the Linux service layer.

Development Platforms

To ensure that all critical requirements are being met on time and on budget, a Linux-based development platform has many benefits when designing a smart wireless device. With Linux, designers have access to vast amounts of open source code software that is adaptable to handheld battery operated devices, offering both processing power and memory.

An example of such a software development platform is Google's recently introduced Android, which provides a designer with everything they need to build and run Android applications including a true device emulator and advanced debugging tools. Given the popularity of Apple's iPhone, many of today's software platforms also provide a GUI designed with a touch screen in mind, and allow input from a keyboard, finger or stylus.

An interesting component of the Android platform is its Dalvik virtual machine. Dalvik is Google's runtime environment written in open source Java software,

allowing Java programmers to feel right at home. Dalvik uses Java syntax but not Java byte code or Java virtual machine to execute. Optimized for low for minimal memory consumption, Dalvik executes files in the Dalvik Executable (.dex) format designed to allow multiple virtual machine instances to run at once, relying on the underlying Linux kernel for process isolation, memory management and threading support. Every Android application runs in its own process, with its own instance of the Dalvik virtual machine.

When developing with open software, designers must understand how it can be legally used in their products and the implications their choices have on hardware. To ensure the design is compatible with licensing agreements, designers can use an interface framework such as GStreamer. Gstreamer is a library that allows construction of medial handling components. Gstreamer supports various applications ranging from a simple playback application to a more complex audio mixing application to a video editing application. Developers can easily add new codecs and filters by writing a simple plug-in with a generic interface. Gstreamer is released under the GNU operating system Lesser General Public License (LGPL) and has been ported to several other operating systems, including Linux.

To further ease development, Navicron offers both software and hardware fusion platforms that have been built from the ground-up specifically for mobile devices. Navicron’s innovative approach maximizes hardware and software functionality, allowing for the development of a higher quality product with better integration. With high power application processor and support for modern wireless standards and multimedia features, the result for designers is an edge in the market with a product that delivers a better end user experience.?

High-level Programming Languages

Smart wireless device designers using Linux also have the flexibility of selecting number of high-level programming languages and development environments for their projects. When using a high-level software framework, the application development becomes fast and easy. The trade-off is that high level languages will consume lots of memory and processing power. Designers can speed development time, but will have the dilemma of adding development costs for processing power and money.

GTK and Python are two popular high-level programming language options. GTK+ is a feature rich toolkit for implementing graphical user interfaces and features. GTK+ was originally written in C language, but has ties to many other popular programming languages such as C++ and Python. GTK+ is licensed under the general public license GNU LGPL 2.1, allowing development of both free and proprietary software with without any license fees or royalties. GTK+ has number of features that today’s wireless developers need including an easy look and feel, object oriented programming, theme support, internationalization, localization and accessibility.

As another option, Python is a powerful and fast object-oriented programming language and a great choice to speed application development for wireless devices.

Python offers strong support for integration with other languages and tools and performance improvements can be wrapped in C or C++, making it flexible. Implementation of Python is also under the open source license, which makes it freely usable and distributable and supported by a set of extensive standard libraries. Python is popular with developers due to the fact that is can be learned in just a few days. In fact, many programmers report substantial productivity gains when using Python. Python runs in many software environments such as Windows, Linux, .NET, Java, and most interestingly, the Nokia Series 60.

Typical application programming interfaces (API) for Python include extension modules that handle all of the required and most popular features for a phone such as dialing, messaging, taking a photo and internet services. With Python, users also enjoy access to a GUI library with menus, styles, dialogs, etc. Python’s core syntax is minimalist, while the libraries are extensive. A trade-off to consider with Python is that applications may require more memory and processing power in order to run seamlessly.

Applications

As mobile handsets rapidly morph into a powerful hand-held computer, there is a clear need for innovation and choice in applications. Applications can be acquired various ways. Open source software is widely available and many applications can be found from several different sources. Developers must make certain that the quality of the software meets the specified design requirements and that all licensing conditions are met. Many applications are commercially available and typically, integration into a Linux platform is fairly straight forward.

There are several pros and cons to consider in regards to applications. Traditionally C or C++ has been used to code applications, and is still a viable option. Newer programming tools allow faster code generation and faster time to market. Once again, speed does not come without the cost of additional memory and performance requirements, so the trade-offs must be evaluated.

Supported by Linux, Java ME (Micro Edition) is the most ubiquitous application platform for mobile devices and provides a tested environment. Java ME includes flexible user interfaces, robust security, built-in network protocols, and support for converged services with mark up and scripting languages for web, application and UI models and various system libraries. Applications based on Java ME are portable across many devices, yet leverage each device’s native capabilities.

Linux is gaining significant traction in the mobile and embedded markets as Linux-based phones and products emerge and gain popularity. The emergence of Google’s Android as well as the commitment of global handset manufacturers such as Nokia to the Linux OS points to a clear direction of open-source development. Both software and hardware designers should understand the trade-offs and the consequences of selecting a Linux-based operating system, associated high level programming languages and applications. Using a proven software and hardware development platform to design a smart wireless device can save both time and money.

Matti Kattilakoski is Chairman of the Board for Navicron. He can be reached at matti.kattilakoski@navicron.com [1]; 469-230-3349.is adaptable to handheld battery-operated devices, offering both processing power and memory. However, when developing with open software, designers must understand how it can be legally used in their products and the implications their choices have on hardware.-->FE

Source URL (retrieved on *12/21/2013 - 7:09am*):
http://www.wirelessdesignmag.com/articles/2008/06/linux-eases-development-advanced-capabilities-smart-wireless-devices?qt-blogs=0&qt-most_popular=0

Links:
[1] mailto:matti.kattilakoski@navicron.com