



[MOBILE LINUX]

OSDL Mobile Linux Initiative

The Linux Platform and the explo\$ive mobile market



BY BILL WEINBERG

ABOUT THE AUTHOR

Bill Weinberg brings over 18 years embedded and open systems experience to his role as Open Source Architecture Specialist and Linux Evangelist at the Open Source Development Labs, where he supports initiatives for meeting developer and end-user requirements for Carrier-Grade, Data Center and Desktop Linux.
bweinberg@osdl.org

■ The global mobile phone market is enjoying explosive growth. With annual unit sales in the hundreds of millions, Gartner analysts estimate that by 2009 the worldwide installed base will top 2.6 billion mobile handsets. For the Linux and Open Source segment of the IT industry, such numbers are tantalizing, orders of magnitude beyond shipments and even the installed base for servers, and far greater in volume than the worldwide desktop market. For the Linux software and related hardware markets, mobile phones are an opportunity to “break out” and enjoy greater market share in client devices, complementing the already important presence of Linux in the voice and data communications infrastructure.

Linux on the Move

In 2004 and 2005, Linux made strong gains as a mobile phone OS. In 2005, OEMs in the Asian marketplace shipped 10 million-15 million phones with almost two dozen phone models based on Linux. Top-brand OEMs LG, Motorola, NEC, Panasonic, and Samsung have a strong commitment to the Open Source OS; so do emerging Chinese brands like Datang, e28, Haier, Huawei, and ZTE.

In the July 2005 issue of *LinuxWorld Magazine*, I wrote an article describing why device OEMs, large and small, are choosing Linux as the strategic platform for their smart phones. That article focused on the mix of technical and economic motivators. This article looks at the challenges that still face Linux in attaining even greater ubiquity and design-wins in this dynamic global marketplace. It also introduces OSDL's newest effort, the Mobile Linux Initiative (MLI), describing how MLI members are striving to fill key gaps in the mobile Linux software stack.

Marketshare Gap

For all of the technical and economic benefits that Linux offers mobile device OEMs, Linux phones today account for less than 5% of the total market. In the fastest growing smart phone segment (85%/year according to Gartner), Linux enjoys a stronger position – 25% in Q2 2005 – far ahead of Windows Mobile, PalmOS, or RIM (but behind the SymbianOS). Mid-tier Linux phones are also making inroads into Japan's giant NTT DoCoMo network, with Panasonic and NEC shipping as many as five million mid-tier "feature phones". Panasonic's December 2005 announcement of its intention to focus on high-end phones based on Linux bodes well for the platform in Japan, where Linux phone share could climb to 15%-20% by 2007.

Technical Challenges

Experienced handset makers like Motorola, NEC, and Panasonic have clearly demonstrated that Linux-based mobile telephony is a reality. However, these companies and other established OEMs, as well as new entrants in the handset market, want the process of building Linux-based handsets to be easier, with faster time-to-market and better price-performance. In particular, they want to reduce the hardware BOM (Bill of Materials) burden needed to support a Linux-based phone stack, and to optimize the performance of key technologies.

As a result, at its most recent face-to-face meeting in Tokyo, OSDL's MLI agreed to focus on the following technical areas:

- Development Tools
- I/O and Networking
- Memory Management
- Multimedia
- Performance
- Power Management
- Security
- Storage

Tools

On one hand, GNU tools like GCC and GDB have formed the basis for all types of embedded development for the last decade. In the last two years, the Open Source Eclipse Project has emerged as the IDE framework of choice, and dozens of vendors now build their cross-development suites as derived Eclipse plug-ins. On the other hand, while GNU and Eclipse are "good

enough," handset developers want more. They require smaller code to fit into limited RAM and flash and they want faster code to meet performance goals on clock-scaled mobile ARM processors. They want more intuitive debugging interfaces and standard hardware bridges to target devices that lack network connections and can have multiple symmetric and asymmetric cores (MCUs, DSPs, etc.). They want to be able to mix native C and C++ Linux coding with Java application development. And they want phone-specific tools that address Flash memory programming, performance analysis, baseband development, and handheld device simulation.

I/O and Networking

It's a fact of OEM life that the processors and peripherals deployed in handsets aren't the same as those found in enterprise equipment. As such, many SoCs (systems-on-chip) devices don't appear in the standard Linux architecture trees (even if CPU their cores do) and the peripherals on those chips lack publicly available drivers. Integrated serial, IrDA, USB, I2S, I2C, SPI/SSP, LCD, DMA, display, interrupt and memory controllers need reliable and readily available support, in Open Source, for CPUs like Intel's XScale, Texas Instruments' OMAP and Motorola's MX processors.

Handset-based networking presents its own additional challenges. Besides having access to MAC-level drivers for "normal" Wi-Fi and Bluetooth, they also need off-the-shelf interfaces for wireless voice networking: CDMA, GPRS, and other WAN chipsets present unique driver and stack requirements, and off-the-shelf call stack implementations today are proprietary and offer limited, poor, or no support for Linux.

Another "exotic" area is the need to bridge unexpected heterogeneous media types. Bluetooth and Wi-Fi-enabled phones may need to route voice streams to/from baseband and Bluetooth cordless phones in the home or Wi-Fi LANs for VoIP calling. They may have to route IP network traffic from a desktop computer connected via IP-over-USB to any of several wireless data channels. I even recall a simple customer request for IP-over-USB turning into a requirement for full-blown support for NFS over USB!

Memory Management

Mobile device memory management has its own unique, non-standard requirements. These include non-contiguous physical memory; heterogeneous memory types like volatile DRAM, battery-backed RAM, NAND and NOR Flash; application and OS execute in place; strong protection of base software, and field upgradable downloads of both platform and application software. Another key memory management concern is out-of-memory handling. In enterprise Linux systems, low memory invokes a "reaper" that terminates "stale" processes to free up RAM; criteria for reaping in a phone must disallow disruption of phone service or other compromised handset performance.

Multimedia

For smart phones and many mid-tier phones, OEMs need to port or fully reimplement complex audio and video capabilities to a Linux platform. Barriers to building next-generation multimedia start with the lack of a unified multimedia framework for Linux (which competing platforms have), and also include the lack of Linux-based DRM software, as well as issues surrounding patent-bearing media formats. Eschewing DRM and using patent-free open media formats isn't a realistic alternative for device OEMs.

Performance

For both the GPRS interface, and for other capabilities like multi-media, Linux still needs to move in the direction of RTOS-like responsiveness. Linux must meet deadlines and switch context adroitly in systems where clocks can slow to conserve battery power from 400MHz peak performance down to 40MHz (or even 0MHz) and back in response to policies and hardware events.

The current generation of ARM-based phone chipsets also feature silicon crammed with peripherals. SoC peripherals and secondary cores can be highly stateful with hard-to-program shared memory interfaces connecting them. These channels constitute a troublesome performance bottleneck.

Real-time and Radio Interfaces

In today's crop of Linux-based smart phones, the GPRS interface resides in an encapsulated "modem" containing a dedicated CPU core, a DSP, and RF hardware

to support baseband communications. Offloading the radio function makes it easier to build a smart phone, but raises the cost by adding significant components to an already heavy BOM. While smart phones offer OEMs sufficient margins to bear this cost, the need for a self-contained modem limits Linux's ability to cover the broader market that includes feature phones and entry-level devices.

Theoretically, OEMs could remove the modem and expose the baseband interface to Linux, but doing so also exposes hard real-time requirements at the edge of the Linux response curve.

Power Management

Mobile device manufacturers today face a mind-boggling set of choices among Linux power and energy management schemes. OEMs can look to the desktop where notebook-centric ACPI and legacy apmd dominate (and indeed occupy most discussion of Linux power management on the kernel mailing list). For non-x86/IA-32 hardware, OEMs can turn to ARM's own energy management framework IEM (Intelligent Energy Management), or work with the various power management schemes present on silicon from over 200 ARM licenses (e.g., XScale or OMAP). There also exist unique and more divergent energy conservation protocols from MIPS, its licensees, from FreeScale for its CPU lines, from IBM for the Power architecture, from Renesas and Hitachi, and so on across the silicon supplier universe. OEMs can also build on software paradigms like MontaVista's DPM.

While choice is a good thing, too much choice leads to fragmentation, with OEMs seeking either an extension of existing schemes to address the held space, or the establishment of an "umbrella" paradigm that bridges desktop, laptop, and handheld.

Security

Mobile device security breaks out into several distinct areas, and mobile OEMs seek standard solutions to the following issues:

- **Wireless Network Security:** While the wireless networks that carry mobile traffic are theoretically "closed," they can be compromised, both in terms of access to client and infrastructure devices, and to the streams of voice and data that they carry. This area, however, is usually the purview of carriers and operators, who

present fairly well-defined requirements to handset OEMs. At present, OEMs seem content with how a Linux-based phone fits into their wireless (GPRS/CDMA) security implementations. However, with the addition of Wi-Fi and Bluetooth to handsets, a new set of concerns arise, especially after accounts of phonebook cracking and other skullduggery via Bluetooth.

- **Content Security:** Handset suppliers have requirements to protect both users' personal content (phonebooks, e-mail, etc. and commercial content (movies, music, and other copyrighted material) that are increasingly ubiquitous on smart phones. While a range of strong encryption technology is readily available for Linux, OEMs aver that the same is not true for broader DRM and other content protection needs.
- **Physical Access:** Unlike the remote servers and clients that populate other networks, phones are by definition "handy" – that is, the so-called "black hats" can use physical means to crack device integrity and security. Opening the "clamshell" plastic that protects a phone's innards can also expose peripheral interface pins and allow probing and malicious signal injection. Phones can also be baked – literally heated up – and dropped, shocked, or similarly abused to induce failure modes that further enable circumvention of security precautions.
- **Exploit Resilience:** Security measures focus on limiting access, often creating a false sense of security on the devices themselves. Moreover, embedded toolkit makers usually ship their tools and platforms with the security minimized or disabled to facilitate development, leaving the final secure configuration and deployment up to OEMs, integrators, and their customers. Misconfiguring firewalls, failing to apply up-to-date security patches, leaving a single physical port exposed, or running multiple functions as root (all common with embedded applications) can lead to deploying easily exploitable devices, and both leaving the phones and networks open to compromise.

Storage

Local storage on mobile devices resides in a mix of Flash, RAM, and remote stores. Linux does present a range of embedded file systems – CramFS, JFFS2, YAFFS, RAMFS, pRAMFS, and also semi-proprietary Flash file schemata from Intel, M-Systems, et

al. However, device OEMs report a range of performance issues around journaling, mount time, wear leveling, and support for both NAND and NOR devices.

OSDL MLI - Bridging Gaps

In October 2005, OSDL launched its fourth and latest working group, the Mobile Linux Initiative. MLI includes members from all levels of the mobile telephony ecosystem – chipset makers, Linux distribution and platform suppliers, middleware ISVs, handset manufacturers, integrators, carriers, and operators. Dubbed "Carrier Grade Linux for handsets" by several OSDL members, MLI will strive to address the platform challenges described in this article "from the kernel up" to accelerate Linux adoption on mobile phones and other converged voice and data devices.

In contrast to other industry groups, MLI intends to focus on solution creation, not merely publishing APIs and new standards that can end up as unfunded mandates. To that end, MLI members are today marshaling resources to create unique implementations to meet handset OEM, carrier, and operator needs, to foster the advance of existing Open Source projects, and to open existing internal technologies for the benefit of the MLI audience and the community in general.

To learn more about OSDL MLI, membership and activities, visit http://www.osdl.org/lab_activities/mobile_linux. You can also join the open MLI mailing list. If your company participates in the mobile-wireless ecosystem, please also consider joining OSDL and contributing to the MLI working group.

MLI Mission:

To accelerate Linux adoption in the mobile space

- Identify and address technical and non-technical industry requirements
- Create and foster implementations in Open Source
- Advocate and explain industry needs to the kernel/Open Source community
- Promote mobile Linux (including educating carriers about the benefits of Open Source)
- Clarify legal and regulatory issues surrounding mobile phones as they relate to Linux and Open Source
- Enable and foster pre-platform developer ecosystem 