Toward Mobile Phone Linux

Yukikazu Nakamoto

NEC Network Development Laboratories

4035, Ikebe-cho, Tsuzuki-ku, Yokohama 224-8555, Japan

TEL:+81-45-939-2694, FAX:+81-45-939-2669

e-mail:nakamoto@ct.jp.nec.com

Abstract— Recently mobile phones provide not only voice service but internet access, multi-media message services, games, local communication controllers and so on. Therefore, more productive software platforms are required. We have developed the next generation software platform based on Linux for mobile phones. In this paper, we describe requirements for mobile phone Linux and solution candidate technologies to satisfy the requirements based on the development experience.

I. Introduction

Mobile phones have provided increasingly sophisticated functions in recent years. First, mobile phones have rich peripherals to give users novel services. The peripherals are display devices, cameras, GPS and local communication to external devices. Bluetooth, USB, wireless LAN, memory cards and IC cards are included in the local communication.

In addition to the rich peripherals, software functions in the mobile phones increase rapidly. The following are examples of the current and future software service functionalities of mobile phones.

Internet Access: browsers for Compact or Full HTML, WAP(WML1.x and XHTML)

Messaging: e-mail, short message service, enhanced message service, multimedia message service

Multimedia: display of still image files, playback of animation (including 2D and 3D), movie and audio files

Applications: personal information management, games, electronic commerce, novel applications to utilize the rich peripherals

Program Execution Engines: Java runtime environment, WMLScript, ECMAScript

The above functionalities have become similar to those of the PC. As a result, a more sophisticated OS is required and one of those is Linux. However, Linux has shortcomings in application to mobile phones.

In this paper, we present requirements and solution candidates in applying Linux to mobile phones. Sec. II mentions the background of requirements. We describe the requirements for Linux kernel and solution candidates in

Fig. 1. Evolution of hardware/software in mobile phones

Sec. III as well as those for middleware for mobile phones in Sec. IV. The future trends in mobile phone software are briefed in Sec. V and conclusion is mentioned in Sec. VI.

II. Backgrounds

Fig. 1 shows an increasing ratio of software development size in mobile phones. The size of mobile phone software has been expanding accordingly, currently exceeding 1 million steps. Software architecture in current mobile phones is shown in Fig. 2 to implement functions described in Sec. I. Requirements of the multifunctionalities and the quality of mobile phones lead to complexity of software structure in mobile phones.

Current mobile phones utilize a real-time OS for embedded systems. Drivers for peripheral devices in the phones, communication software, libraries shared between applications, and application software are placed on top of that real-time OS. In recent years, various innovations for more efficient software development have been achieved in the mobile phones, such as the introduction of window systems similar to those of PCs[1]. However, there are several shortcomings in the real-time OS for developing large-scale software in mobile phones as follows.

- Processors and OS that do not support virtual mem-

Fig. 2. Software architecture in mobile phones

ory and memory protection mechanism are frequently used for mobile phones. It causes various difficulties in the development stage, such as detection of illegal access to memory areas, for which access is not permitted.

- The available software development environments are limited. Application software to be distributed in the market is also more limited than in the PC software field. As the size and the number of mobile phone software keeps on expanding, it has become necessary to introduce a new software platform that allows efficient development and for which it is easy to employ engineers and acquire the software development environments and application software.

In recent years, sophisticated OSs such as Linux for PCs and Symbian OS¹ for mobile equipment such as PDAs can be utilized. A large number of software development environments are also available for these sophisticated OS, and numerous types of application software are distributed in the market. A large number of software development engineers also exist. These sophisticated OSs are also attractive for mobile phones in many respects and they are promising candidates as future OSs for mobile phones. In a sophisticated OSs that provides the virtual memory spaces and the memory protection mechanism, application programs are protected. This will improve development efficiency, for examples by facilitating detection of illegal memory access during the software development.

The mobile phones have, however, the following minimum requirements for the quality and the reliability as a phone which distinguish them from the PC.

1. CPU power in mobile phones is much lower than in PCs (Fig. 1). CPU frequency of the current mobile phones is between 50 MHz and 100 MHz while a PC frequency is 2 ~ 3 GHz. Memory size in mobile phones is also smaller than in PCs. Memory size of the current mobile phones is between 16MB and 32MB while that of PCs is 256 ~ 512 GB. Mobile phones do not have hard disks right now.
2. Mobile phones are driven by secondary batteries, not wired power supply. Thus power consumption in mobile phones is of much concern.

¹<http://www.symbian.com/>

Fig. 3. Power consumption in a wireless communication PDA [7]

3. In terms of reliability, a mobile phone should be able to catch a call anytime. Therefore the phone should notify a user to receive a call even when the user plays a game or uses a browser to access the Internet. A voice communication in a phone should not break. These requirements come from mobile phones being an extension of wired-phone.
4. Users of a mobile phone are generally inexpert in computer technologies. On the other hand, most PC users are considered to be more skillful in usage of computers.

III. Requirements for Linux Kernel and Solutions

One of the sophisticated OSs is considered to be Linux. However, there are weaknesses in the Linux kernel when the kernel is utilized for the mobile phones as follows.

A. Power Consumption

In PCs, static power management has been realized such as suspend/resume mechanism, which is implemented by ACPI (Advanced Configuration and Power Interface)[2]. Only the static power management is not effective for mobile phone usage for two reasons. First, mobile phones are driven by secondary batteries, not a wired power supply. More efficient power usage is required to increase utilization time of the mobile phones. Second, overhead of the static power management is very large and the latency in the activation and de-activation of the system is significant. Moreover, mobile phones have no space to store suspended memory image because of having no disk in the phones. The dynamic power management in which the system power is managed while programs in the system are running is a solution candidate to overcome the drawbacks [3].

Most power management research efforts have been mainly done on the CPU (e.g. [4, 5, 6]). However, the other components beside CPU also consume power in mobile phones and portable devices. Fig. 3 shows the power consumption ratio of components in wireless communication portable devices, as an example[7]. This figure indicates that the system-level power management is required, in which the CPU and the other components are controlled.

In dynamic power management, not only the power consumption of the CPU, but that of the other components in the system is controlled. To show effectiveness of the dynamic power management, we summarize the power consumption model of CPU and peripherals based on [8].

CPU: When the CPU operates at a clock frequency f , the power consumption P_d of the CPU is given by $P_d = c_{sw} \times V_d^2 \times f$, where c_{sw} is the switch capacitance and V_d is the voltage supply. The relationship between the clock frequency and the supply voltage is given: $f = k \times \frac{(V_d - V_t)^2}{V_d}$, where k is a constant and V_t is the threshold voltage. As a result, we can approximate the relationship: $f = a \times V_{dd}$, where a is a constant. Thus we obtain the relationship between the power consumption and the frequency:

$$P_d = C_1 \times f^3$$

where $C_1 = \frac{c_{sw} a}{a}$.

Main Memory: Main memory is operated on at a fixed voltage and its power consumption is proportional to the frequency. The power consumption model of the main memory is given:

$$P_d = C_3 \times f,$$

where C_2 is a constant.

Display devices and speakers: These components consume a constant power from the viewpoint of the frequency. We can show the power consumption model:

$$P_d = C_4,$$

where C_3 is a constant.

Others: To represent the variations of DC-DC regulator efficiency across the range of output power, CMOS leakage currents, other second order effect, we need a term:

$$P_d = C_2 \times f^2,$$

where C_4 is a constant.

We can represent system-wide power consumption as follows.

$$P_d = C_1 \times f^3 + C_2 \times f^2 + C_3 \times f + C_4$$

This equation indicates that scaling frequencies with voltage changes is very effective for system-level power saving.

Recent CPUs have the voltage and frequency scaling capability for low power consumption which is controlled by programs. Other peripherals can be also controlled by the similar capability. For example, frequencies provided to SCI, timer, MMU etc are set to on/off and controlled by programs in SH3[9].

Dynamic power management is performed by application programs while the static power management is done by BIOS. An example of application program control is a

Fig. 4. An example of operating states, operating points, and a policy

video decoder program. The program executes the highest frequency at some period at which video frame arrives and must be decoded. The program is in idle state except at the execution times.

Setting frequencies of the CPU and peripheral components to some values means that the CPU and the peripheral components are in the corresponding states. When the frequencies of the CPU and the components dynamically change, the states also change. For the dynamic power management, programs, which are OS or device drivers, control the states of the CPU and the peripheral components. To implement the dynamic power management effectively, we need a framework in which the program changes the states, that is, the frequencies of the CPU and the peripheral components.

Solution candidates: The dynamic power management (DPM) architecture proposed by IBM and Montavista is one solution for the above requirement[10]. The DPM architecture consists of policies in the kernel and a policy manager in the user or kernel space. In the DPM, a system is modeled as operating points and operating states. A system can be regarded as being executed at multiple operating point represented by parameters such as voltages, bus frequencies, the states of peripheral devices. An operating state is a program execution state in the system such as an interrupt occurrence, application program execution, waiting for event occurrence and so on. A policy maps each state to the corresponding set of operating points. A policy manager coordinates the different type of policies, depending on application program status, user preferences, peripheral device status, and so on. Fig. 4 show an example of operating states, operating points, and policies. In the figure, two policies, which are applicable depending battery condition and managed by the policy manager, contains three operating points each. The operating points show states of CPU and peripherals in a program execution. For example, the rightmost operating point in the full battery policy shows the program is executed in a state, which is controlled with high CPU and BUS frequency, and LCD on. Thus, the DPM provides a flexible framework for system-wide power management.

In addition to dynamic power management, the following are required.

- (1) Different type of memory, for example SRAM and DRAM, require different power consumption. If the current is not provided to unused memories, less power consumption is achieved. So code and data blocks in applications and the kernel should be allocated to specified memory.
- (2) Clock rate should be minimal at idle time.
- (3) Kernel threads are suspended in power save mode.

B. Memory Footprint

Usually the Linux kernel is stored in a compressed form the ROM area or file systems in disks (if any), booted from the area, and its memory images are expanded in the RAM area of system and executed instructions in it. RAM is one of the cost factors in mobile phone and reducing the size of RAM is always required.

As we have shown in Fig. 1, the memory size of current mobile phones is up to 64 MB.

Solution candidates: An execute-in-place mechanism is a candidate to solve the problem [11]. To avoid large RAM usage, codes in Linux kernel, applications and libraries should be directly executed in its ROM area without copying and decompressing the kernel into the RAM area. Application programs and libraries in Linux also are executed in their ROM area. However, the execute-in-place mechanism increases ROM area while saving RAM area. Moreover access speed in ROM area is slower than the speed in RAM area. Note that these tradeoffs exist in the execute-in-place mechanism.

To reduce the memory size, we need the reduction in system-wide approaches. This means that memory reduction efforts are applied to not only the kernel, but middleware, and applications. The middleware issues, especially window system, are described later.

In addition to the above solutions, the following heuristics techniques can be applied.

- (1) The Linux kernel and libraries should be customized configured.
- (2) Memory alignment used in various kernel configuration could be changed.

uClinux, which is suitable for embedded systems and merged into Linux 2.6, is not appropriate for mobile phones because uClinux does not have MMU features which are needed for increasing mobile application productivity.

C. Shortening Boot-up Time

Boot-up time in a Linux system of a PC includes copying of the compressed kernel file on a disk, and uncompressing the copy, starting up daemons(background processes), starting up middleware(including window system,

Fig. 5. Interrupt processing in traditional UNIX

Fig. 6. Interrupt processing in real-time-enhanced LINUX

network system and so on), starting up application programs. During the boot-up time, Linux in a PC takes several minutes to set up device drivers and check consistency of file systems.

Currently, mobile phones should be booted up within tens of seconds.

Solution candidate: The same solutions for shortening the boot-up times as in the memory footprint are also applicable. Deferred initialization of device drivers is also effective if it is possible.

D. Real-time Performance

A long preemptive inhabitation in ordinal Linux degrades real-time response time. An interrupt latency is allowed to be between several tens of msec in PCs and servers. On the other hands, in mobile phones, an interrupt latency is required to be between several μ secs and several hundreds of μ secs.

Solution candidate: An external interrupt arrives at CPU and the corresponding device drivers and process preemption occurs.

Until the process preemption, the following steps are required[12].

- (1) An interrupt handler is executed in response an external interrupt in the kernel.

- (2) An interrupt execution part in the corresponding device driver is executed in the kernel. The part sends data, which arrives when the interrupt occurs, to the corresponding user process.
- (3) The process is preempted and processes the receiving data.

In Linux, (1) is called a top half and (2) is call a bottom half. The bottom half is executed in a interrupt context and all interrupts are enable in the execution.

To shorten an interrupt response time, we have two delays. The first delay is one from the interrupt arrival to calling an interrupt execution part between (1) and (2). The second one is delay from the interrupt arrival to activating the corresponding process between (1) and (3). In traditional UNIX, an execution of the interrupt execution part is serialized in the first delay. As a result, the interrupt inhibit time also becomes longer because enabling the inhibit interrupt is delayed until an end of the interrupt execution part. In the second delay preemption in the kernel is delayed until the end of the system call execution.

To improve the first delay, Linux has provided the following solution by implementing the interrupt execution part concurrently[13].

- A tasklet, which is introduced in Linux 2.4, is a function of which the different type can run concurrently, but of which the same type cannot run concurrently. A function declared as a tasklet is executed.
- A bottom half, which has been introduced in the early time of Linux, is a function of which the same type cannot run concurrently. A bottom half corresponding to flags set by a top half is executed.

These mechanism can localize software control of interrupt controllers in the interrupt handler, or the top half, and shorten the interrupt inhibit time in the controller. Furthermore, these mechanisms can make execution of a bottom half more flexibly. A higher priority bottom half is executed earlier while an execution of a lower priority bottom half is deferrable.

To make the kernel preemptive in the real-time enhanced Linux, there are two approaches. The Linux kernel itself is modified to be preemptive. The first is to insert preemption points into the kernel. In the traditional Linux, critical regions are guarded from other process access by prohibiting process preemptions during a system call execution. To insert preemption points into the kernel, we need to divide the critical regions and prepare the preempted process restarting mechanism.

A more systematic approach is the second one: utilizing Symmetric Multi-Processor (SMP) kernel, which was invented by Montavista[14]. In the SMP kernel, critical regions are already coded and the kernel parts except the critical regions are fully preemptable for SMP. Montavista modifies spinlocks in the SMP kernel into preemptable locks, applying the SMP technology to preemption improvement in real-time processing.

Fig. 7. Implementing Linux on real-time OS

Fig. 8. 2 CPU architecture and their operating systems

In the real-time enhanced Linux kernel, a scheduler should be improved to shorten an overhead of process preemption. The $O(1)$ scheduler is implemented in Linux 2.6.

Besides modifying the kernel, we have another approach: Integrating Linux with real-time OS. The Linux kernel is implemented on the top of real-time OS shown in Fig. 7. In the approach, real-time processing are done in real-time tasks, not execution units in Linux[15, 16].

NEC adopts the real-time enhanced Linux and the integration of Linux and real-time OS based on “2-CPU solution” for the next generation mobile phones. In the 2-CPU solution, mobile phone services are done in an application CPU (A-CPU) and a communication CPU (C-CPU). The A-CPU is dedicated to mobile application services and its OS is Linux for increasing software productivity. The above real-time enhanced Linux is utilized in A-CPU. The communication CPU is for communication processing and OS of the C-CPU is a real-time OS for the processing.

Fig. 9. Structure of journalling file system

Fig. 10. Structure of LSM and SE-Linux

E. Power Down and Reset Tolerance

Battery power supply may stop suddenly in a very short time or a user may remove the battery from a phone. Moreover the battery power may stop suddenly because the battery is dead. Usually the file system inconsistencies may occur and the file system are checked and fixed in Linux when the system restarts after the power stops suddenly. Even if these cases happen, however, the mobile phones should be normally reactivated. Moreover, this requirement applies to flash file system which are utilized in mobile phone.

Solution candidate: As a solution candidate to satisfy these requirements, we have the journailling flash file system (JFFS) [17], which is an implementation of the journailling file system for the flash file system.

There are two requirements for the flash file system. The first is to provide ‘wear levelling.’ In flash chips, erasing a whole of block is required in changing block data. A typical life time of flash chip is 100,000 erases per block so erase operations must not concentrate on one block and must distribute over the flash chip. This is called “wear levelling.” The second is to increase reliability when power down and reset happens. The JFFS is a log-structure file system and can satisfy the two requirements [18, 19]. The structure of the JFFS is shown in Fig. 9. In the JFFS, writing a data block into a file is appending the block to an end of the file sequentially, even in rewriting data block in the file. The number of erasing each block is the same and thus the JFFS implementing a functionality of “wear levelling.”

The first requirement is solved by distributing data blocks over the flash. The JFFS has no metadata (file management data) so that inconsistency if meta data and data written to the flash does not occur. However, the JFFS has the shortcomings that garbage collection in the journailling file system is needed and that a read speed becomes extremely slow when the size of a file becomes very large. The performance data of the log-structure file system, which is a base of the JFFS, is described in [19].

F. Security

In the PC world, bugs of kernel or application programs and DOS(Denial of Service) attacks destroy the system or give functional and performance damage to the system. Such a situations will arrive in the mobile phone world in

Fig. 11. Thread guard pages

the near future if 3rd vendor programs can be installed to the phone.

Solution candidate: To avoid the attacks and protect mobile phone software from the attacks, security functionalities in the kernel must be enhanced. LSM (Linux Security Module) ² is a candidate of security enhancement. LSM provides only a framework and security check mechanisms are separated from the framework because a variety of security policy can be applied. As shown in Fig. 10, LSM provides a hook in the system call executions in which appropriate security checks which are provided in a separate security module are performed [20]. SE-Linux³ is a candidate of security enhanced module on the framework. SE-Linux gives Mandatory Access Control (MAC) and Role Based Access Control (RBAC) [21, 22]. In MAC, access control lists where which program can access to which resources are defined and all processes are checked in accessing resources based on the access control list. A concept of ‘role’ is introduced and access controls are defined in the role in RBAC. For example, the root rights are divided to multiple roles. Thus even a program which has the root access rights has the limited access rights to system resources.

In addition to the above security enhancement, the followings are expected.

- (1) Thread stack areas are allocated in a process. A malicious program or a buggy program expands the thread stack, destroying other thread stack area. To protect the thread violation, a thread is implemented in the kernel, such as Linux 2.6, and a thread guarded page on the top of the stack is allocated (Fig. 11). In a thread guarded page read/ write operations are not permitted and so the thread stack violation can be detected.
- (2) There are very important data such as subscriber data and address information in a mobile phone. The kernel or other program bug possibly destroys such data area. To avoid the destruction, the protected

²<http://lsm.immunix.org/>

³<http://www.nsa.gov/selinux/>

RAM file system⁴ is considered to be solved.

G. Application Program Development Environment

Ideally, programs developed in a PC are ported into mobile phones only with a recompile. However, it is not easy because header files, library files, and system call APIs are different from those in a PC even if the OS in mobile phones is Linux. Providing program development environments in which application programmers can develop easy-to-port application programs on the PC is required.

Solution candidate: To provide the specific Linux, middleware, libraries, and header files, the guest Linux is implemented independent of the host Linux. UML (User Mode Linux⁵, not Unified Modeling Language) is a candidate to implement the guest Linux [23]. UML is a Linux kernel implementation in a user space and can provide application program environments, independent of the host Linux.

H. Kernel Development Environment

To develop Linux kernel optimized for a target product efficiently, development support environments are required.

Requirements for the kernel development tools are summarized: real-time performance, monitoring process consumption of memory and event logging.

IV. Requirements for Middleware

In this section, the term ‘middleware’ is used for a logical software layer located between the kernel and application software. The middleware can be implemented in the protected memory space like the kernel or a server in the user memory space.

Wide varieties of middleware should be required. The middleware will be supplied from third party vendors or open source software development communities.

- Graphical user interface, including window systems
- Multimedia functions
- Network communication protocol stacks

We need to implement some program management mechanism to guarantee to catch a voice call as described in Sec. II. When a call comes to the phone, the management mechanism interrupts the current application services and notifies the user of incoming call by a ring tone, vibrating the phone, or a graphical icon.

A. X window system and toolkit

X window system and toolkit occupy large amount of RAM area. Requirements for X window system and toolkit are summarized as follows.

⁴<http://pramfs.sourceforge.net>

⁵<http://user-mode-linux.sourceforge.net/>

(1) Small size RAM consumption

(2) Double buffer extension to draw image data quickly

(3) Font sets with small size and quicker loaded time

Solution candidate:

(1) As small size X window system, kdrive⁶ is a candidate [24].

(2) As a toolkit, GTK⁷ is the most popular toolkit. The size of GTK, however, is very large. So customizing functionalities of GTK is required.

Note that standardization for customized toolkit APIs will be required because to recruit third party application software easily.

V. Future Trends in Mobile Phone Software

Various types of software are incorporated in mobile phones in order to enable the provision of future new services. Mobile phones in the future will offer not only basic functions, but may also allow users to add applications of their choice. In other words, the future trend is the “personal computerization” of mobile phones. By incorporating the Java execution environment, mobile phones now allow to download applications after releasing the phones. Another aspect of mobile phone evolution is the incorporation of technologies for contents such as images, sound, and moving pictures that are now broadly used in PCs. From the viewpoint of content providers, this trend has the merits to allow reuse of existing contents efficiently and use of contents development environment in PCs.

However, even if the processing performance of mobile phones continues to increase, the evolution of mobile phones is likely to follow a course that differs from that of PCs. Differences in hardware between mobile phones, which have a smaller screen and very few keys, and PCs, will always remain, no matter how much a mobile phone evolves in terms of hardware. Mobile phone applications must meet real-time and software reliability requirements that are more severe than for personal PCs. As one of these, mobile phones require the processing for the voice communication within a limited time. To implement it, CPU, memory and OS resource must be guaranteed to be allocated in the specific duration. Furthermore, personal information such as phone directories in a mobile phone is handled more securely. In PCs, any application can access a phone directory file, for example. Even if a malicious application was executed in a mobile phone, sending the data to the outside and alteration of the data should not be allowed. Restrictions on access to information from applications in mobile phones are most likely to be maintained in the future.

⁶<http://www.xfree.org/current/Xkdrive.1.html>

⁷<http://www.gtk.org/>

VI. Conclusion

We describe requirements for mobile phone Linux and solution candidate technologies to satisfy the requirements based on the development experience.

Note that the technologies described in the paper are not utilized in a native way. We need engineering efforts to customize the technologies and tune up mobile phone software based on the technologies to realize the actual products.

Consumer Electronics Linux Forum ⁸ is established to gather requirements of consumer electronics for Linux and evolve implementation for the requirements. We will contribute our effort in mobile phone Linux to the Forum.

References

- [1] K. Usui, H. Tomimori, J. Takagi, T. Tanaka, and Y. Nakamoto, "Design and implementation of Java application environment and software platform for mobile phones," NEC RESEARCH & DEVELOPMENT, Vol. 42, No. 4., pp.379–383, Nov. 2001.
- [2] Compaq, Intel, Microsoft, Phoenix Technologies, and Toshiba, "Advanced configuration and power interface specification," 2003, <http://www.acpiinfo/>.
- [3] L. Benini, A. Bogliolo, and G. De Micheli, "A survey of design techniques for system-level dynamic power management," IEEE Trans. on Very Large Scale Integration System, Vol.8, No.3, pp.299–316, Jun. 2000.
- [4] M. Weiser, B. Welch, A. Demer, and S. Shenker, "Scheduling for reduced CPU energy", Proc. of First Symposium on Operating Systems Design and Implementation, pp.13–23, IEEE, Nov. 1995.
- [5] T. Ishihara and H. Yasuura, "Optimization of supply voltage assignment for power reduction on processor-based system," Proc. of Synthesis and System Integration of Mixed Technologies, pp.51–58, Dec. 1997.
- [6] P. Pillai and K. Shin, "Real-time dynamic voltage scaling for low-power embedded operating systems," Proc. of 18th ACM Symposium on Operating System Principles, pp.89–102, ACM, Oct. 2001.
- [7] T. Truman, T. Perring, R. Doering, and R. Broderson, "The InfoPad multimedia terminal: a portable device for wireless information access," IEEE Trans. on Computers, Vol.47, No.18, pp.1073–1087, Oct. 1998.
- [8] J. Wang, B. Ravindran, and T. Martin, "A power-aware, best effort real-time task scheduling algorithm," Proc. of IEEE Workshop on Software Technologies for Future Embedded Systems, pp.21–28, May 2003.
- [9] A. Hasegawa, I. Kawasaki, K. Yamada, S. Yoshikawa, S. Kawasaki, and P. Biswas, "SH3: high code density, low power," IEEE Micro, pp.11–19, Dec. 1995.
- [10] IBM and Montavista Software, "Dynamic power management for embedded systems," Nov. 2002, http://www.research.ibm.com/ar1/projects/papers/DPM_V1.1.pdf
- [11] Montavista, "Saving RAM and speeding program launch with execute in place (XIP)," <http://www.mvista.com/>.
- [12] A. Rubini and J. Corbet, "Linux device drivers (2nd ed.)," O'Reilly, 2003.
- [13] D. Bovet and M. Cesati, "Understanding the Linux kernel (2nd ed.)," O'Reilly, 2003.
- [14] Montavista, "MontaVista Linux - real-time performance," 2002 <http://www.mvista.com/dswp/realtime.pdf>.
- [15] E. Bianchi, L. Dozio, and P. Mantegazza, "A hard real-time support for Linux, DIAPM RTAI," <http://www.aero.polimit.it/rtai/>.
- [16] H. Takada, S. Iiyama, T. Kindaichi, and S. Hachiya, "Linux on ITRON: a hybrid operating system architecture for embedded systems," Proc. of 2002 Symposium on Application and the Internet, pp.4–7, 2002.
- [17] D. Woodhouse, "JFFS : the jouralling flash file system," <http://sources.redhat.com/jffs2/jffs2-html/jffs2-html.html>.
- [18] U. Vahalia, "UNIX Internals," Prentice Hall, 1996.
- [19] M. Rosenblum and J. Ousterhout, "The design and implementation of a log-structured file system," ACM Trans. on Computer Systems, Vol.10, No.1, pp.26–52, 1992.
- [20] C. Wright, C. Cowan, J. Morris, S. Smalley, and G. Kroah-Hartman, "Linux security modules: general security support for the Linux kernel," Proc. of 11th USENIX Security Symposium, Aug. 2002.
- [21] P. Loscocco and S. Smalley, "Integrating flexible support for security policies into the Linux operating system", Proc. of the FREENIX Track: 2001 USENIX Annual Technical Conference, Jun. 2001.
- [22] S. Smalley, C. Vance, and W. Salamon, "Implementing SELinux as a Linux Security Module," National Security Agency, May 2002.
- [23] J. Dike, "A user-mode port of the Linux kernel," Proc. of the 4 Annual Linux Showcase & Conference, Usenix, Oct. 2000.
- [24] J. Gettys and K. Packard, "The X Resize and Rotate Extension - RandR," Proc. of 2001 USENIX Annual Technical Conference, Jun. 2001.

⁸<http://www.celinuxforum.org>