지능형 Gadget 시스템을 위한 개발환경 구현

정갑중* · 배창석**

Implementation of Development Environment for Intelligent Gadget System

Gab Joong Jeong* · Changseok Bae**

본 연구는 정보통신부 및 정보통신연구진흥원의 IT신성장동력핵심기술개발사업의 일환으로 수행하였음.
[2006-S032-02, 퍼스널 Life Log 기반 지능형 서비스 기술 개발]

요 약

본 논문은 지능형 Gadget 시스템의 개발환경 구현에 관한 논문이다. 지능형 Gadget 시스템에서 사용된 임베디드 시스템 응용 프로그램과 리눅스 커널의 구조 및 동작에 대해 논하고 지능형 Gadget 시스템에 필요한 기능 및 구성 요소에 대해 조사 및 분석을 통한 리눅스 커널과의 동작 및 기능 검증 구현을 보인다. 새로운 지능형 임베디드 시스템의 하나인 Gadget 시스템에 적용 가능하도록 요구되는 기능과 동작을 구현하고 새로운 소형 운영체제를 위한 개발에 적용가능하다. 그러한 소형 운영체제는 지능형 개인정보서비스를 위한 임베디드 Gadget 시스템으로써 지능형 정보화 기능을 지원하고 새로운 소형 운영체제를 탑재한 시스템의 개발에 적용 가능하다. 본 논문에서는 그러한 지능형 임베디드 Gadget 시스템과 응용 개발을 위한 개발환경 구현에 대하여 기술한다.

ABSTRACT

This paper describes the environment configuration for the development of an embedded gadget system and the architecture and operation of Linux kernel for embedded system applications, which is used for a gadget. It shows and analyzes the operations of Linux kernel to investigate the functions and components for new intelligent embedded gadget systems. The requested functions and operations adaptable for the new intelligent embedded system will be applicable to develop a new small size operating system that supports intelligent operations for the embedded gadget system used for intelligent personal information services. We configure the environment of development for an embedded gadget system and its application.

키워드
Intelligent Gadget System, Light-weighted Operating System, Personal Information Service

Ⅰ. Introduction

In recent years, computers and computing devices are miniaturizing by the contribution of smart processors, sensors, and WiFi devices. Those recent device technologies can realize a networking environment for the era of post-PC which is represented by small device and mobile service. In such computing technology environment, the necessity of an

* 경주대학교
** 한국전자통신연구원

접수일자 : 2007. 7. 23

intelligent gadget device for a personalized information service grows to support ubiquitous computing environment in daily objects and daily human life. Therefore, most users want to have personalized intelligent services from their tangible daily objects. The intelligent environment will be designed for more personalized information service implemented in every daily gadget object [1]-[6].

For the intelligent environment in daily life, the daily gadget object that we use in everywhere should have more light-weight system software. With the minimum system specification, the more intelligent and personalized information service needs a lightly configured operating system. So, we need to study the lightly configured operating system for the smart and personalized information service using daily gadget objects.

In this research, we survey the architecture and operation of the real time Linux kernel, for embedded system applications, which is used for an embedded gadget system. We show and analyze the operations of the Linux kernel to investigate the functions and components for new light-weighted intelligent operating systems [7]-[13]. The light-weighted OS kernel and intelligent gadget system developed in this research is implemented and tested for a new prototype gadget device which supports intelligent personal information services. In this paper, we illustrate the environment configuration for the development of an embedded gadget system and its applications. The developed gadget system uses TI DaVinci processor in which ARM926EJ-S and TMS320 DSP cores are integrated [14]-[16].

II. OS KERNEL FOR EMBEDDED SYSTEM

Technology enhancement using computer was concentrated on host computing by Unix and host programs in 1970's. In 1990's, these technologies had been developed for a centralized server for client-server technology to provide solutions for requests. Recently, the World Wide Web technology has greatly enhanced these client-server technologies. But these technologies are developed by the base of the host computer. These technologies based on host computing have being developed as network computing. In the environment of network computing, there is a need for post-PC intelligent embedded system which provides information service for personal request. Therefore, the development of a smart light-weighted operating system is needed for the intelligent embedded gadget system which provides information service for each individual person.

The kernel of operating system for an embedded system is a main part for developing the intelligent gadget system. It is classified by micro kernel and monolithic kernel. The micro kernel has only minimal functionality like QNX, and MACH operating systems. It minimizes the functions of kernel, and supports simple scheduler and inter-process communication. The monolithic kernel has most services for the kernel functions of operating system. It controls every running program, and manages files and data. Linux and Unix operating systems correspond to the latter case. Linux operating system includes the most functions for operating system. Fig. 1 shows the structure of the Linux system.

Linux system consists of kernel and process modes. The CPU operation mode for running the kernel is called kernel mode and the operation mode for running processes is called user mode. Processes for data exchange operation between process and I/O device, and for operations followed by functions in kernel have to be executed in kernel mode. So, the operating system has to be managed by switching the modes between kernel mode and user mode.

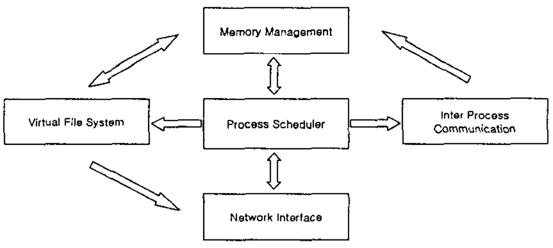


그림 1. 리눅스 커널 구조.
Fig. 1 Structure of Linux kernel.

Linux has the process scheduler module which is one of the important functions in the kernel. It schedules each process using a method of round-robin, etc. It supports inter-process data exchange with the inter-process communication module. It provides huge data storage and recovery functions by using the virtual file system module. It supports available memory allocation and data loading functions for each process by using the memory management module. It supports also network functions like Ethernet with the network interface module.

Ⅲ. ANALYSIS OF LINUX KERNEL

Linux is representative monolithic operating system kernel. The kernel source includes most services that an operating system needs. Fig. 2 shows the directory structure of Linux kernel.

The 'Documentation' directory includes various text files for kernel documents. The 'kernel' directory has main functions for the kernel. It includes signal processing routine, time management routine, and system calls which treat processes like fork() and exit() as well as hardware independent scheduler. The 'ipc' directory provides semaphore for inter-process communication, message queue, and shared memory functions. The 'lib' directory has library functions for kernel, which is implemented in C language. Libraries for application programs are implemented in separate files for each application. Linux provides the special characteristic function of module loading. The object files for kernel modules are managed in the 'modules' directory. The 'scripts' directory includes various script files for building the kernel.

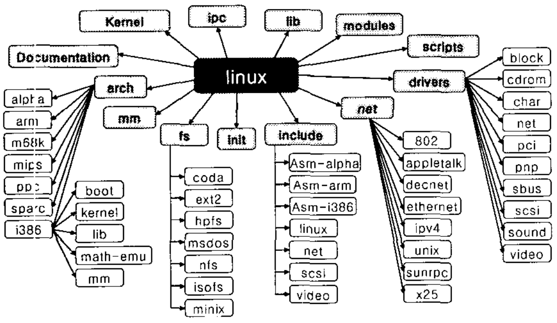


그림 2. 커널디렉토리 구조
Fig. 2 Kernel directory structure.

The 'arch' directory has sub-directories for each CPU. Each sub-directory has source files for handling each different target CPU. When developing a kernel for a specific CPU, it needs to modify the source files in this directory. The 'arch/processor-name/kernel' directory includes trap processing, interrupt processing, context switching, device configuration, and initialization routines. The 'arch/processor-name/mm' directory includes processor dependent routines for the handling routines of memory management. The 'arch/processor-name/tools' directory generates constants for the kernel. The 'arch/processor-name/mach-board-name' directory includes source files for a specific SoC and a target board. It supports memory map configuration for the target board, interrupt controller management for the target SoC, interrupt handling routine, USB device driver, LED controller, PCI interface driver, etc. The 'arch/processor-name/boot' directory has bootstrapping source files before running kernel. It is used for the location of a kernel binary image file after the recompilation of the kernel source.

The 'fs' directory has the source files for a virtual file system and file systems supported by Linux. System call handling routines for the virtual file system are implemented in this directory. It has also special file system directories for ext2, ext3, ramfs, jffs2, nfs, msdos, etc. The directory, 'mm' has source files for the virtual memory management which is independent of hardware, like paging and swapping. The 'init' is a directory which has hardware independent kernel initialization routine. It has 'main.c' file where the start_kernel() function is implemented.

There is the directory, 'include' which has header files for the kernel. The header information provides hardware independent files for the general use of Linux, and specific header files for the specific SoC and the specific target board. The hardware independent header files are stored in the directories, 'include/linux' and 'include/net', and the hardware dependent header files are stored in the directory, 'include/asm-target-processor-name'. It is used for rebuilding the kernel as the name of 'include/asm'. It is used by the symbolic link name of 'include/asm' when the kernel is built. Also, this directory provides header files which are used for the specific SoC and the target development board.

Lastly, there is the 'drivers' directory which has source files for the most input and output device driver programs. Linux device drivers have been implemented in separate sub-directories for character, block, network interface, sound, and video drivers. So, the development of device drivers is performed in the sub-directories of the 'drivers' directory. The 'drivers/char' directory has many character device driver source files for keyboard, mouse, etc. The 'drivers/block' directory has block device drivers. The block devices are large size data communication devices like a hard disk and a CD-ROM. The 'drivers/net' directory has device driver programs for networking devices with other host computers like Ethernet, wireless LAN, etc. The 'drivers/pci' directory provides the PCI bus control driver program. The 'drivers/usb' directory has driver programs for the USB bus controller, and there are 'driver/sound' directory for audio sound interface driver program, 'drivers/video/ directory for frame buffer driver program, and 'drivers/serial' directory for the serial UART driver program.

IV. ENVIRONMENT CONFIGURATION

We configure an efficient environment for the development of an embedded gadget system. We illustrate the configuration of the development environment in Fig. 3. First of all, a main host computer is connected to the general purpose test board which has embedded the target CPU. OS platform for the development host is previously installed in the host computer. As an OS platform, we have selected Linux that has the greatest potential for the development of various embedded gadget systems.

The main host computer manipulates the general purpose test board as a root user via serial terminal connected with COM port by using the cross-compiling tool chain. The general purpose test bed can operates with the recompiled kernel as a stand alone embedded computer. It communicates with the main host computer through the wired network file system. The development of various functions for kernel and application software which can be useful in the gadget system can be performed in the main host computer and it can be tested on the general purpose test bed. Source program coding, cross-compiling and downloading of binary file are performed as a non-root user on the host computer, and the operation test of the developed binary files is performed as root user on the general purpose test bed.

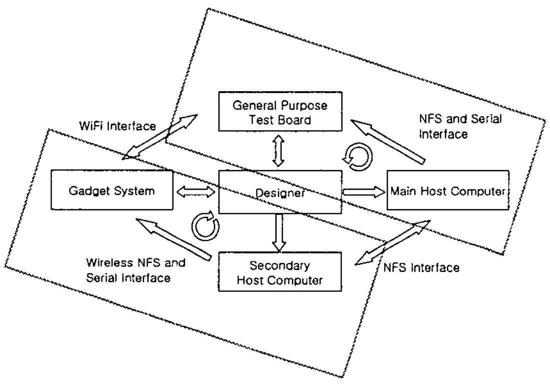


그림 3. 개발환경 구성
Fig. 3 Configuration of the development environment.

After the development of an application in the general purpose test bed, we need to move the source files to the secondary host computer through the wired NFS. The secondary host computer is connected to the target gadget system which is the goal of the development project. It also has Linux platform for the host OS and manipulates the target gadget system as a root user via serial terminal. Source program modification, cross-compilation, and the downloading of binary file are performed as a non-root user, and the operation test of the developed binary files is performed as root user on the target gadget system. Using the separated dual host computers reduces many mistakes for the development of an application and makes easy to recover the environment of the target gadget system. The target gadget system can be operated by the standalone mode and it can communicate with the secondary host computer through the wireless network file system. We use the wireless USB WiFi interface for the gadget system to communicate with the host computer, since the target gadget system we developed has no wired network interface.

To test an application for video and audio, we use the wireless LAN 802.11.g/54Mbps between the target gadget system and the general purpose test bed. On the target gadget system, we encode video and audio data in real time from an NTSC/PAL micro video camera and send the encoded data via the wireless LAN to the general purpose test bed. On the general purpose test bed, we decode and play in real time the video and audio data which are transferred from the target gadget system. Also, we can transfer the video and audio data between the gadget systems. The exchange of data communication can operate between client-server computers through the public network for an intelligent service of personal information.

Fig. 4 shows the general purpose embedded test bed. Fig. 5 shows the target gadget system boards we developed. Fig. 6 shows the operating target gadget system that four gadget boards are stacked. There are main processor module, power module, video/audio module, and interface module. An USB WiFi stick has been installed on the most upper interface gadget module. It provides 802.11g/54Mbps network interface.

Currently, we are developing GPS applications, video/audio applications, LCD screen applications, and micro-kernel applications for a new smaller gadget. The smaller gadget application system can be embedded in daily objects which provide new information oriented personal services.

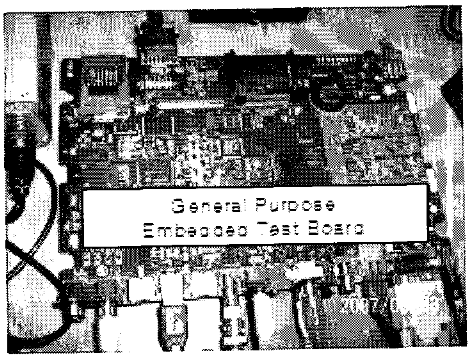


그림 4. 범용 임베디드 테스트보드
Fig. 4 General purpose embedded test board.

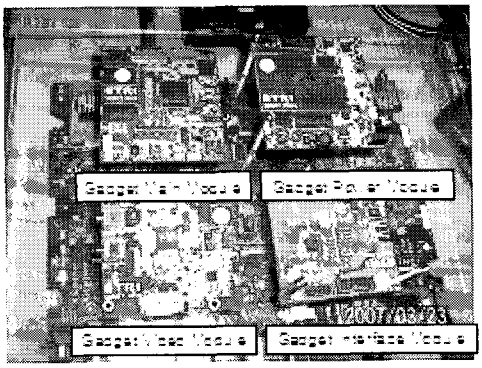


그림 5. 임베디드 가젯 모듈
Fig. 5 Embedded gadget modules.

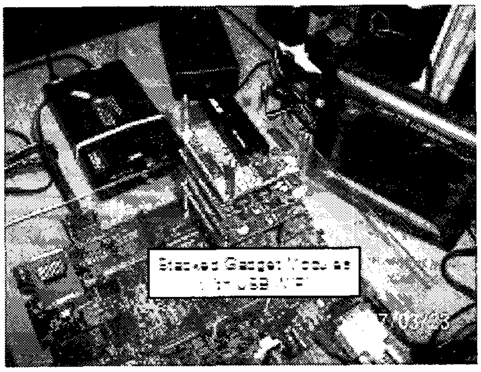


그림 6. USB WiFi탑재후 조립된 가젯 모듈
Fig. 6 Stacked gadget modules with USB WiFi device.

V. CONCLUSIONS

In this research, we have analyzed the architecture and the operation method of Linux which is one of the generalized operating system for an intelligent embedded system. We have illustrated an efficient environment configuration for the development of an embedded gadget system. The environment configuration has been used for the development of an intelligent gadget system. The gadget system under development .has used an ARM9EJ-S processor core and Linux kernel 2.6.10 as a real-time operating system.

In the future, we will enhance and minimize the developed gadget system for implementing in daily objects, and we will support applications to provide new information services based on personal needs.

REFERENCES

[1] Kjell, B., "The rise of embedded processing and the opportunity for open standards," Technology and Society Magazine, IEEE, vol. 23, no. 2, pp. 4 - 5, Summer 2004.

[2] Kameas, A., Bellis, S., Mavrommati, I., Delaney, K., Colley, M., and Pounds-Cornish, A., "An architecture that treats everyday objects as communicating tangible components," Proc. 1st Int. Conf. on Pervasive Computing and Communications 2003 (PerCom 2003), vol. 2, pp. 115 - 122, March 2003.

[3] Scholten, H., Jansen, P., and Hop, L., "Communicating personal gadgets [PAN real-time streaming media protocol]," IEEE 1st CCNC 2004, pp. 630 - 632, Jan. 2004.

[4] Kishino, Y., Terada, T., and Nishio, S., "Ubiquitous Gadgets for Constructing Flexible Ubiquitous Services," Proc. 7th Int. Conf. Mobile Data Management, pp. 100-100, May. 2006.

[5] Lewis, T., "Information appliances: gadget Netopia," IEEE, Computer, vol. 31, no. 1, pp. 59 - 68, Jan. 1998.

[6] Acre, I, "The rise of the gadgets," Security & Privacy Magazine, IEEE, vol.1, no. 5, pp.78-81, Sep. 2003

[7] Craig Hollabaugh, Embedded Linux: Hardware, Software, and Interfacing, Addison-Wesley, Mar. 2002.

[8] Karim Yaghmour, Building Embedded Linux Systems, Sebastol, CA: O'Reilly & Associates, Inc. 2003.

[9] Love and Robert, Linux Kernel Development, Indianapolis: Sams Publishing, 2004.

[10] Jensen, D., "Adventures in embedded development," IEEE Software, vol. 11, no. 6, pp. 116 - 118, Nov. 1994.

[11] Nakajima, T., Sugaya, M., and Oikawa, S., "Operating systems for building robust embedded systems," Proc. 10th IEEE International Workshop on Object-Oriented Real-Time Dependable Systems 2005(WORDS 2005), pp. 211 - 218, Feb. 2005

[12] Bolosky, W.J., Draves, R.P., Fitzgerald, R.P., Fraser, C.W., Jones, M.B., Knoblock, T.B., and Rashid, R., "Operating system directions for the next Millennium," Proc. 6th Workshop on Hot Topics in Operating Systems, pp. 106 - 110, May. 1997.

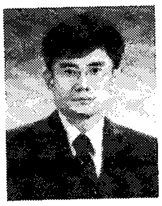
[13] Steiger, C., Walder, H., and Platzner, M., "Operating systems for reconfigurable embedded platforms: online scheduling of real-time tasks," IEEE Transactions on Computers, vol. 53, no. 11, pp. 1393-1407, Nov. 2004

[14] Steve Furber, ARM system-on-chip architecture second edition, Addison-Wesley, 2000.

[15] TI, TMS320DM6446, SPRS283D-DEC. 2005-REV. SEPT. 2006, Texas Instruments, Available: http://www.ti.com

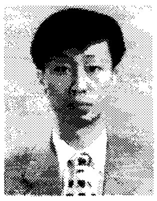
[16] Zhihui Xiong, Maojun Zhang, Sikun Li, Shaohua Liu, and Yafei Chao, "Virtual embedded operating system for hardware/software co-design," Proc. 6th Int. Conf. On ASIC 2005(ASICON 2005), vol. 2, pp. 858-861, Oct. 2005.

저자소개



정 갑 중(Gab Joong Jeong)

1987년 경북대학교 전자공학과 공학사
1989년 경북대학교 대학원 전자공학과 공학석사
1999년 연세대학교 대학원 전자공학과 공학박사
1989년 1월 ~ 1999년 3월 LG반도체 책임연구원
1999년 4월 ~ 2001년 2월 ETRI 선임연구원
2001년 3월 ~ 현재 경주대학교 조교수
※관심분야 : 임베디드 시스템, 지능형 가젯, 컴퓨터 그래픽스, 컴퓨터 게임, 패킷 스위칭, VLSI, SoC



배 창 석(Changseok Bae)

1987년 경북대학교 전자공학과 (공학사)
1989년 경북대학교 대학원 전자공학과 (공학석사)
2003년 연세대학교 대학원 전기전자공학과 (공학박사)
1989년 ~ 현재 ETRI 컴퓨터소프트웨어연구소 책임연구원
※관심분야 : 디지털 영상신호 처리, 멀티미디어 코덱, 지능형 가젯, 라이프로그 시스템