퍼지 및 지능시스템학회 논문지 2004, Vol. 14, No. 4, pp. 424-430

# 유비쿼터스 컴퓨팅을 위한 임베디드 파일시스템

# Embedded File System for Ubiquitous Computing

이병권* · 주영관* · 김석일* · 전중남*

Byung-Kwon Lee, Sukil Kim and Joong-Nam Jeon

*충북대학교 전자계산학과

## 요 약

본 논문에서는 유비쿼터스 컴퓨팅의 시스템 구현 방법으로 임베디드 시스템을 활용시 다양한 플랫폼에 알맞은 파일시스템을 구축하는 방법에 대하여 기술한다. DOC(Disk-On-Chip) 파일시스템과 MTD(Memory Technology Devices)를 기반으로 플래시 메모리를 사용하는 파일시스템에 대한 정형화된 계층 구조를 구성하였다. DOC 파일시스템의 경우, 루트 파일시스템과 유저 파일시스템은 모두 M-Systems가 제공하는 TrueFFS로 구성한다. MTD 파일시스템의 경우, 루트 파일시스템은 속도가 빠른 램 디스크로 구성하고, 유저 파일시스템은 큰 용량을 지원할 수 있는 JFFS2로 구성한다. 또한, 두 가지 경우 모두 GUI(Graphic User Interface) 파일시스템의 구성을 위하여 Qt/E를 포팅하는 과정도 함께 제시한다.

## Abstract

This paper explains the construction of the filesystems which could be utilized in embedded systems as an implementation of ubiquitous computing. It includes the formal architecture of filesystem hierarchy for the DOC (Disk-On-Chip) filesystem and the flash filesystem based on the MTD (Memory Technology Devices). For DOC, the root filesystem and the user filesystem are constructed by the TrueFFS supported by the M-Systems. For MTD filesystem, the root filesystem is implemented in the fast RAM disk, and the user filesystem is implemented in the JFFS2 that supports large capacity. In order to support the GUI filesystem, the porting process of Qt/E is also included in this paper.

**Key words** : 유비쿼터스 컴퓨팅, 임베디드시스템, TrueFFS, DOC파일시스템, 플래시파일시스템

## 1. 서 론

컴퓨팅 환경이 더 이상 컴퓨터 중심이 아닌 사용자의 편리성을 극대화시키는 사용자 중심의 패러다임으로 변화되고 있다. 미래의 사회는 디지털 기술의 급속한 발전으로 사람, 사물, 기계 등이 실시간으로 정보를 주고받는 디지털 주도의 사회로 진보할 것이다[1]. 또한, 사람과 컴퓨팅 기기 및 환경이 서로 상호 작용하여 컴퓨터가 사람의 필요 사항을 알아서 처리하는 인간 중심의 유비쿼터스 컴퓨팅 환경으로 변화될 것이다[2][3]. anytime, anywhere, anynetwork, anydevice, anyservice를 지향하는 유비쿼터스 사회에 따른 장치들의 네트워크화가 가속화되어, 지능화된 임베디드형 장비들이 출시되고 있다. 임베디드 컴퓨팅 기술은 첨단 네트워크 기술을 접목한 새로운 컴퓨팅 기술이 유비쿼터스 컴퓨팅 환경을 실현하기 위한 핵심 기술들 중 하나이다[4].

본 연구에서는 임베디드 시스템의 파일시스템을 유비쿼터스 컴퓨팅 환경에 적합한 구조로 구축하는 방법들을 제시하고 비교한다. 임베디드 리눅스 환경에서 파일시스템 플랫폼으로 x86기반의 DOC(Disk-On-Chip) 파일시스템과 플래시 메모리 기반의 JFFS2 파일시스템을 구축 및 비교 대상으로 선정하였다. 이를 위해 액세스속도가 빠른 램 디스크를 루트 파일시스템으로 구성하였고, 큰 용량을 제공할 수 있는 JFFS2로 유저 파일시스템을 구성하였다. 시스템과 사용자 간의 원활한 의사소통을 위해 GUI(Graphic User Interface) 파일시스템으로써 임베디드 GUI 툴셋인 Qt/E를 포팅하는 과정도 포함하였다.

접수일자 : 2004년 5월 25일
완료일자 : 2004년 6월 23일

## 2. 파일시스템

임베디드 시스템의 파일시스템은 환경에 맞게 여러 가지 파일시스템을 선택적으로 사용할 수 있다. 파일시스템의 종류는 DOC TrueFFS, Ramdisk, JFFS, JFFS2, Cramfs, Ramfs 등이 있고, 각각 장단점이 있으므로 개발자는 사양에 알맞은 파일시스템을 적절히 선택하여야 한다.

임베디드 시스템은 커널과 루트 파일시스템, 그리고 유저 파일시스템으로 나누어지고, 그 외에도 GUI 파일시스템이 추가되어 하나의 모듈 형태로 존재한다. 루트 파일시스템은 리눅스 커널이 동작하기 위한 공간과 library, util 등을 포함하도록 설계되어야 하고, 유저 파일시스템과 GUI 파일시스템은 시스템 사용자에게 추가적인 어플리케이션을 제공할 수 있도록 설계되어야 한다.

본 연구에서 단일 디스크 형태의 플래시 디스크에 TrueFFS 사용하는 DOC 파일시스템과 일반 플래시 메모리로 사용되는 램 디스크 및 플래시 파일시스템에 대하여 구성 방법을 비교 설명한다. 마지막으로 사용자와의 시스템과의 연결을 위한 환경의 구성으로 GUI 인터페이스 역할을 하는 Qt/E를 포팅하여 임베디드 시스템 컴퓨팅 환경에 적합한 플

랫폼을 구성한다.

### 2.1 DOC 시스템

DOC은 다양한 용량이 지원되는 고성능, 단일 칩 플래시 디스크로서, 빠르고 크기가 작아 적은 공간을 차지하는 저가형 플래시 디스크 솔루션이다. 또한, DOC는 특별한 외부장치나 점퍼를 세팅하지 않고 소켓에 장착하고 전원을 넣는 것만으로 동작하며 EDC/ECC (Error Detection Code/Error Correction Code) 구조를 포함해 에러 탐색 및 수정할 수 있고, 부트블록(IPL, Initial Program Loader)을 가지고 있어 스스로 시스템 부팅이 가능하다. 그림 1는 M-SYS사의 DOC 블록이다. DOC는 TrueFFS 드라이버를 포함해 wearing leveling을 통해 플래시 배열의 동일한 위치에 연속적으로 데이터를 쓰는 것을 방지하여 일반 플래시 콤퍼넌트에서 있을 수 있는 부분적인 열화(烈火)를 방지할 수 있는 단일 칩 플래시 디스크 솔루션이다[5].

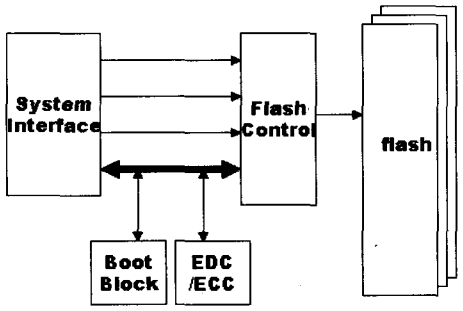


그림 1. DOC 블록도
Fig 1. DOC Block Diagram

그림 2(a)는 시스템 계층에 위치한 TrueFFS로 OS 파일시스템과 플래시 디스크 장치인 DOC 사이에 존재하면서 파일 관리를 돕는다. 또한, 상위계층에 GUI 파일시스템은 시스템과 사용자간의 인터페이스 역할을 한다.

### 2.2 플래시 메모리 시스템

플래시 메모리는 커널, 루트 파일, 사용자 파일 등이 전원이 없는 상태에서 보관하기 위한 저장장소로 사용된다. 일반적으로 루트 파일은 램 디스크 형태로 압축하여 보관하지만, 램인 경우 휘발성이기 때문에 이를 보완하기 위한 방법으로 JFFS란 플래시 파일시스템을 사용한다. 플래시 메모리는 포맷하기 위한 MTD 디바이스 시스템이 필요하다(그림 2(b)). MTD는 커널 수준에서 메모리 영역을 할당한다. 커널 옵션 설정시 "Memory Technology Device(MTD) support" 활성화하면 서브 항목들이 활성화 되어 옵션을 선택할 수 있다.

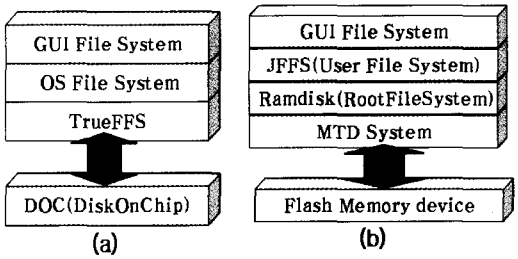


그림 2. DOC(a)와 MTD(b)파일시스템
Fig 2. DOC(a) and MTD(b) of File System

하지만 파티션 분할시 MTD에 DOC가 붙어 있을 경우 파티션 분할이 가능하지 않다. DOC는 일반 디스크처럼 파티션을 분할해야 한다. 자세한 사항은 커널 옵션 설정 설명서를 참조하기 바란다[9].

표 1. MTD/dev 엔트리 메이저 넘버
Table 1. MTD/dev entries, Major number

| | | | |
|---|---|---|---|
| mtdN | char device | char | 90 |
| mtdrN | char device | char | 90 |
| mtdblckN | block device, JFFS, and JFFS2 | block | 30 |
| nftlLN | NTFL | block | 93 |
| ftlLN | FTL | block | 44 |

그림 3은 ROM, RAM, flash, DOC를 포함할 수 있는 MTD(Memory Technology Devices) 서브시스템 구조이다. 또한, MTD 서브시스템은 erase, read, write, and sync와 같은 동작을 수행하기 위하여 콜백(callback) 함수들을 호출한다. 이러한 MTD을 조작하는 방법으로 커널에 이미 MTD를 제어하기 위한 MTD 코드가 포함되어 있다. 또한, 리눅스에 포함되어 있는 MTD는 서로 다른 기능에 대하여 공통된 기능을 제공하기 위하여 MTD 서브시스템을 제공한다. 사용자 모듈(user modules)은 소프트웨어 모듈로서 유저 영역과 인터페이스를 제공함으로써 MTD 칩 드라이버를 액세스 할 수 있도록 만든다.

표 2. MTD/dev 엔트리와 마이너 넘버
Table 2. MTD/dev entries, minor number

| | | |
|---|---|---|
| mtdN | 0 to 32 per increments of 2 | N=minor /2 |
| mtdrN | 1 to 33 per increment of 2 | N=(minor-1)/2 |
| mtdblockN | 0 to 16 per increment of 1 | N=minor |

MTD를 사용하는 기본적인 과정은 먼저 커널 환경을 설정하고, 다음에 MTD에 장치에 필요한 유틸리티를 설치한다. 마지막으로 /dev폴더에 장치에 알맞은 엔트리(Major/Minor)를 만들어 주어야 한다. 표 1과 표 2를 참조하여 간단한 스크립트를 작성하면 장치 디바이스의 메이저 넘버(Major Number)와 마이너 넘버(Minor Number)를 쉽게 결정할 수 있다.

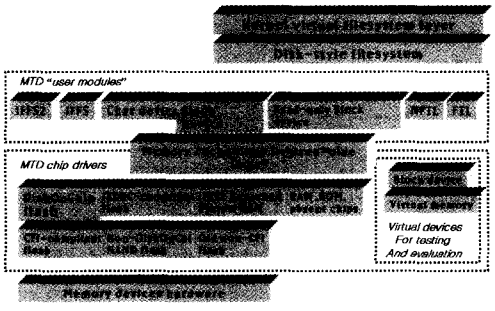


그림 3. MTD 서브시스템 블록도
Fig 3. MTD subsystem block diagram

### 2.3 Qt/E GUI 파일시스템

Qt는 데스크톱 개발 시스템 분야에서 시스템 구축에 급속한 성장을 보인 GUI 툴킷 중 하나이다. 현재 리눅스 데스크톱 환경인 KDE(K Desktop Environment)의 개발의 근간이 되기도 했다. Qt는 GNU C++ 기반의 GUI Library로서 MS-Windows, Unix, Linux, Mac, Zaurus, iPaq, Cassiopeia, Generic PDA 등을 지원하며 서로 다른 플랫폼에서 소스코드의 호완을 보장한다[10][13]. Qt의 그래픽 X환경은 Qt/x11과 Qt/Embedded의 형태의 서로 구분되는 계층구조를 갖는다(그림4). 현재 일반적으로 사용하는 리눅스는 데스크톱 환경은 XFree86를 사용하고 있다. XFree86은 x86용 UNIX 계열의 OS용으로 개발된 그래픽 환경 전용 소프트웨어이다. 여러 가지 비디오 카드 칩에 대응하고 기능면도 충실하지만 일부는 비디오 칩에는 대응하지 않고 칩에 따라 리눅스용 드라이버의 호완성에 차이가 있다. 또한, 인스톨이나 트러블 슈팅에 어느 정도의 전문 지식이 필요하다. 리눅스의 데스크톱으로 XFree86를 사용하여 임베디드 시스템의 GUI를 구현하기란 트러블 슈팅의 문제는 물론, XFree86의 가용 용량이 너무 커 자원을 많이 차지하는 문제를 갖는다.

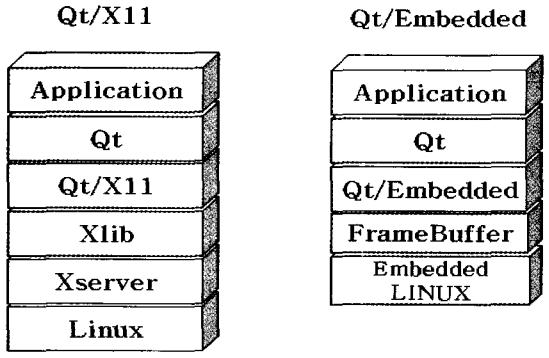


그림 4. Qt 플랫폼 계층구조
Fig 4. hierarchy of Qt platform

현재 XFree86는 약 20Mbyte 정도로 매우 크다. 이런 단점을 보안하기 위해 Qt/Embedded에서는 프레임버퍼(FrameBuffer)계층을 제공하여 XServer, X11을 제거해 성능을 높이고, GUI에 필요한 라이브러리의 저장 공간을 절약하는 효과를 제공한다. 프레임버퍼를 이용하면 X-윈도우를 설치할지 않더라도 커널 수준에서 직접 그래픽 카드의 비디오 램에 접근하는 방식을 사용할 수 있다. 프레임버퍼는 Gray Scale, 8bit Color, 24bit Color 해상도를 제공하고 호스트 환경에서 타겟 환경처럼 시뮬레이션해 볼 수 있는 VFB(Virtual Frame Buffer)를 제공한다. 임베디드 시스템에서 Qt/E 라이브러리를 이용해 개발된 프로그램 동작을 위한 GUI를 지원할 수 있는 VGA환경인 프레임버퍼에 관한 환경을 설정해야 한다. LILO 혹은 GRUB와 같은 부트로더에 옵션을 설정하여 인자로 넘겨주는 것이 일반적인 방법이다.

각각의 옵션 이름은 보통 VGA인데, 그래픽 카드 제조회사에 따라 옵션 이름이 바뀔 수 있다. 기본적인 VESA2.0 규격을 준수하는 그래픽 카드라면, VGA 옵션으로 표 3과 같은 해상도가 주어진다. 적당한 컬러는 16bit, 해상도는 800X600 혹은 1024x768을 선택하면 충분하다.

표 3. 해상도 프레임버퍼 설정
Table 3. setting of value for frame buffer resolution

| Color | 640x480 | 800x600 | 1024x768 | 1280x1024 | 1600x1200 |
|---|---|---|---|---|---|
| | 0301 | 0303 | 0305 | 0307 | 031C |
| | 0310 | 0313 | 0316 | 0319 | 031D |
| | 0311 | 0314 | 0317 | 031A | 031E |
| | 0312 | 0315 | 0318 | 031B | 031F |

## 3. 리눅스용 DOC 설치

DOC가 장착된 임베디드 플랫폼으로 SBC는 AXIOM-TEK(주)에서 개발한 산업용 전용 보드로 기본 베이스는 x86이다. 산업 전용 보드이기 때문에 보드에 기본적인 장치만 있을 뿐, 주변장치는 거의 분리된 상태로 동작한다. SBC에는 기존 데스크톱 리눅스를 그대로 사용하여 커널 옵션 및 패치로 환경을 설정한 후 사용할 수 있다.

### 3.1 부팅디스크 작성

DOC는 NAND 형의 플래시 디바이스로 제조 중에 불량블록이 포함될 수 있다. 불량블록을 저장하기 위한 테이블(BBT)이 DOC안에 파일 형태로 포함되어있다. 현재 리눅스에서는 BBT 테이블의 내용을 읽지 못하므로, DOC 디바이스를 사용하려면 x86용 DOS DOC 툴을 적재한 DOS로 부팅 가능한 디스켓이 필요하다. DOS로 부팅 후 아래 명령어를 사용해 세그먼트 d000에 있는 BBT(Bad Block Table)를 docbbt.txt에 명령어(1)로 복사한다.

A:\dformat/win:d000/noformat/log:docbbt.txt (1)

명령어(1)로 리눅스에서 DOC 장치를 포맷시 사라지는 DOC의 정보를 파일 형태로 저장 후 다음에 DOC의 정보를 복원한다.

### 3.2 DOC를 위한 커널 작성

리눅스 플랫폼에서 DOC 설치를 위해 제공하는 더미(dummy) 오브젝트 형태의 TrueFFS 드라이버와 리눅스 커널인 kernel-2.4.2-i386.tar.gz를 획득한 후 특정 폴더(/tmp)에 압축해제 한다. 리눅스 커널이 준비되면 DOC을 커널에 추가하기 위해 패치 작업이 필요하다[11].

#### 3.2.1 커널 패치

DOC패치 명령(2)로 패치 작업이 끝나면 make xconfig, make menuconfig, make config 세 가지 방법 중 한 가지를 선택하여 DOC 인식을 필요한 옵션을 표4와 같이 설정한다.

#>patch_linux linux-2_2-patch driver-patch (2)

표 4. DOC를 위한 커널옵션 선택
Table 4. selection kernel option for DOC

```
▸ Loadable module support
 ->[*] Enable loadable module support
 ->[*] Set version information on all symbols for modules
 ->[*] Set version information on all symbols modules
▸ Block devices
 ->[*] Loopback device Support
 -><*> RAM disk support
 ->[*] Initial RAM disk(initrd) support
 ->[M] M-System DOC device support
```

### 3.2.2 장치 노드(node) 만들기

DOC 인식을 위해 장치들이 위치한 폴더(/dev)에 명령어(3)으로 노드를 생성한다. mknod_fl은 미리 작성된 쉘(shell) 스크립트로 메이저 넘버, 마이너 넘버를 생성하고, 노드 작성 시 세심한 작업이 필요하다.

#> mknod_fl (/dev/msys/fla, fla[1-4]) (3)

### 3.2.3 커널 컴파일

DOC가 포함된 커널을 이미지로 생성하기 위해 명령(4)-(9) 순서로 컴파일 한다.

#>make dep //의존성검사 (4)
#>make clean //컴파일부산물 제거 (5)
#>make bzImage //커널이미지 생성 (6)
#>make install //컴파일된 커널 복사 (7)
#>make modules //모듈 컴파일 (8)
#>make modules_install //모듈 설치 (9)

### 3.2.4 DOC 멀티 부팅 설정

DOC가 추가된 커널로 재시동해야 DOC를 포맷, 부팅 디스크 생성, 루트 파일시스템을 생성할 수 있다. 이를 위해 파일(/etc/lilo.conf)에 표 5와 같이 추가한다. 또한, 파일(lilo.conf)을 수정 후 부트로더에 추가된 커널 정보의 적용을 위해 명령(10)을 반드시 수행해야 한다.

#>./sbin/lilo (10)

리눅스를 재 부팅한 후 커널 컴파일 작업에서 생성된 DOC모듈(doc.o)의 추가 명령(11)을 수행한다.

#>insmod doc (11)

표 5. 부트로더에 DOC 마운트
Table 5. DOC mount from the bootloader

```
image=/boot/flash-2.4.0 //DoC 커널이미지
label=DOC(DiskOnChip) // 커널 이름
root=/dev/hda5 //linux root filesystem
```

### 3.2.5 DOC을 위한 부트로더 설정

DOC을 위한 그래픽을 프레임 버퍼를 사용한다는 것을 커널에 알려야 한다. 파일(/etc/lilo.conf)을 열어서 표 6과 같이 수정한다.

표 6. DOC 부팅을 위한 LILO 설정
Table 6. setting of LILO for DOC booting

```
boot=/dev/msys/fla1
compact
install=/boot/doc.b
map=/boot/System.map-2.2.14-5.0
disk=/dev/msys/fla1
bios=0x80
prompt
delay=50
time=50
vga=0x314 //800x600, 65535 Color 16bit
image=/boot/vmlinuz-2.2.14-5.0
label=linux
root=/dev/msys/fla1
initrd=/boot/initrd-2.2.14-5.0.img
```

### 3.2.6 커널 이미지 복사 및 루트파일시스템 작성

현재 DOC가 추가된 리눅스로 부팅된 상태이고 부팅과정에 사용된 리눅스 운영체제 관련 이미지를 DOC 장치에 복사(커널이미지, 루트 파일시스템)해야 한다. 명령(12)를 사용해 미리 정의한 쉘 스크립트로 명령을 수행한다.

#>./mkdocimg redhat-kernl.root.files.copy (12)

### 3.2.7 DOC를 통한 부팅

DOC를 부팅하기 위한 부트로더 적용 명령어(13)를 실행 후 운영체제를 재시동 한다. 부팅 환경이 변경되어 좌측 상단에 펭귄 모양의 LOGO를 볼 수 있다. VGA 모드가 맞지 않는다면 적절한 다른 숫자를 넣으라는 화면과 함께 부팅이 멈춘다.

#>./sbin/lilo (13)

## 4. 루트 파일시스템

루트 파일시스템은 커널이 사용할 유틸리티와 환경설정 파일들로 구성되어 있다. 주로 속도가 빠른 램 디스크 방법으로 플래시 메모리에 저장되어 사용된다. 요즘에는 플래시 파일시스템을 루트 파일시스템으로 사용하는 경우가 많다. 램 디스크는 별다른 물리적인 장치를 지칭하는 것이 아니라, 메모리 일부를 디스크로 인식시킨 것이다. 램 디스크를 루트 파일시스템으로 사용하는 것이 임베디드 리눅스 시스템에서 가장 일반적인 방법이다. 이것은 램 디스크는 램에서 동작하기 때문에, 속도가 빠를 뿐만 아니라, gzip 알고리즘으로 압축하기 때문에 용량을 줄일 수 있다는 장점이다.

### 4.1 램 디스크 만들기

명령어(14)와 같이 dd명령을 통한 램 디스크를 생성 한다. 1024byte의 한 블록을 4096번 반복하여 총 4MByte의 램 디스크를 생성하게 된다.

#>dd if=/dev/zero of=./ramdisk count=4096 bs=1024 (14)

### 4.2 ext2 파일 포맷

각각의 파일로서 존재하는 램 디스크는 자신만의 파일 시스템을 가지고 있다. 현재 리눅스의 경우 ext2와 ext3를 가장 많이 사용한다.

#>mkfs -t ext2 ramdisk.fs (15)

명령어(15)와 같이 생성된 램 디스크에 파일 포맷을 결정하고 필요한 파일을 복사한다.

### 4.3 파일 마운트

명령어(16)의 mount 명령을 통한 하나의 디렉토리에 마운트해 램 디스크에 필요한 파일의 내용을 채우고, gzip으로 압축하면 포팅 가능한 램 디스크 이미지 파일이 만들어 진다.

#>mount -t ext2 -o loop ramdisk.fs /mnt/ramdisk (16)

램 디스크에 저장되는 파일로 루트 파일시스템 필요한 바이너리 파일과 기본 유틸리티 파일이 복사되어 포함된다.

## 5. JFFS (Journalling Flash File System)

JFFS는 스웨덴의 Axis Communications에서 개발한 저널링 플래시 파일시스템으로 디스크가 없는 임베디드 장치에서 플래시 메모리를 이용한 전원 파손 등에 안전한 파일시스템이다. JFFS2로 버전이 업데이트 되면서 압축, 하드 링크 등 향상된 기능을 제공하며, 리눅스에서 제공하는 MTD 디바이스를 통해 접근이 가능하다.

### 5.1 JFFS 유틸리티 생성

JFFS2 이미지를 생성하기 위해서는 이미지에 삽입할 내용을 미리 확보하고 필요한 유틸리티를 설치하여야 한다. 필요한 유틸리티의 위치는 FTP://ftp.uk.linu x.org/pub/people/dwmw2/mtd/cvs/에서 가져올 수 있다. 다운로드가 완료/압축 해제 하였으면 생성된 util 디렉토리로 이동 후 이미지 생성에 필요한 유틸리티를 명령어(17)과 같이 컴파일 작업을 거치면 JFFS2가 생성된다.

#>make mkfs.jffs2 (17)

### 5.2 JFFS 이미지 생성

플래시 이미지를 생성할 수 있는 JFFS2 유틸리티가 확보되었으면 명령어 (18,19)로 JFFS2 이미지를 만들 수 있다.

#>cp mkfs.jffs2 /usr/local/bin/ (18)
#>mkfs.jffs2 -o usr.jffs -e 0x40000 -r tmpdir (19)

명령어(19)에서의 옵션은 -o는 생성 파일 이름, -e는 Flash erase 블록 사이즈, -r은 대상 디렉토리 지정(하위 디렉토리 포함)을 나타낸다. 이후에 JFFS2를 이용한 usr 이미지 생성 및 폴더에 필요한 내용을 복사 후 명령어(20)을 통해 이미지를 생성한다.

#>mkfs.jffs2 -o my.jffs -e 0x10000 -r myjffs (20)

정상적으로 jffs2이미지가 생성 되었다면 부트로더의 tftp 기능을 이용한 다운로드 사용가능을 할 수 있다.

## 6. Qt/E GUI 파일시스템 작성

Trolltech 사에서 제공하는 임베디드 Qt/E 툴킷을 다운받아 압축을 해제한다. 표 6와 같이 설치시 라이브러리 경로와 매크로 정의를 .bash_profile(Readhat 경우)에 작성하면 Qt/E GUI 파일시스템을 작성할 수 있다. 설치시 옵션 변경으로 불필요한 GUI 컴포넌트를 제외하면서 설치할 수 있다.

표 7. Qt/E 환경설정
Table 7. setting of Qt/E environment

```
QTDIR=~/qte-2.3.2
LD_LIBRARY_PATH=~/qte-2.3.2/lib:$LD_LIBRARY_PATH
export QTDIR LD_LIBRARY_PATH
```

Qt 임베디드 툴킷을 설치시 타겟 환경에 맞는 해상도, Qt 라이브러리, VFB 지원여부를 선정하여 Makefile을 작성한다. 자세한 환경 설정 관련은 파일 INSTALL에서 확인할 수 있다. 또한, Qt로 작성된 프로그램에 대하여 자동으로 Makefile를 작성하게 만들어 줄 수 있는 tmake를 설치한다. 표 6에서는 tmake를 bash_profile에 정의한 부분이다.

표 8. tmake 환경설정
Table 8. setting of tmake environment

```
TMAKEPATH=~/tmake-1.8/lib/qws/linux-x86-g++
PATH=$PATH:~/tmake-1.8/bin
export TMAKEPATH PATH
```

최종적으로 설치명령(make)를 수행해 Qt 툴킷을 설치할 수 있다.

## 7. DOC와 플래시 파일시스템의 비교

최종적으로 구성된 파일시스템의 계층 구조는 단일 칩 플래시 디스크인 DOC 기반 플랫폼(그림 5)과 MTD를 사용한 일반적인 플래시 메모리의 플랫폼(그림 6)으로 이루어진다.

그림 5는 x86기반의 임베디드 리눅스 시스템 상에서 DOC를 사용한 파일시스템 계층도이다. 이 경우 계층 구조에 TrueFFS 구조를 포함하고 있어 임베디드 시스템의 중요한 특징 중 신뢰성을 높이고 정확한 동작을 위한 파일시스템 관리로 에러의 수집 및 수정이 기능을 할 수 있는 장점을 가지고 있다.

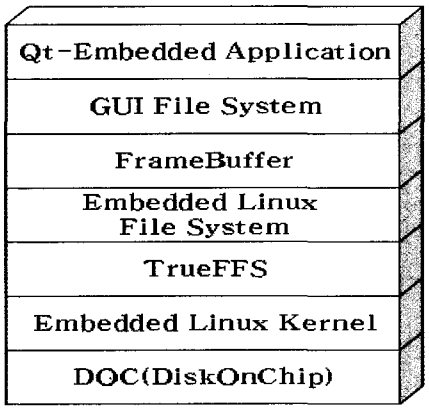


그림 5. DOC 기반 파일시스템
Fig. 5. Filesystem on DOC

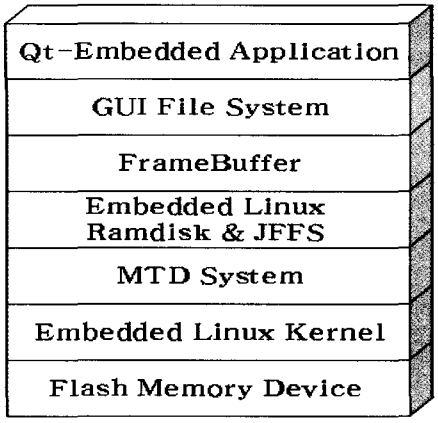


그림 6. 플래시 메모리 기반 파일시스템
Fig. 6. Filesystem on Flash Memory Devices

그림 6는 일반적인 플래시 메모리를 MTD 시스템을 사용하여 파일시스템 계층을 구성하는 방법으로 특별한 MTD 시스템을 커널 수준에서 중간 매개체로 두어 플래시 메모리와 데이터의 전달을 도와준다. 또한 루트 파일시스템은 속도가 빠른 램 디스크로 사용하고, 유저 파일시스템은 고용량을 지원할 수 있는 JFFS2의 향상된 플래시 파일시스템을 사용하여 파일 관리를 효과적으로 할 수 있다. 두 가지 경우 모두 최상위에 GUI 파일시스템인 Qt/E 추가하여 시스템 사용자가 쉽게 어플리케이션을 작성할 수 있게 구성할 수 있다.

## 8. 결 론

유비쿼터스 컴퓨팅 시스템의 구성 방법으로 임베디드 시스템을 이용한다. 임베디드 시스템에 가장 중요한 파일시스템의 선정은 시스템 전체 성능에 중요한 부분을 차지 할 것이다.

본 논문에서 임베디드 파일시스템의 사용되는 루트 파일시스템, 유저 파일시스템 및 GUI 파일시스템의 구성에 있어 서로 다른 플랫폼(DOC, MTD)을 갖는 구성에 적합한 계층 구조를 정형화하여 시스템 사용자에 대한 편의성을 제공할 수 있도록 하였다.

유비쿼터스 컴퓨팅 환경은 임베디드 시스템들이 센서 네트워크를 통하여 연결되어 통신하는 형태로 발전할 것이다. 이에 대비하여 소규모 운영체제가 필요할 것이며, 이 환경에서 사용될 수 있는 파일시스템이 선정은 중요한 이슈로 대두될 것이 예상된다. 향후 연구 과제로 RTOS를 이용한 실시간 임베디드 시스템 상에서 동작할 수 있는 실시간 파일시스템의 구성 대한 연구가 이루어져야 할 것이다.

## 참 고 문 헌

[1] 하원규, 김동환, 최남희, "유비쿼터스 IT 혁명과 제3공간," 전자신문사, 2002.
[2] 김홍남, "유비쿼터스 컴퓨팅 기기들을 위한 임베디드 S/W 플랫폼 기술 개발 계획,"정보처리학회지 10권, 제4호, 2003. 7.
[3] 임채덕 외6인, "임베디드 소프트웨어의 기술동향 및 산업발전전망," Institute of Information Technology Assessment, 2002.
[4] 최병욱 외3인, 임베디드리눅스, 홍릉출판사, 2002.
[5] Karim Yaghmour, *Building Embedded LINUX SYSTEMS*, O'RELLY, 2003.
[6] M-SYS, http://www.m-sys.com/content/products/
[7] Gatliff, Bill, "The How-To's of Flash: Implementing Downloadable Firmware," Embedded Systems Conference, San Jose, 25 September 2000.
[8] Matt Welsh, Matthias Kalle Dalheimer, Terry Dawson, Lar Kaufman, *Running Linux*, ORELLY, 1999.
[9] http://option.kernel.pe.ku/view.php3
[10] Qtopia, http://www.trolltech.com
[11] M Beck, H Bohme, M Dziadzka, *LINUX KERNEL programming*, ADDISON WESLEY, 2001.
[12] 이연조, *임베디드 리눅스 프로그래밍*, PC BOOK, 2002.
[13] Patrick Ward, Qt programming for linux and windows 2000, PH PTR,2001.

## 저 자 소 개

**이병권(Lee Byung Kwon)**
1995년 : 대덕대학 전자과(전문학사)
1999년 : 한밭대학교 전자계산(학사)
2002년 : 한남대학교 컴퓨터공학(석사)
2003~현재 : 충북대학교 전자계산대학원 박사과정

관심분야 : 임베디드시스템, 컴퓨터구조
Phone : 043-272-7725
E-mail : sonic747@hanmail.net

**주영관(Ju Young-Kwan)**
1999년 : 청주대학교 전자계산학과(학사)
2004년 : 충북대학교 전자계산학과(석사)
2004~현재 : 충북대학교 전자계산대학원 박사과정

관심분야 : 병렬컴퓨터 구조, 병렬처리
Phone : 043-272-7725
E-mail : juyg@john.chungbuk.ac.kr

**김석일(Kim Sukil)**
1975년 : 서울대학교 전기공학과(학사)
1975년-1990년 : 국방과학연구소
1989년 : North Carolina 주립대학(박사)
1990년-현재 : 충북대학교 전기전자 컴퓨터공학부 교수

관심분야 : 병렬컴퓨터구조, 슈퍼컴퓨팅, 병렬처리언어, 시간
장애인 사용자 인터페이스
Phone : 043-272-2328
E-mail : ksi@chungbuk.ac.kr

**전중남(Jeon Joong Nam)**
1981년 : 연세대학교 전자공학과(학사)
1985년 : 연세대학교 전자공학과(석사)
1990년 : 연세대학교 전자공학과(박사)
1990년-현재 : 충북대학교 전기전자
컴퓨터공학부 교수

관심분야 : 임베디드시스템, 컴퓨터구조, 병렬처리
Phone : 043-272-2264
E-mail : joongnam@chungbuk.ac.kr