실시간 운영체제 하에서의 임베디드 시스템을 위한 효율적인 FAT 모델 구현

조정철, 이호송[0], 성영락[†], 권택근, 이철훈
충남대학교 컴퓨터공학과, [†]국민대학교 전자정보통신공학부
{jcjo, hslee[0], tgkwon, chlee}@ce.cnu.ac.kr, [†] yeong@mail.kookmin.ac.kr

Implementation of efficient FAT for Embedded System on Real-Time Operating Systems

Jung-Chul Jo, Ho-Song Lee[0], Yeong Rak Seong[†], Taeck-Geun Kwon, and Cheol-Hoon Lee
Dept. of Computer Engineering, Chungnam National Univ.
[†] School of Electrical Engineering, Kookmin Univ.

요 약

최근 실시간 운영체제하에서의 특정한 기능을 수행하는 임베디드 시스템의 급속한 발전함에 따라 유, 무선 네트워크 기능뿐만 아니라 PDA(Personal Digital Assistants), 디지털 TV 등에 사용되는 멀티미디어 처리가 가능한 시스템이 필요하게 되었다. 이런 시스템은 효율적인 파일시스템을 필요로 하게 된다. 본 논문에서는 많은 운영체제에서 사용되는 FAT(File Allocation Table) 파일 시스템을 실시간 운영체제와 함께 동작하도록 구현하는 방법을 제시한다.

1. 서 론

최근 임베디드 시스템은 기존의 독립적인 시스템(stand-alone system)에서 점차 유, 무선을 이용한 네트워크 기능뿐만 아니라 PDA, DVD(Digital Versatile Disc), 디지털 TV 등의 멀티미디어 처리 기능을 필요로 하게 되었다. 이런 시스템은 많은 보조기억장치를 필요로 하기 때문에 기존의 임베디드 시스템에서 지원하던 내부 메모리만을 이용한 In-Memory 파일 시스템만으로는 작은 저장 공간으로 인하여 문제가 발생할 소지가 크다.

FAT 는 하드디스크 및 플로피 디스크뿐만 아니라 스마트 카드 등 다양한 보조기억장치를 지원하며, 간단한 구조를 가지고 있으며, 기존 장치와의 호환성에서도 많은 이점을 가지고 있다. 또한 FAT 는 현재 가장 많이 사용되는 파일 시스템 중 하나로 거의 모든 운영체제에서 지원하고 있다.

현재 널리 사용되고 있는 FAT32 의 경우에는 FAT12, FAT16 과 비교하여 대용량 디스크를 지원하고, 보다 나은 신뢰성과 성능을 보장한다.

본 논문에서 제안하는 파일 시스템은 기존의 FAT32 에 충남대학교에서 개발한 실시간 운영체제를 지원하는 기능을 추가한 모델이다.

본 논문의 구성은 2 장에서 FAT 의 구조를 간략히 소개하고 각 FAT 마다의 특성을 기술하며, 3 장에서는 실제 실시간 운영체제, 가상 파일 시스템, FAT 의 전체 구조를 설명한다. 4 장에서 실제 구현된 파일 시스템을 설명하고, 5 장에서 결론 및 차후 과제를 알아보도록 한다.

2. FAT 소개

2.1 FAT

FAT 는 운영체제가 하드 디스크 등의 저장장치 내에 유지하는 파일 배치표로서, 파일들이 저장되어 있는 클러스터들의 위치도를 제공한다. 즉 파티션의 처음 몇 개의 클러스터에 저장된 File Allocation Table 을 사용해서 파티션에 저장되어 있는 파일들을 유지하고 관리하는 기능을 수행한다.

2.2 FAT 기본 구조

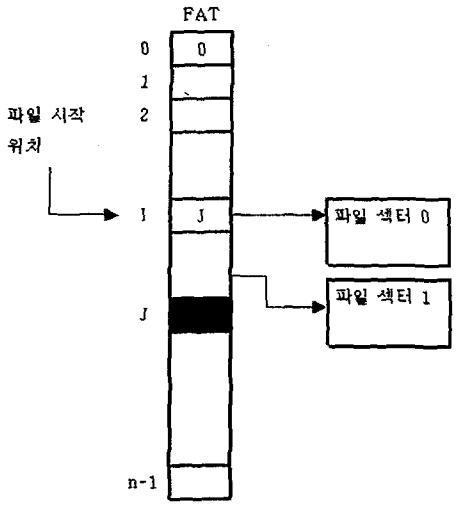
Master Boot Record	Reserved Sector	Partition Boot Record	FAT1	FAT2	Root 디렉토리

[그림 1] FAT 구조

전체적인 FAT 의 구조로 Master Boot Record 는 부트 코드로서 사용되며 FAT2 의 경우에는 FAT1 의 백업 용도로 사용된다. 논리 섹터 영역 중 처음 두개의 섹터는 특별한 용도로 사용하도록 정해져 있다.

[그림 2]은 FAT 에서 실제 논리섹터를 찾는 방법을 보여주고 있다. 디렉토리 엔트리에 저장된 파일 위치를 통해 파일에 접근하며 각 테이블 내에서는 다음 테이블 위치를 가리키는 리스트 구조로 되어있다. 실제적인 논리 섹터는 클러스터 사이즈를 통해 계산할

수 있다. 파일의 끝은 FAT 테이블 내에 특정 값을 주어 표시한다.



[그림 2] FAT 구성

2.3 FAT 분류

FAT 는 테이블을 구성하는 bit 수에 의해서 분류된다. 이는 전체 클러스터의 수와 동일하다.

	최대 크기	매체	클러스터 크기	특성
FAT 12	32MB	플로피 디스크	512B ~ 8KB	3 바이트를 이용하여 2 개의 테이블 인덱스를 구성.
FAT 16	4GB	소용량 저장 매체	512B ~ 64KB	스마트 카드나 소용량 하드 디스크에서 주로 사용. 단일 파일의 최대 크기는 4GB
FAT 32	8TB	대용량 저장 매체	512B ~ 32KB	상위 4 비트는 예약된 비트로 28 비트만 사용, 이론상 8TB 까지 가능하나 효율적 관리를 위해 32GB 로 제한

<표 1> FAT 분류

<표 1>은 각각의 FAT 의 특성을 서술하고 있다. 각 FAT 의 저장 사이즈는 클러스터 사이즈에 의하여 변경된다. VFAT 는 FAT 에서 최대 8 문자의 파일명만을 지원하는 것을 수정하기 위해 사용된다. FAT32 의 경우에는 기존의 FAT12 나 FAT16 에 비해 대용량의 매체를 접근하기 용이하며 빈공간 재계산 억제에 의해 디스크의 이용효율이 좋아진다.

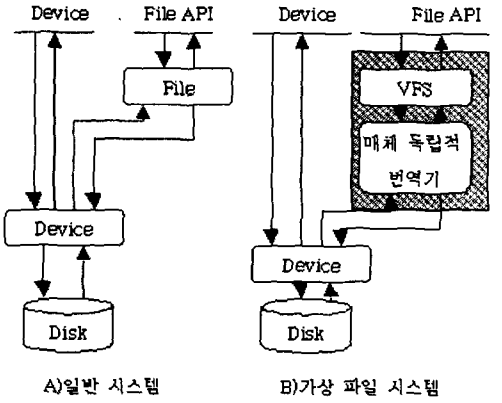
3. 실시간 운영체제 및 가상 파일 시스템 구조

3.1 임베디드 시스템

실시간 운영체제(RTOS : Real Time Operating System)는 임베디드 시스템이 대부분을 차지하고 있다. 실시간 운영체제의 경우 자원이 제한된 상황에서 특정 기능을 수행하도록 설계된 시스템으로 핵심 기능으로는 태스크(Task)를 관리하는 스케줄러(Scheduler)를 들 수 있다.

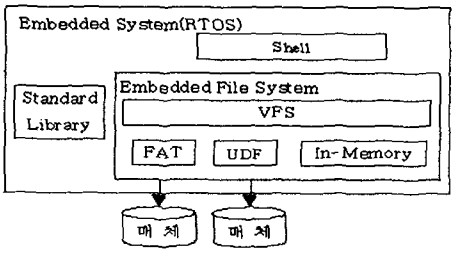
3.2 가상 파일 시스템

가상 파일 시스템(VFS)는 일반적으로 하위 파일시스템과 디바이스 장치를 관리하기 위한 모듈로 하위에 여러 파일시스템을 마운트(mount)하여 사용할 수 있도록 구성되어 있다. 이는 각각의 파일 시스템을 관리하는 것보다 효율적이며 확장성을 용이하게 한다. 가상 파일 시스템은 각 저장 장치에서 사용하는 파일 시스템에 상관없이 사용자에게 일정한 인터페이스를 제공해주고 내부적으로는 접근하는 파일 시스템에 적합한 함수를 호출하여 파일 입출력 연산을 수행하게 하여 특정 파일 시스템들을 일관된 방법으로 사용할 수 있게한다.



[그림 3] 일반 시스템과 가상 파일 시스템

[그림 3]은 일반 파일 시스템과 가상 파일 시스템을 사용할 때의 차이를 보여준다.



[그림 4] 전체 시스템 구조

임베디드 시스템, 가상 파일 시스템, 특정 일반 파일 시스템의 전체 구조는 [그림 4]과 같다.

4. 구현 내용

본 논문에서 구현한 임베디드 FAT 파일 시스템은 Linux 기반의 환경에서 작성되어 테스트 되었다. 컴파일러는 gcc-2.96 을 사용하였다. 차후 ARM920T 기반의 삼성 S3C2800™ 32-bit RISC Micro Processor 에서 테스트될 예정이다.

4.1 FAT 중요 함수

FAT12, FAT16 과 FAT32 파일 시스템을 구성하고 FAT32 의 클러스터 관리를 위한 데이터 구조를 추가하였다. FAT 는 다른 파일 시스템과 마찬가지로 슈퍼 블록(Super Block)에 매체의 모든 정보를 담고 있다. 따라서 처음 슈퍼 블록을 읽어들여야 하며 이 정보를 통하여 클러스터 사이즈와 블록 사이즈를 알아낼 수 있다. 다음 <표 2>는 FAT 의 기본 내부 동작을 위하여 필요한 중요 함수들이다.

<표 2> FAT 중요 함수

함수	기능
fat_read_super	FAT 부트 섹터에 접근한다. FAT 파일 시스템의 클러스터 사이즈 및 물리적 블록 사이즈를 읽어온다.
fat_read_root	디렉토리 정보를 포함하고 있는 루트 디렉토리 정보를 읽어온다.
fat_bmap	클러스터 값과 offset 을 이용하고 fat_access 함수를 호출하여 실제 클러스터의 내용에 접근한다.
fat_add_entries	디렉토리 엔트리(Directory Entry) 정보를 해당 클러스터 내에 쓴다.
fat_access	주어진 클러스터 값과 offset 을 이용하여 물리적인 섹터을 찾아 그 위치의 FAT 엔트리값을 반환한다.

4.2 가상 파일 시스템을 위한 인터페이스

FAT 파일 시스템은 추후 가상 파일 시스템과의 통합을 위하여 가상 파일 시스템의 함수와 동일한 매개 변수와 기능을 수행하도록 하였다. 다음 <표 3>에서는 가상 파일 시스템과의 연동에 필요한 표준 인터페이스들을 설명한다.
가상 파일 시스템과의 통합은 FAT 파일 시스템을 가상 파일 시스템에 등록한 후 사용자가 마운트(mount)과정을 요청하면 msdos_read_super <표 3>를 통해 호출된 fat_read_super<표 2>함수를 통해 파일 시스템을 구성하고, 가상 파일 시스템을 통해 호출된 요청은 FAT 인터페이스를 통해 변환 후 FAT 파일 시스템을 사용하게 된다.

임베디드 시스템은 적은 자원으로 인하여 메모리 관리와 컴파일된 오브젝트 사이즈가 작아야 한다. 이를 위하여 디렉토리 엔트리를 최소화하고 있으며 디렉토리 엔트리에 연관된 아이노드만을 저장하고 있다.

<표 3> 가상 파일 시스템을 위한 FAT 인터페이스

인터페이스	기능
msdos_read_super	fat_read_super 를 호출하기 위한 표준 인터페이스로 루트 디렉토리 엔트리와 루트 아이노드를 생성한다..
msdos_lookup	디렉토리나 파일명을 이용하여 해당 아이노드 정보를 읽어온다.
msdos_mkdir	주어진 아이노드 정보를 이용하여 디렉토리를 생성한다.
msdos_rmdir	아이노드와 디렉토리 엔트리 정보를 이용하여 디렉토리를 삭제한다.
msdos_find	주어진 디렉토리 엔트리를 이용하여 해당 위치를 찾는다.
msdos_init	가상 파일 시스템에 FAT 파일 시스템을 등록한다.

5. 결론 및 향후 과제

본 논문에서는 실시간 운영체제하에서의 임베디드 시스템을 위한 효율적인 FAT 모델을 구현하였다. 이 파일 시스템은 상위의 가상 파일 시스템에서 제공되는 표준 인터페이스를 따르고 있다.
향후 연구 과제로는 사용되는 파일명을 255 문자까지 지원 가능한 VFAT 모델을 구현할 계획이며, 또한 임베디드 시스템은 대부분 제한된 자원을 사용하기 때문에 성능 개선에 대한 연구도 필요하다.

6. 참고문헌

[1] Daniel P. Bovet and Macro Cesati, *Understanding the LINUX KERNEL*, O' REILLY, 2002.
[2] GARY NUTT, *Kernel Projects FOR Linux*, Addison Wesley, 2001.
[3] 강석민 " 임베디드 시스템을 위한 In-memory 파일 시스템 구현" 한국정보과학회 춘계학술발표 논문집, pp.61~63, Apr. 2002.