실시간 운영체제에서 가상 파일시스템 설계 및 구현*

류현수[0] ,유용선, 김용희, 권영훈‡, 이철훈
충남대학교 컴퓨터공학과, ‡삼성탈레스㈜ R&D 팀
{hsryu, ysryu, yonghee, chlee}@ce.cnu.ac.kr, yh1004.kwon@samsung.com‡

The Design and Implementation of Virtual File system on RTOS

Hyeon-Soo Ryu[0], Yong-Seon Ryu, Yong-Hee Kim, Young-Hoon Kwon‡, Cheol-Hoon Lee
Dept. of Computer Engineering, Chungnam National Univ.
‡Dept. of R&D Team, SAMSUNG THALES CO.,LTD

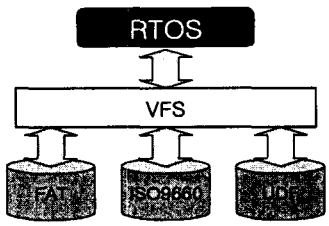
요 약

오늘날 임베디드 시스템은 우리들의 생활에 커다란 변화를 가져왔으며, 많은 적용분야와 다양한 기능을 갖추고 있어서 직장 생활 및 여가 생활 등에서 널리 사용되고 있다. 임베디드 시스템에 적용되는 운영체제는 높은 신뢰성과 빠른 수행속도, 적은 메모리를 특징으로 하는 실시간 운영체제(RTOS)이며, 임베디드 환경에서 저장장치를 관리하기 위한 파일 시스템은 필수 요구 조건이 되고 있다. 파일 시스템은 프로그래머가 시스템 내부의 저장장치나 네트워크상의 저장장치에 효율적인 접근을 할 수 있도록 해주며, 이러한 저장장치들로는 CD-ROM, 플로피 디스크, 하드 디스크, 플래시 메모리 등이 있는데, 이러한 현존하는 많은 파일 시스템을 통합 관리할 수 있는 가상 파일 시스템(Virtual File System : VFS)을 필요로 하게 되었다. 본 논문에서는 실시간 운영체제와 함께 동작하는 가상 파일 시스템을 설계하고 구현하는 방법을 제시한다.

1. 서론

사람들은 임베디드 시스템의 존재를 잘 인식하지 못하지만, 공장 자동화, 방어 시스템, 수송장치, 항공장치 등 여러 분야에서 사용되고 있으며, 또한 방대한 적용 분야와 다양한 기능을 갖추고 있어서 가정과 직장 생활 및 여가 생활에서 널리 사용되고 있다. 이러한 시스템에 탑재될 수 있는 운영체제는 높은 신뢰성을 보장하고, 수행 속도가 빠르며 적은 메모리를 요구하는 실시간 운영체제(RTOS)이며 점점 많은 임베디드 시스템에서 RTOS 의 활용이 증가하고 있다. 또한 시스템 내에서 다양한 저장매체의 활용이 증가하고 있으며, 따라서 파일 시스템은 임베디드 시스템 환경에서 반드시 필요한 컴포넌트라 할 수 있다. 이러한 다양한 파일 시스템을 통합 관리할 수 있고, 공통 인터페이스를 제공하는 것이 가상 파일 시스템(Virtual File System : VFS)이다.

가상 파일 시스템은 사용자와 여러 개의 실제 파일 시스템 사이의 추상적인 개념이다. Linux™는 MS-DOS 나 EXT2 등의 많은 다른 파일 시스템을 지원하며, 마운트되어 있는 모든 파일과 파일 시스템을 하나로 통합하는 가상 파일 시스템을 제공한다. 따라서, 사용자와 프로세스는 파일이 어떤 파일 시스템에 속해 있는지 알 필요 없이 파일에 접근한다. 예를 들어, 플로피 디스크나 CD-ROM 등의 파일을 사용하려는 요청이 들어오면, 이 요청은 VFS 에 전해지고, VFS 는 그에 맞은 파일시스템 드라이버를 선택한다. 각각의 파일 시스템 드라이버는 그에 해당하는 파일 시스템 조작방법을 실제로 구현함으로써 사용자는 파일 시스템의 형태나 그 하부의 물리적인 장치의 특징에 상관없이 시스템의 저장장치에에 있는 파일이나 디렉토리를 인식할 수 있게 된다.



[그림 1] 가상 파일 시스템(VFS)

[그림 1]은 서로 다른 파일 시스템에서 공통 인터페이스를 제공하는 VFS 와 이를 포함하는 실시간 운영체제를 도식화한 그림이다. 본 논문의 2 장에서는 관련 연구를, 3 장에서는 가상 파일 시스템의 자료구조 및 설계를, 4 장에서는 테스트 환경 및 결과를, 5 장에서는 결론 및 향후 연구 과제에 대해 기술한다.

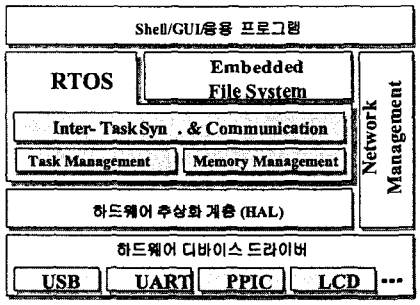
* 이 논문은 산업자원부 중기거점과제 연구비 지원에 의한 것임

2. 관련 연구

2.1 실시간 운영체제 (RTOS)

실시간 시스템이란 시스템의 수행 결과가 기능적으로 정확해야 할 뿐만 아니라, 결과가 도출되는 시간 역시 주어진 제약조건을 만족시켜야 하는 시스템이다. 이러한 실시간 시스템에 사용되는 운영체제가 실시간 운영체제이다. 범용 운영체제는 CPU 의 효율성과 각 프로세스의 공평성에 중점을 두고 스케줄링 하지만, 실시간 운영체제는 시간적 제약사항에 더욱 중점을 두고 스케줄링 한다. 이러한 시간 제약사항으로 몇 가지 주요한 특징들이 있다.

첫째, 멀티태스킹을 지원하고 선점 가능해야 한다. 이는 실시간 시스템이 예측 가능하여 우선순위가 높은 태스크에 대하여 시간 결정성을 보장함을 의미한다. 둘째, 태스크간의 우선순위에 따른 동작을 보장하여 종료시한을 만족할 수 있는 스케줄링 정책을 지원해야 한다. 셋째, 독립적인 태스크들 사이에서 자원의 공유와 통신이 필요하므로, 이들간의 동기화 및 통신방법이 마련되어야 한다. 넷째, 실시간 운영체제의 시간적인 행동이 정확해야 한다. 여기에는 인터럽트 지연시간, 시스템 콜 처리시간이 포함되며, 시스템과 디바이스 드라이버의 인터럽트 레벨도 고려해야 한다.



[그림 2] 실시간 시스템 전체 구성도

실시간 운영체제는 실행 속도와는 상관이 없다. 즉, 운영체제의 각종 동작이 종료시한 내에 이루어진다면 그 시간이 아무리 길어도 실시간 운영체제라고 말할 수 있다. 실시간 운영체제는 UNIX™, Linux™과 같은 범용시스템과 기본적인 스케줄러의 기능이 크게 다르지 않지만, 태스크 생성 시간, 태스크 전환 시간, 커널 모드에서 소비되는 시간, 임계영역(Critical section)의 최대 크기, 인터럽트 응답 시간 등을 최소화하고, 최악의 경우(Worst-Case)를 계산할 수 있게 만든다는 점에서 큰 차이점이 있다. [그림 2]는 실시간 시스템 전체 구성도이다.

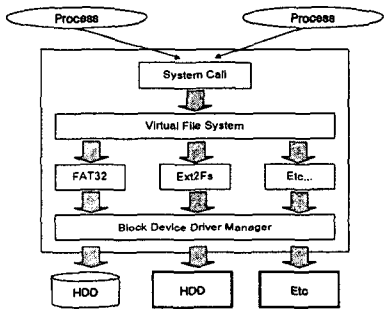
2.2 파일 시스템

파일 시스템은 파일에 이름을 붙이고, 저장이나 검색을 위해 논리적인 위치를 나타내는 방법이다. 도스, 윈도우, OS/2, 매킨토시 및 유닉스 기반의 운영체제들은 파일들이 계층적 구조를 가지고 있고, 파일은 계층구조 내의 디렉토리 또는 서브디렉토리 내에 놓여진다. 파일 시스템은 파일의 이름을 붙이는 규칙을 가지고 있는데, 이러한 규칙에는 파일이름의 길이 제한, 사용 가능한 문자 제한 등이 포함되며, 몇몇 파일 시스템에서는 파일이름 확장자의 길이에도 제한을 두고 있다. 또한 파일 시스템은 디렉토리 구조를 통하여 파일까지의 경로를 설정하는 형식을 포함한다. 현존하는 파일시스템을 예로 들면, 플로피 디스크나 하드 디스크, 램디스크는 FAT12/16/32 파일시스템을, CD-ROM 은 ISO9660 파일 시스템을, DVD 는 UDF 파일시스템을 사용한다.

3. VFS 자료구조 및 설계

3.1 VFS 개요

가상 파일 시스템은 구체적인 파일 시스템들을 관리하고 커널의 요청이 있을 때마다 개별 파일 시스템의 의존적인 인터페이스를 호출한다. 새로운 파일 시스템이 추가되더라도 VFS 를 제외한 다른 부분에서는 이를 인식할 필요가 없다. [그림 3]는 사용자 프로세스가 시스템 콜(System Call)을 통하여 VFS 에 접근하고, 적당한 파일 시스템을 연결시켜주고, 블록 디바이스 드라이버를 통하여 디바이스에 접근하는 모습을 보여준다.



[그림 3] 가상 파일 시스템

3.2 VFS 오브젝트

VFS 를 표현할 수 있는 공통 모델은 다음과 같은 객체 유형으로 이루어지며, 객체는 자료 구조와 이에 대한 연산을 수행하는 메소드를 정의하고 있다.

- Superblock
  마운트된 파일 시스템에 대한 정보를 저장한다. 디스크 기반 파일 시스템의 경우, 이 객체는 디스크에 저장한 파일시스템 제어 블록에 대응한다.
- Vnode
  특정 파일에 대한 일반 정보를 저장한다. 디스크 기반 파일 시스템의 경우, 일반적으로 디스크에 저장한 파일 제어 블록에 대응한다. Vnode 번호가 할당되어 파일 시스템 내에 각 파일을 유일하게 식별한다.
- File
  열린 파일과 프로세스 사이의 상호 작용과 관련한 정보를 저장한다. 이 정보는 각 프로세스가 파일에 접근하는 동안 커널 메모리에만 존재한다.
- Dentry
  디렉토리 항목과 이에 대응하는 파일의 연결에 대한 정보를 저장한다.

3.3 VFS 구조체

[그림 4]에서 FileSystemStt 구조체는 커널에 등록된 모든 구체적 파일 시스템들을 관리기 위한 구조체이며,

VFSStt 구조체는 하나의 구체적인 파일 시스템을 표현하기 위한 구조체이다.

```
typedef struct VFSTag{
        char            szName[32];
        DWORD           dwType;
        VFSOPStt        op;
        BlkDevObjStt    *pDevObj;
        DWORD           dwMode;
        VNodeStt        *pMountNode;
        VNodeStt        *pRootNode;
        VNodeManStt     *pVNodeMan;
        void            *pPrivate;
        struct  VFSTag  *pPre;
        struct  VFSTag  *pNext;
};
typedef struct VFSTag VFSStt;
typedef struct FileSystemTag {
        int             nTotal;
        struct VFSTag   *pStart,*pEnd;
        struct VNodeTag *pRootNode;
};
typedef struct FileSystemTag FileSystemStt;
```

[그림 4] VFS 구조체 소스 코드

3.4 VFS 시스템 콜

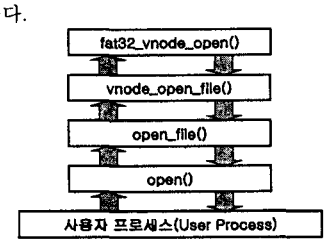
사용자가 VFS 를 통해 파일 또는 디렉토리에 어떠한 동작을 요구할 경우 시스템 콜을 이용한다. [표 1]은 대표적인 시스템 콜이며, 본 연구에서 지원할 수 있도록 실험 중이다.

[표 1] VFS 의 시스템 콜

System Call	Description
mount, unmount	Mount/unmount file systems
sysfs	Get file system information
statfs, fstatfs, ustat	Get file system statistics
chdir, fchdir, getcwd	Manipulate current directory
mkdir, rmdir	Create and destroy directory
stat,fstat, lstat	Read file status
open, close, create	Open and close file
lseek	Change file pointer
read, write	Carry out file I/O operation

3.5 시스템 콜을 이용한 파일 열기

사용자 프로세스에서 open() 함수를 호출하면 4 가지 단계를 거쳐 파일에 대한 vnode 가 생성되고 파일 핸들을 통해 vnode 에 접근할 수 있다. [그림 5]은 FAT 파일 시스템을 기준으로 하나의 파일을 열 때 호출되는 함수의 계층적 구조를 보여준다.

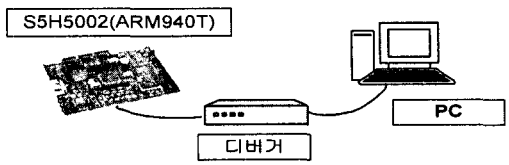


[그림 5] 함수의 계층적 호출

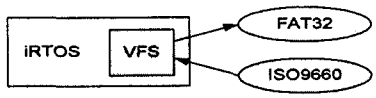
프로세스는 접근하려는 파일의 이름을 인자로 주어 open() 함수를 호출한다. 시스템 콜에서 open_file() 함수를 호출하고, 탐색된 부모 디렉토리 내의 원하는 파일을 오픈한다(vnode_open_file()). 파일 디스크립터 구조체를 초기화하고, 부모 디렉토리에 대한 참조 카운터를 하나 감소시킨다.(fat32_vnode_open())

파일 읽기(read()), 파일 쓰기(wirte()), 파일 닫기(close()) 도 위와 비슷한 계층적 함수호출이 이루어진다.

3. 테스트 환경 및 결과



[그림 6] 테스트 환경



실행 예) $cp /cdrom/test.txt /ramdisk/TEST.txt

본 연구팀에서 개발한 실시간 운영체제인 iRTOS™에 VFS 모듈을 구현하였으며, ARM940T(32-bit RICS CPU)가 탑재되어 있는 삼성 S5H5002 보드에서 SDT2.51 통합 개발 환경을 사용하여 테스트하였다.

테스트 결과 안정적으로 파일 복사(cp)가 이루어짐을 확인하였다.

5. 결론 및 향후 연구 과제

본 논문에서는 임베디드 시스템을 위한 가상 파일 시스템을 구현하였다. 현재는 DVD 를 지원하는 UDF 파일 시스템, CD 를 지원하는 ISO9660 파일 시스템, MS-DOS 에서 사용하는 FAT16/32 파일 시스템을 VFS 에 등록하고 마운트하여 파일 읽기, 쓰기를 실험 중이다. 향후 과제로는 파일 시스템의 다양한 시스템 콜을 지원하고, 현존하는 다양한 파일 시스템을 지원하는 것이 필요하다.

6. 참고문헌

[1] http://www.inestech.com

[2] David Stepner 외 2 명, " Embedded Application Design Using a Real-Time OS" , IEEE, 1999

[3] Jean, J. Labrosse, " μ C/OS The Real-Time Kernel" R&D Publications, 1992.

[4] Daniel P. Bovet, Macro Cesati, " Understanding the Linux Kernel, 2nd Edition" , published by HANBIT Media, Inc. 2003.

[5] 오재준, " OS 제작의 원리 그리고 Codes" , 기남사, 2004.