Audio Player를 위한 경량 파일시스템의 설계 및 구현*

송예진 0, 조문행, 손필창, 이철훈
충남대학교 컴퓨터공학과
{yjsong, root4567, pcson, chlee}@cnu.ac.kr

The Design and Implementation of Light-Weight Real-Time Operating System for Audio Player

Ye-Jin Song0, Moon-Haeng Cho, Pil-Chang Son, Cheol-Hoon Lee
Dept. of Computer Engineering, Chungnam National University

요 약

임베디드 시스템은 용도에 따라서 다양한 기능을 수행하도록 설계된 전용 컴퓨팅 시스템인데, 임베디드 시스템 중 최근 등장한 많은 포터블기기에서 중요시 되는 하드웨어 자원이 저장장치이다. 따라서 임베디드 환경에서 저장장치를 관리하기 위한 파일 시스템은 필수 요구 조건이 되고 있다. 파일 시스템은 프로그래머가 시스템 내부의 저장장치에 효율적인 접근을 할 수 있도록 해주는데, 적은 크기의 SRAM 메모리를 사용하는 Audio Player 에는 경량의 파일 시스템이 필요하다. 본 논문에서는 Audio Player 시스템을 위한 경량 파일 시스템을 설계 및 구현하였다.

1. 서 론

최근 임베디드 시스템은 PMP, MP3 와 같은 포터블 기기 쪽으로 많은 발전을 이루었다. 이러한 포터블 기기에서 CPU, 메모리와 함께 중요한 하드웨어 자원이 바로 저장장치이고 이런 하드웨어 자원은 제한적이다. 이와 같은 임베디드 시스템의 한정적인 하드웨어 자원을 효율적으로 관리하기 위하여 실시간 운영체제(Real-Time Operating System)가 많이 사용되고 있다.

실시간 운영체제는 시스템이 정확한 계산 결과를 출력해야 한다는 기능정확성 그리고 이러한 결과를 주어진 시간 안에 얻을 수 있어야 한다는 시간정확성에 모두 의존적이다 [2].

임베디드 시스템에서 대부분 저장장치로 NAND Flash 메모리를 많이 사용하고 있으며, 이런 저장장치를 효율적으로 관리하기 위한 파일 시스템은 임베디드 시스템 환경에서 반드시 필요한 컴포넌트이다.

본 논문에서는 적은 크기의 메모리에 실시간 운영체제와 파일 시스템, 애플리케이션이 모두 탑재되어야 하는 Audio Player 시스템에서 경량 파일 시스템을 설계 및 구현하였으며, 실시간 운영체제로 UbiFOS™을 사용하였다.

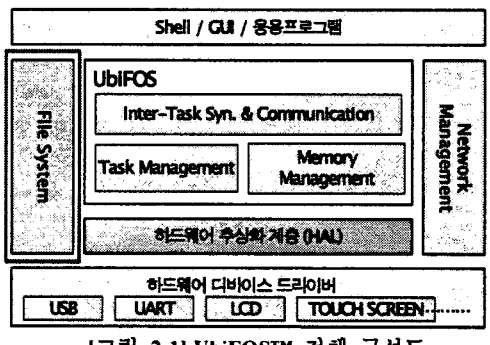
본 논문의 구성은 2 장에서 관련연구로 실시간 운영체제 UbiFOS™의 전반적인 구성과 파일 시스템에 대하여 기술하고, 3 장에서는 Audio Player 를 위한 경량 파일 시스템의 설계 및 구현 내용을 기술한다. 4 장에서는 테스트 환경 및 결과를 기술하고, 5 장에서는 결론 및 향후 연구과제에 대해서 기술한다.

* 본 연구는 정보통신부의 선도기반기술개발사업의 지원으로 수행되었습니다

2. 관련 연구

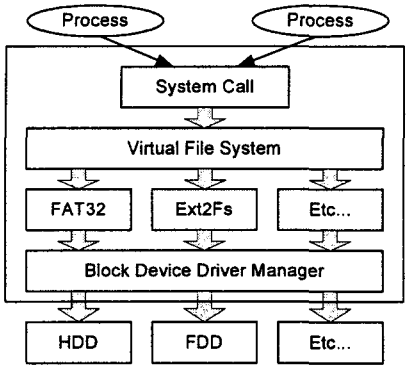
2.1 UbiFOS™

실시간 운영체제 UbiFOS™ 은 멀티태스킹을 지원하기 위해 256 단계의 우선순위를 제공하는 선점형 스케줄러로 동일 우선순위에 대해 FIFO 와 Round-Roubin 기법을 제공한다 [1][3].



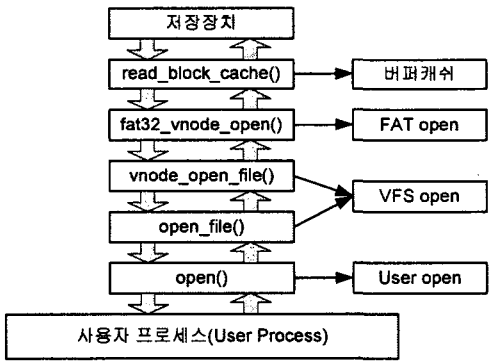
[그림 2-1] UbiFOS™ 전체 구성도

2.2 가상 파일 시스템(VFS)



[그림 2-2] 가상 파일 시스템

[그림 2-2] 가상 파일 시스템은 개별 파일 시스템들을 관리하고 커널의 요청이 있을 시 개별 파일 시스템의 의존적인 인터페이스를 호출한다. 새로운 파일 시스템이 추가되더라도 VFS(Virtual File System)를 제외한 다른 부분에서는 이를 인식할 필요가 없다 [4].



[그림 2-3] 함수의 계층적 호출

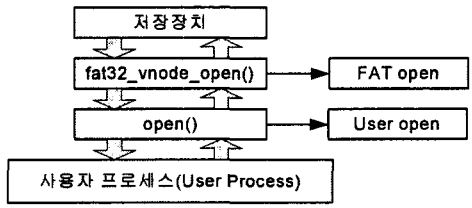
[그림 2-3]은 사용자 프로세스에서 open() 함수를 호출하는 경우로 여러 단계를 거쳐 파일에 접근하는 계층적 구조를 보여준다 [4].

3. 파일 시스템의 설계 및 구현

3.1 Audio Player 시스템을 위한 경량 파일 시스템의 필요성

구현한 경량 파일시스템은 Audio Player 의 NAND flash 에 저장된 음악 파일에 대한 오퍼레이션을 수행한다. 일반적으로 Audio Player 는 크기와 가격 문제로 적은 크기의 메모리를 가지고 있다. 예를 들어, 본 논문의 테스트 기반이 되는 S5L841F 오케스트라 평가 보드는 512KB 의 SRAM 을 가지고 있고 이 SRAM 에 실시간 운영체제인 UbiFOS™와 VFS(Flash Translation Layer 포함), 그리고 MP3, OGG, WMA 파일 포멧을 지원하기 위한 코덱, 키 입력과 USB 기능을 위한 애플리케이션이 들어가야 하는데 총 크기가 SRAM 의 용량을 넘어서는 문제가 발생한다. 하지만 실시간 운영체제와 코덱, 기타 애플리케이션이 차지하는 메모리 공간을 축소할 수는 없다. 본 논문에서는 이런 SRAM 의 용량 제한을 만족시켜 주고 시스템에서 필요로 하는 기능만을 갖도록 기존의 VFS 를 경량화 하였다.

3.2 경량 파일 시스템의 설계 및 구현



[그림 3-1] 경량 파일 시스템 함수의 계층적 호출

구현한 경량 파일 시스템에서는 [그림 3-1]에서와 같이 버퍼캐쉬 단계와 VFS 단계를 생략하고 FAT16/32 파일 시스템을 지원하며, 파일을 접근하는 계층 구조를 축소하여 수행 속도를 향상 시켰다. 또한, NAND flash 의 저장장치에 접근하기 위한 FTL 을 파일 시스템 내부에 추가하였다.

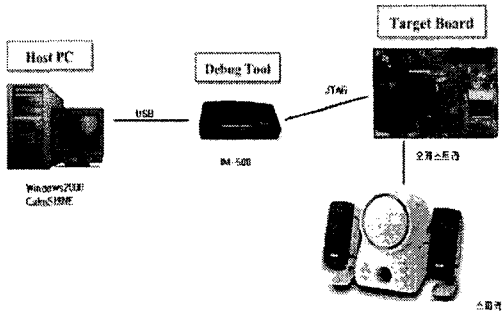
```
#define BufSize 512          // 블록의 기본 크기

typedef struct mf_fat_struct {
        MK_INT_t        FatIndex;
                        // FAT entry 인덱스
        MK_S8_t         FatTable[BufSize];
                        // FAT entry 저장 테이블
}MF_FAT;
// FAT entry를 관리하는 구조체

MK_U32_t MF_FatClusterToBlock(MK_U32_t dwCluster )
{
        dwblock = ClusterToBlock(dwcluster);
                // 클러스터 번호를 블록 번호로 변환
        LogicalToPhysical(dwblock);
                // 논리적인 블록 번호를 물리적인 주소로 변환
}
// FTL을 통한 주소변환

#define  MAX_VNode_Num    4
                // 파일 open 최대갯수
```

[그림 3-2] 경량 파일 시스템 구조체와 함수, 변수

[그림 3-2]는 본 논문에서 구현한 경량 파일 시스템의 구조체와 함수, 변수로 MF_FAT 구조체는 FatIndex를 통해 FAT entry 를 효과적으로 관리하여 메모리 낭비를 줄이고, MF_FatClusterToBlock()은 논리적인 주소를 FTL 을 통하여 물리적인 주소로 변환하는 함수이며, MAX_VNode_Num 는 파일을 open 할 수 있는 최대 갯수이다.

본 논문에서 구현한 경량 파일 시스템이 지원하는 기본 API 함수로 MF_Open(), MF_Close(), MF_Create(), MF_Read(), MF_Write(), MF_Remove(), MF_Rename(), MF_MkDir(), MF_OpenDir(),MF_CloseDir(), MF_ReadDir(), MF_Lseek()이 있다. 그리고 파일 시스템의 이미지 크기는 FTL 포함하여 34KB 로 Audio 애플리케이션 구동만을 위해 최적화 되었다.

4. 테스트 환경 및 결과



[그림 4-1] 실험 환경

본 논문에서 구현한 경량 파일 시스템은 실시간 운영체제인 UbiFOS™의 커널이 탑재된 CalmRISC16 CPU 기반의 S5L841F 오케스트라 평가 보드에서 CalmShine을 사용하여 Audio 애플리케이션의 정상동작 여부를 테스트하였다.

[그림 4-2]는 구현한 경량 파일 시스템의 기본 API 를 테스트 한 결과로 song 이라는 디렉토리를 생성하고 1.txt 라는 파일에 "This is a test "라는 메시지를 쓴 후, 1.txt 파일의 이름을 song.txt 로 변경하여 출력하였다. 마지막으로, 생성된 디렉토리와 파일의 List 를 출력한 결과이다.

```
*************************************************
 1 : Create Dir        2 : Delete File
 3 : Write File        4 : Read File
 5 : Rename File       6 : List Dir        0 : EXIT
*************************************************
Enter Directory name to Write : song
mkdir(/song) - 0

*************************************************
 1 : Create Dir        2 : Delete File
 3 : Write File        4 : Read File
 5 : Rename File       6 : List Dir        0 : EXIT
*************************************************
Enter File name to write:1.txt
Enter Text  to write:This is a test

*************************************************
 1 : Create Dir        2 : Delete File
 3 : Write File        4 : Read File
 5 : Rename File       6 : List Dir        0 : EXIT
*************************************************
Enter Old File Name: 1.txt
Enter New File Name: song.txt

*************************************************
 1 : Create Dir        2 : Delete File
 3 : Write File        4 : Read File
 5 : Rename File       6 : List Dir        0 : EXIT
*************************************************
Enter File name to read from : song.txt
Contents ....:This is a test

*************************************************
 1 : Create Dir        2 : Delete File
 3 : Write File        4 : Read File
 5 : Rename File       6 : List Dir        0 : EXIT
*************************************************
[  0] (00000002)       2 /     0 SONG          song
[  0] (00000003)       3 /  4024 SONG.TXT      song.txt
```

[그림 4-2] 테스트 결과

5. 결론 및 향후 과제

본 논문에서는 Audio 애플리케이션을 위한 경량 파일 시스템을 설계 및 구현하였다. 논문의 주 목적이 파일 시스템의 크기를 줄이는데 있기 때문에 추가적인 기능이나 성능의 향상 또한 더욱 고려해 봐야 할 사항이다. 따라서 향후 과제로는 크기도 작고 성능 면에서도 우수한 파일 시스템에 대한 연구가 필요하다.

참고 문헌

[1] Embedded System & RTOS, http://www.aijisystem.com

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

[3] 박희상, 안희중, 김용희, 이철훈, "Design and Implementation of Memory Management Facility for Real-Time Operating System, iRTOS™", 한국정보과학회, 2002

[4] 류현수, 유용선, 김용희, 권영훈, 이철훈, "The Design and Implementation of Virtual File system on RTOS", 한국정보과학회, 2004. 가을