편집중인 파일을 자동으로 저장하는 파일 시스템의 설계

조원호*, 이인환**
한양대학교 전자통신컴퓨터공학과
e-mail: * whcho@csl.hanyang.ac.kr, ** ihlee@hanyang.ac.kr

Automatically Saving Editing-Files in File System Layer

Wonho Cho*, Inhwan Lee**
Dept. of Electrical and Computer Engineering
Hanyang University

요 약

본 논문에서는 현재 편집중인 파일을 주기적으로 저장하여 파일이 저장되지 못한 상황에서도 쉽게 복구할 수 있는 파일 시스템을 설계하였다. 제안하는 파일 시스템은 Ext3 파일 시스템을 기반으로 자동 저장에 필요한 자료 구조와 기능을 추가하여 구현하였으며 리눅스 환경에서 사용할 수 있도록 구현하였다. 또한 파일 시스템을 사용하는 데 필요한 시스템 콜과 API를 추가하였다.

1. 서론

컴퓨터를 사용하다 보면 정전으로 인해 전원이 꺼지거나 하드웨어 또는 소프트웨어의 문제로 인해 컴퓨터가 멈춰 버리는 일이 종종 발생한다. 만약 사용자가 편집중이던 파일을 미리 저장해 놓았다면 다행이지만, 그렇지 않다면 그 내용을 모두 잃어버리게 된다. 몇 시간동안 작성하던 문서나 정성들여 수정하던 중요한 소스코드의 내용을 잃어버린다는 것은 매우 치명적인 일이다.

문서의 내용을 잃지 않기 위해서는 사용자가 작업 중간 중간에 파일을 자주 저장해야만 한다. 하지만 모든 문서작업을 할 때마다 저장 버튼을 수시로 누르는 것은 상당히 번거롭고 집중력을 떨어뜨릴 수 있다.

현재의 버전 관리 파일 시스템이나 응용 소프트웨어에서는 이러한 문제를 해결하기 위해 각 파일 또는 디렉토리의 변경 사항을 버전으로 만들어 관리하는 등의 노력을 기울이고 있다. 하지만 현재의 버전 관리 파일 시스템은 전체 파일시스템을 대상으로 하고 있으므로 편집중이던 파일의 복구만을 위한 파일 시스템으로는 적합하지 않다. 몇몇 응용 프로그램에서도 파일을 주기적으로 저장하는 기능을 제공하고 있지만 그 프로그램이 아닌 다른 응용 프로그램에서는 해당 기능을 사용할 수 없기 때문에 활용성이 떨어진다.

본 논문에서는 현재 편집중인 파일을 자동으로 저장함으로서 파일을 저장하지 못한 채 시스템이 종료된 상황에서도 파일의 내용을 쉽게 복구할 수 있도록 하는 파일 시스템을 설계해 보고자 하였다. 복구 기능을 응용 프로그램이 아닌 파일 시스템에서 제공함으로써 응용 프로그램의 개발이 쉬워지고 모든 편집용 프로그램에서 복구 기능을 사용할 수 있는 장점이 있다.

2. 관련 연구

2.1 응용 프로그램에서의 버전 관리

CVS[1]는 소프트웨어의 버전 관리를 위한 프로그램이다. CVS는 각 사용자가 프로그램의 사용법을 익혀야 하고 각 파일의 버전을 만들기 위해서는 사용자가 명시적으로 서버에 등록해야 하는 불편함이 있다.

Microsoft Word와 리눅스의 vi 에디터는 현재

작업중인 파일을 하드디스크 안에 임시 파일의 형태로 자동 저장한다. 이 프로그램들은 사용자가 자동 저장에 관여할 필요가 전혀 없다는 장점이 있다. 하지만 프로그램 내에서 구현된 기능이기 때문에 다른 프로그램에서 사용하는 파일에 대해서는 자동 저장을 할 수 없다.

2.2 파일 시스템에서의 버전 관리

Elephant 파일 시스템[2]은 가지고 있는 모든 파일들에 대해서 버전을 관리한다. 이 파일 시스템의 가장 큰 특징은 중요한 버전만을 골라서 저장할 수 있는 정책이 있다는 것이다.

Wayback[3] 파일 시스템은 파일 시스템에 대한 쓰기 작업이 있을 때마다 버전을 만든다. 이 파일 시스템은 사용자 레벨에서 구현되었기 때문에 커널 레벨에서 구현하는 것에 비해 속도가 느려지는 단점이 있다. 하지만 구현이 쉽고 다양한 종류의 기존 파일 시스템에 마운트하여 사용할 수 있다는 장점도 가지고 있다.

마지막으로, Ext3cow 파일 시스템[4]은 버전을 만들고 관리하는 별도의 프로그램을 사용한다. 이 파일 시스템은 버전을 만들기 위해 사용자가 직접 명령을 해야 하는 단점이 있다. 그리고 이전 버전을 찾기 위해서는 버전이 만들어진 시간에 대한 정보를 입력해야 하는 불편함도 가지고 있다.

본 논문에서는 파일의 복구 기능을 파일 시스템에서 구현하였다. 따라서 위에서 제시한 응용 프로그램에서의 복구 기능을 해당 응용 프로그램에서만 사용 가능하다는 단점을 극복할 수 있다. 또한 일반적인 모든 파일들이 아니라 현재 편집중인 파일들을 사용자의 명령 없이 자동으로 저장한다. 따라서 사용자가 미처 저장하지 못했던 파일들을 복구 가능하다.

3. 파일 시스템의 설계

이 장에서는 파일 시스템의 외부 인터페이스와 내부 자료 구조에 대해 정의한다. 그리고 내부 동작 메커니즘에 대해 설명한다.

3.1 외부 인터페이스 정의

3.1.1 모든 파일이 정상적으로 종료된 경우

모든 파일이 정상적으로 종료된 경우에는 모든 인터페이스가 일반 파일 시스템과 같다. 단 한 가지 다른 것은 사용자에게 파일 자동 저장 주기를 조절할 수 있는 기능을 제공하는 것이다. 파일 자동 저장을 자주 할수록 나중에 저장하지 않은 파일을 복구할 때에 시스템 이상 전의 파일에 더 근접하도록 복구할 수 있다. 하지만 자동 저장을 자주 실행하면 디스크로의 트래픽이 많이 발생하기 때문에 컴퓨터의 전체적인 속도가 느려질 수 있다. 그렇기 때문에, 파일 시스템을 설계할 때에 사용자가 파일의 자동 저장 주기를 설정할 수 있도록 하여, 되도록 많은 정보를 복구하고 싶은 사용자는 자동 저장을 자주 실행하도록 하고, 컴퓨터의 속도가 더 중요한 사용자는 자동 저장 주기를 늘릴 수 있도록 한다.

주기를 변경시키는 명령어의 형태는 setper <seconds>이다. 자동 저장 주기를 60초로 만들고 싶다면 터미널에서 다음과 같이 입력한다.

[root@whcho /root]# setper 60

3.1.2 임시 파일이 생성된 경우

일부 파일이 정상적으로 저장되지 못하거나 정상적으로 종료되지 못하면 임시 파일이 생성된다. 이 임시파일이 있으면 다음번에 해당 파일을 실행할 때 사용자에게 자동 저장된 임시 파일이 있다는 메시지와 함께 파일의 복구 여부를 묻는 메시지를 보여준다. 사용자는 'Y' 또는 'N'을 입력하여 파일의 복구 여부를 결정할 수 있다.

3.2 내부 자료 구조

3.2.1 편집중인 파일의 목록

편집중인 파일들을 주기적으로 저장하기 위해서는 현재 어떤 파일들이 편집 상태에 있는지 알아야 한다. 이를 위해 파일 시스템 안에 현재 편집중인 파일의 목록을 유지하도록 한다.

편집중인 파일을 목록에 추가하기 위해서는 각 파일을 편집하는 응용프로그램에서 편집중인 파일을 목록에 추가하는 API를 이용한다. 파일을 닫을 때에도 마찬가지로 응용 프로그램에서 API를 이용하여 닫으려는 파일을 목록에서 삭제하도록 한다.

위에서 언급한 두 API의 인자로는 파일의 이름과 위치해 있는 디렉토리 경로명이 필요하다. 한 디렉토리 안에 같은 이름을 가진 파일이 두 개 이상 존재할 수 없기 때문에 이 두 가지 정보만 가지고 있으면 어떤 파일인지 구별이 가능하기 때문이다. 따라서 각 API는 다음과 같은 형태를 갖는다.

addfile(<filename>,<full path of the directory>)

deletefile(<filename>,<full path of the directory>)

3.2.2 각 편집중인 파일에 대한 임시 파일

각 편집중인 파일은 setper 명령어에 의해 정해진 시간이 지나면 임시 파일에 저장된다. 이 임시 파일은 각 편집중인 파일이 있는 디렉토리와 같은 디렉토리에 하나씩 생성되며, 파일이 정상적으로 종료된 경우 삭제된다. 임시 파일의 이름은 편집중인 파일 뒤에 '.tmp'를 붙여 만든다.

편집중인 파일을 임시 파일에 저장할 때에는 파일을 통째로 임시 파일에 저장한다. 이렇게 하면 변경된 부분만을 저장하는 방법에 비해 디스크 공간을 많이 차지하게 되는 단점이 있다. 하지만 편집중인 모든 파일을 주기적으로 디스크에 저장하는 작업에 파일의 어느 부분이 변경되었는지 검색하는 작업까지 더하면 컴퓨터가 해야 할 작업이 너무 많아진다. 그리고 임시 파일들은 파일이 정상적으로 종료됨과 동시에 삭제되기 때문에 디스크에서 공간을 차지하는 비율이 매우 미미할 것이다.

편집중인 파일을 임시 파일에 저장할 때에는 이전에 저장했던 임시 파일이 있더라도 이전 기록을 보존하지 않고 그 위에 덮어쓰게 된다. 본 논문에서 제시하고자 하는 파일 시스템의 목적은 시간대별로 파일의 버전을 관리하여 나중에 되돌릴 수 있도록 하는 것이 아니라 파일이 정상적으로 저장되지 못한 상황에서도 최근의 파일로 되살리는 것이기 때문이다.

3.2.3 임시 파일의 존재 여부를 알리는 플래그

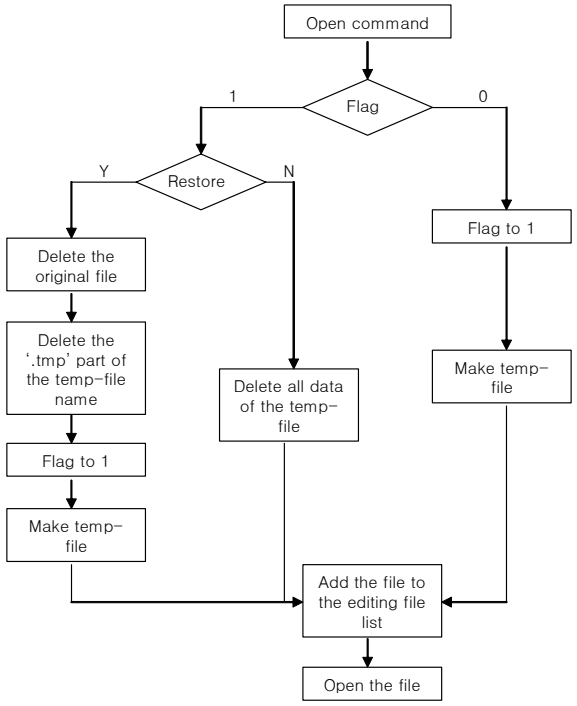
파일 시스템 내의 각 파일에는 하나의 플래그가 추가된다. 이 플래그는 해당 파일에 대하여 임시 파일이 존재하는지 아닌지를 알려준다. 사용자가 파일을 사용하기 위해 열 때 내부적으로 우선 이 플래그를 확인하여 임시 파일의 존재 여부를 판단할 수 있다. 이 플래그가 없다면 파일을 열 때마다 디렉토리 안의 모든 파일을 검색해서 임시 파일이 있는지 확인해야 할 것이다.

3.3 내부 동작 메커니즘

3.3.1 파일을 열 때

파일을 열 때에는 먼저 flag가 0인지 확인한다. 플래그가 0이면 파일이 정상적으로 저장되고 종료된 것이므로 임시 파일이 없다. 따라서 플래그를 1로 만들고 임시 파일을 생성한 후 열고자 하는 파일을 편집중인 파일 목록에 추가하면 된다.

플래그가 1이라면 정상적으로 저장되지 않은 임시 파일이 존재하는 것이다. 이 경우엔 사용자에게 복구할 것인지를 묻게 된다. 사용자가 복구를 하지 않으면 임시 파일의 모든 내용을 지우고 열고자 하는 파일을 편집중인 파일 목록에 추가한다.

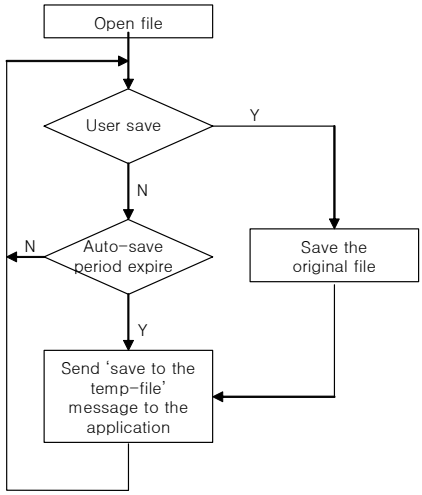


(그림 1) 파일을 열 때의 과정

사용자가 임시 파일의 복구를 원할 때에는 원래 열고자 했던 파일을 지우고 임시 파일의 이름에서 '.tmp' 부분을 지운다. 이렇게 하면 임시 파일이었던 것이 자연스럽게 원본 파일이 되면서 쉽게 복구가 된다. 그 다음 flag를 1로 만들고 새 임시 파일을 만든 후 편집중인 파일 목록에 추가한다. 이 과정을 그림으로 나타내면 그림 1과 같다.

3.3.2 파일을 편집할 때

파일을 편집할 때에는 두 가지 종류의 파일 저장이 실행된다. 하나는 사용자가 파일을 저장하는 경우이고 다른 하나는 파일의 자동 저장 주기만큼 시간이 흘러 임시 파일에 저장되는 경우이다. 사용자가 파일을 저장할 경우엔 원본 파일을 저장하면서 임시 파일에도 같이 저장을 해야 한다. 왜냐하면, 임시 파일이 더 오래된 파일 정보를 가지고 있다면 더 오래된 파일로 복구할 수도 있기 때문이다. 파일 자동저장 주기가 다 되었을 때에는 임시 파일에만 저장한다. 이 과정을 그림으로 나타내면 그림 2와 같다.



(그림 2) 파일을 편집할 때의 과정

3.3.3 파일을 닫을 때

파일을 닫을 때에는 먼저 임시 파일을 삭제한 후 원본 파일을 닫는다. 파일을 닫기 전에 flag를 0으로 만들고 편집중인 파일 목록에서 삭제하는 작업이 필요하다.

4. 파일 시스템의 구현

파일 시스템의 구현은 리눅스에서 많이 사용되는 Ext3 파일 시스템을 수정하여 구현한다. 리눅스 커널은 2.4.20-8 버전을 사용한다.

setper 시스템 콜의 경우 /usr/src/linux/include/asm-i386/unistd.h 파일과 usr/include/asm/unistd.h 파일, 그리고 /usr/src/linux/arch/i386/kernel/entry.S 파일을 수정하여 등록한다. 그리고 /usr/src/linux/fs/ 디렉토리에 setper.c 파일을 추가하여 기능을 구현하였다.

각 임시 파일에 대한 플래그는 /usr/src/linux/ext3_fs.h 파일에서 ext3_inode 구조체에 추가한다. 플래그는 많은 데이터를 저장하지 않으므로 정의된 데이터 타입 중 가장 작은 __u8을 사용한다.

편집중인 파일의 목록은 /usr/src/linux/ext_fs.h 파일의 ext3_super_block 구조체에 추가한다. 이 목록의 두 가지 필드인 파일 이름과 디렉토리 경로명은 각각 문자 32개, 64개의 크기이다. 수퍼블록의 여유 공간이 크지 않으므로 파일명의 길이를 32문자로 제한하였다. 또한 이 목록은 8개까지 만들 수 있다.

파일을 여는 명령을 내리면 /usr/src/linux/fs/open.c 파일에 있는 sys_open() 시스템 콜이 호출된다. 파일을 닫는 명령에는 같은 파일에 있는 sys_close() 시스템 콜이 호출된다. 따라서 플래그 변경과 편집중인 파일의 목록 수정 등 파일을 열고 닫을 때 필요한 작업들은 이 시스템 콜에서 작성한다.

5. 결론 및 추가 과제

컴퓨터에서 문서를 편집하던 중 정전이나 하드웨어 또는 소프트웨어의 문제로 인해 컴퓨터가 멈춰버리는 일이 종종 발생한다. 이때 사용자가 편집중이던 문서를 미리 저장해 놓지 않으면 편집된 내용을 잃어버리기 쉽다.

본 논문에서는 Ext3 파일 시스템을 이용하여 사용자가 편집하는 파일을 자동으로 저장하고 관리하는 파일 시스템을 설계하였다. 이 파일 시스템을 사용하면 편집중인 파일을 저장하지 못한 채 시스템이 종료된 상황에서도 편집된 파일의 내용을 쉽게 복구할 수 있다. 또한 파일 시스템에서 기능이 구현되기 때문에 응용 프로그램의 설계가 간단해진다. 그리고 모든 응용 프로그램에서 파일의 자동 저장 기능을 사용할 수 있다는 이점이 있다.

현재 연구의 진행 상황은 전체 시스템의 설계를 마무리하였고 각 API와 함수들, 그리고 테스트용 프로그램을 구현중이다. 앞으로는 구현을 마무리하고 파일 시스템의 크기와 속도에 대하여 성능 측정과 평가를 진행할 것이다.

참고문헌

[1]Dick Grune, Brian Berliner, Jeff Polk. Concurrent Versions System. Web site: http://www.cvshome.org/

[2]Douglas S. Santry, Michael J. Feeley, Norman C. Hutchinson, Alistair C. Veitch, Ross W. Carton and Jacob Ofir. Deciding when to forget in the Elephant file system. 17th ACM Symposium on Operating Systems Principles, 34(5), 110-123, Dec.1999.

[3]Brian Cornell, Peter A. Dinda and Fabian E. Bustamante. Wayback: A User-level Versioning File System for Linux. The 2004 USENIX Annual Technical Conference, FREENIX track, 19-28, June 2004.

[4]Zachary Peterson and Randal Burns. Ext3cow: A Time-Shifting File System for Regulatory Compliance. ACM Transactions on Storage, Vol.1, No.2, 190-212, May 2005.