Windows 2000 기반 가상 디스크의 기능 개선

백승식*°, 최수원**, 고정국*
*동명정보대학교 컴퓨터공학과
**㈜정소프트

Improvement of Windows 2000-based Virtual Disk

Seung-Sik Baek*, Su-Won Choi**, Jeong-Gook Koh*
*Dept. of Computer Engineering, Tongmyong University of Information Technology
**Jungsoft Co., Ltd.

요 약

일반적으로 사용자들은 효율적인 파일 관리를 위해 하드 디스크를 다수의 파티션으로 분할하여 사용하는데, 컴퓨터 사용에 익숙하지 않은 사용자들은 체계적인 파일 관리를 하지 못해 많은 애로를 겪기도 한다. 본 논문에서는 초보자나 파일 관리에 어려움을 겪는 사용자에게 편리한 파일 관리 기능을 제공하는 개선된 가상 디스크 프로그램을 구현하고 기능을 시험하였다. 가상 디스크를 사용하면 기존 디스크 파티션의 변경없이 별도의 디스크가 존재하는 것처럼 느끼게 되어 파일 관리에 많은 도움이 될 것이다.

1. 서론

PC에 윈도우가 채택되면서 컴퓨터에 익숙하지 않은 사람들도 컴퓨터를 쉽게 사용할 수 있게 되었다. 또한 저장 기술의 발전에 따라 디스크의 용량이 커지면서 대용량 디스크의 사용도 늘고 있다.

일반적으로 사용자들은 효율적인 파일 관리를 위해 하드 디스크를 다수의 파티션으로 분할하여 사용한다. 그러나 초보 사용자에게는 이러한 작업들이 어렵기 때문에 손쉬운 디스크 관리 방법이 필요하다. 또한, 하나의 컴퓨터를 여러 사람이 공동으로 사용하면서 체계적인 파일 관리가 되지 못할 경우 컴퓨터 사용에 많은 애로를 겪기도 한다.

이러한 문제에 대한 해결책으로 기존의 디스크 파티션은 변경하지 않으면서 파일 관리 기능을 제공하는 가상 디스크 프로그램을 개발하였다[1]. 본 논문은 기존 가상 디스크의 문제점들을 개선하여 구현하고 기능을 시험하였다.

본 논문의 구성은 다음과 같다. 2장에서는 가상 디스크에 관련된 기술을 기술하고, 3장에서는 가상 디스크의 기능 개선 및 구현 내역을 설명한다. 4장에서는 결론 및 향후 연구 방향에 대하여 기술한다.

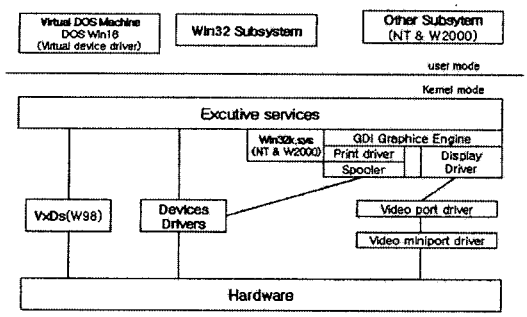
2. 관련 기술

2.1 윈도우 시스템

[그림 1]의 윈도우 시스템에서 응용 프로그램은 Win32에서 실행되거나 DOS/Win16 프로그램용 가상 DOS 머신에서 실행된다. 윈도우 NT/2000은 OS/2나 Posix 등의 서브 시스템도 지원한다.

커널 모드의 실행부(Excutive) 서비스는 커널의 다른 부분에 요청을 전달하기 전에 보안 확인과 인자 확인을 한다. 윈도우 NT/2000은 성능 향상을 위

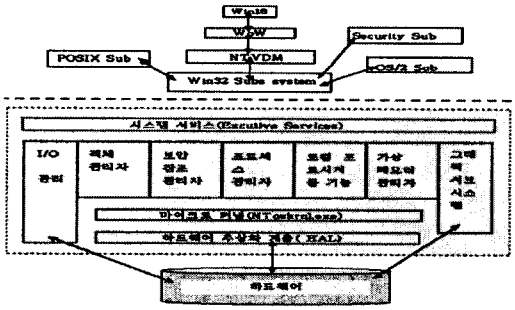
해 Win32기능의 일부를 커널에 구현하였다. 윈도우 95/98은 하드웨어와 통신하기 위해 VxD를 사용하지만, 윈도우 NT/2000은 NT형태의 디바이스 드라이버나 WDM드라이버를 사용하여 하드웨어와 통신한다[2].



[그림 1] 윈도우 시스템의 구조

2.2 윈도우 2000 시스템

윈도우 2000은 [그림 2]와 같이 하드웨어 추상화 계층(Hardware Abstraction Layer)과 실행부, 마이크로 커널로 구성되어 있다.



[그림 2] 윈도우 2000의 구조

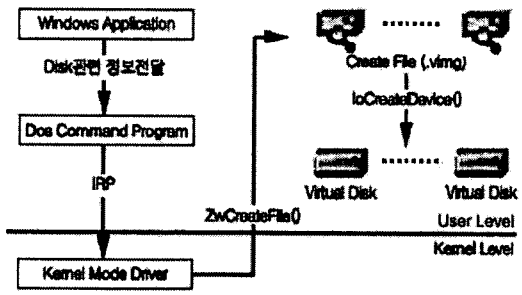
HAL 계층은 실행부와 하드웨어 플랫폼 사이에 위치하며, 하드웨어로부터 윈도우 2000의 나머지 부분을 격리시켜 시스템 이식을 용이하게 한다. 실행부는 시스템 서비스와 내부 루틴으로 구성되며, 응용 프로그램과 서브 시스템으로부터 보호된다. 커널 모드에서 실행되는 컴포넌트들은 하드웨어와 소프트웨어 자원을 직접 액세스하는데, 시스템의 보안과 안정을 위해 운영체제의 핵심 부분만 커널에서 실행한다. 마이크로 커널은 실행부의 일부로서 실행부와 저수준의 운영체제 프리미티브를 통해 통신한다[3,4].

3. 가상 디스크의 기능 개선 및 구현

3.1 가상 디스크의 기능 개선

개선된 가상 디스크 프로그램은 윈도우 2000/XP Professional에서 실행되며 가상 디스크 이미지 파일을 이용하여 가상 디스크를 생성하고 삭제할 수 있다.

가상 디스크 생성 방식은 [그림 3]과 같다. 새로운 가상 디스크를 생성하거나 생성된 가상 디스크를 개방하려면 가상 디스크 프로그램에서 "가상 디스크 생성" 항목을 선택한다. 가상 디스크 프로그램은 디스크 생성 정보를 DOS 명령어 프로그램에게 전달하고, DOS 명령어 프로그램은 디스크 생성 정보를 수록한 IRP(I/O Request Packet)를 커널내의 가상 디스크 구동기에게 전달한다. 가상 디스크 구동기는 IRP를 수신한 후 가상 디스크 이미지 파일을 생성/개방하기 위해 ZwCreateFile 루틴을 통해 파일 핸들을 개방한다. 그 후에는 IoCreateDevice 루틴을 이용하여 가상 디스크를 생성한다. 가상 디스크 구동기는 가상 디스크를 제거하거나 시스템 종료전까지 생성된 가상 디스크의 이미지 파일을 이용하여 가리키게 된다.



[그림 3] 가상 디스크의 생성 방식

그러나 기존의 가상 디스크는 시스템이 재시작될 때 가상 디스크를 생성하지 않기 때문에 사용자가 가상 디스크를 다시 생성해야만 하였다. 또한 생성가능한 가상 디스크 수도 한 개뿐이므로 가상 디스크를 여러 개 생성할 수가 없었다.

개선된 가상 디스크는 이러한 문제점을 보완하기 위해 "마운트(Mount) 테이블" 기능을 제공한다. 즉, 마운트 테이블을 이용하여 시스템이 재시작될 때 가상 디스크를 재생성할 수 있다. 윈도우가 시작되면 시스템 종료전의 가상 디스크 정보를 ZwCreatrFile 루

틴에 제공하여 핸들을 개방하고, IoCreateDevice 루틴을 통해 이미지 파일을 가리키게 하여 가상 디스크를 복원한다. 또한 가상 디스크 생성시 고유 번호를 부여하여 다수의 가상 디스크를 생성할 수 있다.

개선된 가상 디스크도 기존 가상 디스크처럼 윈도우가 초기화된 후 가상 디스크 구동기의 서비스를 시작한다. 그러나 기존 가상 디스크는 시스템 종료 및 재부팅시 가상 디스크 구동기의 서비스도 중단되어 가상 디스크가 제거되었다. 다만, 가상 디스크의 데이터는 이미지 파일에 이진 데이터 형태로 저장되므로 이미지 파일을 삭제하지 않는다면 디스크에 존재한다. 따라서 기존의 가상 디스크에서는 시스템이 재시작되면 가상 디스크를 다시 생성해야 하는 단점이 있다. 그리고 하나의 가상 디스크만 생성할 수 있기 때문에 가상 디스크간 데이터 복사와 이동은 불가능하였다.

개선된 가상 디스크는 이러한 단점을 보완하여 다수의 가상 디스크를 생성하고, 가상 디스크간의 파일 이동도 가능하다. 또한 시스템 종료시 가상 디스크 정보를 자동으로 저장하므로 시스템 재시작 후 가상 디스크를 사용할 수도 있고, 다수의 가상 디스크 중에서 원하는 디스크만 삭제할 수도 있다.

3.2 개선된 가상 디스크의 구현

개선된 가상 디스크의 구현 및 기능시험 환경은 다음과 같다.

1) 운영체제
   - Windows XP Professional (구현/기능시험)
   - Windows 2000 Professional (기능시험)
   - Windows 2000 Server (기능시험)
2) 개발도구
   - Visual Studio 6.0, Windows DDK 2600, C++ Builder 6
3) 디버깅도구
   - WinDbg6.0, DebugView 4.13, SoftICE 2.5

[그림 6]에서는 기존 가상 디스크와 개선된 가상 디스크에서 이미지 파일을 생성하기 위해 ZwCreateFile 루틴을 호출하는 코드를 비교하였다.

기존 가상 디스크의 VdiskOpenFile() 함수

```
NTSTATUS
VDiskOpenFile (IN PDEVICE_OBJECT DeviceObject, IN PIRP Irp)
{
    ...
    status = ZwCreateFile(
        &device_extension -> file_handle,
        device_extension->read_only ?      GENERIC_READ :
            GENERIC_READ | GENERIC_WRITE,
        &object_attributes,
        &Irp -> IoStatus,
        NULL,
        FILE_ATTRIBUTE_NORMAL,
        0,
        FILE_OPEN,
        FILE_NON_DIRECTORY_FILE |
        FILE_RANDOM_ACCESS |
        FILE_NO_INTERMEDIATE_BUFFERING |
        FILE_SYNCHRONOUS_IO_NONALERT,
        NULL,
        0 );
...
```

개선된 가상 디스크의 VdiskOpenFile() 함수

```
NTSTATUS
VDiskOpenFile(IN PDEVICE_OBJECT     DeviceObject, IN PIRP Irp)
{
    ...
    status = ZwCreateFile(
        &device_extension->file_handle,
        device_extension->read_only ? GENERIC_READ:
            GENERIC_READ | GENERIC_WRITE,
        &object_attributes,
        &Irp->IoStatus,
        NULL,
        FILE_ATTRIBUTE_NORMAL,
        device_extension->read_only ? FILE_SHARE_READ : 0,
        FILE_OPEN,
        FILE_NON_DIRECTORY_FILE |
        FILE_RANDOM_ACCESS |
        FILE_NO_INTERMEDIATE_BUFFERING |
        FILE_SYNCHRONOUS_IO_NONALERT,
        NULL,
        0);
...
```

[그림 6] 가상 디스크 이미지 파일 생성 코드 비교

입출력 관리자는 장치 구동기가 시스템에 적재될 때 장치 구동기 객체를 생성하고 구동기의 초기 루틴을 호출한다. 적재된 장치 구동기는 장치 객체를 생성하기 위해 IoCreateDevice 루틴을 호출한다. [그림 7]에서는 기존 가상 디스크와 개선된 가상 디스크에서 가상 디스크 장치를 생성하기 위해 IoCreateDevice 루틴을 호출하는 코드를 비교하였다.

기존 가상 디스크의 VDiskCreateDevice() 함수

```
status = IoCreateDevice(
        DriverObject,
        sizeof(DEVICE_EXTENSION),
        &device_name,
        FILE_DEVICE_DISK,
        0,
        FALSE,
        &device_object
        );
    ...
```

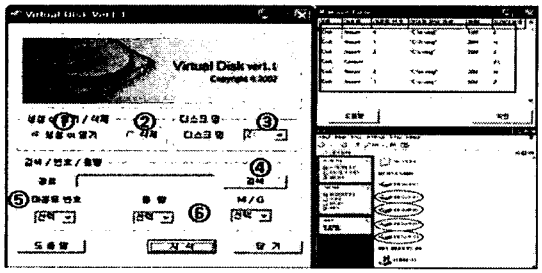
개선된 가상 디스크의 VdiskCreateDevice() 함수

```
status = IoCreateDevice(
        DriverObject,
        sizeof(DEVICE_EXTENSION),
        &device_name,
        DeviceType,
```

```
        0,
        FALSE,
        &device_object
        );
    ...
```

[그림 7] 가상 디스크의 장치 생성 코드 비교

구현된 프로그램의 활용법은 다음과 같다. 가상 디스크 생성시 [그림 8]의 가상 디스크 프로그램을 구동시킨 후 번의 "생성 or 열기"를 선택하고 번의 "디스크 명"을 지정한다. 그리고 가상 디스크 이미지 파일이 저장될 경로를 번의 "검색" 버튼으로 지정하고 마운트 테이블을 참조하여 번의 "마운트 번호"가 중복되지 않게 선택하고 번에서 가상 디스크 "용량"을 지정한다. 끝으로 "시작" 버튼을 누르면 가상 디스크가 생성된다. 본 논문에서는 개선된 가상 디스크의 기능 시험을 위해 4개의 디스크(E:, H:, M:, Z:)를 생성하였으며, [그림 8]의 오른쪽 하단 부분은 가상 디스크 생성 결과를 보여준다.



[그림 8] 개선된 가상 디스크의 기능 시험

한편, 가상 디스크 생성시 마운트 번호가 중복되지 않도록 하기 위해 [그림 8]과 같이 ComboBox를 이용하여 마운트 번호를 선택하게 하였다. 또한, 잘못된 마운트 번호를 선택하면 오류 메시지가 출력되는 중복방지 검사코드를 [그림 9]와 같이 삽입하였다.

```
if(strStore3 == " 0")
{
    StorageTemp_3 = 0;
}
else if(strStore3 == " 1")
{
    StorageTemp_3 = 1;
}
else if(strStore3 == " 2")
{
    StorageTemp_3 = 2;
}
else if(strStore3 == " 3")
{
    StorageTemp_3 = 3;
}
else
{
    MessageBox("\" 마운트 번호 \" - 확인하세요 !", "알림",
    MB_ICONSTOP|MB_ICONERROR|MBICONHAND|MB_OK);
}
```

[그림 9] 마운트 번호 중복 방지 검사 코드

가상 디스크를 제거할 때는 [그림 8]에서 번 항목인 "삭제"를 선택하고, 삭제 대상을 번의 "디스크 명"으로 지정한 후 "시작" 버튼을 누르게 된다.

4. 결론

본 논문에서는 편리한 파일 관리 기능을 제공하는 개선된 가상 디스크 프로그램을 구현하고 기능을 시험하였다. 가상 디스크를 사용하면 디스크 파티션의 변경없이 별도의 디스크가 존재하는 것처럼 느끼게 되어 파일 관리에 도움이 될 것이다. 또한, 기존 가상 디스크의 문제점을 개선하여 다수의 가상 디스크를 생성하거나 삭제할 수 있으며, 시스템이 재시작되어도 가상 디스크가 유지되도록 하였다.

향후 연구 방향은 가상 디스크 이미지 파일의 암호화 기능을 제공할 예정이다. 또한 윈도우 파일시스템 외에도 리눅스 파일시스템을 생성할 수 있도록 하여 리눅스 사용자도 편리하게 사용하도록 할 예정이다. 그리고 가상 CD-ROM 생성 기능도 제공하여 가상 CD-ROM 이미지 파일도 저장할 수 있도록 구현할 계획이다.

[참고문헌]

[1] 최수원, 고정국, "Windows NT 기반의 가상 디스크 설계 및 구현", 한국정보처리학회 추계학술발표논문집, 제9권 제 2호, 2002. 11

[2] Chris Cant, Writing Windows WDM Device Drivers, 에이콘출판사, 2000

[3] Art Baker Jerry Lozano, The Windows 2000 Device Driver Book 2nd Ed., Prentice-Hall Computer Books, 2001

[4] Walter Oney, Programming the Windows Driver Model, 정보문화사, 2001

[5] 박광우 외, Visual C++.NET Programming Bible, 삼양미디어, 2002

[6] 이상엽, Visual C++ Programming Bible Ver 6.x, 영진출판사

[7] Windows System Programmer, http://cafe.daum.net/programmer

[8] DevPartner/Driver Studio, http://www.devpia.com