

## 2장. 윈도우즈와 드라이버 맛보기

### (Flavors of Windows and Drivers)

오랫동안의 시스템 프로그래머로서, 필자는 작업에 착수하는데 있어 실질적인 세부항목에 집중하기를 좋아한다. 예나 지금이나, 작업대상 시스템이 운영체제의 구성 중 어느 부분에 가장 잘 맞아떨어지는지를 이해했을 때, 가장 효과적인 결정을 내릴 수 있다고 생각한다. 이 장에서는 독자들이 벌써 알고 있을 만한 것에 대하여 다시 살펴볼 것이다.

윈도우즈 95는 윈도우즈 NT로 가는 경로의 몇 단계 중의 하나이다. 이 윈도우즈 NT는 마이크로소프트가 데스크탑 컴퓨터의 표준으로 자리 잡기를 희망하는 운영체제이다. 윈도우즈 NT가 아닌 다른 윈도우즈 버전에 있어 시스템 프로그래밍이란 가상 디바이스 드라이버(Virtual Device Driver; VxD)제작인 32비트 프로그래밍과 0순위 권한 모듈(ring-zero module) 제작이란 의미가 점점 증가되어 왔다. 원래 VxD란 MS-DOS 드라이버나 윈도우즈 DLL 드라이버 같은 3순위 권한 모듈(ring-three module)을 위해 하드웨어를 *가상화(virtualizing)*하는 일을 맡고 있다. 윈도우즈 95에서 VxD는 하드웨어에 대하여 일차적인 인터페이스를 제공하며 다양한 응용 프로그램이 필요로 하는 소프트웨어 서비스를 제공한다.

---

**0순위 권한 소프트웨어(ring-zero software)**는 인텔 프로세서에서 제공하는 가장 높은 특권 레벨을 소유하고 동작하는 소프트웨어이다.

**3순위 권한 소프트웨어(ring-three software)**는 인텔 프로세서에서 제공하는 가장 낮은 특권레벨을 소유하고 동작하는 소프트웨어이다.

---

필자의 의견으로는, 윈도우즈 3.0은 견고하고 능력이 충분해서 응용 프로그램 플랫폼으로써 깊은 주의를 끌었던 윈도우즈 최초의 버전이었다. 윈도우즈 3.0은 오랜 기간 동안 장황하고 대대적인 광고를 한(그러나 그 내용은 아마도 비밀이었을 것이다) 이후인 1990년 베타 프로그램을 공개했다. 당시 인텔의 80386 기계는 데스크탑 컴퓨팅에 있어 최신이었으며 향상모드(enhanced mode, 80386 프로세서와 같이 소개되었다)에서 실행하는 것은 인텔에서 제시한 규칙보다 더 많은 예외적인 경우를 사용하고 있었다. 윈도우즈 3.0은 표준모드(standard mode)에서 최고의 성능으로 실행되었으며 (비록 80386 기계라고 하더라도), 계속해서 고색 창연한 오래된 리얼모드(원래의 8086 프로세서는 이 모드만 사용 가능하다)를 지원하였다. 마이크로소프트는 아무도 당황해 하거나 애들하게 하지 않고서도 윈도우즈 3.1에는 리얼모드를 떼어 버렸다. 더구나 윈도우즈 95는 표준모드를 떼어 버렸다. 이제 윈도우즈 95는 향상모드에서만 실행되기 때문에 최소한 80386 프로세서 이상이 필요하다.

---

#### 리얼모드(real mode)

80x86 프로세서의 디폴트 실행모드로써, 한번에 한 개의 프로그램만 지원하는 환경을 응용 프로그램에게 제공하며 시스템 메모리와 IO 디바이스 액세스를 자유롭게 할 수 있다.

---

## 1.1 윈도우즈 시스템의 분류

### (A Taxonomy of Windows Systems)

윈도우즈 95는 운영체제의 생태에 있어 특별한 위치를 차지하고 있다. 윈도우즈 체계는 그 자체적으로 다양한 능력 수준의 마이크로프로세서에 따라 여러 가지가 있다. 윈도우즈 NT는 최고의 시스템을 그 대상으로 한다. 윈도우즈 3.x는 80386에서도 잘 동작한다. 윈도우즈 95는 오늘날의 보통의 데스크탑 시스템(대략 인텔 i486 프로세

서와 8MB의 메모리)을 그 대상으로 한다. 그러나 마이크로소프트는 80386에 4MB 메모리를 가진 시스템에서도 동작한다고 믿고 싶어한다.

## (1) 윈도우즈 NT (Windows NT)

윈도우즈 NT는 인텔 프로세서를 바탕으로 하는 데스크탑 컴퓨터에서 실행되는 운영체제 중 가장 강력하다. 그러나 또한 잘 돌리기 위해서는 강력한 플랫폼이 필요하다. (필자는 여기서 OS/2가 미래의 대세가 될지도 모른다는 주장의 허점을 공격하려고 하는 것은 아니다. 그러나 개인적으로 OS/2가 대세가 되지 못한다고 생각한다. 오래 전 OS/2 카드놀이(Solitaire) 게임의 그래픽을 한번보고 필자의 마음은 이미 굳어졌다.) 윈도우즈 NT는 몇 가지 점에서 다른 윈도우즈와 구별된다. 우선 윈도우즈 NT에는 보안 시스템이 있다. 만약 하드디스크가 윈도우즈 NT의 관리하에 있다면 프로그램이나 데이터를 불법적으로 액세스하려고 하는 것을 방지할 수 있다. 또한 윈도우즈 NT는 마이크로소프트의 운영체제 중 유일하게 다른 마이크로프로세서 구조에 이식되었다. (이 말은 맞는 말이기도 하지만 틀리는 말이기도 하다. 윈도우즈 NT가 일반적인 데스크 탑 컴퓨터에 모두 이식 될 수 있는 것은 아니다. 얼마 전에 마이크로소프트에서 Power PC의 NT 이식을 포기한다는 뉴스가 그 좋은 일례일 것이다)

윈도우즈 NT는 32비트 Win32 API(Application Programming Interface)를 소개했다. 만약 개발자가 이 API를 이용해 응용 프로그램을 개발한다면 윈도우즈 NT가 운영되는 어떠한 마이크로프로세서라도 간단히 컴파일만 새로 해서 이식시킬 수 있다. Win32 API는 그래픽을 사용하지 않는 비만한 MS-DOS 프로그램과 구분할 수 없을 정도로 비슷한 문자모드의 윈도우즈 응용 프로그램뿐만 아니라 그래픽컬한 응용 프로그램도 지원한다. Win32 API와 다른 윈도우즈 API의 중요한 차이점은 그것이 32비트라고 하는 것이다.

---

### 설명 (Note)

물론 API는 API의 구체적인 수행과는 기술적으로 구별되는 추상적인 개념이다. 그래서 재능이 뛰어나고 시간이 남아도는 어떤 사람은 Win32 API가 16비트에서 실행 되도록 하는 돈키호테식의 무모한 방법을 만들기도 했었다.

---

윈도우즈 API는 대단히 풍부하고 다양하다. 여기에 대한 최초의 책은 두 권의 *Microsoft Win32 Programmer's Reference (Microsoft Press, 1993)*이며 총 85장으로 구성되어 있었다. 각 장은 API를 분류하고 종합적으로 설명한다. 이 중 27장은 "System Service"라는 제목으로 되어 있는데 Win32 프로그래머가 사용할 수 있는 서비스가 몇 가지 있다. 이것은 가상 메모리 관리, 프로세스 및 스레드 생성과 관리, 4가지의 서로 다른 동기화 프리미티브, 3가지의 서로 다른 파일 시스템, 3가지 이상의 네트워크를 통한 통신방법 등등이 그것이다.

윈도우즈 NT는 5년 전 처음으로 소개되었으나 너무 느려서 주목을 받지 못했다. 이 느리다는 것에는 두 가지 요인이 있는 것 같다. 우선 성능 문제이다. 윈도우즈 NT 3.5는 초기의 3.1 버전보다 훨씬 좋아지긴 했지만, 아직도 강력한 컴퓨터가 필요하다. 어제의 최신 컴퓨터가 오늘은 보통의 컴퓨터가 되고 내일은 저 물밑에서 쉬고 있는 닢이 되어 있을 것이다. 따라서 성능 지표의 중요성은 시간이 지남에 따라 줄어들 것이다.

두 번째 요인은 윈도우즈 NT의 출발이 느렸다는 것이며, 따라서 필요로 하는 32비트 응용 프로그램이 많지 않았다는 것이다. 대부분의 윈도우즈 응용 프로그램은 16비트였으며, 윈도우즈 NT는 솔직히 윈도우즈 3.1처럼 이들을 잘 실행시켜 주지 못하는 것이 많았다. 마이크로소프트는 윈도우즈 3.1을 위한 다소 절름발이의 Win32s 라고 하는 부속 시스템을 고안했는데, 이것은 응용 프로그램 제작자들이 32비트 응용 프로그램을 만들도록 유혹하기 위함이었다. Win32s 아이디어의 뒤에는 윈도우즈 NT와 윈도우즈 3.1 모두를 목표로 Win32 응용 프로그램을 개발할 수 있도록 하자는 데 있었다. 그러나 제약조건으로 인해 호환은 불가능했고 눈에 뻘히 보이는 버그는 소프트웨어 제작자들에게 Win32s가 호소력이 없게 만들었다. 따라서 32비트 응용 프로그램을 위한 이 생명력 없고 싸구려 진열용 플랫폼으로 인해 제작자들은 32비트 응용 프로그램 개발을 기피했다.

## (2) 윈도우즈 3.1 (Windows 3.1)

성능 순위에 있어서 윈도우즈 3.1은 윈도우즈 NT 바로 아래에 있다. 윈도우즈 3.1은 요즘의 저수준에 속하는 80386 컴퓨터를 위해 설계되었으며, 32비트 운영체제로서 16비트 응용 프로그램, MS-DOS 응용 프로그램, 확장

DOS 응용 프로그램을 실행시킬 수 있다.

---

#### **확장 DOS 응용 프로그램 (extended DOS applications)**

이것은 도스 확장자를 바탕으로 만들어지는 16비트나 32비트 프로그램이다. 따라서 MS-DOS와 BIOS 서비스 시스템에 의존하면서도 보호모드에서 실행할 수 있다. 윈도우즈 그 자체 - 응용 프로그램에서 사용하는 KERNEL, USER, GDI 모듈 - 는 이런 면에서 볼 때 실제로는 16비트 확장 도스 응용 프로그램이다.

#### **보호모드 (protected mode)**

인텔 80286 이상의 프로세서의 운영체제로써 멀티테스킹, 데이터 보호, 가상 메모리 등을 지원한다.

---

윈도우즈 3.1 응용 프로그램은 윈도우즈 시스템과의 연결에 16비트 API를 사용하며, MS-DOS와 BIOS 연결에 소프트웨어 인터럽트를 사용한다. 프로그래머는 사실 보통의 MS-DOS 응용 프로그램에 사용되는 같은 런타임 라이브러리를 사용하며 16비트 윈도우즈 프로그램을 만든다. 윈도우즈 3.1 응용 프로그램을 관리하는 32비트 운영체제의 주요 목적은 보호모드에서 발생하는 소프트웨어 인터럽트를 처리하는 것과 이들을 지능적으로 리얼모드 인터럽트로 바꾸어 리얼모드에 있는 MS-DOS와 BOOS에게 전달하는 것이다.

윈도우즈 3.1에는 많은 문제가 있다. 가장 커다란 문제중의 하나는, 공동으로 동작하는 멀티테스킹을 지원하기 위해 제어권이 윈도우즈 시스템으로 되돌아 갈 수 있도록 정기적으로 컴퓨터 제어권을 양보해야 한다는 것이다. 잘못 동작하는 응용프로그램은 기계를 폭주시킬 수 있다. 어떤 종류의 응용 프로그램- 두 가지를 들어, 계산이 많은 프로그램과 통신을 통한 상호처리에 의존하는 프로그램- 은 이 양보에 대한 규칙을 위반하지 않고서는 프로그램 작성하기가 대단히 어렵다. 윈도우즈 3.1의 문제는 USER와 GDI 모듈 같은 시스템 핵심 요소들이 제한된 힙 공간을 가지고 있다는 것이다.

시스템 수준에서 본다면 윈도우즈 3.1은 MS-DOS에 재진입하지 않는다는 것, 리얼모드나 가상 8086(Virtual 8086; V86)모드에서만 실행한다는 것, 하위 1MB의 메모리만 액세스 할 수 있다는 사실 등과 계속해서 충돌을 일으킨다. 이러한 사실은 중요한데, 왜냐하면 윈도우즈 3.1은 디스크 드라이버 액세스와 파일 시스템에 MS-DOS와 BIOS에 의존하기 때문이다. 이렇게 리얼모드 소프트웨어에 의존한다는 제한 사항으로 인해 다음과 같은 문제가 발생한다.

- 만약 사용자가 하드디스크에 영구적인 스왑공간을 만들어 놓지 않으면 윈도우즈 3.1은 MS-DOS나 BIOS가 페이지 I/O를 수행해야 하기 때문에 단일 쓰레드가 필요하다.
- CD-ROM을 액세스하는데 MSCDEX TSR이 필요하다. 이것은 일종의 피어 네트워킹 소프트웨어를 지원하는 인터페이스이다.
- 윈도우즈가 디스크를 액세스하는데 피어 네트워킹 서버 인터페이스를 사용함으로 인해 서버 기계는 영구 스왑파일을 사용할 수 없고 영구 스왑파일을 통한 빠른 페이지를 사용할 수 없는 결과가 된다.
- 파일 시스템 I/O는 보호모드에서 V86모드로의 전환이 필요하며 어드레스 영역에서 하위 1MB에 버퍼를 할당해 놓아야 한다.
- 여러 개의 리얼모드 드라이버는, 특히 네트워크 연결을 관리하는 드라이버는 하위 1MB를 매우 많이 사용하기 때문에 간혹 어떤 응용프로그램은 실행시킬 수 없을 때가 있다.

---

#### **재진입 가능 코드(reentrant code)**

다른 쓰레드에 의해 정지되거나 실행할 수 있는 프로그램 코드

---

### **(3) 윈도우즈 95(Windows 95)**

윈도우즈 95는 윈도우즈 3.1에서 야기된 여러 가지 문제들을 겨냥했고 사용자에게 친근한 컴퓨팅을 발전시켜 독창성을 부여하도록 만들었다. 사용자(end user)입장에서 볼 때 윈도우즈 95는 긴 파일 이름, 도킹 태스크 바, 좀더 통합된 인터페이스 셸, 시작메뉴에 문서항목 인터페이스 추가 등의 많은 편리한 요소들이 추가되었다. 비록 이런 것들이 윈도우즈 3.1과 같은 가상 머신 기술을 바탕으로 하고 있지만 윈도우즈 95는 32비트 파일시스템, 32비트

디스크 드라이버, 32비트 네트워크 인터페이스 등을 포함하고 있다. 이러한 기술들은 윈도우즈 NT와 같은 프로세스, 쓰레드 모델을 사용하여 Win32응용프로그램에게 선점적 멀티태스킹을 지원한다. 플러그 앤 플레이 하부 시스템은 사용자(그리고 필자 같이 손재주 없는 시스템 프로그래머)들이 새로운 하드웨어 추가를 쉽도록 해 주었다.

필자는 윈도우즈 95에 있어서 두드러진 양상을 발견했는데 그것은 오래된 소프트웨어와의 호환성 측면이다. 필자가 항상 즐기는 토막만화는 이를 명쾌히 보여준다. 이 토막만화의 첫 번째 칸에는 미군병사 '씩'은 사병계시관을 읽기 위해 멈추어 선다. 여기에는 메모들로 가득 덮여 있다. 다음 칸에는 '씩'이 해가 들지 않는 겹겹이 쌓인 종이를 뒤적거리다. 이 계시관에는 분명히 버려진 메모가 하나도 없었다. 다음의 그림 2-1과 같이 토막만화의 마지막 칸에는 '씩'이 성냥불을 들고서 조지 워싱턴장군이 델라웨어 강을 건너라고 내린 명령을 읽고 있다.



그림 2-1. 윈도우즈 95의 호환성

즉, 윈도우즈 95도 이와 같다. 소프트웨어에 있어서 델라웨어 강을 횡단한다는 것이란 PSP(Program Segment Prefix) 데이터 영역과 비슷하다고 할 수 있다. PSP는 MS-DOS가 응용프로그램(또한 매우 중요한 것으로 파일 핸들 세트)을 각각 구별하는데 사용한다.

윈도우즈 95에서는 비록 Win32응용프로그램이라 하더라도 PSP를 가지고 있으며, MS-DOS가 사용할 수 있도록 하위 1MB안에 있다. 오래 전의 응용프로그램에게 중요한 MS-DOS, BIOS, 윈도우즈 그 자신의 어떤 요소는 여전히 살아남아 있으며 윈도우즈 95에서 잘 동작된다. 예를 들면 다음과 같다.

- 윈도우즈 95는 네트워킹 작업을 수행하는데 있어 가능한 완전히 새로운 32비트 드라이버를 사용한다. 그러나 리얼모드 네트워크 드라이버 밖에 없다면 이것도 사용할 수 있다.
- 윈도우즈 95는 모든 하드웨어를 액세스하는데 완전히 새로운 32비트 드라이버를 사용하거나 필요에 따라서 구식 리얼모드 드라이버를 사용한다. 가능한 예 중에서 하나를 들자면, 필자의 시스템에는 CD-ROM과 하드디스크가 있는데 여기에 일부러 리얼모드 드라이버를 설치한 적이 없다. 이때 윈도우즈 95가 시작하기 전까지 이 디스크들을 액세스할 수 없으며, 그 뒤 윈도우즈 95가 자동으로 하드웨어를 검출하고 이를 시스템에 맞도록 설정시키는 등 분명히 요청하지도 않은 32비트 드라이버로 시작한다.
- 16비트 윈도우즈 응용 프로그램은 가끔 서로간에 공유 메모리와 이전 버전의 윈도우즈의 멀티태스킹 모델 등에 의존하기 때문에, 윈도우즈 95는 16비트 프로그램을 근본적으로 윈도우즈 3.1이 했던 것과 같은 방

법으로 실행한다.

- 중요한 16비트 상용 프로그램들이 윈도우즈 버전을 부적절한 방법으로 테스트하기 때문에, 윈도우즈 95는 이들을 실행시키기 위해 또한 부적절한 방법을 사용한다.

이 호환성은 시스템 프로그램을 작성하려는 사람들에게 특별한 문제를 야기 시킨다. 일면으로 볼 때, 32비트의 가상 디바이스 드라이버 형태로 운영체제로 확장하기가 상대적으로 쉽다. 다른 면에서 볼 때, 리얼모드 MS-DOS 응용 프로그램, 확장 DOS 응용 프로그램, 16비트 윈도우즈 응용 프로그램, 32비트 응용 프로그램들이 모두 각각 충분히 성능을 발휘하도록 폭 넓은 시스템 프로그램을 만들기가 대단히 어렵다. 예를 들어, 만약 디스크 디바이스를 지원하고 있다면 IOS(Input/Output Supervisor) 아키텍처에 산뜻하게 집합시켜주는 레이어 드라이버를 제공하는 것으로도 충분하겠지만, 그러나 일반적인 면에서 볼 때, 하드웨어를 지원하는 것은 가끔 V86 프로그램으로부터 트랩과 시뮬레이트 액세스가 필요하며 이러한 작업들은 다소 복잡하다.

## 2.2 윈도우즈 디바이스 드라이버의 역사 요약 (A Brief History of Device Drivers for Windows)

시스템 프로그래머가 수행하는 많은 작업들은 특정한 하드웨어의 디바이스 드라이버를 궁리해내서 작성하는 일이다. 보편적으로 볼 때 이러한 작업들이 윈도우즈 95에서는 이전 버전의 윈도우즈 비해 더 쉽다. 디바이스 드라이버 프로그래밍에 있어서 그 역사를 이해한다면 많은 도움이 될 수 있다.

### (1) 리얼모드 윈도우즈 (Real-Mode Windows)

MS-DOS와 시스템 BIOS는 처음부터 많은 하드웨어 디바이스에 대한 드라이버를 제공한다. BIOS는 몇 가지 잘 알려진 소프트웨어 인터럽트를 통해서 드라이버 서비스를 익스포트 한다. 여기에는 비디오 서브 시스템을 위한 INT 10h, 디스크 서브 시스템을 위한 INT 13h, 키보드를 위한 INT 16h가 있다. BIOS는 또한 하드웨어 인터럽트를 처리하고 PIC(Programmable Interrupt Controller)를 관리하는 책임을 진다. 또한 MS-DOS도 소프트웨어 인터럽트 21h, 25h, 26h등과 같은 방법을 통해 시스템 서비스를 익스포트하며, 어떤 메커니즘(CONFIG.SYS에 device= 문장)을 제공해서 새로운 드라이버나 임포트 된 드라이버가 시작 시에 로드 될 수 있도록 한다. 랠프 브라운(Ralf Brown)과 짐 카일(Jim Kyle)의 *PC Interrupts(Addison-Wesley, 1994)*는 이러한 MS-DOS와 BIOS 인터페이스에 대하여 포괄적인 논의를 하고 있으며, 수년동안 많은 PC 프로그래머들에게 인기가 있어 왔다.

### (2) 표준 모드 윈도우즈 (Standard-Mode Windows)

옛날의 윈도우즈에서는 MS-DOS와 BIOS가 최고로 중요했었다. 윈도우즈는 MS-DOS에다가 그래픽 오버레이를 제공하는 순전한 리얼모드 "운영 환경(operating environment)" 이었다. 시스템 수준에서 볼 때, 윈도우즈는 커다란 그래픽 프로그램 정도에 지나지 않았다. 인텔 80286 프로세서의 출현은 윈도우즈가 보호모드에서 실행될 수 있게 해 주었기 때문에 무리 메모리의 16MB까지 액세스할 수 있었다. 프로세서가 보호모드로 진입하고 혹은 보호모드에서 빠져 나오는 등의 모드전환을 통하여 윈도우즈는 계속적으로 필요로 하는 MS-DOS와 BIOS를 사용할 수 있었다. 이 운영모드는 나중에 **표준 모드(standard mode)**라 알려졌다.

80286 프로세서에서 리얼모드와 보호모드 전환간에는 끔찍할 정도로 비싼 대가를 필요로 했다. 인텔은 한가지 모드로의 전환만을 제공했는데(리얼모드에서 보호모드로 전환하는 것이 상대적으로 많이 빨랐음), 이것은 아무도 리얼모드로 되돌아가지 않으리라고 확실히 단정 지었었기 때문이다. 따라서 표준모드 윈도우즈 같은 보호모드 프로그램이 MS-DOS 같은 리얼모드 소프트웨어를 액세스하기 위한 유일한 방법은 프로세서를 리셋 하는 방법이었다. 리셋 신호를 발생시키기 위한 유명한 방법들로써는 보통 Ctrl-Alt-Delete가 눌러 졌을 때 발생하는 것과 같은 외부 신호를 발생시키도록 키보드 콘트롤러를 자극하는 방법과 애초부터 처리 불가능한 "3중 폴트(triple fault)"를 일으키는 것이다. 이 모든 방법들이 모드 비싼 대가를 지불해야 하는데, 왜냐하면 이런 것은 최소한 BIOS 부트스

트랩 코드를 통한 짧은 우회 통로를 필요로 하기 때문이다. 사실 80286에서의 모드 전환은 밀리 초를 모두 써 버린다.

윈도우는 키가 눌러지거나 마우스가 움직이는 것과 같은 입력 이벤트마다 리얼모드로 전환하는 것을 피하는 방법이 분명히 필요했다. 이 해결책은 보호모드에서 I/O 인터럽트를 완전히 처리할 수 있는 보호모드 디바이스 드라이버를 작성하는 것이었다. 이런 드라이버들은 아직도 주위에 남아 있다. 바로 SYSTEM 디렉토리에서 찾을 수 있는 .DRV의 확장자를 가진 파일이다. 여기에는 MOUSE.DRV, COMM.DRV 등이 있다(여전히 그렇다). 이들은 내부적으로 16비트 DLL(dynamic-link library)과 비슷해 보이고 인텔칩의 보호 권한에 있어 가능 낮은 신뢰도로 취급되어 실행된다는 사실로 미루어 볼 때 3순위 권한의 DLL 드라이버로 간주할 수 있다. 이들의 주요 목적은 보호모드를 떠나지 않고 하드웨어와 표준모드 윈도우의 KERNEL, USER, GDI 사이를 연결하는 것이다.

### (3) 향상 모드 윈도우 (Enhanced-Mode Windows)

인텔 80386이 윈도우 운영모드에 있어 세 번째 모드가 가능하도록 했다. 이 모드를 **향상 모드(enhanced mode)**라 했다. 이 모드에서 윈도우는 페이지링을 사용했고 가상머신(virtual machine; VM)을 생성하는데 V86 요소들을 사용했다. 응용 프로그램 입장에서 본다면, VM은 완전히 자체적인 키보드, 마우스, 디스플레이 등을 소유한 보통의 개인용 컴퓨터처럼 보인다. 실제로 VM은 가상화(virtualization) 방법을 이용한 프로세스를 통해 단일의 물리 기계를 공유하는 것이다. 사용자(end-user)의 입장에서 본다면, 표준 모드 윈도우와 리얼모드 윈도우에 비해 향상 모드 윈도우의 두드러진 점은 그래픽컬 데스크탑의 윈도우에서 실행되는 MS-DOS 세션을 가지고 있다는 점이었다.

가상화를 지원하는 것은 가상 디바이스 드라이버(virtual device driver; VxD)의 몫이다. VxD라는 이름은 어떤 일반적인 디바이스(generic device)를 가상화 하는 드라이버를 뜻하는 "virtual x device"라는 것에서 왔다(여기서 "일반적인"이란 것은 x로 표현되었다). VKD라 이름 붙은 드라이버는 윈도우와 다중 MS-DOS 세션들이 마치 그들 자체적인 키보드와 상호 작용하는 것처럼 동작할 수 있도록 키보드를 가상화 한다. 다른 VMD라 이름 붙은 드라이버는 마우스에 대해 비슷한 마술을 부린다. 어떤 VxD는 아무 하드웨어도 가상화하지 않는데, 대신 여러 저수준 시스템 서비스에 대하여 편리한 창구 역할을 지원한다. 이러한 "디바이스 없는" VxD의 예는 함께 스왑 파일을 관리하는 PAGESWAP과 PAGEFILE이다. 스왑 파일은 향상모드 윈도우가 넓은 가상 어드레스 공간을 제공하기 위해 물리 RAM을 증대시키는데 사용해 왔다.

이러한 강한 기술적 토대에도 불구하고 향상 모드 윈도우는 디스크와 파일 I/O에 계속적으로 MS-DOS와 BIOS를 사용했다. 그래서, 예를 들면, 윈도우 3.0이 페이지를 스왑 해야할 때, MS-DOS와 BIOS가 I/O 동작을 명확히 처리하도록 하기 위해 프로세서를 V86모드로 전환한다.

보호모드, 리얼모드, V86모드간의 모든 모드 전환은 윈도우를 느리게 만들었다. 더구나 MS-DOS도 BIOS도 제진입 할 수 없다는 사실은 윈도우를 더 느리게 만들었다. 그래서 윈도우는 리얼모드 서비스에 대해서는 모든 응용 프로그램이 한 라인 위에서 강제적으로 기다리도록 해야만 했으며, 그래서 페이지링과 파일 시스템 I/O가 동시에 실행되는 프로그램(역자 주 : 윈도우 95나 윈도우 NT의 파일 맵핑 서비스 같은 것)은 만들 수가 없었다.

### (4) 윈도우 95 (Windows 95)

윈도우 95의 설명을 마지막으로 윈도우 역사에 대한 논의를 마무리한다. 윈도우 95는 하드웨어 디바이스 드라이버에 있어 몇 가지 다른 모델을 사용하는데, 대부분이 32비트 사용을 지향하며 3순위 권한 대신 0순위 권한 가상 디바이스 드라이버를 지향한다는 것이다. 윈도우 95에서 디바이스를 핸들링하는 일반적인 모델은 VxD가 인터럽트를 처리하고 모든 데이터 전송을 수행하며, 응용 프로그램은 필요로 하는 VxD 함수를 호출해 통신한다.

디바이스 프로그래밍에 있어서 VxD 지향적인 접근의 좋은 예 중의 하나는 바로 직렬 통신이다. 윈도우에서 통신은 관례적으로 3순위 권한 드라이버(COMM.DRV)를 사용하는데 이것은 하드웨어 인터럽트 핸들러와 UART(universal asynchronous receiver-transmitter) 칩을 운영하는데 필요한 모든 로직을 포함하고 있다. 두 개의 VxD(VCD와 COMBUFF)는 하드웨어 인터럽트와 소프트웨어적인 IN, OUT 명령을 가로채 처리하는 등의 작업을

통하여 각 포트를 가상화하고 멀티 테스킹 때문에 난처했던 문제점들을 개선하였다. 윈도우즈 95도 역시 comm.drv라는 이름의 3순위 권한 드라이버를 가지고 있으나, 이것은 VCOMM이라는 이름의 새로운 VxD를 둘러싼 얇은 껍데기에 지나지 않으며 단지 16비트 응용 프로그램과 VCOMM 사이를 연결하는 간단한 기능만을 수행한다. VCOMM은 응용 프로그램, VxD 클라이언트 프로그램, VxD 포트 드라이버 등이 달라붙어 있는 거미줄의 중심에 위치하고 있다. 이제 포트 드라이버들은 하드웨어와 의사 소통을 하는 모든 인터럽트와 실제의 IN, OUT 명령을 처리한다.

윈도우즈 95 파일 시스템은 VxD로 이동되고 있다는 또 다른 예 중의 하나다. 옛날에 16비트 보호모드 프로그램들은 파일 시스템 요청에 대하여 INT 21h 지향적이었다. VxD는 INT 21h 호출을 처리하고 MS-DOS에 의해 처리될 것에 대해서는 이를 V86 모드로 넘겨주었다. MS-DOS는 하드디스크 입출력에 대해서는 BIOS 함수를 사용하기 위해 INT 13h를 발생시키며, 네트워크인 경우 이를 네트워크로 전송하기 위하여 네트워크 재 설정 모듈(network redirector module)을 사용하기 위해 INT 2Fh를 발생시킨다. 윈도우즈 95는 응용 프로그램에 대해 호환성 있는 인터페이스를 제공하고 있기 때문에 여전히 파일 시스템 운영에 INT 21h 호출을 사용한다. 그렇지만 그 하부 구조는 많은 부분이 다르다.

윈도우즈 95에서 IFS(Installable File System) 관리자는 INT 21h 요구를 처리하며, 비록 이 요구가 V86 모드에서 발생한다고 하더라도, 그 진행 경로에 따라 FSD(file system driver)를 제어한다. VFAT라고 하는 FSD가 있는데 이것은 MS-DOS 파일 할당 테이블(file allocation table; FAT) 파일 시스템을 알고 있다. CDFSD라고 하는 다른 FSD는 CD-ROM의 형식을 알고 있으며, 또 다른 FSD는 다양한 네트워크에서 통신하는 방법을 알고 있다. VFAT 같은 로컬 파일 시스템 드라이버에 대한 디스크 동작은 전적으로 IOS(Input/Output Supervisor)의 관리하에 있는 다량의 VxD를 통하여 전달된다. 비록 V86 모드에서의 INT 13h도 결국 IOS 내부를 호출하는 것이다. 다시 말하면, 리얼모드 프로그램이나 보호모드 프로그램 모두다 VxD에 의해서 서비스되는 로컬 디스크 드라이브나 원격 디스크 드라이브에 대하여 파일 시스템 요구를 발생시키는 것이 가능하다.

윈도우즈 95에서 VxD 중심적인 디바이스 드라이버 모델의 가장 큰 이점은 윈도우즈 95 시스템 프로그래머가 더 이상 MS-DOS와 BIOS에 대하여 디바이스 드라이버를 작성하고 익스포트 할 필요가 없다는 것이다. 특정 응용 프로그램을 위해 시스템 확장을 제공하던 프로그래머도 이와 비슷하게 좋아졌다. 그에 비해 이전에는 DPMI를 알고 있어야 했으며, 윈도우즈 핵심 모듈에 대하여 비밀스럽고 문서화되지 않은 많은 요소들을 알고 있어야 했다. 하지만 이제는 Win32 DeviceControl API 함수와 소위 말하는 “경보 가능한(alertable)” 대기를 지원하는 Win32 API 들만을 이해하면 된다. 이러한 2개의 인터페이스는 VxD를 32비트 응용 프로그램처럼 사용할 수 있도록 해 준다.

시스템 수준에서 윈도우즈 95를 확장하는 방법으로써, VxD 프로그램에 분명히 역점을 두었음에도 불구하고, 윈도우즈 95는 여전히 윈도우즈 3.1과 이전버전의 윈도우즈에 훌륭한 호환성을 유지하고 있다. DPMI는 여전히 이를 사용하는 16비트 응용 프로그램을 위해 존재하고 있으며, 만일 원한다면 리얼모드용 네트워크 드라이버나 리얼모드용 파일 시스템 드라이버를 실행시킬 수 있다. 실제로, 종종 어떤 디바이스와 네트워크 드라이버, 16비트 응용 프로그램, 이를 지원하는 VxD가 세트로 윈도우즈 3.1과 윈도우즈 95에서 모두 문제없이 실행되어야 하는 일을 해야 할 때가 있는 것이다.