# Linux Partition HOWTO

Kristan Koehntopp, kris@koehntopp.de $Id: howto.sgml,v 2.4 1997/11/03 06:27:22 kris Exp $

번역: 박민석, dolman@correl1.snu.ac.kr

이 글은 리눅스 시스템을 위해 디스크 공간을 계획하고 설계하는 방법을 가르쳐준다. 디스크와 파티션, 스왑 공간의 크기 및 위치 결정에 있어 고려할 점들, 파일 시스템과 그 유형, 관련된 화제들에 대해서 이야기한다. 이 글의 의도는 절차가 아니라 배경 지식을 가르치는 것이다.

## 목 차



## 제 1 절 소개

### 1.1 이 글에 대하여.

이 글은 리눅스 미니 하우투 중의 하나이다. 미니 하우투란 리눅스의 설치와 관리에 관한 자습서 형식의 짧은 글이다. 정식 하우투나 책이 되기에는 글의 길이나 다루는 주제가 너무 작기 때문에 '미니' 하우투다. 하우투는 레퍼런스가 아니므로 레퍼런스가 필요하면 매뉴얼 페이지들을 보기 바란다.

### 1.2 이 문서의 내용 및 관련된 하우투 문서들

이 미니 하우투에서는 리눅스 시스템을 위해 디스크 공간을 계획, 설계하는 방법을 알려주고자 한다. 디스크와 파티션, 스왑 공간의 크기 및 위치 결정에서 고려할 점들, 파일 시스템과 그 유형, 관련된 화제들에 대해서 이야기한다. 이 글은 배경 지식을 가르치고자 하는 것이므로, 도구들에 대한 이야기보다는 주로 원리들에 대해 다룰 것이다.

당신이 처음 리눅스를 설치하기 전에 이 문서를 읽었으면 하지만, 대부분의 사람들에게는 어려운 일이다. 또 초심자라면 디스크 최적화 이외의 문제들도 겪게 된다. 따라서 당신은 아마 막 리눅스 설치를 마치고, 어떻게 하면 설치를 최적 화할 수 있을지 생각하고 있거나 다음에는 귀찮은 계산 착오를 피할 방법을 궁리하고 있는 사람이리라고 생각한다. 어떤 경우에도, 이 글을 다 읽고 당신이 전에 설치한 것을 싹 밀어버리고 새로 깔아야겠다는 욕구가 생겨나길 기대하는 바이다. :-)

이 미니 하우투의 대부분은 디스크 공간을 계획하고 설계하는 것에 국한되어 있으며 fdisk나 LILO, mke2fs, 각종 백업 프로그램의 사용법에 대해서는 다루고 있지 않다. 이런 문제들은 다른 하우투 문서들이 다루고 있다. 리눅스 하우투의 최신 정보가 필요하면 리눅스 하우투 목록(Index)를 보기 바란다. 목록에는 하우투 문서들을 구하는 방법도 나와있다.

파일 시스템의 여러 부분에 요구되는 크기와 속도를 알아볼 방법이 필요하면, by Gjoen Stein <gjoen@nyx.net>의 "리눅스 다중 디스크 설계 미니 하우투"를 보기 바란다. 1024개 이상의 실린더를 가진 디스크에 대한 내용이 필요하면 Andries Brouwer <aeb@cwi.nl>의 "리눅스 대용량 디스크 미니 하우투"를 보도록 하라.

이용자 별로 사용 가능한 디스크 공간을 제한하는 방법(쿼타, quotas)에 대한 내용은 Albert M.C. Tam <bertie@scn.org>의 "리눅스 쿼타 미니 하우투"를 보라.

현재 디스크 백업에 대한 일반적인 문서는 없지만, 특정한 백업 기법에 대한 문서가 몇 가지 있다. Thomas Koenig <Thomas.Koenig@ciw.uni-karlsruhe.de>의 "리눅스 ADSM 백업 미니 하우투"는 리눅스를 IBM의 ADSM 백업 환경에서 쓰고자 할 때 보라. Christopher Neufeld <neufeld@physics.utoronto.ca>의 "MS-DOS에서의 리눅스 백업 미니 하우투"에는 MS-DOS에서의 리눅스 백업에 대한 정보가 있다.

하우투 문서를 써서 투고하는 방법에 대해서는 Greg Hankins <gregh@sunsite.unc.edu>의 리눅스 하우투 목록을 참고하라.

/usr/src/linux/Documentation의 내용을 살펴보는 것도 교육적인 측면에서 큰 도움이 된다. 디스크 드라이버의 특성에 대한 정보나 파일시스템 혹은 하위 디렉터리들을 살펴보려면 ide.txt와 scsi.txt를 보라.

## 제 2 절 파티션이란 무엇인가?

PC의 하드디스크가 발명되고 얼마 지나지 않아, 사람들은 하나의 디스크밖에 없는 시스템에 여러 운영 체제를 설치하고 싶어하게 되었다. 이에 따라 하나의 물리적 디스크를 여러 개의 논리적 디스크로 분할하는 기술이 필요하게 되었는데, 이것이 바로 파티션이다. 대부분의 운영 체제가 하드디스크 상의 인접한 블록 섹션들을 완전히 별개의 디스크인 양 취급한다.

파티션이 겹쳐서는 안 된다는 것은 자명하다. 같은 컴퓨터에 설치된 다른 운영 체제가 구획 중복 때문에 중요한 정보를 덮어 써 버린다면, 어떤 운영 체제이건 기뻐할 리 만무하다. 인접한 파티션 사이에 빈 공간이 있어서도 안 된다. 이런 배치가 해로운 것은 아니지만, 구획 사이의 값비싼 디스크 공간을 낭비하게 되는 까닭이다.

디스크 전체를 파티션으로 분배할 필요는 없다. 디스크 끝에 지금 설치된 어떤 운영 체제에도 속하지 않은 빈 공간을 남겨두기로 할 수도 있다. 나중에 당신이 어떤 운영 체제를 가장 많이 쓰는지 분명해지면, 이 자투리 공간을 분배해서 파일 시스템을 설치하면 된다.

파티션을 옮기거나 그 크기를 바꾸면 그 안의 파일 시스템은 파괴된다. 따라서 파티션을 변경할 때에는 대개 영향을 받는 모든 파일들을 백업해서 보관하게 된다. 실제로 파티션을 변경하면 많은 것들이 뒤죽박죽이 되는 일이 보통이므로, `fdisk` 같은 것을 건드리기 전에 그 컴퓨터 상의 모든 디스크의 모든 것들을 백업해야 한다.

(혹시 운이 좋다면) 특정한 파일 시스템의 파티션은 자료 손실 없이 둘로 나눌 *수 있다*. 예를 들어 "fips"라는 프

로그램은 MS-DOS를 다시 설치하지 않고도 리눅스 설치를 위한 공간을 마련하기 위해 MS-DOS 파티션을 둘로 나눌 수 있다. 하지만 아직은 이런 일을 하기 전에 컴퓨터 안의 모든 것들을 조심해서 백업해두어야만 한다

## 2.1 백업이 중요하다.

테이프는 가장 친근한 백업 장치이다. 테이프는 빠르고 믿을 만 하며 사용하기 편하므로 백업을 자주 쉽게 자동으로 받을 수 있다.

Step on soapbox: 필자는 디스크 컨트롤러에 의해 작동되는 ftape 따위가 아니라 진짜 테이프를 이야기하고 있다. SCSI를 사는 것도 고려해 볼 만 하다. 리눅스는 SCSI를 기본적으로 지원하고 있다. 리눅스에서는 ASPI 드라이버를 띄울 필요가 없으므로, 귀중한 HMA를 잃어버릴 일이 없다. 그리고 일단 SCSI 호스트 어댑터가 설치되면 디스크나 테이프, 시디롬 등을 어댑터에 붙이기만 하면 된다. 더 이상 I/O 어드레스나 IRQ 조작이 필요없고, 마스터/슬레이브와 PIO 레벨을 맞출 필요도 없다.

덧붙임: 적절한 SCSI 호스트 어댑터는 별다른 CPU 부담 없이 입출력 성능의 향상을 가져다준다. 디스크를 심하게 써도 괜찮은 반응 속도를 경험할 수 있다. 리눅스 시스템을 유즈넷 뉴스 서버나 ISP 사업용으로 쓸 계획이라면, SCSI 없는 시스템은 꿈도 꾸지 말라. Climb of soapbox.

## 2.2 장치 번호와 장치의 이름

인텔에 기반한 시스템의 파티션 개수는 애초부터 제한되어 있었다. 원본 파티션 테이블이 부트 섹터의 일부로 설치되어 있고 네 개의 파티션이 들어갈 자리밖에 없다. 이 파티션들은 프라이머리 파티션이라 불린다. 시스템에 더 많은 파티션들이 필요하다는 것이 분명해 지자, 논리 파티션이 고안되었다. 논리 파티션의 개수에는 제한이 없다. 각각의 논리 파티션은 다음 논리 파티션을 가리키는 포인터를 하나씩 가지고 있다. 따라서 파티션은 꼬리에 꼬리를 물고 끝없이 이어질 수도 있다.

호환성 때문에, 논리 파티션이 차지하는 모든 공간은 프라이머리 파티션에 속해야만 한다. 논리 파티션을 쓰고 있다면, 프라이머리 파티션 하나가 "확장 파티션"으로 표시되어서 논리 파티션들이 차지하는 영역의 처음부터 끝까지 덮고 있게 된다. 이것은 모든 논리 파티션들에 할당된 공간이 연결되어 있어야만 한다는 것을 뜻한다. 확장 파티션은 하나 뿐이다. 어떤 `fdisk` 프로그램도 둘 이상의 확장 파티션을 만들 수 없다.

리눅스는 드라이브 당 제한된 수의 파티션만을 다룰 수 있다. 리눅스는 4개의 프라이머리 파티션(논리 파티션을 쓰고 있다면 이 가운데 3개를 쓸 수 있다)을 가질 수 있고, 디스크 하나 당 SCSI의 경우 최대 15개, IDE라면 최대 63개의 파티션을 가질 수 있다.

리눅스에서는 디바이스 파일이 파티션을 나타낸다. 디바이스 파일은 c(버퍼 캐쉬를 쓰지 않는 "character" 디바이스)나 b(버퍼 캐쉬를 사용하는 "block" 디바이스) 형식을 갖는 파일이다. 리눅스에서는 모든 디스크가 block 디바이스로만 표시된다. 다른 유닉스 시스템들과는 달리 리눅스는 디스크와 파티션에 대해 "버퍼를 거치지 않는" character 디바이스를 제공하지 않는다.

디바이스 파일에서 중요한 것은 파일 크기 대신 표시되는 주 장치 번호(major device number)와 부 장치 번호(minor device number) 뿐이다.

```
$ ls -l /dev/hda
brw-rw----   1 root     disk       3,   0 Jul 18  1994 /dev/hda
                                   ^    ^
                                   |    부 장치 번호
                                   주 장치 번호
```

디바이스 파일에 접근할 때, 주 장치 번호가 입/출력을 수행하기 위해 호출될 디바이스 드라이버를 결정한다. 이 호출은 부 장치 번호를 매개변수로 수행되며, 부 장치 번호가 해석되는 방식은 전적으로 드라이버에 달려있다. 보통 드라이버에 관한 문서에 이 부 장치 번호가 사용되는 방식이 기술되어 있다. IDE 디스크라면 `/usr/src/linux/Documentation/ide.txt`에 기술되어 있다. SCSI 디스크에 대한 문서는 `/usr/src/linux/Documentation/scsi.txt`일 것이라고 추측하겠지만, 이런 문서는 없다. 확실한 것을 알아

보려면 드라이버의 소스(/usr/src/linux/driver/scsi/sd.c:184-196)를 살펴보아야만 한다. 다행히도 Peter Anvin의 디바이스 번호 및 이름 목록인 /usr/src/linux/Documentation/devices.txt이 있다. IDE는 주 번호 3, 22, 33, 34, SCSI는 주 번호 8인 block 디바이스 목록을 보라. 주 번호와 부 번호는 각각 한 바이트로 지정되며, 이런 까닭으로 디스크 당 파티션 수가 제한되는 것이다.

관례에 따라 디바이스 파일은 특정한 이름을 가지며, 많은 시스템 프로그램들은 컴파일될 때부터 이 이름을 알고 있다. IDE 디스크는 /dev/hd*, SCSI 디스크는 /dev/sd*라는 이름을 갖는 것이 관례다. 디스크들은 a, b, c 등으로 번호를 부여받는다. 따라서 /dev/hda가 첫 번째 IDE 디스크, /dev/sda가 첫 번째 SCSI 디스크의 이름이다. 이 두 디바이스들은 모두 1번 블록에서 시작되는 전체 디스크를 나타낸다. 잘못된 도구로 이 디바이스들에 쓰는 것은 이 디스크들에 있는 master boot loader와 파티션 테이블을 파괴할 것이며, 디스크 상의 모든 자료를 사용 불가능하게 하고 당신의 시스템을 부팅 불가능하게 만들 것이다. 디바이스들에 쓰는 일을 하기 전에, 무엇을 하고 있는지 알아야 하며, 그리고 다시 한 번 말해 두거니와, 백업을 해 두어야만 한다.

디스크 상의 프라이머리 파티션은 1, 2, 3, 4이다. 따라서 /dev/hda1가 첫 번째 IDE 디스크의 첫 파티션이 된다. 논리 파티션은 5 이상의 번호를 가지며, 따라서 /dev/sdb5가 두 번째 SCSI 디스크의 첫 번째 논리 파티션이다.

각 파티션에는 할당된 공간의 시작 및 끝 블록 주소와 형식이 등록되어 있다. 형식이란 특정 파티션을 어떤 형식의 운영 체제에 지정하는 (1 바이트의) 수치 부호이다. 컴퓨터 참조(computing consultant) 상의 편의를 위해 파티션 형식 부호는 단일하지 않으며, 언제나 두 개의 운영 체제가 같은 형식 부호를 가질 확률이 있다.

리눅스는 형식 부호 0x82를 스왑 파티션으로, 0x83를 "고유의(native)" 파일 시스템 (대부분의 경우 ext2) 정해두고 있다. 한 때 인기 있었지만 이제는 구식이 된 Linux/Minix 파일 시스템의 파티션은 0x81 부호를 쓰며, OS/2와 윈도즈 NT의 NTFS의 파티션은 0x07의 형식 부호로 표시된다. MS-DOS는 다양한 FAT 파일 시스템의 취향에 따라 여러 가지 형식 부호를 할당한다. 0x01, 0x04, 0x06이 알려져 있다. DR-DOS는 보호된 FAT 파티션을 표시하기 위해 0X81을 사용하며, 이 경우 Linux/Minix아 충돌을 일으킨다. 하지만 Linux/Minix와 DR-DOS는 이제는 널리 사용되지 않으므로 별 문제가 아니다. 논리 파티션을 담기 위해 쓰이는 확장 파티션은 0x05의 형식을 갖는다.

fdisk 프로그램으로 파티션을 만들거나 지우게 된다. 쓸만한 운영 체제라면 fdisk 프로그램을 포함하여 배포된다. 이 프로그램은 거의 모든 운영 체제에서 전통적으로 fdisk(혹은 FDISK.EXE)라고 부른다. DOS 등의 fdisk는 다른 운영 체제의 파티션을 다루는데 한계를 갖는다. 이 한계에는 다른 운영 체제의 부호와 관련된 것을 전혀 다룰 수 없는 점이나 1024를 넘는 실린더 수를 다룰 수 없다는 점, 끝이 실린더 경계와 일치하지 않는 파티션은 만들 수도 인식할 수도 없다는 점등이 포함된다. 예를 들어 MS-DOS의 fdisk는 NTFS 파티션을 지울 수 없으며, OS/2의 fdisk는 리눅스의 fdisk가 만든 끝이 실린더 경계와 일치하지 않는 파티션을 묻지도 않고 "수정"한다고 알려져 있다. DOS와 OS/2의 fdisk는 모두 1024 실린더 이상의 디스크를 다루는데 문제가 있다. (이런 디스크에 대한 자세한 내용이 필요하면 "large-disk" 미니 하우투를 보라.)

# 제 3 절 내가 필요로 하는 파티션은 무엇인가?

## 3.1 나는 얼마나 많은 파티션이 필요할까?

당신은 어떤 파티션이 필요한가? 어떤 운영 체제들은 얼토당토않은 이유로 논리 파티션에서 부팅하지 못하도록 되어 있다. 따라서 당신은 아마 MS-DOS, OS/2, 리눅스 등등 사용 중인 운영 체제의 부트 파티션으로 쓰기 위해 프라이머리 파티션을 남겨 두기를 바랄 수도 있다. 프라이머리 파티션 하나는 확장 파티션으로서 필요하다는 것도 염두에 두도록 하라. 확장 파티션은 나머지 디스크 공간을 논리 파티션으로 쓰기 위해 담아두는 그릇 노릇을 한다.

부팅 가능한 운영 체제는 BIOS와 1024 실린더의 한계를 포함하는 real-mode여야 한다. 때문에 당신은 말썽을 피하기 위해 부트 파티션들을 모두 앞쪽 1024 실린더 이내에 두기를 원할 것이다. 자세한 내용은 역시 "large-disk" 미니 하우투를 참조하라.

리눅스를 설치하려면 최소한 하나의 파티션이 필요하다. 커널이 이 파티션으로부터 (예를 들면 LILO에 의해) 적재된다면, 이 파티션은 BIOS가 읽을 수 있는 것이어야만 한다. 만일 (예를 들어 부트 디스크나 MS-DOS에

기반한 리눅스 로더인 LOADLIN.EXE 같은) 다른 방법으로 커널을 적재한다면, 리눅스 파티션은 어디 있어도 괜찮다. 어떤 경우이건 이 파티션은 0x83 "Linux native" 형식이어야 한다.

시스템에는 스왑 공간이 필요하다. 스왑을 파일에다 하지 않겠다면, 스왑 전용 파티션이 하나 있어야 한다. 이 파티션은 리눅스 커널에 의해서만 사용되고 커널은 PC BIOS의 의존성에 구속받지 않으므로, 스왑 파티션은 어디에 있어도 좋다. 필자는 스왑 파티션으로는 논리 파티션(/dev/?d?5나 그 이상)을 쓰도록 권장한다. 리눅스 스왑 전용 파티션은 0x82 "Linux swap" 형식이다.

이상이 최소로 요구되는 파티션이다. 하지만 리눅스용 파티션을 더 만들어 두는 편이 좋다. 계속 읽도록 하라.

## 3.2 파티션의 크기는 얼마나 되어야 할까?

스왑 전용 파티션을 쓰기로 했다면, 대개의 경우 옳은 선택이다. 다음 안내를 따라 그 크기를 결정하도록 하라.

- 리눅스에서는 램과 스왑 공간이 더해진다. (모든 유닉스 시스템에서 이런 것은 아니다.) 예를 들어 8메가의 램과 12메가의 스왑이 있다면, 당신은 총 20메가 정도의 가상 메모리를 갖게 된다.
- 스왑 공간의 크기를 정할 때, 최소한 총 16메가의 가상 메모리를 갖도록 해야 한다. 즉 램이 4메가 있는 경우라면 최소한 12메가의 스왑을 잡아야 하고, 8메가의 램이 있다면 적어도 스왑 8메가를 잡아야 한다.
- 리눅스에서는 한 스왑 파티션의 크기가 128메가를 넘을 수 없다. 파티션의 크기는 128메가 보다 클 수도 있지만, 128메가 이상의 공간은 절대로 사용되지 않기 때문이다. 따라서 128메가 이상의 스왑이 필요하면, 스왑 파티션을 여러 개 만들어야 한다.
- 스왑 공간의 크기를 정할 때에는 너무 많은 스왑 공간은 전혀 쓸 데 없다는 점을 명심해야 한다. 모든 프로세스는 "working set"을 갖는다. Working set이란 프로세서에 의해 곧 참조될 메모리에 올려진 페이지들의 집합이다. 리눅스는 (최근 사용된 페이지들이 가까운 장래에 다시 사용되리라고 가정함으로써) 이 메모리 접근을 예측하고자 한다. 그리고 가능한 한 이 페이지들을 램 내에 남겨두려고 노력한다. 프로그램의 "참조의 국소성"이 좋으면 이 가정은 참이 되고, 예측 알고리즘이 제 몫을 할 것이다.

  Working set을 주 메모리에 남겨 두는 것은 충분한 주 메모리가 있을 경우에만 작동한다. 한 컴퓨터에서 지나치게 많은 프로세스를 실행시키면, 커널은 매우 가까운 장래에 다시 참조될 페이지들을 디스크에 넣어 두도록 강요받게 된다. (다른 working set에서 한 페이지를 메모리에서 내리고, 막 참조된 페이지를 메모리에 올리도록 한다.) 대개 이런 상황은 페이지 관리 작업을 매우 증가시키고, 작업 수행 능력을 현저히 저하시킨다. 이런 상태에 있는 컴퓨터를 "몸부림치고 있다(thrashing)"고 한다. (For you german readers: That's "thrashing" ("dreschen", "schlagen", "haemmern") and not trashing ("muellen"))

  몸부림치고 있는 컴퓨터에서는 프로세스들이 램이 아니라 디스크로부터 수행되고 있다. 메모리 접근 속도와 디스크 접근 속도의 비 정도로 수행성능이 저하되리라고 생각하면 된다.

  PDP와 Vax가 쓰이던 옛 시절의 경험적인 원칙 하나는 한 프로그램의working set의 크기는 그 프로그램의 가상적인 크기의 1/4정도라는 것이다. 따라서 램의 세 배 이상 스왑을 잡는 것은 대개 소용없는 일이다.

  하지만 이 것은 단지 한 가지 경험 법칙일 뿐이라는 점을 명심하기 바란다.

  엄청나게 크거나 지독히 작은 working set을 갖는 상황을 만들기란 쉬운 일이다. 예를 들어 매우 불규칙적으로 이용되는 커다란 데이터집합을 갖는 시뮬레이션 프로그램이라면, 데이터 참조에 있어서 눈에 띌만 한 국소성이란 거의 없다. 따라서 이런 프로그램의 working set은 상당히 클 것이다.

  한편 여러 JPEG 파일들을 동시에 열어 놓았지만 하나 만 빼고는 모두 아이콘화시켜 놓은 xv 프로그램은 매우 큰 데이터를 갖는다. 그러나 이미지 전환은 하나의 이미지에서만 이루어지고, xv가 차지하고 있는 메모리의 대부분은 전혀 건드리지 않는다. 여러 개의 편집기 창을 가지지만 한 번에 하나의 창에서만 편집이 되는 편집기도 마찬가지 경우다. 이런 프로그램들은 제대로 설계되었다면 매우 높은 참조의 국소성을 가지며, 프로그램의 대부분이 큰 성능 저하 없이 스왑 공간으로 내려질 수 있다.

  커맨드 라인 시대에 통하던 1/4이라는 working set 크기가 요즘처럼 여러 개의 문서를 편집하는 GUI 프로그램에서도 맞을지 의심스러울 수도 있지만, 필자가 아는 바로는 이 수치를 확인하려고 시도한 새로운 논문은 없다.

따라서 16메가의 램을 가진 상황이라면, 최소한의 설정을 위해서는 스왑이 필요 없고, 48메가 이상의 스왑은 아마 쓸모 없을 것이다. 정확히 얼마나 메모리가 필요한지는 응용 프로그램과 컴퓨터에 달려있다. (다른 무엇을 기대했나?)

## 3.3 스왑 공간의 위치는 어디가 좋을까?

- 기계적인 것은 느리고, 전자적인 것은 빠르다. 요즘의 하드디스크에는 많은 헤드가 있다. 같은 트랙의 헤드 사이를 오가는 것은 완전히 전자적인 것이어서 빠르다. 반면 트랙들 사이를 오가는 것은 느린데, 실제 물체를 움직이는 일이 포함되기 때문이다.

  따라서 헤드가 많은 디스크와 적은 디스크가 있다면, 다른 사양이 같을 경우에는 헤드를 많이 가진 쪽이 빠를 것이다.

  하지만, 스왑 공간을 나누어서 두 개의 디스크에 두는 편이 더 빠를 것이다.

- 구형 디스크에는 모든 트랙에 같은 숫자의 섹터가 있다. 이런 디스크의 경우에는, 디스크 헤드가 임의의 트랙에서 스왑 영역으로 간다고 가정하면 스왑을 디스크 중간에 잡는 것이 가장 빠를 것이다.

- 신형 디스크는 ZBR(zone bit recording)을 사용한다. 이런 디스크는 바깥 쪽 트랙에 더 많은 섹터를 갖고 있다. Rpms가 일정하다면, 이런 구조에서는 바깥 쪽 트랙이 안쪽 보다 훨씬 우수한 성능을 보인다. 스왑은 빠른 트랙에 두어야 한다.

- 물론 디스크 헤드가 무작위로 움직이지는 않을 것이다. 늘 바쁜 home 파티션과 거의 사용되지 않는 보관용 파티션 사이에 스왑 공간을 두고 있다면, 헤드가 덜 움직이도록 스왑을 home 파티션 중간에 두는 편이 낫다. 스왑을 거의 스왑 전용인 다른 디스크에 잡는다면 훨씬 더 좋을 것이다.

**요약:** 스왑은 빠르고, 헤드를 많이 가지고 있으며, 다른 작업에 바쁘지 않은 디스크에 잡아라. 디스크를 여러 개 가지고 있다면, 스왑을 쪼개서 디스크마다 혹은 제어기(controller)마다 두도록 하라.

**더 나은 방법:** 램을 더 사라.

## 3.4 파일 시스템과 파편화에 대한 몇 가지 것들.

운영 체제는 디스크 공간을 블록과 블록의 조각(fragmentation) 단위로 관리한다. Ext2 파일 시스템에서는 조각과 블록이 같은 크기이기 때문에, 이 글에서는 블록에 대해서만 이야기하도록 하겠다.

파일의 크기는 다양하다. 파일이 블록의 크기에 딱 맞는 일은 없다. 따라서 모든 파일의 마지막 블록 가운데 일부는 낭비되게 된다. 파일의 크기가 불규칙하다면 디스크에 들어 있는 모든 파일들은 각각 반 블록 정도의 낭비되는 부분을 갖게 된다. 탄넨바움 씨는 자신의 저서 "운영 체제"에서 이것을 "파편화(fragmentation)"이라고 불렀다.

파일의 개수는 대략 디스크에 있는 할당된 inode의 개수와 같다고 추정할 수 있다. 필자의 디스크에는

```
# df -i
Filesystem           Inodes   IUsed   IFree  %IUsed Mounted on
/dev/hda3             64256   12234   52022     19%  /
/dev/hda5             96000   43058   52942     45%  /var
```

/에 약 12000 개의 파일이 있고, /var.에는 약 44000 개의 파일이 있다.

블록 하나의 크기가 1KB인 경우, 파일 꼬리에 붙은 약 6+22 = 28MB의 디스크 공간이 손실된다. 만약 블록 하나의 크기가 4KB였다면, 필자는 네 배의 공간을 손해보았을 것이다.

반면에 데이터의 전송은 인접 데이터 덩어리가 큰 경우에 더 빠르다. 때문에 ext2 파일 시스템에서는 점점 커지는 파일에 대해서는 8 개의 인접 블록을 한 단위로 하여 공간을 미리 배분하도록 한다. 사용되지 않은 사전 할당 공간은 파일이 닫혀질 때 놓여나므로, 공간의 낭비는 없다.

한 파일 안의 블록들이 인접해 있지 않다면, 파일이 종종 차례로 접근되기 때문에 성능에 좋지 않다. 이렇게 되면 운영 체제가 디스크 접근을 나눠서 해야 하고, 디스크도 헤드를 움직여야 하게 된다. 이런 상황을 "외적 파편화" 혹은 간단히 "파편화"라고 부르며, 도스 파일 시스템에서 흔한 문제다.

Ext2 파일 시스템에는 외적 파편화를 방지하기 위한 몇 가지 전략이 있다. 보통 ext2에서는 유즈넷 뉴스 스풀처럼 혹사되는 파티션의 경우에도 파편화가 큰 문제가 되지 않는다. Ext2 파일 시스템에도 파편화된 것을 정리해주는 도구가 있지만, 아무도 이 도구를 쓴 일이 없고 현재 쓰이는 ext2에 비하면 구식이다. 한 번 해 봐도 되지만, 문제가 생기면 당신이 책임져야 한다.

MS-DOS 파일 시스템은 병적인 디스크 공간 관리로 유명하다. MS-DOS에서 쓰는 최악의 버퍼 캐쉬와 더불어, 파일 파편화가 수행 성능에 끼치는 영향은 정말 엄청나다. DOS 사용자들은 몇 주마다 디스크의 파편화 상태를 정리하는데 익숙해진 나머지, 파편화를 정리하는데 대해 약간은 종교적인 신념까지 생기게 되었다. 리눅스와 ext2 파일 시스템에서는 이런 습관이 필요 없다. 정상적으로 사용한다면 리눅스 본래의 파일 시스템에서는 파편화를 정리할 필요가 없다. 디스크에 최소한 5%만 빈 공간이 있다면 어떤 경우에도 괜찮다.

MS-DOS 파일 시스템은 내적 파편화 때문에 많은 디스크 공간을 낭비하는 것으로도 잘 알려져 있다. 256메가 이상의 파티션에서는, DOS 블록의 크기가 너무 커져서 더 이상 쓸모가 없어져 버린다. (이런 점은 FAT32에서는 어느 정도 고쳐졌다.)

Ext2는 0.5 TB (1테라 바이트는 1024 기가 바이트와 같다) 이상의 엄청나게 큰 파일 시스템만 아니라면, 대형 파일 시스템에서도 큰 블록을 선택할 필요가 없다. 0.5 테라 이상인 경우에는 작은 크기의 블록이 효율적이지 않게 된다. 따라서 DOS와는 달리 블록 크기를 줄이려고 큰 디스크를 여러 파티션으로 쪼갤 필요가 없다. 가능하다면 1 킬로바이트의 디폴트 블록 크기를 쓰도록 하라. 어떤 파티션에 대해서는 2 킬로바이트 짜리 블록 크기를 시험해 보고 싶을 수도 있지만, 흔치 않은 버그와 맞닥뜨리기 십상이다. 대부분의 사람들은 디폴트를 쓴다.

## 3.5 파티션 결정 기준으로서의 파일 수명과 백업 주기

Ext2에서는 백업 계획과 다양한 파일 수명에 따른 외적 파편화를 줄일 것을 염두에 두고 파티션을 결정해야 한다.

파일은 수명을 갖는다. 한 파일은 만들어진 다음 시스템에 어느 시간 동안 남아 있다가 지워지게 된다. 파일의 수명은 시스템에 따라 크게 다르고, 부분적으로는 파일의 경로 명에도 의존하게 된다. 예를 들어 `/bin`, `/sbin`, `/usr/sbin`, `/usr/bin`나 이 비슷한 디렉터리에 있는 파일들은 대개 여러 달 이상의 매우 긴 수명을 갖는다. `/home`에 있는 파일들의 수명은 여러 주쯤으로 중간 정도이다. `/var`의 파일들은 보통 수명이 짧다. `/var/spool/news`에 있는 파일들은 며칠 이상 남아있는 것이 드물고, `/var/spool/lpd`에 속한 파일들의 수명은 몇 분 남짓이다.

백업을 위해서는 하루에 백업할 양이 백업 매체 하나의 용량 이하인 쪽이 편하다. 매일 이루어지는 백업은 전체 백업일 수도 있고, 바뀐 부분만 추가해 가는 식(incremental)일 수도 있다. 따라서 (매일 전체 백업을 하기 위해서) 파티션의 크기를 한 백업 매체에 완전히 들어갈 정도로 작게 잡을 수 있다. 어떤 경우이건, 파티션 하나의 크기는 매일 바뀐 파일 전부가 백업 매체 하나에 들어갈 만한 크기여야만 한다. (추가식 백업을 택하고 백업 매체는 주 혹은 월 단위의 전체 백업 때 바꾸도록 한다. 이 경우 자동 백업은 불가능하다.)

백업 전략은 이 결정에 달려있다.

디스크 공간을 계획하고 구입할 때, 백업에 쓸 돈을 충분히 남겨 놓도록 해야 한다. 백업되지 않은 데이터는 쓸모 없다! 아마 누구든 데이터를 다시 만들어내는 비용이 백업 비용에 비해서 훨씬 비쌀 것이다.

수행 성능을 위해서는 파일의 수명에 따라 다른 파티션에 두는 것이 유용하다. 이렇게 하면 뉴스 파티션에 있는 수명이 짧은 파일들이 심하게 파편화 되더라도, `/`나 `/home` 파티션의 수행 성능에는 영향이 없다.

# 제 4 절 한 가지 예

## 4.1 열성적인 초보자에게 추천하는 모형

위에서 이야기한 것처럼 /와 /home, /var 파티션을 만드는 것은 공통적인 모형이다. 이 정도면 설치하고 관리하기에 쉬우면서도 파일들 사이의 수명 차이로 인한 부작용을 막기에 충분히 분리되었다고 할 수 있다. 또 백업하기에도 좋다. 유즈넷 뉴스 스풀을 백업하려고 고생할 사람은 아마 아무도 없을 것이다. /var/에 있는 파일 가운데 백업할 만한 가치가 있는 것은 얼마 되지 않는다. (/var/spool/mail 정도가 떠오른다.) 한편 / 디렉터리의 내용은 좀처럼 바뀌지 않고 (설정을 바꾼 다음처럼) 필요가 있을 때만 백업해도 좋다. 그리고 설치된 소프트웨어의 양에 따라 다르지만 대략 250에서 500메가 정도로, 최신 백업 매체 한 장에 전체 백업을 받을 수 있을 만한 크기이기도 하다. /home은 귀중한 사용자의 데이터들을 포함하고 있으므로 매일 백업을 받아야 한다. 어떤 경우에는 /home이 매우 크므로 이때에는 추가식 백업을 써야만 한다.

어떤 시스템에서는 /tmp가 별도의 파티션에 위치하기도 하고, 다른 경우에는 /tmp를 /var/tmp에 심볼릭 링크 시켜서 같은 효과를 내기도 한다. (이 경우 단일 사용자 모드에서는 문제가 생길 수 있다는 점에 주의해야 한다. 이 때에는 /var를 사용할 수가 없고, /tmp를 만들거나 /var를 수동으로 마운트 시킬 때까지는 시스템에 /tmp가 없는 상황이 된다.) 혹은 (솔라리스에서처럼) 램 디스크에 /tmp를 두기도 한다. 이렇게 /tmp를 /에서 빼내 두는 것도 좋은 생각이다.

이런 모형은 업그레이드나 재설치에도 편리하다. 설정 파일들을 (혹은 /etc 디렉터리 전체를) /home 디렉터리에 저장해 두고, /를 밀어버린다. 재설치를 하고 나서 /home의 저장 디렉터리에서 예전의 설정 내용을 불러오면 된다.

# 제 5 절 필자의 경우

필자에게는 쓸모가 없어서 두 해 전에 포기하고 선반에 치워둔 ISA 버스 방식의 낡은 386/40 컴퓨터가 있었다. 필자는 이 컴퓨터를 X 윈도우 없는 조그만 가정용 LAN 서버로 만들 계획을 세웠다.

필자는 우선 이 386 컴퓨터에 16메가의 램을 설치했다. 그리고 가능한 한 가장 작고 값싼 EIDE 디스크(800 메가)와 이더넷 카드를 장착했다. 필자가 아직 허큘리스 용 모니터를 가지고 있었기 때문에 구식 허큘리스 카드를 달았다. 다음에는 리눅스를 설치해서 메일 라우터와 POP3 서버를 비롯해서 NFS, SMB, HTTP, LPD/LPR, NNTP 서버를 띄웠다. ISDN 카드를 달아서 이 컴퓨터는 필자의 TCP/IP 라우터 겸 방화벽 역할도 하게 되었다.

이 컴퓨터의 디스크 공간은 대부분 /var 디렉터리 아래의 /var/spool/mail와 /var/spool/news, /var/httpd/html에 주었다. /var 디렉터리는 따로 큼직하게 잡아둔 파티션에 두었다. 이 컴퓨터에는 사용자가 거의 없을 테이므로 홈 파티션을 만드는 대신, 다른 워크스테이션에서 NFS를 통해 /home 디렉터리를 마운트 시켰다.

/로 250메가 파티션을 잡으면 X 없이 따로 몇 가지 유틸리티를 설치한 리눅스에는 충분할 것이다. 램은 16메가가 있지만, 이 컴퓨터는 많은 서버를 돌릴 것이므로 16메가 정도의 스왑이 필요할 것이고, 32메가 정도면 넉넉할 것이다. 디스크 공간이 부족하지는 않으므로 32메가의 스왑을 잡았다. 인정 때문에 약 20메가의 MS-DOS 파티션도 만들었다. /home을 다른 컴퓨터로부터 마운트하기로 했기 때문에, 남은 500메가 이상의 공간은 /var로 다 할당했다. 이 정도면 집안에서 쓸 유즈넷 뉴스 서버로는 충분하고도 남는다.

결국 다음과 같이 되었다.

```
Device      Mounted on                      Size
/dev/hda1   /dos_c                           25 MB
/dev/hda2   - (Swapspace)                    32 MB
/dev/hda3   /                               250 MB
/dev/hda4   - (Extended Container)          500 MB
/dev/hda5   /var                            500 MB
homeserver:/home /home                      1.6 GB
```

필자는 네트웍을 통해 homeserver에서 테이프로 이 컴퓨터를 백업할 계획이다. 이 컴퓨터의 모든 것들이 시디롬에서 설치되었기 때문에, /etc 디렉터리의 설정 파일들 몇 가지와 /root/Source/Installed에 있는 필자가 따로 설치한 *.tgz 파일들, /var/httpd/html과 /var/spool/mail 정도만 백업하면 된다. 필자는 이 파일들을 매일 밤 따로 만들어 둔 homeserver의 /home/backmeup 디렉터리에 복사하고, homeserver를 정식으로 백업할 때 이 디렉터리도 함께 백업한다.