손쉬운 리눅스 관리

Steve Frampton

frampton (at) LinuxNinja.com

도운이

고양우

newcat (at) splinux.co.kr

손쉬운 리눅스 관리
지은이 Steve Frampton

도운이 고양우

펴냄 1999년 10월 21일

“손쉬운 리눅스 관리 (Linux Administration Made Easy)” (LAME) 는 리눅스 시스템 운영자들이 매일 매일 부딛히게 되는 관리와 유지에 관한 이슈를 설명해주는 책이다. 이 책은 리눅스 문서화 프로젝트 (Linux Document Project) 의 일부로서 작성되었다.

차례

1. 서문 **5**
1.1. 감사의 말 5
1.2. 저작권 정보와 관련 법적 정보 5
1.3. 도움 요청 5
2. 소개 **6**
2.1. 범위 6
2.2. 리눅스 배포판 선택하기 6
3. 리눅스 개요 **9**
3.1. 리눅스는 무엇인가? 9
3.2. 미신 깨뜨리기 9
3.3. 사용자의 관점 11
4. 설치와 하드웨어 설정 **13**
4.1. 설치 디스켓 만들기 13
4.2. 리눅스 설치 프로그램으로 부팅하기 14
4.3. 하드 드라이브 파티션하기 15
4.4. 스왑 공간 셋업하기 20
4.5. 포맷할 파티션 선택하기 20
4.6. 설치할 패키지 선택하기 20
4.7. 하드웨어 설정 21
4.8. LILO 로 부팅하기 21
4.9. 레드햇 업데이트를 내려받아서 설치하기 22
5. X 윈도우 시스템 설정하기 **24**
5.1. X-Configurator 를 이용하여 X 윈도우 시스템 설정하기 24
5.2. X 데스크탑 매니저 사용하기 25
5.3. X 에서 글꼴의 모양을 좋게 하는 법 26
5.4. X 용 윈도우 매니저 선택하기 27
5.5. 그놈의 설치와 설정 27
5.6. KDE 설치와 설정 28
6. 일반 시스템 관리의 이슈 **30**
6.1. 루트 계정 30
6.2. 사용자 계정 만들기 30
6.3. 사용자 패스워드 바꾸기 32
6.4. 사용자 계정 막아두기 32
6.5. 사용자 계정 제거하기 33
6.6. 리눅스 패스워드 & 섀도우 파일 형식 33
6.7. 시스템의 종료와 재시작 35
7. 설정과 관리 **37**
7.1. 웹 서버와 HTTP 캐싱 프록시의 관리 37
7.2. 도메인 네임 서버 (DNS) 의 설정과 관리 38
7.3. TACACS 를 이용한 인터넷 사용자 인증 42

7.4. 삼바를 이용한 윈도 스타일의 파일과 인쇄 서비스 43
7.5. Netatalk 를 이용한 매킨토시 스타일의 파일과 인쇄 서비스 47
7.6. 네트웍 파일 시스템(NFS) 서비스 49
7.7. Linuxconf 를 이용한 설정의 모든 것 50
8. 백업과 복구 설차 **51**
8.1. 서버의 백업 절차 51
8.2. 서버 복구 절차 56
8.3. 시스코 라우터 설정 백업 58
9. 잡다한 관리 업무 **61**
9.1. 저장 공간 검사 61
9.2. 프로세스 관리 63
9.3. 프로세스의 시작과 중단 64
9.4. Cron 과 Crontab 파일을 이용하여 작업을 자동화하기 64
10. 리눅스와 응용 프로그램을 업그레이드하기 **66**
10.1. 레드햇 패지지 관리자 (RPM) 사용하기 66
10.2. RPM 없이 설치하고 업그레이드 하기 67
10.3. 시스템을 최신으로 유지하는 전략 69
10.4. 리눅스 커널 업그레이드 69
10.5. 레드햇 제공 커널로 업그레이드하기 70
10.6. 맞춤 커널 만들기 71
10.7. 리눅스 2.2.x 커널로 옮아가기 75
10.8. 아파치 웹 서버 설정하기 77
10.9. Squid HTTP 캐싱 프록시 데몬 설정하기 77
10.10. Sendmail 전자우편 데몬 설정하기 78
11. 리눅스를 기업차원에서 쓰기 **81**
11.1. 성능 튜닝 81
11.2. RAID 를 이용한 고가용성 구현 81
11.3. 서버 이관과 확장성 82
12. 서버를 안전하게 지키기 위한 전략 **85**
13. 사람 살려**!** 천국에 문제가 생겼어요**!** **89**
13.1. 리눅스를 새로운, 지원되지 않는 하드웨어에 설치하기 89
13.2. 정전이나 시스템의 파괴에 따른 파일 시스템의 손상 89
13.3. 어디에서 도움을 받을까 ? 90
13.4. 추가 문서에 대한 포인터 92

1장. 서문

1.1. 감사의 말

리눅스 커뮤니티에 감사드린다; 특히 유즈넷과 메일링 리스트를 통하여 리눅스를 최대한 활용하는 방법에 대한 많은 팁과 답과 제안을 제시한 사람들에게 감사드린다. 여러분들의 공헌은 우리들 모두에게 혜택을 주고 있다.

이 문서는 DocBook SGML 형식으로 작성된 후 SGMLTools 2.x를 이용하여 HTML, 포스트스크립트, 리치 텍스트 포맷, PDF 등 다양한 형태로 변환되었다. SGMLTools에 대한 자세한 장보는 해당 프로젝트의 웹 사이트인 http://www.sgmltools.org/ 를 참고하기 바란다.

1.2. 저작권 정보와 관련 법적 정보

Copyright ŒR997-1999 by Steve Frampton. This material may be distributed only subject to the terms and conditions set forth in the Open Publication License, v0.4 or later (the latest version is presently available at http://www.opencontent.org/openpub/).

나는 이 문서를 공공 서비스로서 리눅스 커뮤니티에 작성하여 무료로 제공한다. 나는 여기에 포함된 정보가 최신이며 정확하고 도움이 되도록 하기 위하여 최선의 노력을 하였지만 여기에 포함된 잘못된 정보에 의한 직간접적인 손실에 대하여 어떠한 책임도 지지 아니한다.

실수나 누락으로 인하여 비난받고 싶지는 않다. 어쨌든 눈의 띄게 틀린 내용이나 개선할 사항에 대한 제안이 있으면 알려주기 바란다. 물론 최신 버전이 맞는지는 확인하기 위하여 (목차에 나오는) 이 문서의 버전 번호와 날짜를 검사해보기 바란다. 만약, 3개월 이상 지난 문서라면 http://metalab.unc.edu/LDP/ 에 있는 리눅스 문서화 프로젝트의 홈페이지에 새 버전이 있는 지를 확인하기 바란다.

현재의 이 문서는 적당한 베타 수준이라고 보면 된다. 나는 이 문서를 1997년에 쓰기 시작하였고 시간이 허락하는한 계속 갱신해 왔다. 오픈 소스 커뮤니티의 개발은 빠른 속도로 계속되고 있기 때문에 이 문서를 최신판으로 유지한다는 것은 대단히 어려운 일이다. 따라서 일부 절에는 쓸모없는 정보가 들어 있을 수 있다.

간단히 말하자면 여기에 있는 정보가 정확하다는 것을 나는 보장할 수 없다. 그렇지만 도움이 된다면 좋은 일 아니겠는가 !

1.3. 도움 요청

만약 이 문서가 도움이 되어서 그 감사의 표시를 하고 싶다면 지역의 푸드 뱅크에 음식을 보내주는 것을 생각해보기 바란다.

2장. 소개

리눅스 **2.2.0 (1999**년 **1**월 **25**일 발표**):** 전세계의 석권을 향하여**...**

아마 당신은 리눅스 초보일 것이다. 그래서 때때로 필요한 리눅스의 설정방법과 관리업무에 대한 요약본을 찾고 싶었을 것이다. 정말로 그렇다면 이 문서야 말로 당신이 찾던 문서다 !

2.1. 범위

이 문서는 리눅스의 설치와 설정에 관한 사항과 리눅스 기반의 서버나 데스크탑 컴퓨터가 정상적으로 실행되도록하기 위하여 지켜야 할 관리와 유지보수 절차에 대하여 요약할 생각이다. 개인 사용자는 물론이고 기업 사용자도 대상으로 한다. 유닉스에 대하여 상세한 정보가 필요하다면 좋은 책들이 이미 나와있고 심지어는 인터넷에서도 구할 수 있기 때문에 유닉스 전반에 대한 설명을 하지는 않을 생각이다.

일반적으로 여러분의 리눅스 시스템은 조금만 사용자가 유지보수를 해주면 운영할 수 있다. 반복적인 일들 예컨대 시스템 로그 파일을 순환시키거나 버리는 일 등은 자동화가 가능하다. 따라서, 대부분의 경우에 극히 조금만의 사용자 간섭만으로도 리눅스는 잘도 업무를 수행할 수 있다. 하지만 특별한 요구사항이 있거나 시스템의 장애가 발생한 경우 이 문서가 도움이 될 것이다.

나는 리눅스를 집과 직장에서 동시에 쓰고 있다. 나는 리눅스를 아주 잘 쓰고 있으며 인터넷 서비스와 파일/인쇄 서비스를 나의 사장님에게 지난 4년간 안정적으로 제공하고 있다.

2.2. 리눅스 배포판 선택하기

시중에는 다양한 리눅스 배포판이 나와 있다. 모든 배포판은 같은 기본 리눅스 커널과 시스템 도구를 제공하지만 설치하는 방법과 끼워주는 응용 프로그램이 다르다. 각 배포판에는 일장일단이 있다. 따라서, 선택하기에 앞서 각각의 배포판이 어떤 기능을 제공하는지 조사하는데 시간을 투자할 필요가 있다.

각 배포판에 대한 정보와 어떻게 다운로드 또는 구매할 수 있는지를 설명한 웹 사이트의 목록을 열거하면 다음과 같다 :

http://www.redhat.com/

상용 리눅스 배포판 업체인 레드햇 소프트웨어사가 제공하는 레드햇 배포판은 가장 인기있는 배포판이다. 설치 과정은 그래픽 사용자 인터페이스를 사용할 것인지 텍스트 기반의 인터페이스를 사용할 것인지 선택할 수 있으며, 레드햇 6.1은 가장 설치하기 쉬운 리눅스 배포판이다. 레드햇 배포판은 "RPM" 유틸리티를 사용하여 쉽게 패키지를 업그레이드하거나 관리할 수 있는 기능을 제공하며 X 윈도우 시스템을 위한 구이 윈도우 관리자로 인기가 있는 그놈 - GNU Network Object Model Environment(**GNOME**) - 과 **KDE** - "K Desktop Environment" - 를 동시에 제공한다. 이 배포판은 인텔과 알파 그리고 스파크용 버전이 있다.

http://www.debian.org/

데비안 배포판은 “The Debian Project”로 알려진 비영리기구가 제공하는 것이며 오픈 소스 진영에서 인기가 있다. 이 배포판도 “dpkg” 유틸리티를 이용한 쉬운 업그레이드와 패키지 관리기능을 제공한다. 이 배포판은 인텔, 알파, 스파크와 모토롤러 (매킨토시, 아미가, 아타리) 용 버전이 나와있다.

http://www.suse.com/

수세 배포판은 상용 소프트웨어 업체인 S.u.S.E. 가 제공하는 것으로서 유럽에서는 선두의 자리를 지키고 있다. 이 배포판에는 **KDE** 가 포함되어 있으며 쉬운 업그레이드와 패키지 관리를 위하여 “YaST” 유틸리티를 제공한다. 이 배포판은 인텔용과 알파용 버전이 있다.

http://www.caldera.com/

오픈리눅스 배포판은 상용 소프트웨어 업체인 칼데라사가 제공하며 기업 사용자들 목표로 하고 있다. 새 버전인 오픈리눅스 2.2 을 통하여 칼데라사는 쉬운 리눅스 배포판 설치의 기준을 한단계 높였다. 그리고 표준으로 **KDE** 를 제공한다. 이 배포판은 인텔용 버전만 있다.

http://www.linux-mandrake.com/

맨드레이크 배포판은 상용 소프트웨어 업체인 MandrakeSoft S.A. 가 제공하는 것이며 사용자의 선택에 따라 레드햇 또는 데비안 배포판에다가 원래 배포판에 들어있는 것 보다 더 가치있는 소프트웨어 패키지를 포함하여 제공한다.

http://www.slackware.com/

슬랙웨어 배포판은 월넛 크리크 소프트웨어사의 패트릭 폴커딩에 의하여 제공된다. 패트릭 폴커딩은 현재와 같은 리눅스 배포판의 원조에 해당하는 사람이다. 설치과정은 꽤 간단하지만 업그레이드와 패키지 관리 기능은 취약하다. 여전히 libc 라이브러리에 기반하고 있다. 물론 다음 버전에서는 새 버전인 glibc로 옮겨갈 것으로 알려지고 있다. 이 배포판은 리눅스에 친숙하고 기술이 뛰어난 사람들에게 추천할 만 하다. 이 배포판은 인텔 버전만 있다.

모든 배포판을 열거하는 것은 이 문서의 범위를 넘어서는 것이기 때문에 가장 인기있는 몇개만 열거하였다. 물론, 더 많은 정보를 원한다면
http://metalab.unc.edu/LDP/HOWTO/Distribution-HOWTO.html 에서 찾을 수 있는
“**Distribution-HOWTO**” 가이드를 참조하기 바란다.

작은 정보**:** 팁 : 씨디롬으로 된 배포판을 사기로 결정했다면 여러 리셀러를 둘러 봄으로써 싸게 살 수 있을 것이다. (예를 들어, 나는 인터넷 기반의 소프트웨어 판매회사인 http://www.cheapbytes.com/ 와 여러 번 거래하였는데 상당히 만족스러웠다.) 한편, 배포판 제공 회사에 상대적으로 많은 돈을 지불함으로써 그 배포판이 계속 개선되기를 기대하는 방법도 있다.

내가 선택한 배포판은 레드햇 리눅스다. (게다가 재론의 여지없이 리눅스 사용자에게 가장 인기있는 배포판이기도 하다.) 거의 3년간 나는 끈질긴 슬랙웨어 추종자였다. (그 전에 커널 0.90a 시절까지 거슬러 올라가면 tsx-11에서 나온 자그마한 배포판에 얽혀 있기도 하였다.) 예

전에도 레드햇을 시험삼아 써보기는 하였지만 그에 대하여 좋은 얘기를 할 만하지는 않았다. 그러다가 레드햇 5.1을 써보고는 생각이 완전히 변하였다! 내 의견에 5.1에 이르러서야 레드햇은 마침내 "제대로 방향을 잡았"다.

내가 레드햇 배포판의 팬이 된 이유로는 쉬운 설치, 다양한 플랫폼 지원 (최근까지도 레드햇은 인텔, 알파, 솔라리스 - 스파크가 맞는 표현 (역자) - 플랫폼을 지원하는 유일한 업체였다.) 그리고 무엇보다도 RPM 패키지 관리자를 들 수 있다. 게다가 레드햇은 배포판에 포함된 패키지의 개정판이 나오는 대로 회사의 FTP 사이트 (ftp://ftp.redhat.com/redhat/updates/) 에 올려두는데 이는 시스템을 최신판으로 유지하고 때때로 발견되는 버그나 보안상의 문제점을 제거하는 좋은 방법이다.

처음에는 별다른 용도로 쓸데가 없던 컴퓨터에 테스트 목적으로 설치하였다가 우리의 인터넷 / 파일 & 인쇄용 서버 중 두대를 슬랙웨어에서 레드햇 5.1로 바꾸었으며 그 뒤로 후회를 해본 적이 없다. 나는 내 시스템과 집에 있는 시스템과 다른 세개의 간단한 서버에도 설치하였다. 게다가 인텔 버전은 물론이고 알파 버전과 스파크 버전도 갖고 놀 기회를 가졌다. 최근에 나는 내가 관리하는 모든 리눅스 시스템을 레드햇 6.1로 바꾸었다.

따라서, 이 문서는 명백히 레드햇 "풍"이 될 것이며 그것도 인텔 기반의 6.1 버전과 관련이 깊을 것이다. 하지만 기대하기로는 대부분의 또는 최소한 일부의 정보는 다른 배포판의 사용자에게도 유용할 것이다.

3장. 리눅스 개요

리눅스 세계에 오신 것을 환영합니다!

3.1. 리눅스는 무엇인가?

리눅스는 인텔, 스파크, 알파, 파워피씨를 포함한 다양한 플랫폼에서 실행가능한 진정한 32비트 운영체제이다. (플랫폼 중 일부는, 예컨대 알파에서는, 실제로는 64비트이다.) 다른 기계에 포팅된 사례도 있지만 나는 그에 대한 경험이 없다.

리눅스가 당시 핀란드의 대학생이었던 리누스 토발즈에 의하여 처음 만들어진 것은 1990년대 초반으로 거슬러 올라간다. 리누스는 집에 "예술의 경지"인 386 박스를 갖고 있었으며 286 기반의 미닉스 시스템 (작고 유닉스와 유사한 운영체제로 운영체제 수업에 사용할 목적으로 만들어졌다) 의 대체품을 당시로서는 새로운 이 칩에만 있는 추가의 명령을 활용하기 위하여 만들기로 결정하였다. 그리고는 조그마한 알맹이만 있는 커널을 작성하기 시작하였다.

우연히 그는 그의 작은 프로젝트에 관심이 있는 사람들이 살펴볼 수 있도록 하고 어쩌면 공헌을 할 수 있도록 하게 하기 위하여 유즈넷 그룹 comp.os.minix (news:comp.os.minix)에 발표하였다. 그 결과는 엄청난 것이었다!

리눅스에 관련하여 재미있는 것은 그것이 완전히 공짜라는 것이다! 리누스는 프리 소프트웨어 재단의 GNU 카피레프트 라이센스를 채택하기로 결정하였는데 이는 코드가 저작권에 의하여 보호되기는 되는데 항상 다른 모든 사람들에게 제공될 수 있도록 보호된다는 것을 의미한다.

여기서 프리는 자유로움을 의미한다 -- 당신은 그것을 자유로이 얻을 수 있고, 자유로이 사용할 수 있으며, 심지어는 이득을 취하기 위하여 자유로이 팔 수도 있다! (공짜 소프트웨어를 판다는 것이 어쩌면 이상하게 들리겠지만 레드햇을 포함하여 여러 회사가 표준 리눅스 커널, GNU 유틸리티, 자기 "취향" 에 따라 포함시킨 응용 프로그램을 묶어서 배포판으로 팔고 있다. 널리 알려지고 인기있는 배포판으로는 슬랙웨어, 레드햇, 수세, 데비안이 있다.) 대단한 것은 대부분의 상용 소프트웨어 회사들이 자신들의 "대상시장"에 맞춰 프로그램을 만드는 데 반하여, 누구나 소스 코드에 접근할 수 있기 때문에 자신의 필요에 맞춰 운영체제를 고칠 수 있다는 것이다.

리눅스는 유닉스의 완전히 성숙한 구현이라고 생각될 수 있고 생각되어야만 한다. 물론 우리는 리눅스를 "유닉스"라고 부를 수는 없다. 그 이유는 호환성이 없거나 기능이 부족하여서가 아니라 "유닉스"라는 단어가 AT&T사의 등록상표이고 따라서 그 단어의 사용은 라이센스 계약을 통하여만 허가되기 때문이다.

리눅스는 어떤 면에서건 다른 어떤 운영체제만큼은 지원을 받을 수 있고, 믿을만 하며, 성장할 수 있다. (글쎄 내 의견으로는 더하면 더했지 덜 하지는 않는다!) 어쨌든 리눅스는 그 근본과 그 뒤에 깔려있는 철학과 수백만 달러를 퍼붓는 마케팅 캠페인 행사의 부족 때문에 수많은 미신에 시달리고 있다. 사람들은 아직도 이 멋진 운영체제에 대해 배워야할 것이 많다!

3.2. 미신 깨뜨리기

나는 리눅스를 수년간 써왔다. 그리고 나는 내가 운영체제가 무엇이며 무엇을 할 수 있고 없는지 좀 안다고 생각하고 싶다. 열렬한 유즈넷 독자로서 최근의 개발상황은 물론이고 언제나 튀어나오는 다양한 불꽃튀는 논쟁 (갑론을박하는 사람들 같으니라고! ;-)) 을 계속 따라 왔다. 나는 적지 않은 사람이 믿고 있는 내가 맡아야 할 미신들 (퍼드 **FUD** -- "두려움, 불확실, 의심(Fear, Uncertainty, Doubt)" 라고 불리기도 하는데 이는 상업적인 기술 회사들이 자신의 시장이 경쟁하는 기술을 멀리하도록 겁을 주기위하여 흔히 사용되는 수법이다.) 을 알고 있다. 내가 이러한 미신을 향해 돌진해서 박살을 내도록 해보겠다. :-)

- 리눅스는 프리웨어이기 때문에 장난감이다.

사람들은 이윤을 동기로 갖지않은 자원자들에 의해 작성된 소프트웨어는 상용의 소프트웨어보다 질이 떨어질 것이라고 생각하는 경향이 있다.

이는 과거에는 사실이었을런지도 모른다. (내 말은 도스와 초창기 윈도 시절에 진짜 쓰레기 같은 프리웨어가 많이 있었다 는 의미이다) 그러나 요즘에는 확실히 사실이 아니다.

인터넷의 힘은 지구상의 가장 똑똑한 두뇌 중 일부가 모여서 자신들이 재미있다고 생각하는 일을 협력해서 하는 것을 가능하게 하였다. 리눅스나 수많은 GNU 유틸리티, 응용 프로그램 패키지에 도움을 주고 있는 사람들은 다양한 배경을 갖고 있으며 공헌하고자 하는 이유도 각자 다르다.

일부 진짜 골수 해커들은 코딩을 좋아하기 때문에 개발에 참여하고 어떤 사람들은 필요한 소프트웨어(예를 들어, 회사에서 쓸 랜용 트래픽 모니터)가 있어서 그걸 직접 작성하기로 결정했기 때문에 개발을 하고 또 다른 부류인 연구자나 전산학을 하는 사람들은 연구 성과를 내기 위해서 리눅스를 이용하기도 한다.

상용 제품의 경우 패키지가 개발되고 판매될 때 소스 코드는 빼고 최종 사용자에게 판매되는데 반하여 리눅스의 코드는 관심과 능력을 가진 사람들에 의하여 속속들이 관찰되고 디버깅되며 개선된다. 이러한 상세한 검토는 리눅스가 높은 신뢰도와 성능을 발휘하게 되는 원인 중 하나가 된다.

잊지 말것: 인터넷은 그 자체가 오픈 소스 프로젝트에 기반하여 만들어지고 운영되고 있다. 전세계 사람들과 당신이 매일같이 주고 받는 전자우편의 80%가 Sendmail에 의하여 한쪽 또는 양쪽에서 처리되며 당신이 "웹을 서핑"하며 보는 웹 페이지의 50%이상이 아파치에 의하여 처리된다. 이 정도면 신뢰도가 충분한가요?

- 리눅스는 지원을 받을 수 없다.

이 미신을 들으면 구역질이 나려고 한다. 그렇다면 "다른" 회사들은 실제로 지원을 한단 말인가 ? 나는 매우 널리 사용되는 상용 운영체제와 관련하여 개인적인 경험을 갖고 있는데 그 업체의 소위 "지원"이라는 것은 완전히 쓸모없는 것이었다.

무엇보다도 먼저 얘기할 점은 리눅스를 위한 지원이 있다는 것이다. 물론 상업적인 지원이다. 당신이 돈을 내고서라도 지원을 받을 만한 지원을 해주는 회사가 몇군데 있다. 그들은 전화나 전자메일을 통해서 지원을 하기도 하고 집에까지 찾아와서 문제를 해결하기도 한다!

물론 리눅스에서 맞닥뜨리는 상황의 99%는 단 한두번의 대답을 듣고서도 원하는대로 할 수 있을 것이다. 이런 대답을 찾으려면 유즈넷이나 여러 메일링 리스트를 참고하면 된다!

내가 겪은 문제 중에 http://www.dejanews.com/을 검색하거나 comp.os.linux.* newsgroups 중 한곳에 질문을 해서 해결책을 찾지 못한 문제는 없었다. 대개 내가 글을 올리고 세시간내지 열두시간안에 답을 들을 수 있었다.

리눅스의 또 한가지 재미있는 측면은 커널 전체와 운영체제 컴포넌트 대부분의 소스 코드를 자유로이 구할 수 있기 때문에 보안, 서비스 거부, CPU 오류(예를 들어, 인텔의 **F00F** fatal exception)와 같은 결정적인 지원 이슈들을 재빨리 추적하여 해결할 수 있다. 해결에 걸리는 시간은 유사한 또는 같은 문제에 대하여 상용 소프트웨어들이 해결하는 시간과는 단위자체가 다르게 빠르다. 그렇다면, 상용 지원 서비스는 어디에 갔단 말인가?

그 외에도 내가 그 실체를 밝히고 싶은 미신들은 수없이 많다. 하지만 이는 이 문서의 범위를 넘어선다. 어쨌든 미신 추방에 대하여는
http://www.KenAndTed.com/KensBookmark/linux/index.html에 있는 “Linux Myth Dispeller”과
http://www.geocities.com/SiliconValley/Hills/9267/fud2.html에 있는 “The Linux FUDfactor FAQ”
를 참조하기 바란다.

3.3. 사용자의 관점

나는 회사와 집에서 모두 리눅스를 사용한다.

우리 회사에서는 수백명의 사용자에게 인터넷 서비스를 제공하기 위하여 리눅스를 사용하고 있다. 제공되는 인터넷 서비스에는 SMTP와 POP은 물론이고 TACACS (모뎀을 통하여 다이얼인 하는 사용자) 인증, 웹 페이지 호스팅, 프록시 캐싱 등이 포함된다. 그 외에도 NFS 서비스와 SMB 프로토콜(WfW/Win95/WinNT)을 이용한 파일 & 인쇄와 팩스 서비스를 삼바 패키지를 이용하여 제공하고 있다.

집에서 나는 리눅스를 인터넷 서비스, 소프트웨어 개발 그리고 물론 게임(리눅스 박스에서 실행되는 퀘이크II는 아름다움 그 자체이다!)에 이르기까지 개인 용도로 사용한다. 리눅스와 관련하여 내가 좋아하는 것 중 하나는 아무리 못살게 굴어도 죽지 않는다는 점이다. 게다가 유닉스 기술을 배우고 개발하며 유지하는데에도 큰 도움이 된다.

현재 내가 사용하는 리눅스는 레드햇 6.1이다. (더 자세한 정보는 http://www.redhat.com/를 참조하기 바란다.) 이 배포판에는 쉘, 컴파일러 & 인터프리터, 네트웍 지원, X 윈도우 시스템, 모든 인터넷 서비스 (예를 들어, 전자메일, 뉴스, 웹 서버, 텔넷 등) 등 완전한 유닉스 시스템을 구성하는데 필요한 모든 소프트웨어가 들어있다. 이 배포판은 표준 리눅스 커널 2.2.12를 포함하고 있다.

우리 회사에서 기본 인터넷 서버로 사용하고 있는 리눅스 기반 시스템의 구성은 다음과 같다:

- 커널: 2.2.12
- 기계: 펜티엄 II @ 300 MHz (bogo-mips 299.83) PCI-bus, 256 Mb RAM
- 3 Gb 후지쯔 IDE 하드 디스크 하나(/dev/hda)
- 4.4 Gb 퀀텀 파이어볼 스카시 하드 디스크 네개(/dev/sd0 부터 /dev/sd3 까지),

- 24배속 스카시 씨디롬(/dev/scd0),
- 어댑텍 AHA-131 스카시 콘트롤러
- HP 스카시 DAT 테이프 드라이브(/dev/st0 와 /dev/nst0),
- 인텔 EtherExpress Pro 10/100 이더넷 카드

또 다른 시스템은 - 이 시스템은 더 좋은 인텔 박스 - 역시 레드햇 5.2를 사용하고 있으며 다른 사무실에 놓여있다. 이 시스템은 삼바를 이용한 네트웍 파일 & 인쇄 서비스, 스퀴드를 이용한 로컬 웹 캐싱, 보조 DNS 서비스 등을 제공한다. 불행히도 이 기계는 내가 보통 일하는 곳과 50 킬로미터나 떨어진 곳에 있기 때문에 항상 외톨이로 있다 - 그럼에도 불구하고 여전히 나의 자랑이자 기쁨이다! 이 시스템의 간단한 규격은:

- 커널: 2.2.12
- 기계: 펜티엄 II 350 MHz (bogo-mips 349.80) PCI-bus, 256 Mb RAM
- 4.1 Gb 퀀텀 파이어볼 스카시 하드 디스크 하나(/dev/sda)
- 9.4 Gb 퀀텀 파이어볼 스카시 하드 디스크 네개를 하드웨어 RAID 레벨 5 어레이로 묶음(/dev/rd/c0d0, /dev/rd/c0d1)
- 36배속 스카시 씨디롬(/dev/scd0),
- 버스로직 BT-948 스카시 컨트롤러
- Mylex AcceleRAID 250 (DAC960) RAID 컨트롤러
- HP 스카시 DAT 테이프 드라이브(/dev/st0 와 /dev/nst0),
- 인털 EtherExpress Pro 10/100 이더넷 카드

24 기가 바이트 이상의 엄청난 저장공간을 하드웨어 RAID 5 어레이로 묶는 것은 어쩌면 겸손한 느낌도 든다. Mylex RAID 컨트롤러는 매우 잘 되며 하드웨어 RAID 솔루션을 찾는 사람에게 나는 주저없이 추천할 수 있다! (당신의 리눅스 시스템을 RAID 어레이로 구성하고 싶다면 11.2절을 참조하기 바란다.)

그 외에도 알파 하나, 스파크 하나, 인텔 두 개 그리하여 총 네대의 리눅스 시스템이 더 있는데 그 중 두개는 개발용으로 사용되고 있다. 그 외 우리집의 시스템도 있지만 괜히 주저리 주저리 더 쓰고 싶진 않다.

이 문서는 최대한 하드웨어에 무관하도록 작성하겠지만 하드웨어에 대한 얘기가 나올 때에는 내가 어떤 하드웨어를 쓰고 있는지를 아는 것이 도움이 될 것이다.

4장. 설치와 하드웨어 설정

이 장에서는 인텔 시스템에 레드햇 6.1을 설치하는데 필요한 과정을 상세히 설명한다; 설치 과정은 구이(GUI) 기반의 설치를 선택하든 텍스트 기반의 설치를 선택하든 유사하다. 설치에 관한 정보의 대부분은 레드햇의 사용자 가이드 (인쇄된 매뉴얼은 레드햇 "정품" 박스에 들어 있고 파일로는 씨디의 "/doc" 디렉토리에 있으며 온라인으로는 ftp://ftp.redhat.com/pub/redhat/redhat-6.1/i386/doc/rhinst/index.htm 에서 구할 수 있다) 에 잘 설명되어 있으므로 상당부분은 대충 건너뛸 생각이다. 하지만, 내 생각으로는 레드햇 가이드에서 부족하게 설명되었다고 생각되는 부분이 있기 때문에 이러한 부분에 대하여는 더 상세히 다룰 생각이다.

4.1. 설치 디스켓 만들기

리눅스 레드햇 배포판을 시스템에 탑재하기 위한 첫번째 단계는 설치 프로그램을 시작시킬 방법을 찾는 것이다. 이때 주로 사용되는 방법은 설치 디스켓을 만드는 것이다. 물론 당신이 씨디를 갖고 있고 사용하려는 시스템의 바이오스가 씨디로부터 직접 부팅하는 것을 지원한다면 씨디롬으로부터 직접 설치할 수 있다.

그렇지 않다면 설치 디스켓을 만들어야 하는데, 그러려면"boot.img" 파일 (이 파일은 ext2 포맷의 리눅스 부트 디스켓과 추가의 설치 프로그램의 이미지다) 을 플로피 디스켓으로 복사하여야 한다. "boot.img" 파일은 레드햇 씨디롬의 /images 디렉토리에 들어있으며 ftp://ftp.redhat.com (ftp://ftp.redhat.com/) 에서 /pub/redhat/redhat-6.1/i386/images (ftp://ftp.redhat.com/pub/redhat/redhat-6.1/i386/images) 파일 (인텔 박스에 리눅스를 설치하는 경우) 을 다운로드 받아도 된다.

이제 도스, 윈도, 다른 리눅스 또는 유닉스 시스템에서 부트 디스켓을 만들 수 있다. 만들 디스켓은 포맷되지 않은 디스켓을 써도 되고 (도스의 경우)미리 포맷된 디스켓을 써도 된다 - 차이는 없다.

도스에서: 씨디 롬이 드라이브 D: 라면 다음과 같이 타이핑하면 된다:

```
d:
cd \images
..\dosutils\rawrite
```

소스 파일의 이름에는 "boot.img"이라고 입력한다. 목적 파일의 이름에는 "a:" (만들려는 설치 디스켓이 A: 드라이브에 들어있는 경우)이라고 입력한다. "rawrite" 프로그램은 "boot.img" 파일을 디스켓으로 복사한다.

리눅스/유닉스에서: "boot.img" 파일이 현재 디렉토리에 있다고 가정하면 (이렇게 하려면, /mnt/cdrom에 씨디롬을 마운트하고 난 뒤 /mnt/cdrom/images 디렉토리에서 파일이 있는지 보아야할 것이다.) 다음과 같이 타이핑하면 된다:

```
dd if=boot.img of=/dev/fd0
```

"dd" 유틸리티는 입력 파일 ("if") 인 "boot.img" 파일을 출력 파일 ("of") /dev/fd0 (플로피 드라이브가 /dev/fd0 에서 접근가능하다고 가정) 에 복사할 것이다.

당신이 사용하는 리눅스 또는 유닉스 시스템이 플로피 장치에 대하여 쓰기 퍼미션을 주지 않는 이상 이 명령은 수퍼유저로 실행시켜야 한다. (만약 루트의 암호를 알고 있다면, "su" 라고 타이핑하여 수퍼유저가 된 뒤, "dd" 명령을 실행하고나서 "exit" 라고 타이핑하여 일반 사용자 상태로 돌아오면 된다.)

위에서 설명한 어떤 방법을 썼던지 간에 새로운 레드햇 리눅스 시스템을 설치하는데 사용할 부팅가능한 레드햇 6.1 설치 디스켓을 갖게 되었다!

4.2. 리눅스 설치 프로그램으로 부팅하기

새로운 레드햇 시스템의 셋업을 시작하기 위하여 설치 씨디로부터 부팅을 하든지 아니면 설치 디스켓을 시스템의 A: 드라이브에 넣고 시스템의 전원을 켠다. 잠깐 후에 레드햇 설치 프로그램 화면이 나타날 것이다.

대부분의 경우에 설치 과정을 시작하기 위하여 그냥 <Enter>키를 누르면 된다. 그러나 만약 경험이 많은 사용자로서 하드웨어 장치를 정확하게 어떻게 셋업해야하는지 아는 경우에는 "expert" 모드를 선택하여 여기에만 나오는 추가의 정보와 기능을 이용할 수도 있다. (첫번째 설치화면이 나온 상태에서 만약 당신이 아무것도 선택하지 않고 10내지 15초 정도 내버려두면 디폴트 설치 과정이 자동으로 시작된다.)

그러면 언어 (대개는 "English") 와 키보드 종류 (나는 캐나다에 있지만 나는 "US 101-key" 를 선택한다), 그리고 어디에서 (씨디롬인지 아니면 네트웍을 통해서인지) 설치하려는 지를 선택하라고 물어볼 것이다. 레드햇은 어디에서 설치할 것인가에 관한한 매우 유연하다.

대개의 경우에는 (씨디롬 장치에 들어있는) 레드햇 씨디롬으로부터 설치하기 때문에 "Local CDROM" 를 선택하면 된다. 물론 당신의 시스템에 씨디롬 장치가 없다면 그 외 선택가능한 여러가지 방법이 있다.

만약 다른 리눅스 시스템 (또는 NFS 파일 마운팅을 지원하는 다른 시스템)을 갖고 있다면 NFS 마운트로부터 설치하기 위하여 "NFS"를 선택할 수 있다. 이를 위하여는, 리눅스 씨디롬을 다른 시스템에 마운트시키고 (또는, 레드햇 배포판을 다른 시스템의 어딘가에 깔아두어도 된다 -- 모든 것을 FTP를 통하여 다운로드한 뒤 다른 시스템의 하드 드라이브에서 설치할 수 있다) 새로운 시스템이 해당 디렉토리를 접근하는 것을 허용하도록 /etc/exports 파일에 필요한 항목을 넣어주고 (어떻게 NFS를 셋업하고 사용하는지에 대한 상세내용은 7.6절 을 보기바란다) 필요한 세부내용을 입력하면 된다. 다음은 실제 사례이다:

- 레드햇 씨디를 다른 시스템 (예를 들어, "spock"이라고 하자)에 넣는다.
- 씨디를 마운트하기 위하여 다음과 같이 타이핑한다:

  mount /dev/cdrom /mnt/cdrom -t iso9660

- 수퍼 유저로 "/etc/exports" 파일을 편집하여 다음과 같은 항목을 넣는다:

  /mnt/cdrom newsys.mydomain.name(ro)

(이는 newsys.mydomain.name에 있는 새로운 시스템이 "`/mnt/cdrom/`" 디렉토리와 그 이하의 디렉토리에 대하여 읽기전용으로 접근을 할 수 있도록 허가된다는 것을 의미한다.

당신의 새로운 시스템에 할당된 도메인 이름이 없다면 대신 IP 주소를 사용할 수도 있다:

```
/mnt/cdrom 10.23.14.8(ro)
```

(당신의 새 시스템이 10.23.14.8 이라는 IP 주소를 갖고 있다고 가정하고 있다.)

- 다시 한번, 수퍼유저로서 다음과 같이 타이핑한다:

  ```
  killall -HUP rpc.nfsd ; killall -HUP rpc.mountd
  ```

  이 명령은 NFS를 통하여 새로이 내보내기가 된 것이 적용되도록 NFS 데몬과 mountd 데몬을 다시 시작시킨다.

- 이제, 당신의 새로운 시스템에서 "`NFS`" 를 설치 소스로 선택한다. 그러고 나면 네트웍 카드와 IP 셋팅에 대한 정보를 입력하라고 할 것이다. 만약 당신의 시스템이 랜에 연결된 경우에는 스태틱 IP 셋팅을, 케이블 모뎀을 통해서 연결되어 있다면 DHCP 셋팅을 입력하는 것이 보통이다. 당신의 새 시스템에 적절한 정보를 입력하면 된다.
- 그러고 나면 NFS 서버의 이름과 레드햇 디렉토리를 물어볼 것이다. 우리의 경우에는, NFS 서버의 이름으로 "`spock`" 를 레드햇 디렉토리로 "`/mnt/cdrom/`" 를 입력하면 된다.

레드햇을 설치하는 또 다른 방법으로는 당신의 하드 디스크에서 기존의 파티션으로부터 또는 FTP를 통하여 삼바 (윈도 스타일의 네트워킹) 연결을 이용하는 것이다. 이러한 방법을 사용하는 법에 대하여는 레드햇 사용자 가이드를 참조하든지 아니면 될 때까지 애를 쓰던지 (절차는 그렇게까지 복잡하지는 않다!) 하기 바란다.

설치 소스를 선택하였으므로 이제는 "`설치`" 를 할 것인지 아니면 "`업그레이드`" 를 할 것인지를 물어본다. 새로운 시스템을 설치하고자 하는 경우라면 물론 "`설치`" 를 선택하여야 한다. (한편, 나는 기존의 시스템에 새로운 배포판을 결코 업그레이드 하지 않는 사람이다 -- 마이크로소프트 제품에서 너무나도 많은 문제를 겪었기 때문에 시스템의 업그레이드라는 것 전체를 불신하게 된 것 같다. 나는 처음부터 다시 설치하고 나의 개인 파일, 사용자 파일 등은 백업해 두었다가 다시 까는 방법을 선호한다.)

설치 프로그램은 이제 스카시 어댑터를 갖고 있는지를 물어볼 것이다. 여기에서 예라고 대답하면 적절한 드라이버를 선택하라고 물어볼 것이다. 어떤 경우에는 설치 프로그램이 당신이 갖고 있는 어댑터를 자동으로 탐지해주기도 한다.

다음에는 파일 시스템 (즉, 리눅스용 파티션) 을 셋업하라고 할 것이다. 파티션을 만드는데 사용되는 도구는 레드햇이 제공하는 "`Disk Druid`" 와 리눅스의 표준인 "`/fdisk`" 유틸리티가 있다.

이 두 도구는 파티션의 종류와 크기를 지정하는 기능에서는 유사하다. 하지만 디스크 드루이드가 fdisk에 비하여 조금 더 "사용자에게 친숙"하며 조금 더 완전하다. 사실 드라이브를 파티션하기 위하여 fdisk를 사용한다고 하더라도 마운트 위치를 지정하기 위하여 어차피 디스크 드루이드 화면이 뜨게 된다. 그럼에도 불구하고 과거 슬랙웨어의 사용자였던 나는 개인적으로는 항상 fdisk를 사용하는데 이는 습관때문이 아닌가 생각한다! :-)

다음 섹션에서는 파티션 정보를 어떻게 그리고 왜 셋업해야 하는지를 자세히 다룰 것이다.

4.3. 하드 드라이브 파티션하기

왜 어쨌든 파티션을 해야만 하는거지? 글쎄, 하나의 파티션으로 구성된 시스템에서도 리눅스 시스템은 완벽하게 동작할 수 있으며 이렇게 설정하는 것이 약간 더 쉽다. 하지만 저장 장치를 여러개의 파티션으로 쪼개두는 것은 많은 잇점이 있다.

하나의 큰 파티션만 있던 디스크에서도 리눅스가 잘 돌아간다는 것은 사실이지만, 디스크를 최소한 네개의 주요 파일 시스템 (root, usr, home, swap) 으로 파티션해두는 것은 몇가지 장점이 있다. 그 장점은 다음과 같다:

첫째 (부팅하는 중에 또는 수작업으로 fsck를 돌려서) 파일 시스템을 검사하는데 필요한 시간을 줄여준다. 왜냐하면 이러한 검사를 병렬로 처리할 수 있기 때문이다. (그런데, 절대로 마운트된 파일 시스템에 대하여 fsck를 실행하면 않된다!!! 당신은 그 결과 때문에 확실히 후회하게 될 것이다. 예외로서 파일 시스템이 읽기전용으로 마운트된 경우에는 그래도 안전하다.) 또한, 여러 파티션이 있는 시스템에서는 파일 시스템 검사를 하기가 훨씬 수월하다. 예를 들어, /home 파티션에 문제가 있다는 것을 알았다면 그 파티션의 마운트를 해제하고 파일 시스템 검사를 한 뒤 수리된 파일 시스템을 다시 마운트하기만 하면 된다. (반대로, 파티션이 나뉘어 있지 않다면 응급 복구 디스켓으로 시스템을 부팅한 뒤 싱글 유저 모드에서 이러한 수리작업을 진행해야 한다.)

둘째, 여러 파티션이 있으면 당신이 원하기만 한다면 몇개의 파티션은 읽기전용으로 마운트할 수 있다. 예를 들어, /usr 파티션에 있는 파일은 심지어 루트라고 하더라도 건드리지 못하도록 하고 싶다면 /usr 파티션을 읽기 전용으로 마운트하면 된다.

마지막으로, 파티션으로 나누는 것의 가장 중요한 혜택은 파일 시스템을 보호할 수 있다는 것이다. (사용자의 실수나 시스템의 오류로) 파일 시스템에 어떤 일이 벌어지게 되었을 때, 파티션으로 나뉜 시스템에서는 어떤 한 파일 시스템에 있는 파일만 날리게 된다. 만약 파티션을 하지 않은 시스템이라면 모든 파일 시스템을 날리게 될 것이다.

이 작은 사실은 대단한 강점이 될 수 있다. 예를 들어, 당신의 root 파티션이 깨져서 부팅을 할 수 없다면 복구 디스켓으로 부팅하고 root 파티션을 마운드한 뒤 복사가능한 파일들을 home과 같은 다른 파티션으로 복사하고 (또는 백업으로부터 가져와도 된다; 어떻게 파일을 백업하고 다시 복원하는지에 대하여는 8장을 보기 바란다) 다시 복구 디스켓으로 부팅한 다음, "`mount root=/dev/hda3`" (임시로 root 파일 시스템으로 사용할 파티션이 hda 디스크의 세번째 파티션이라고 가정) 라고 타이핑하면 완전히 동작하는 리눅스 시스템으로 부팅할 수 있다. 그러고 나서 마운트 되지 않은 깨진 root 파티션에 대하여 fsck를 실행하면 된다.

나는 파일 시스템이 완전히 망가지는 것을 개인적으로 경험한 적이 있다. 그리고 여러 파티션으로 나누어 두었기 때문에 손상이 특정 파티션에 국한되었다는 것이 매우 기뻤다.

마지막으로, 리눅스는 다른 운영체제(들) (윈도 95/98/NT, BeOS, 그외 무엇이든)을 셋업하는 것을 허용하여 듀얼 (또는 트리플, ...) 부트를 할 수 있기 때문에 이의 장점을 살리기 위해서 추가의 파티션을 셋업할 필요가 있을 것이다. 각각의 운영체제를 위해서 최소한 하나의 파티션은 셋업해야 한다. 리눅스에는 훌륭한 부트 로더(인텔버전은 LILO, 알파버전은 MILO, 스팍버전은 SILO 라고 부른다)가 들어있는데 전원을 켰을 때, 어느 운영체제를 부팅할 것인지를 지정할 수도 있고 적당한 시간이 지나면 선호하는 디폴트 운영체제(아마, 리눅스겠지?)로 부팅이 된다.

디스크는 필요에 맞추어 파티션을 하여야 한다. 인텔, 알파, 스팍 플랫폼에서 나의 경험에 비추어 볼 때, (집에서 쓰는 데스크탑 시스템이건 회사에서 쓰는 인터넷 서버이건 간에) 적절한 양의 작업을 처리하는 적절한 수준의 로드가 걸리는 (기능을 제대로 써먹는) 시스템이면 파티션 크기를 결정하는데 다음과 같은 추정치를 사용하면 꽤 효과적이라는 사실을 발견하였다.

```
주어진 상황:
X Mb/Gb 의 디스크           (예. 2 Gb)
(또는 하나 이상의 디스크가 있는 경우 그 용량의 합이 X Mb/Gb)
계산방법:
(swap)     대략 메인 램의 2배   (예. 64 Mb 시스템이면 swap은 128 Mb)
/ (root)   전체 용량의 대략 10% (예. 200 Mb)
/home      전체 용량의 대략 20% (예. 400 Mb)
/usr       남은 공간 전체        (예. 1272 Mb)
/var      (선택사양 -- 아래를 참조하시요)
/boot     (선택사양 -- 아래를 참조하시요)
/archive (선택사양 -- 아래를 참조하시요)
```

물론 위에서 제시한 크기는 대략의 가이드라인일 뿐이다. 물론 당신의 리눅스 시스템을 무슨 일에 쓸 것인지에 따라 위의 비율을 적당히 조절해나가야 할 것이다. 워드퍼펙트나 넷스케이프와 같은 큰 응용 프로그램이나 일본어 폰트와 같이 많은 양을 추가로 깔고 싶다면 /usr 공간을 좀 많이 확보하는 것이 좋을 것이다.

내가 볼 때에는 항상 /home 에 많은 공간이 비어 있었다. 따라서 사용자가 너무 많은 공간을 쓰지 않도록 하거나 (또는 엄격한 쿼터를 부여하거나) 또는 쉘 계정과 개인 웹 페이지를 제공하려는 것이 아니라면 /home 공간은 줄이고 대신 /usr 를 키우는 것이 좋다.

다음은 다양한 마운트 위치와 파일 시스템 정보에 대한 설명이다. 이를 통하여 어떻게 사용자의 요구에 맞춰 파티션의 크기를 최적으로 배분할 것인지에 대한 아이디어를 얻게 될 것이다:

- **/ (root)** - 임시 파일, 리눅스 커널과 부트 이미지, 중요한 바이너리 파일 (/usr 파티션을 마운트하기 전에 필요한 것들) 그리고 더 중요하게는 로그 파일, 인쇄 작업과 보내는 메일을 위한 스풀 영역, 사용자가 받는 메일 등을 저장하는데 쓰인다. 소스 RPM 파일로부터 RPM 패키지를 만드는 것 처럼, 특정 기능을 수행하는데 필요한 임시 공간으로도 사용된다. 따라서, 전자메일을 많이 사용하는 사용자가 많거나 임시 공간이 많이 필요할 것 같으면 이 공간을 크게 하여야 한다. 파티션 종류는 디폴트 값인 83 (리눅스 네이티브) 으로 두어야 한다. 그리고 부팅가능 플래그를 켜서 여기에 부팅 정보를 저장하도록 하여야 한다.
- **/usr/** - 리눅스에 필요한 대부분의 바이너리 파일과 별도로 설치된 소프트웨어, 웹 페이지, Squid 프록시 캐시, 삼바 공유 서비스, 일부 별도로 설치된 소프트웨어의 로그 파일 등이 여기에 저장되기 때문에 가장 큰 파티션이다. 파티션 종류는 디폴트 값인 83 (리눅스 네이티브) 로 두어야 한다.
- **/home/** - 사용자들에게 쉘 계정을 제공하지 않는다면 이 파티션을 매우 크게 할 필요는 없다. (학교 웹 페이지 처럼) 사용자에게 홈 페이지를 제공하는 경우에는 예외로서 이 파티션의 크기가 큰 것이 보탬이 된다. 마찬가지로 이 파티션의 종류도 디폴트 값인 83 (리눅스 네이티브)으로 두어야 한다.

- **(swap)** - 리눅스는 시스템에 설치된 실제 램보다 더 많은 메모리를 사용할 수 있도록 하는 "가상 메모리" 기능을 제공한다. 스왑 파티션은 이 기능을 수행하는데 사용된다. 대충의 규칙은, 스왑 파티션은 시스템에 설치된 실제 램의 최소 두배가 되어야 한다.

  만약 시스템에 두개 이상의 실제 하드 드라이브가 있다면 여러개의 스왑 파티션을 만들 수 있다. 이렇게 하면 병렬 디스크 액세스의 장점을 살려 스와핑의 성능을 개선할 수 있다. 예를 들어, 네개의 드라이브가 있고 램이 256 Mb인 시스템이 있다면 128 Mb 짜리 스왑 파티션을 네개를 만들어 전체적으로는 256 Mb 램에 512 Mb의 스왑을 만들겠다. (이렇게 하면 전체 가상 메모리의 크기는 768 Mb가 된다.) 파티션의 종류는 82 (리눅스 스왑) 으로 바꾸어 주어야만 한다.

  > 참고**:** 주의 : 리눅스에서 스왑의 크기는 128 Mb로 제한된다고 흔히 오해하고 있다. 이는 과거에는 사실이었지만 최근의 리눅스 배포판에서는 사용하는 시스템의 아키텍쳐에 따라 다르다. (예를 들어, 인텔 시스템에는 스왑 크기를 2 Gb까지 잡을 수 있다.) 그 외의 정보는 "`man mkswap`" 라고 타이핑하면 얻을 수 있다.

- **/var/** (선택 사항) - / (root) 파티션을 더 쪼개고 싶을 때가 있다. /var 디렉토리는 메일 스풀 (보내는 메일과 받는 메일 모두), 인쇄 작업, 프로세스 락 등과 같이 실행중에 필요한 저장공간으로 사용된다. 이 디렉토리를 / (root) 아래에 그냥 두게 되면 (예를 들어) 받는 메일이 많이 와서 파티션을 채울 수도 있으므로 위험하게 된다. / (root) 파티션이 꽉 차게 되면 나쁜 일 (예컨데 시스템 크래시?) 이 벌어질 수 있는데 /var 를 별도의 독자적인 파티션으로 만들어서 이 문제를 피할 수 있다. / (root) 에 할당을 얼마를 하건간데 이를 두배로 하여 절반은 / (root) 로 나머지는 /var 로 하면 나의 경우에는 항상 성공적이었다. 파티션의 종류는 디폴트 값인 83 (리눅스 네이티브) 로 두어야만 한다.
- **/boot/** (선택 사항) - 특별한 환경에서는 (예를 들어 소프트웨어 RAID 구성으로 시스템을 셋업하는 경우) 리눅스 시스템이 부팅하기 위한 파티션을 분리해둘 필요가 있는 경우가 있다. 부팅을 하고 다른 파일 시스템을 읽어들이는데 필요한 드라이버들을 탑재하는데 이 파티션이 사용된다. 이 파티션의 크기는 겨우 몇 Mb 정도로 작다; 나는 대략 10 Mb 정도 (이 정도면 커널과 초기 램디스크 이미지, 백업커널 한 두개 정도를 저장하는데 충분하다) 를 추천하고 싶다. 파티션의 종류는 디폴트인 83 (리눅스 네이티브) 로 두어야 한다.
- **/archive/** (선택 사항) - 여분의 공간이 있다면 별도의 파티션을 예컨데 /archive 라는 이름으로 만들어 두는 것이 유용하기도 하다. 이 /archive 디렉토리는 백업 파일, 크기가 크거나 자주 사용되지 않는 파일들, 삼바 파일 서비스 그 외 아무 것이나 필요하다고 생각되는 것을 저장하는데 사용하면 된다. 이 파티션의 종류는 디폴트 값인 83 (리눅스 네이티브)로 둘 수도 있고 만약 리눅스와 다른 운영체제에서 동시에 활용하고 싶다면 다른 값 이를테면 6 (16비트 도스, 32 M 이상 크기지원) 으로 할 수도 있다.

추가로 드라이브를 더 달게 되면 새 드라이브에는 추가의 파티션이 만들어지고 필요에 따라 다양한 위치에 마운트하면 된다 -- 따라서, 리눅스 시스템에서는 저장공간이 부족한 경우를 걱정할 필요가 없다. 예를 들어, 만약 앞으로 sda6 가 꽉 찰 것이 명백하다면 다른 드라이브를 추가한 다음 넉넉한 크기로 파티션을 만들어 /usr/local 에 마운트하고 현재 /usr/local 에 들어있던 모든 것을 새 드라이브로 옮기면 된다. 이렇게 하더라도 어떤 시스템이나 응용 프로그램도 "깨지지"는 않는다. 왜냐하면 리눅스는 어느 드라이브에 있든지 /usr/local 을 볼 수 있기 때문이다.

파티션의 사례를 보여주기 위하여 나는 (윈도 95와 리눅스 듀얼 부트) 인텔 시스템을 다음과 같은 파티션 구조로 만들어 보았다:

```
   Device Boot    Begin     Start       End    Blocks   Id  System
/dev/hda1   *         1         1       254  1024096+   6  DOS 16-bit >=32M
/dev/hda2           255       255       782  2128896    5  Extended
/dev/hda5           255       255       331   310432+  83  Linux native
/dev/hda6           332       332       636  1229728+  83  Linux native
/dev/hda7           637       637       749   455584+  83  Linux native
/dev/hda8           750       750       782   133024+  82  Linux swap
```

첫번째 파티션인 **/dev/hda1** 는 다른 운영체제 (윈도 95) 를 저장하는데 사용되는 도스 포맷의 파일 시스템이다. 위의 예에서는 그 운영체제에 1 Gb의 공간을 할당하였다.

두번째 파티션인 **/dev/hda2** 는 드라이브의 나머지 부분 전체를 차지하는 ("확장 파티션" 이라고 불리는) 실제 파티션이다. 이 파티션은 나머지 논리 파티션을 묶는데 사용된다. (한 디스크에는 실제 파티션이 4개밖에 있을 수 없는데 위의 경우에는 4개를 초과하는 파티션이 필요하므로 나머지 파티션을 위해서는 논리 파티션 기법을 사용해야만 한다.)

세번째부터 다섯번째 파티션인 **/dev/hda5**, **/dev/hda6**, **/dev/hda7** 는 모든 e2fs 포맷의 파일 시스템으로서 각각 / (root), /usr, /home 로 사용된다.

마지막에 있는 여섯번째 파티션인 **/dev/hda8** 는 스왑 파티션으로 사용된다.

또 하나의 예는 이번에는 알파 박스인데 두개의 하드 드라이브를 갖고 있으며 리눅스로만 부트되는 시스템으로서 다음과 같은 파티션 구조를 갖고 있다:

```
   Device Boot    Begin     Start       End    Blocks   Id  System
/dev/sda1             1         1         1      2046    4  DOS 16-bit <32M
/dev/sda2             2         2       168    346859   83  Linux native
/dev/sda3           169       169       231    130851   82  Linux swap
/dev/sda4           232       232      1009   1615906    5  Extended
/dev/sda5           232       232       398    346828   83  Linux native
/dev/sda6           399       399      1009   1269016   83  Linux native
/dev/sdb1             1         1       509   2114355   83  Linux native
/dev/sdb2           510       510      1019   2118540   83  Linux native
```

첫번째 파티션인 **/dev/sda1** 는 도스 포맷의 파일 시스템으로서 MILO 부트 로더를 저장하는데 사용된다. 알파 플랫폼은 인텔 시스템과는 부팅할 때 사용되는 방식이 약간 다르다. 따라서, 리눅스는 부트 정보를 FAT 파티션에 저장하였다. 이 파티션은 가능한 가장 작은 크기의 파티션이면 충분한데 여기에서는 그 크기가 2 Mb 이다.

두번째 파티션인 **/dev/sda2** 는 e2fs 포맷의 파일 시스템으로서 / (root) 파티션으로 사용된다.

세번째 파티션인 **/dev/sda3** 는 스왑 파티션으로 사용된다.

네번째 파티션인 **/dev/sda4** 은 "확장" 파티션이다. (상세한 내용은 이전의 예를 참조하시요.)

다섯번째와 여섯번째 파티션인 **/dev/sda5** 과 **/dev/sda6** 는 e2fs 포맷의 파일 시스템이며 각각 /home 과 /usr 파티션으로 사용된다.

일곱번째 파티션인 **/dev/sdb1** 은 e2fs 포맷의 파일 시스템이며 /archive 파티션으로 사용된다.

여덟번째이며 마지막 파티션인 **/dev/sdb2** 는 e2fs 포맷의 파일 시스템으로서 /archive2 파티션으로 사용된다.

파티션 정보를 셋업하고 나면 새로운 파티션을 디스크에 기록해야 한다. 이렇게 하고 나면 레드햇 설치 프로그램은 파티션 테이블을 메모리로 다시 가져와서 설치 과정의 나머지 과정을 진행할 수 있도록 한다.

4.4. 스왑 공간 셋업하기

일단 파티션 정보를 셋업하고 "마운트 포인트" (/usr 은 /usr 파일 시스템의 마운트 포인트이다.) 를 지정하고 나면 설치 프로그램은 스왑 공간으로 어느 파티션을 사용할 것인지를 물어본다. 스왑 파티션은 이미 (파티션 번호 # 82 로) 알 수 있기 때문에 <Enter> 을 눌러서 스왑에 사용할 파티션(들)의 포맷을 시작할 수 있다. 나는 혹시 파티션에 숨어있는 문제점이 없도록 하기 위하여 "포맷하는 동안 불량 블럭 검사" 기능을 사용할 것을 권한다. 이 기능을 사용하면 포맷하는 과정은 상당히 느려지지만 그만한 가치가 충분히 있다고 믿는다.

4.5. 포맷할 파티션 선택하기

이제 설치 프로그램은 당신이 리눅스에 할당한 파티션의 목록을 보여주고 새로운 파일 시스템으로 포맷할 파티션이 있는지, 있다면 어느 파티션인지를 선택하라고 할 것이다. 아마, 기존의 시스템을 업그레이드 하는 것이거나 (예를 들어 /home 에) 어떤 정보가 이미 들어있는 경우를 제외하고는 모든 파티션을 포맷하기를 원할 것이다.

여기에서도 마찬가지로 "포맷하는 동안 불량 블럭 검사" 옵션을 사용하기를 권한다.

4.6. 설치할 패키지 선택하기

그 다음에는 시스템 구성부분의 목록이 나오면서 어느 것을 설치할 것인지 물어볼 것이다. 숙달된 리눅스 사용자라면 필요에 맞춰 꼭집어 선택할 수 있을 것이다. 하지만 리눅스가 처음이라면 맨 아래 있는 "전체" 옵션을 선택하면 될 것이다.

나는 보통 내가 필요한 구성부분을 선택하고 설치를 좀 더 세밀하게 조종할 수 있는 "개별 패키지 선택" 옵션을 사용한다.

필요한 구성부분을 선택하였다면 설치를 시작하기 위하여 "Ok" 를 선택한다. 만약, "개별 패키지 선택" 을 사용한다면 어느 패키지를 설치할 것인지 지정하라고 할 것이다. 선택하는 방법은 아주 간단하다. 만약 어떤 패키지가 뭐하는 것인지 확실히 모르면 <**F1**> 키를 눌러서 그 패키지에 대한 간단한 설명을 참조하면 된다.

한두개의 패키지를 잘못 선택하는 것에 대하여 걱정할 필요는 없다. 결국 모든 패키지는 씨디롬 (또는 다른 소스 메체) 에 들어있고 손쉬운 레드햇 RPM 도구를 사용하여 시스템이 실행되는 도중에라도 조정할 수 있다. (상세한 내용은 10.1절 을 참조하시요)

설치하기를 원하는 패키지를 선택하고나면 설치 프로그램은 이제 당신이 정의한 파티션을 포맷할 것이다. 이 과정은 몇분이 걸릴 수 있다. 특히 파티션이 크거나 불량 블럭 검사 기능을 사용하면 오래 걸릴 수 있으므로 이 과정에서 시스템이 멈춰버린 것이 아닌가하고 생각하지 말기 바란다!

포맷이 끝나고 나면, 레드햇 리눅스는 선택된 패키지의 설치를 시작할 것이다. 이 과정은 끝나는데 까지 당신 시스템의 속도에 따라 5내지 15분 정도 걸린다.

4.7. 하드웨어 설정

패키지 설치가 끝나면 레드햇은 시스템에 붙은 장치를 설정하기 시작한다. 최신의 하드웨어라서 아직 리눅스가 완전히 지원하지 못하는 경우를 제외하고, 대부분의 경우에 설치 프로그램은 자동 설정을 제대로 해낸다.

당신이 보게될 프롬프트는 매우 뻔한 것이다:

- 마우스 탐지 (버튼이 두개짜리인지 세개짜리인지도 선택한다. 버튼 두개짜리 마우이면 3버튼 에뮬레이션 기능을 사용하기를 원할 수도 있다.)
- 비디오 카드 탐지
- 모니터 선택
- X 윈도우 시스템을 설정하기 위하여 "`XConfigurator`" 를 실행 (당신은 당신의 카드를 "탐지" 하기를 원할 것이다. 만약 여기에서 에러가 나더라도 걱정할 필요는 없다. 다음에 시스템이 실행되고 난 뒤에라도 X 설정은 할 수 있다; 상세한 내용은 5장 을 참조하시요.)
- 비디오 모드의 선택 (X 윈도우에서 사용할 비디오 모드를 선택하는데 있어 디폴트 값을 선택할 수도 있고 좀 더 튜닝할 수도 있다)
- 랜 설정
- 시계와 시간대 설정
- 초기 기동 서비스 (디폴트 선택이 아마도 최선이겠지만 여기에서도 <**F1**> 키를 눌러서 각각의 서비스가 무슨 일을 해주는지를 볼 수 있다)
- 프린터 설정
- 루트 패스워드 지정 (안전한 것을 선택할 것!)
- 부트 디스크 만들기 [게으름 피우지 마시요! 꼭 하나를 만들어 두세요! :-)]

4.8. LILO 로 부팅하기

다음에는, 설치 프로그램이 당신의 하드 드라이브에 부트 로더를 써야 한다. 부트 로더 (인텔 시스템에서는 **LILO**) 는 리눅스의 부팅을 담당할 뿐만 아니라 멀티부트로 시스템을 셋업한 경우에는 다른 운영체제의 부팅도 담당한다. (이에 대한 상세한 내용은 4.8.1절 을 참조하시요)

"`Lilo 설치`" 대화상자는 부트 로더 이미지를 어디에 기록하여야 하는지를 물어본다. 아마 당신은 첫번째 드라이브 (IDE 라면 /dev/hda, 스카시라면 /dev/sda) 의 마스터 부트 레코드에 설치하고자 할 것이다.

부트 로더를 기록할 장소를 선택하고 나면 두번째 대화상자가 나타난다. 여기에서는 그 외 부팅에 필요한 설정 인자를 입력할 수 있다. 대개의 경우 여기에서 뭔가를 입력할 필요는 없다. 그러나 램이 64 Mb 를 넘으면 리눅스가 나머지 램 영역도 활용하도록 하기 위하여 (그렇지 않으면 리눅스는 맨앞의 64 Mb 만 사용한다) 특별한 인자를 입력하여야 한다. 예를 들어, 시스템에 128 Mb 의 램이 있다면 다음과 같이 입력하여야 한다:

```
append="mem=128M"
```

시스템에 스카시 드라이브가 있고 1024 실린더 이상의 크기를 갖는 파티션에 LILO 가 설치하고 싶다면 "리니어 모드 사용" 옵션을 켜주어야 한다. 만약 그렇지 않은 상황이라도 이 옵션이 뭔가를 해치지는 않으므로 아마 괜찮은 선택이 될 것이다.

4.8.1. 다른 운영체제와 멀티부트로 만들기

마지막으로, 만약 다른 운영체제와 리눅스가 멀티 부트가 되도록 셋업하였다면 파티션의 목록을 보여주는 세번째 대화상자가 나타날 것이다. 여기에서 당신은 다른 운영체제의 이름을 지정할 수 있다. (이 이름은 부팅할 때 "LILO" 가 어느 운영체제로 부팅할 것인지를 물어보는 장면에서 입력할 때 사용한다.) 설치 프로그램은 부팅 가능한 모든 파티션에 대하여 디폴트로 어떤 이름 지정하기 때문에 디폴트 값이 싫지 않다면 굳이 바꿀 필요는 없다.

시스템이 처음 기동할 때 부트할 디폴트 운영체제는 물론 리눅스이다. 하지만 필요하다면 디폴트를 다른 어떤 운영체제로도 바꿀 수 있다.

부트 로더를 하드 드라이브에 설치하고 나면 설치 프로그램은 리눅스가 성공적으로 설치되었다는 것을 나타내기 위하여 "축하" 대화상자를 보여줄 것이다. (만약 있다면) 설치용 플로피 디스켓을 제거하고 <**Enter**> 키를 눌러서 시스템을 리부팅 시켜서 ... 리눅스로 들어간다!

리눅스가 부팅되고 모든 것이 순조롭다면 "login" 프롬프트를 보게될 것이다. 여기서 당신은 설치할 때 지정한 패스워드를 이용하여"루트(root)" 로 로그인할 수 있을 것이다.

4.9. 레드햇 업데이트를 내려받아서 설치하기

레드햇은 이때까지 매우 인상적인 배포판을 만들어왔다. 그러나 "시간이 완전히 무르익" 기전에 배포하는 역사가 있다. 따라서 리눅스 시스템을 완전히 활용하기 위해서는 업데이트 패키지를 내려받아서 적용할 필요가 있다. 이들 패키지는 RPM 유틸리티를 이용해서 적용하기 때문에 "rpm 파일" 이라고 부르기도 한다. (상세한 내용은 10.1절 을 참조하시요)

이 과정은 리눅스 시스템을 준비하는 과정 중에서 (엄청나게 빠른 인터넷 연결을 갖고 있지 않는한) 상당히 시간을 많이 소모하는 과정이 될 것이다. 하지만 시간을 내서 꼭 이 일을 하라! 그 대신 많은 고통을 덜게 될 것이다!

우선 여기에서 모든 파일을 내려받는다:

ftp://ftp.redhat.com/redhat/updates/6.1/i386/

(앞의 예는 인텔 박스에서 리눅스를 사용하고 있다고 가정하고 있다.)

모든 것을 한 디렉토리로 다 내려받은 다음 "`rpm -Uvh *`" 라고 타이핑하기만 하면 모든 패키지를 업데이트해준다. 만약 커널 rpm 파일을 내려받았다면 이는 다른 디렉토리에 옮긴다. 커널을 업그레이드하거나 커스터마이즈하는 것은 좀 복잡하며 매우 주의깊게 하여야 한다. (이에 대한 상세한 내용은 10.4절 을 참조하시요.) 따라서, 업그레이드를 적용하기에 앞서 모든 kernel-*.rpm 파일은 다른 디렉토리로 옮기는 것이 좋다.

업그레이드를 적용하기 위해서 단순히 "`rpm`" 을 모든 패지지에 대하여 한번에 실행 ("`rpm -Uvh *`") 할 수도 있고 만약 원한다면 한번에 하나씩 업그레이드 ("`rpm -Uvh 업그레이드할파일.rpm`") 할 수도 있다. 뒤의 방법은 각각의 업데이트가 에러없이 잘 적용되는지를 확인하고 싶어하는 특이한 사람들이 쓰는 방식이다. :-)

아마 어떤 패키지를 업그레이드하기에 앞서 설치가 되어 있기는 한지 궁금할 수 있다. 또는, 현재 설치되어 있는 패키지의 버전이 무엇인지 알고 싶을 수도 있다. 이러한 것은 모두 RPM 유틸리티를 이용해서 할 수 있다; 자세한 내용은 10.1절 을 참조하시요.

5장. X 윈도우 시스템 설정하기

X 윈도우 시스템 또는 "X" (흔히들 X-Windows 또는 X 윈도라고 잘못 알고 있지만 Windows/윈도는 마이크로소프트 윈도에서 쓰는 용어이다) 는 리눅스 상에서 돌아가는 그래픽 사용자 인터페이스이다. 마이크로소프트 윈도와는 달리, X 윈도우 시스템은 매우 다양한 방법으로 보이거나 운영될 수 있다. 아주 원시적으로 또는 아주 수준높게 동작할 수 있으며, 추하게 또는 아름답게 보일 수 있으며, 매끄럽고 빠를 수도 있고 둔하고 느리게 동작할 수도 있다. (이러한 차이는 "리눅스 대 마이크로소프트 NT" 간의 논쟁이 그렇듯이 품질에 대한 주관의 차이에서 오는 것 같다.)

X 를 제대로 동작하게 하는 것은 쉬울 수도 있고 머리칼이 다 빠지도록 복잡할 수 있다. 이는 리눅스를 처음 사용하는 대부분의 사람들이 불평하는 점이고 나도 설정을 놓고 이루헤아릴 수 없을만큼 많이 싸워왔기 때문에 절감하고 있는 문제이다. 다행히도 리눅스 배포판 새 버전이 계속 나오면서 설정은 점점 쉬워지고 자동화되고 있다. 사실 만약 레드햇 6.1을 사용한다면 아마도 이 문제점에 대하여 걱정하지 않아도 될 것이다.

비록 대부분의 경우에 X 가 자동으로 설정될 수 있지만 예외도 있다; 내 생각에는 비디오 카드의 종류, 시스템에 설치된 비디오 램의 크기, 모니터의 종류와 수평 / 수직 동기화 주파수 (이 정보는 모니터 사용자 설명서의 맨 뒤 페이지에 나온다. 혹시 없으면 웹에서 뒤져보아야 한다.) 정도는 알아야 한다고 본다.

5.1. X-Configurator 를 이용하여 X 윈도우 시스템 설정하기

리눅스 레드햇 배포판에서 X 가 동작하게 하는 방법에는 크게 두가지가 있다. 첫번째 방법은 가장 쉬운 방법으로서 레드햇의 "`Xconfigurator`" 유틸리티를 사용하는 것이다. 이 유틸리티는 하드웨어를 감지하여 필요한 X 소프트웨어를 설치하고 적절히 설정을 해주려고 시도한다.

Xconfigurator 를 이용하여 여러가지 셋팅을 시도하였으나 여전히 성공하지 못하였다면, "`xf86config`" 유틸리티를 사용해야 운이 따를 모양이다. 비록 Xconfigurator 만큼 사용하기 쉽고 매력적이지는 못하지만 설정 과정에 대하여 좀 더 세밀한 조종을 할 수 있다.

마지막으로 만약 여전히 운이 따라 주지 않는다면 수작업으로 "`/etc/X11/XF86Config`" 파일을 수정하여 여러가지 셋팅을 비틀어보는 수 밖에 없다. 만약 이런 경우라면 리눅스 커뮤니티 (상세한 내용은 13.3절 을 참조하시요) 로부터 도움을 받아야할 수도 있다. 진정하시라, 어쨌든 -- 대부분의 경우 Xconfigurator 가 적절히 해주니까!

X 가 동작하기는 하는데 색깔이 풍부하지 않아서 실망할지도 모른다. 이는 X 가 디폴트로 픽셀당 8-비트("**bpp**")를 사용하기 때문이다. 만약 비디오 하드웨어가 지원해준다면 더 높은 컬러 수를 사용할 수 있다.

다양한 색상 설정에 대하여는 "`/etc/X11/XF86Config`" 파일에 다음과 같은 식으로 나와있다:

```
Subsection "Display"
    Depth       24
    Modes       "800x600" "1024x768"
```

```
        ViewPort    0 0
        Virtual     1024 768
    EndSubsection
```

위의 내용은 24-비트를 사용할 때 선택가능한 해상도를 보여주고 있다("Modes" 줄에 있는 800x600과 1024x768); 이들 해상도는 <**Alt**><**+**> 과 <**Alt**><**-**>키를 누름으로써 "실행중에" 전환할 수 있다.

> 작은 정보**:** 팁: 디폴트로 X 는 시작할 때 가능한 한 낮은 해상도에서 시작한다. 만약 나만큼이나 이를 싫어한다면 "`/etc/X11/XF86Config`" 파일을 편집하여 해상도의 차례를 바꾸어주면 (즉, "1024x768" "800x600") 간단히 해결된다.

셋업을 하는 중에 각각의 색상수를 시험해보려면 수작업으로 (24-비트 색상을 실험하고자 한다면)"`startx -- -bpp 24`" 이런 식으로 타이핑하여 내가 원하는 색상수에서 X 가 정상적으로 동작하는지 확인하면 된다.

높은 색상수를 실험해본 결과가 성공적이라서 그것을 디폴트로 쓰고 싶다면, "`/etc/X11/xinit/xserverrc`" 파일을 다음과 같이 만들어야 한다:

```
exec X :0 -bpp 24
```

위의 내용은 X 가 픽셀당 24 비트를 사용하도록 한다. (만약 문제가 있다면 그대신 16이나 32를 시도해야 한다.)

X 설정을 제대로 하였다면 아무 사용자로나 간단히 "`startx`" 라고 타이핑하기만 하면 X 를 시작시킬 수 있다. 그러면 X 구이 (GUI) 가 시작되고 모든 세션을 끝내고 X 를 끝내고 나면 보통의 리눅스 콘솔로 되돌아 온다.

선택적으로 시스템이 부트될 때 X 가 시작되고 항상 실행되게 (이에 대한 상세한 내용은 5.2절 을 참조하시요) 할 수 있다. 이는 "지루한" 흑백 화면을 보고 싶어하지 않는 사람이나 명령어 라인을 가급적 다루고 싶어 하지 않는 사람에게 유용하다.

5.2. X 데스크탑 매니저 사용하기

만약 원한다면 시스템이 부팅할 때 X 데스크탑 매니저("`xdm`") 가 X 윈도우 시스템을 자동으로 시작시키게 할 수 있다. 이렇게 하면 리눅스가 언제나 X 에서 돌아가게 할 수 있다. (물론, <**Ctrl**>-<**Alt**>-<**F1**> 키를 눌러 구이 화면에서 보통의 콘솔로 그리고 <**Alt**>-<**F7**> 키를 눌러 다시 구이 화면으로 전환하는 것은 가능하다.) 이는 매력적이고 사용하기 편리한 환경을 사용자에게 제공하고 번번이 "`startx`" 를 타이핑하는 것을 피할 수 있는 좋은 방법이다.

xdm 이 동작하게 하려면 "`/etc/inittab`" 파일을 편집하여 "`id:3:initdefault:`" 라고 되어 있는 줄을 다음과 같이 바꾸어주면 된다:

```
id:5:initdefault:
```

위와 같이 바꾸어주면 시스템이 부팅할 때 리눅스를 실행 레벨 5 로 바꾸어준다; 이 실행 레벨은 디폴트로 xdm 를 시작시킨다. 확실히 하기 위하여 "/etc/inittab" 파일에서 맨 아래쪽 어딘가에 다음과 같은 줄이 있는지 확인한다:

```
x:5:respawn:/usr/bin/X11/xdm -nodaemon
```

만약 xdm 를 사용하는 상황에서 "bpp" 값을 디폴트인 8 보더 더 높은 값으로 하고 싶다면 (물론 비디오 카드와 모니터가 이를 지원해준다면) "/etc/X11/xdm/Xservers" 파일을 다음과 같이 바꾸어야 한다:

```
:0 local /usr/X11R6/bin/X -bpp 24
```

위와 같이 바꾸어주면 xdm 은 픽셀당 24 비트로 동작한다.

그리고 "/etc/X11/xdm/Xsetup_0" 파일을 편집하여 "xbanner" 로 시작하는 줄에 "#" 글자를 넣어서 다음과 같이 주석처리 하여야 한다:

```
#/usr/X11R6/bin/xbanner
```

이렇게 하면 KDE 세션간에 순간적으로 xdm 배너가 디폴트로 뜨는 것을 막을 수 있다. 심미적인 문제라는 걸 나도 알지만...

> 작은 정보**:** 팁: 가끔 콘솔 화면으로 돌아갈 필요가 있다는 것을 알게 될 것이다. (예를 들어, 어떤 게임은 콘솔에서는 실행되지만 X 에서는 실행되지 않는다.) 콘솔로 전환 하는 방법에는 두가지가 있다. X 에서 콘솔로 잠시 갈 때에는, <**Alt**><**F1**> 키를 누르고 되돌아 오려면 <**Alt**><**F7**> 키를 누르면 된다. 또는, X 를 완전히 끝내고 (좀 더 많은 메모리를 확보하기 위하여) 싶다면 시스템의 실행 레벨을 바꾸도록 "/sbin/telinit 3" 라고 타이핑하면 xdm 이 중단된다. 다시 되돌아오려면 "/sbin/telinit 5" 라고 타이핑하면 된다.

5.3. X 에서 글꼴의 모양을 좋게 하는 법

아주 솔직히 말하자면, X 는 특별히 매력적인 폰트를 갖고 있다고 알려진 적이 결코 없다. 사실 많은 사람들이 못생기고 고약한 글꼴이 X 인생의 팔자라는 주장에 굴복하였다.

다행히도, 극적으로 X 의 모양을 개선하고 사용가능한 글꼴의 수를 증가시키는 것이 가능하다. 사실, 만약 윈도를 갖고 있다면 그 플랫폼에서 트루타입 글꼴을 복사하여 X 에서 사용할 수 있다! 이러한 글꼴 지원기능은 "xfstt" 또는 "xfs" 를 사용함으로써 가능하다.

레드햇 6.1 은 이제 "xfs" 지원기능을 내장하고 있기 때문에 설치하자 마자 매력적인 글꼴 지원이 가능하다. 따라서, 만약 이 버전의 리눅스를 사용한다면 상당히 만족스러울 것이다. 하지만 트루타입 글꼴을 활용하는 것 외에도 좀 더 좋게 개선할 수 있는 부분이 있다.

트루타입 글꼴을 지원하려면, 디렉토리를 하나 만들고 (예를 들어, "/usr/local/share/ttfonts") 윈도 시스템에서 글꼴 파일 ("c:\windows\fonts" 디렉토리에 들어 있다) 의 전부 또는 일부를 그 디렉토리로 복사한다.

작은 정보: 팁: 사용할 수 있는 트루타입 글꼴이 없다면 마이크로소프트 사이트인 http://www.microsoft.com/typography/fontpack/default.htm 에서 직접 내려받을 수도 있다.

글꼴을 이용하기 위해서는, 새로 만든 "ttfonts" 디렉토리에서 (루트로서) 다음과 같이 타이핑한다:

```
ttmkfdir -o fonts.scale
mkfontdir
```

그런 다음, "`/etc/X11/fs/config`" 파일을 편집하여 새로운 글꼴 디렉토리를 기존의 디렉토리 목록에 추가한다. 그리고 `default-point-size` 를 120 에서 140 으로 바꾸어 줘서 글꼴이 좀 더 크게, 읽기 좋게 되도록 해준다.

마지막으로, (아직 X 를 빠져나가지 않은 상태라면) X 를 빠져나가서, xfs 서버를 다음과 같이 다시 시작한다:

```
/etc/rc.d/init.d/xfs restart
```

그리고나서, X 를 다시 시작시켜 멋진 글꼴을 즐기면 된다!

X 에서의 글꼴 지원을 개선하는데 대하여는 http://www.frii.com/~meldroc/Font-Deuglification.html 라는 곳에 "**XFree86 Font Deuglification Mini HOW-TO**" 라는 훌륭한 자료가 있다.

5.4. X 용 윈도우 매니저 선택하기

이제 윈도우 매니저를 선택하여야 한다. X 윈도우 시스템은 컴퓨터 하드웨어에 그래픽이 나타나게 해주는 기능만을 담당한다; 그리고 윈도우 매니저는 X 가 어떤 모습으로 보일 것인지 그리고 사용자와 응용 프로그램과는 어떻게 상호 동작할 것인지를 책임진다.

리눅스의 레드햇 버전에는 fvwm, olvm, twm, AfterStep 등 여러개의 윈도우 매니저가 들어있다. 처음 X 를 띄울 때 보게 될 디폴트 윈도우 매니저는 윈95와 유사한 환경을 제공하는 fvwm95 이다.

개인적으로 보면 내 취향과 일반적인 제안은 다르긴 하지만, 나는 그놈 (GNOME) 또는 KDE (또는 둘다!) 를 추천한다. 이들의 설치에 대하여는 다음 두개의 섹션에서 다룰 것이다.

5.5. 그놈의 설치와 설정

GNU 네트웍 객체 모델 환경 (GNU Network Object Model Environment, GNONE, 그놈) 은 X 윈도우 환경을 개선시키는 윈도우 환경이다. 쓸모 있는 대량의 응용 프로그램을 포함하고 있으며 완전히 갖춰진 환경이다. 물론 이 글을 쓰고 있는 시점을 기준으로 보면 그놈에는 여전히 몇몇의 사소한 버그가 있어서 가끔 종잡을 수 없는 행태를 참아야 할 수도 있다. 하지만, 꽤 안정적이며 확실히 쓸모가 있다!

레드햇 6.1을 쓰고 있다면 (최소한 이 글을 쓰는 시점을 기준으로) 가장 최신 버전의 그놈이 배포판에 포함되어 있다. 그 외의 경우에는 최신판 패키지의 RPM 배포판을 내려받아야 한다. 이 글을 쓰는 시점을 기준으로 레드햇 6.0 i386 시스템을 위한 RPM 파일은 ftp://ftp.gnome.org/pub/GNOME/RHAD/redhat-6.0/i386/ 또는 그의 미러 사이트에 들어 있다.

> 참고: 주의: 레드햇 6.0 을 쓰고 있다면 여기에는 상당히 버그가 많은 그놈 버전이 들어있으므로 주의하여야 한다. 따라서, 앞에서 설명한 FTP 사이트에서 최신판의 RPM 을 내려받아야만 한다.

필요한 모든 파일을 받고나면, "루트" 로서 다음과 같이 간단히 타이핑함으로써 그놈 패키지를 설치할 수 있다:

```
rpm -Uvh gtk*.rpm *.rpm
```

(위의 명령은 GTK 라이브러리가 먼저 설치되어 있도록 해주기 때문에 상호의존성에 의한 오류를 피할 수 있다.)

통상적인 믿음과는 달리, 그놈은 사실 윈도우 매니저가 아니라, 사용자가 선호하는 윈도우 매니저 위에 올라앉아 기능을 더해주는 역할을 한다. 따라서, 일단 그놈을 설치하고 나면 어느 윈도우 매니저를 쓸 것인지 선택하여야 한다. 그리고 당신의 디렉토리에 "`.xinitrc`" 파일을 만들어서 적당한 윈도우 매니저를 로드하고 나서 그놈을 시작하도록 해주어야 한다. 파일은 다음과 같은 모양이 될 것이다:

```
afterstep &
exec gnome-session
```

위의 파일에서는 먼저 윈도우 매니저로서 AfterStep 을 로드하고 그 다음에 그 위에서 그놈을 실행한다.

GNU 네트웍 객체 환경에 대한 더 자세한 정보는 그놈의 웹 사이트인 http://www.gnome.org/ 를 참조하면 된다. 그리고 잊지말고 http://www.gnome.org/screenshots/ 에서 스냅샷도 감상하시라.

5.6. KDE 설치와 설정

K 데스크탑 패키지 (KDE) 는 또 하나의 인기있는 윈도우 매니저이며 이 글을 쓰는 시점에는 그놈보다 어느정도 더 성숙된 프로그램이다. 하지만 그놈 보다는 조금 더 많은 메모리를 필요로 하는 것으로 보이므로 시스템에 들어 있는 램의 크기를 고려할 필요가 있다. (당신의 시스템이 64 Mb 미만의 램과 128 Mb 미만의 스왑을 갖고 있다면 그놈을 사용하는 것이 나을 것이다.)

KDE 를 설치하는 첫번째 단계는 패키지의 최신 RPM 배포판을 내려받는 것이다. 그러려면 http://www.kde.org/mirrors.html 에 들어가서 적당한 FTP 미러를 골라야 한다. 지리적으로 당신과 가장 가까운 미러를 선택하되 자주 갱신되는 사이트인지 (목록을 보면 알 수 있을 것이다) 확인해 보아야 한다.

적당한 미러를 찾았다면 당신의 플랫폼과 당신이 사용하는 레드햇 버전에 맞는 모든 RPM 파일을 내려받는다. 예를 들어, 만약 당신이 인텔 플랫폼에서 레드햇 5.2 (또는 그 이상) 를 사용하고 있다면, FTP 미러의 디렉토리에서 "`/pub/mirrors/kde/stable/latest/distribution/rpm/RedHat-5.2/i386/`" 에 있는 모든 파일을 내려받아야 할 것이다.

필요한 모든 파일을 받았다면, 다음과 같은 (KDE rpm 파일이 들어있는 디렉토리 있다고 가정하고) 간단한 명령을 "루트" 로 실행함으로써 KDE 패키지를 설치할 수 있다:

```
rpm -Uvh qt*.rpm
install-kde-1.1-base
```

위의 명령은 Qt 라이브러리를 먼저 설치하고나서 KDE 기본 패키지를 설치한다. 이 작업이 끝나고 나면 로그오프 한 뒤 다시 로그인하여 (또는 "su" 명령을 이용하여 루트가 된 것이라면 exit 하여 나간뒤 다시 "su" 하여 루트로 들어와서) 실행 경로 환경을 적절히 바꾸어 주기 위하여 다음과 같이 타이핑한다:

```
install-kde-1.1-apps
```

위의 명령은 응용 프로그램을 설치한다.

설치 과정에 대한 더 자세한 설명은 내려받은 KDE 파일에 포함되어 있는 "`readme-redhat-rpms.txt`" 파일에 들어있다.

모든 것이 순조롭게 진행되고 KDE 가 에러없이 설치되고 난 뒤, 필요하다면 "루트" 로서 다음과 같이 타이핑함으로써 KDE 를 모든 사용자의 디폴트 윈도우 매니저로 ("`startx`" 라고 타이핑하면 보게될 화면으로) 설정할 수 있다:

```
/opt/kde/bin/usekde userid
```

(**userid** 를 실제 사용자의 id로 꼭 바꾸어 주어야 한다!)

K 데스크탑 환경에 대한 자세한 정보는 KDE 의 웹 페이지인 http://www.kde.org/ 를 참조하기 바란다. 그리고, http://www.kde.org/kde2shots.html 에 있는 화면도 잊지 말고 보기 바란다.

6장. 일반 시스템 관리의 이슈

6.1. 루트 계정

"루트" 계정은 유닉스 시스템에서 가장 강력한 권한을 갖는 계정이다. 이 계정은 계정의 추가, 사용자 패스워드의 변경, 로그 파일의 검사, 소프트웨어의 설치 등 시스템 관리의 모든 측면을 수행할 수 있는 힘을 준다.

이 계정을 사용할 때에는 최대한 조심하여야 한다는 점에 유의하여야 한다. "루트" 계정에는 어떤 보안상의 제약도 가해지지 않는다. 따라서 관리 업무를 수행하는데 있어서 어떤 혼란도 없이 쉽게 작업을 할 수 있다는 것을 의미한다. 하지만, 시스템은 당신이 하고 있는 일을 당신이 정확하게 알고 있다고 가정하고 있으며 따라서 당신이 요청하는대로 모든 일을 실행한다 -- 아무것도 묻지 않고. 따라서, 명령어를 잘못 타이핑하여 중요한 시스템 파일을 지워버리는 것도 쉬운 일이다.

만약 "루트" 로 로그인하거나 역할을 하게 되면 쉘은 프롬프트의 (만약 bash를 사용한다면) 마지막 글자로서 '#'를 보여준다. 이는 이 계정이 절대적인 권력을 갖고 있다는 사실을 당신에게 알려주기 위한 역할을 한다.

가장 간단한 규칙은 절대적으로 필요한 경우가 아니라면 "루트" 로 들어가지 않는 것이다. 그리고 일단 "루트" 가 되었다면 명령어를 조심스럽게 타이핑하고 리턴키를 치기 전에 다시 한번 확인하는 것이다. 그리고 필요한 일을 마치자마자 "루트" 에서 빠져나와야 한다. 마지막으로 (다른 계정에 대하여도 마찬가지이기는 하지만 특별히 이 계정에 있어서 중요한 것은) 패스워드를 안전하게 지키는 것이다!

6.2. 사용자 계정 만들기

> 주의
>
> (주의: 슬랙웨어 기준임. 레드햇을 위해서는 갱신하여야 할 필요가 있음)

이 섹션에서는 당신의 리눅스 시스템에서 섀도우 패스워드를 사용하고 있다는 것을 전제로 하고 있다. 만약 현재 사용하지 않고 있다면, 이 기능이 보안을 어느 정도 강화해주므로 사용하는 것을 고려해 볼 필요가 있다. 섀도우 기능은 설치하기가 간단하며 기존의 섀도우가 아닌 패스워드 파일을 새로운 섀도우 포맷으로 자동으로 변환까지 해준다.

새로운 사용자 계정을 만든 것은 두단계로 구성된다. 첫번째는 계정 그 자체를 만드는 것이고 두번째는 전자메일 주소를 위한 별명을 (우리 회사에서는, "이름.성@도메인.이름" 이라는 규칙을 사용한다) 만들어주는 것이다.

계정을 만들려면 사용자에게 할당할 사용자 이름을 결정하여야 한다. 사용자 이름은 최대 8자까지이며 가급적 성을 이용하거나 이미 같은 성을 쓰는 사람이 있으면 (adduser 스크립트는 계정 이름이 중복되지 않도록 확인해준다) 이름의 첫자를 따서 만드는 것이 좋다.

그러고 나면 다음과 같은 기타 정보를 입력하라고 할 것이다: 사용자의 진짜 이름, 사용자 그룹 (대개는 디폴트 값을 씀), 사용자 번호 # (자동으로 할당됨), 홈 디렉토리 (자동으로 할당됨), 사용자 쉘, 패스워드 만기일 관련 정보 그리고 마지막으로 원하는 패스워드 (패스워드를 입력할 때에는 화면에 나타나지 않는다; 패스워드는 보안상의 이유로 6 ~ 8 자로 하도록 하여야 한다).

모든 것 을 소문자로 입력해야만 한다는 점에 유의하여야 한다. 예외로 사용자의 진짜 이름은 "어떤 형태로든" (예. Joe Smith) 입력해도 되고 패스워드도 마찬가지이다. 사용자 계정 정보에서는 대소문자를 가리기 때문에 사용자 이름이나 패스워드를 입력할 때 대소문자를 정확히 맞추도록 하여야 한다.

다음은 Joe Smith 라는 사용자를 추가하는 과정을 보여주는 사례이다:

```
mail:~# /sbin/adduser
User to add (^C to quit): smith
That name is in use, choose another.
User to add (^C to quit): smithj
Editing information for new user [smithj]
Full Name: Joe Smith
GID [100]:
Checking for an available UID after 500
First unused uid is 859
UID [859]:
Home Directory [/home/smithj]:
Shell [/bin/bash]:
Min. Password Change Days [0]:
Max. Password Change Days [30]: 90
Password Warning Days [15]:
Days after Password Expiry for Account Locking [10]: 0
Password [smithj]:</ FL1539
Retype Password:</ Fl1539
Sorry, they do not match.
Password:</> FL1539
Retype Password:</ FL1539
Information for new user [smithj]:
Name: Joe Smith
Home directory: /home/smithj
Shell: /bin/bash
Password: <hidden>
Uid: 859        Gid: 100
Min pass: 0     maX pass: 99999
Warn pass: 7    Lock account: 0
public home Directory: no
Type 'y' if this is correct, 'q' to cancel and quit the program,
or the letter of the item you wish to change: Y
```

다음 단계는 사용자의 전자메일에 대하여 별명을 만들어주는 것이다. 이는 사용자들이 전자메일 주소를 계정 이름으로 할 것인지 아니면 다른 사람들이 전자메일 주소를 "좀 더 쉽게" 추

측할 수 있도록 하기 위하여 자기의 이름을 (이름.성 형식으로) 쓸 것인지 선택할 수 있게 해준다.

전자메일 별명을 추가하려면 "/etc/aliases" 파일을 다음과 같이 수정하면 된다:

```
mail# pico -w /etc/aliases
```

파일의 맨 아래에다가 새로운 별명을 추가한다. 별명을 추가하는 형식은 다음과 같다:

```
이름.성:사용자이름
```

사용자에게 이름 표기에 있어 어떤 것을 선호하는지 물어보아야 할 것이다 (예컨대 Joseph.Smith 또는 Joe.Smith 중 어느 것으로 할 지) 예의 Joe Smith 라는 사용자에 대하여는 다음과 같이 입력하면 될 것이다:

```
Joe.Smith:smith
```

별명의 추가를 끝내고 나서, <**Ctrl**>-<**X**> 를 눌러서 파일을 저장한다. 그리고 나서 "newaliases" 라고 타이핑하여 별명 데이터베이스를 갱신해준다.

바로 이 시점부터 사용자 계정이 만들어져서 사용할 수 있게 된다. 새로운 사용자에게 사용자 이름은 소문자로 입력하여야 하며 패스워드는 대소문자를 정확히 구분해서 입력해야 하고 전자메일의 주소는 무엇인지 (예를 들어, "**Joe.Smith@mail.**도메인**.**이름") 를 알려주는 것이 좋다.

6.3. 사용자 패스워드 바꾸기

사용자를 대신하여 패스워드를 바꾸기 위해서는 우선 "루트" 계정으로 로그인하든지 "su" 하여야 한다. 그런 다음, "passwd 사용자" (여기서 사용자는 패스워드를 바꾸려는 사용자의 계정이름이다) 라고 타이핑한다. 시스템은 새 패스워드를 넣으라고 할 것이다. 패스워드를 입력하는 동안에는 입력하는 내용이 화면에 나타나지 않는다.

사용자가 자기 자신의 패스워드를 바꾸고자 하면, (사용자 이름을 지정하지 않고) "passwd" 라고 타이핑하면 된다. 그러면 확인을 위하여 원래의 패스워드를 입력하라고 하고 그 다음에 새로운 패스워드를 입력하라고 할 것이다.

6.4. 사용자 계정 막아두기

어떤 사용자 계정을 막으려면, 루트로서 "/etc/shadow" 파일을 (섀도우 패스워드를 쓰고 있다고 가정한 경우; 만약 섀도우 패스워드를 쓰지 않는다면 "/etc/passwd" 파일을) 편집하여 (암호화된 형태로 저장된) 패스워드를 "*" 별표로 바꾸어주면 된다. 모든 유닉스 패스워드는 (최대 8자까지 가능하지만) 길이와는 상관없이 13자의 암호화된 문자열로 저장된다. 따라서, 패스워드를 "*" 한글자로 바꾸어 버리면 사용자는 더 이상 로그인할 수 없게 된다.

> 참고: 주의: 이 방법을 사용하면 다시 그 사용자 계정을 쓸 수 있게 하려면 이미 암호화된 패스워드 필드가 변경되었기 때문에 새로운 패스워드를 부여하여야 한다. 이를 해결하는 방법으로 시스템 관리자들이 애용하는 방법은 계정을 막을 때에는 암호화된 패스워드의 맨 앞에 "*" 별표 문자를 덧붙이고 다시 쓸 수 있게 할 때에는 별표만 지워주는 것이다.

"/etc/passwd" 파일과 "/etc/shadow" 파일에 대한 좀 더 자세한 정보는 아래의 6.6절 을 참조하시요.

6.5. 사용자 계정 제거하기

경우에 따라서, 어떤 사용자의 계정을 서버에서 완전히 제거해야 할 때가 있다.

레드햇을 사용하고 있다면, 필요없는 사용자 계정을 제거하는 가장 쉬운 방법은 "userdel" 명령을 사용하는 것이다. 물론, "루트" 로서 타이핑하여야 한다. 예는 다음과 같다:

```
/usr/sbin/userdel baduser
```

위의 명령은 사용자 이름 "**baduser**" 에 해당하는 항목을 "/etc/passwd" 파일과 만약 섀도우 패스워드를 사용하고 있다면 (섀도우 패스워드는 반드시 사용하여야 한다; 상세한 내용은 6.6절 을 참조하시요) "/etc/shadow" 파일에서 찾아서 지워준다.

> 참고: 주의: "/etc/group" 파일은 다른 사용자가 속해있는 그룹을 삭제할 위험성이 있기 때문에 수정하지 않는다. 이는 별일이 아니지만 만약 신경이 쓰인다면 수작업으로 그룹 파일을 편집하여 해당 항목을 삭제하면 된다.

만약 사용자의 홈 디렉토리까지 지우고 싶다면, "userdel" 명령에 "-r" 옵션을 추가하면 된다. 예를 들어:

```
/usr/sbin/userdel -r baduser
```

나는 계정을 곧바로 제거하지 말고 대신 처음에서 그냥 막아두도록 권장한다. 특히, 많은 사용자를 가진 회사 서버인 경우에는 꼭 그래야 한다. 결국 예전의 사용자가 그 계정을 다시 사용하여야 하는 경우도 생길 수 있고 자기의 홈 디렉토리에 들어있는 파일을 요구할 수도 있다. 또는 심지어는 (업무 변경에 따라) 새로운 사용자가 전임 사용자의 파일을 필요로 할 수도 있다. 어떤 경우에라도 "만약의 경우를 대비하여" 이전 사용자의 홈 디렉토리의 백업본을 만들어 두어야 한다. 사용자 계정을 막아두는 상세한 방법에 대하여는 6.4절 을 그리고 백업을 실행하는 상세한 방법에 대하여는 8장 을 참조하시요.

6.6. 리눅스 패스워드 & 섀도우 파일 형식

전통적인 유닉스 시스템에서는 단방향 암호화가 된 패스워드를 포함하여 사용자 계정 정보를 "/etc/passwd" 라는 텍스트 파일에 보관한다. 이 파일은 (예를 들어 "ls"와 같은) 수많은 도구에 의하여 -- 예를 들어, 파일의 소유자 이름을 보여주기 위하여 사용자 번호와 사용자 이름

을 매칭하는데 -- 사용되므로 누구든지 읽을 수 있도록 되어있다. 결과적으로 이는 어느 정도 보안상의 위험이 된다.

계정 정보를 저장하는 또 하나의 방법은, 내가 항상 사용하는 방법인데, 섀도우 패스워드 형식을 같이 사용하는 것이다. 이 방식은 전통적인 방식과 결합하여 /etc/passwd 파일은 호환성 있는 형식으로 정보를 저장한다. 단, 패스워드는 (실제로 /etc/passwd 파일에 저장하지 않고) 대신 "x" 한글자만 넣어둔다. 두번째 파일인 "/etc/shadow" 파일이 암호화된 패스워드와 그 외의 정보, 즉 패스워드 유효기간 등을 저장한다. /etc/shadow 파일은 루트 계정에 의하여만 읽을 수 있기 때문에 보안상의 위험은 줄어들게 된다.

다른 리눅스 배포판은 섀도우 형식을 사용하기 위해서는 섀도우 패스워드 수트를 설치하도록 되어 있는데 비하여 레드햇에서는 간단히 할 수 있다. 두개의 형식을 바꾸기 위하여는 (루트로서) 다음과 같이 타이핑하면 된다:

```
/usr/sbin/pwconv    섀도우 형식으로 변환
/usr/sbin/pwunconv 전통형식으로 되돌아 감
```

섀도우 패스워드를 사용하는 경우에, "/etc/passwd" 파일은 계정 정보를 담고 있으며 그 모양은 다음과 같다:

```
smithj:x:561:561:Joe Smith:/home/smithj:/bin/bash
```

한 줄의 각 항목은 ":" 콜론 문자에 의하여 분리되며 그 내용은 다음과 같다:

- 사용자이름. 최대 8자까지. 대소문자를 구분하며 대개의 경우에는 소문자로만 구성
- 패스워드 항목에는 "x" 문자가 들어있음. 패스워드는 "/etc/shadow" 파일에 저장됨.
- 숫자로된 사용자 번호. 이 번호는 "adduser" 스크립트에 의하여 부여된다. 유닉스는 이 항목과 그 다음의 그룹 항목을 이용하여 어느 파일이 누구에게 속한 것인지를 구분한다.
- 숫자로 된 그룹 번호. 레드햇은 파일 보안을 향상 시키기 위하여 꽤 특이하게 그룹 번호를 사용한다. 보통 그룹 번호는 사용자 번호와 같다.
- 사용자의 진짜 이름. 이 항목의 최대 길이가 얼마인지는 모르겠지만 합리적인 수준으로 (30자 이내) 관리하도록 해야 한다.
- 사용자의 홈 디렉토리. 보통 /home/사용자이름 (예를 들어, /home/smithj) 이 된다. 모든 사용자의 개인 파일과 웹 페이지, 메일 전달 지정 등이 여기에 저장된다.
- 사용자의 "쉘 계정". 대개의 경우에 (내가 개인적으로 선호하는 쉘이기도 한) bash 쉘을 사용하도록 하기 위하여 "/bin/bash" 로 지정한다.

사용자에게 쉘 계정을 제공하고 싶지 않을 때도 있다. 그럴 때에는 에러 메시지를 보여주고 사용자가 로그 오프 시켜버리는 스크립트를 만들어서 그 이름을 예를 들어 "/bin/sorrysh" 라고 한 뒤 이 스크립트를 그런 사용자의 디폴트 쉘로 지정하면 된다.

> 참고: 주의: 예를 들어, 웹 페이지를 갱신할 수 있도록 "FTP" 전송을 허용하여야 한다면 쉘 계정은 "/bin/bash" 로 해주어야 한다 -- 대신 사용자의 홈 디렉토리로 로그인하는 것은 막을 수 있도록 특별한 방법으로 셋업을 해주어야 한다. 이에 대한 자세한 내용은 7.1 절 을 참조하시요.

"/etc/shadow" 파일은 패스워드와 사용자 계정의 유효기간 등을 담고 있으며 그 모양은 다음과 같다:

```
smithj:Ep6mckrOLChF.:10063:0:99999:7:::
```

passwd 파일과 마찬가지로 섀도우 파일도 ":" 콜론 문자로 각 항목을 구분하며 각각은 다음과 같다:

- 사용자이름. 최대 8자. 대소문자를 구분하며 대개의 경우에는 소문자로만 구성된다. /etc/passwd 파일의 사용자이름과 짝을 이룬다.
- 패스워드. 13 글자로 암호화 됨. 비어있는 경우 (즉, ::) 는 (보통은 좋은 방법이 아니지만) 로그인하는데 패스워드가 필요없다는 것을 나타내며, "*" 라고 되어 있으면 (즉, :*:) 그 계정은 막아두었다는 것을 나타낸다.
- 패스워드를 마지막으로 바꾼 날이 (1970년 1월 1일로부터) 며칠째의 날인지 나타내는 항목.
- 최소 며칠이 지난 뒤에 패스워드를 바꿀 수 있는지를 나타내는 항목 (0이면 언제든지 바꿀 수 있다는 의미)
- 패스워드를 반드시 바꿔야 하는 날로 부터 며칠이나 지났는지를 나타내는 항목 (99999 는 무지무지 오랫동안 패스워드를 바꾸지 않았다는 것을 나타낸다)
- 패스워드가 만기가 되었음을 며칠 동안 사용자에게 계속 주의를 줄 것인지 나타내는 항목 (1주일 동안 알려주려면 7)
- 패스워드가 만기가 되어서 계정이 사용불가가 된지 며칠이나 되었는지 나타내는 항목
- 계정이 사용불가가 된 것이 1970년 1월 1일부터 계산하여 며칠째였는지 나타내는 항목
- 미래의 사용을 위해서 잡아둔 항목

6.7. 시스템의 종료와 재시작

터미널 상태에서 시스템을 종료시키려면, "루트" 계정으로 로그인하거나 "su" 하여야 한다. 그리고 나서 "`/sbin/shutdown -r now`" 라고 타이핑한다. 모든 프로세스가 종료되고 난 뒤 리눅스가 종료되기까지는 몇 분이 걸린다. 컴퓨터는 스스로 재부팅할 것이다. 만약 콘솔앞에 앉아 있다면 좀 더 빠른 방법으로서 <**Ctrl**>-<**Alt**>-<**Del**> 를 눌러도 된다. 리눅스가 완전히 끝나는데 까지는 몇 분이 걸리므로 제발 인내심을 갖고 기다려주기 바란다.

시스템을 종료시켜서 멈추도록 (즉, 종료시키고 난 뒤 곧바로 재부팅하지 않는 것) 할 수도 있다. 다시 전원을 넣어주거나 <**Ctrl**>-<**Alt**>-<**Del**> 를 눌러서 재부팅하지 않는한 시스템을 사용할 수 없는 상태로 된다. 이는 시스템을 완전히 꺼서 다른 곳으로 옮긴다든지 할 때 유용하다. 이렇게 하려면, "루트" 계정으로 로그인하거나 "su" 한 뒤 "`/sbin/shutdown -h now`" 라고 타이핑하면 된다. 리눅스는 스스로를 종료시킨 후 "System halted" 라는 메시지를 보여준다. 이때부터는 컴퓨터의 전원을 꺼도 된다.

관리자가 콘솔에 있는 경우에만 시스템을 종료시키는 것이 좋다. 물론 원격지에서 쉘 세션을 통하여 종료시키는 것도 가능하면 문제가 생겨서 시스템이 제대로 재시작하지 못하는 경우 어차피 시스템이 있는 곳까지 가서 어떤 조처를 취해야 하기 때문이다. (하지만, 나의 경우에는 한번도 이런 문제가 생긴 적은 없었다.)

시스템이 부팅하면, 리눅스는 자동으로 시작되며 네트웍 지원, 인터넷 서비스를 포함한 모든 필요한 서비스를 자동으로 올려준다.

작은 정보**:** 팁: 만약 모든 온라인 사용자 (온라인이란 쉘 계정으로 로그인하였다는 의미이다) 에게 어떤 경고 메시지를 보내고 싶다면, “now” 대신에 적당한 시간 값을 넣어주면 된다. 물론, 시스템 종료 경고 메시지도 바꿀 수 있다. 예를 들어, “`/sbin/shutdown -r +5 Hardware upgrade`” 라고 하면 하드웨어 업그레이드를 위해서 시스템을 종료한다는 것을 사용자에게 알릴 수 있다. 그리고 사용자가 결정적인 시간이 다가오기 전까지 필요한 파일을 닫고 로그 오프하도록 주기적으로 계속 경고 메시지를 보내주게 된다.

7장. 설정과 관리

집에서건 회사에서건 나는 레드햇 리눅스 배포판의 표준 설치의 디폴트 설정 셋팅을 전혀 또는 거의 바꾸지 않고 "박스에서 꺼내서 곧바로" 서비스를 시작할 수 있었다.

물론, 모든 인터넷, 파일 & 인쇄 서비스, 기타 우리 회사에서 필요로 하는 모든 서비스를 제공하기 위해서는 조금씩 고쳐야할 것이나 추가로 설치해야할 서비스가 많이 있다. 관리자는 다음과 같은 내용을 알고 있어야 한다:

- "`/etc/rc.d/rc.local`" 파일은 시스템이 시작될 때 실행되며 부팅되면서 시작되어야 할 추가 서비스를 담고 있다.
- 사이트에 따라 수정이 필요한 부분을 확인하기 위하아여 /etc 를 들여다보아야 한다. 이러한 내용에는 다음과 같은 것이 포함된다:
  - "`/etc/inetd.conf`" (finger, echo, chargen 과 같이 필요하지 않은 서비스가 실행되지 않도록 되어있는지 확인해야 한다; 또한 필요에 따라 서비스를 추가하거나 수정하여야 한다)
  - "`/etc/exports`" (NFS 볼륨을 마운트할 수 있는 호스트의 목록을 갖고있다; 자세한 내용은7.6절 을 참조하시요)
  - "`/etc/organization`", "`/etc/nntpserver`", "`/etc/NNTP_INEWS_DOMAIN`" (적절히 설정한다)
  - "`/etc/lilo.conf`" (LILO 부트 로더 -- 부팅할 때 리눅스 커널을 탑재시켜주는 프로세스 -- 가 필요로 하는 정보를 갖고 있다; 자세한 내용은 4.8절 을 참조하시요)
  - "`/etc/sudoers`" (특별한 권한을 주어야할 사용자의 목록과 그들이 실행할 수 있는 명령어를 담고 있다)
  - "`/etc/named.boot`" (DNS 를 위해 사용된다; 자세한 내용은 7.2절 을 참조하시요)
- "`/usr/local/`" 과 그 서브디렉토리에 있는 것들은 추가 패키지이거나 (RPM 이 아니라 타르 파일 같은 것으로 설치한) 기존 버전에 대한 수정판들이다. (또는, 그곳에 설치할 수 밖에 없는 어떤 것들이다.) 이들 파일은, 그 중에서도 특히 /usr/local/src/ 에 있는 파일은, 항상 최신판으로 유지하여야 한다. 이에 대한 상세한 내용은 10장 을 참조하시요.

7.1. 웹 서버와 HTTP 캐싱 프록시의 관리

> 주의
>
> (주의: 이 섹션의 내용은 무시할 것!)

1. 인터넷 사용자를 정상적으로 만든다. "쉘" 계정은 "`/bin/bash`" 이어야 한다 (FTP 를 이용하려면 정상적인 쉘로 되어 있어야 한다).
2. "`cd /home ; chown root.root theuser`" 이는 "theuser" 라는 사용자의 디렉토리를, 보안상의 이유로, 루트의 소유로 만들어준다.

3. "`cd /home/theuser ; mkdir www ; chown theuser.theuser`" 이렇게 하면 사용자의 "www" 디렉토리를 만들고 사용자가 읽고/쓸 수 있도록 소유관계를 지정한다.
4. "`echo "exit" > .profile`" 이렇게 하면 "`.profile`" 파일이 만들어지고 그 내용은 "`exit`" 한 줄이 된다. 사용자가 텔넷을 통하여 로그인하려고 하면 곧바로 끊어지게 될 것이다.
5. "`ls -l`" 명령을 실행하여 디렉토리에 ("." 과 ".."를 제외하고) 딱 두 개의 파일만 있는지 확인한다:
   - **.profile** (root.root가 소유)
   - **www** (theuser.theuser가 소유)

   그 외 다른 모든 파일은 지워도 된다 (예를 들어, "`rm .less ; rm .lessrc`")
6. 만약 사용자가 자신의 전자메일이 자동으로 다른 전자메일 주소로 전송되기를 원한다면 그 전자메일 주소가 달랑 한 줄 들어있는 .forward 라는 파일을 만들어주면 된다.

그게 다다. 사용자는 홈 페이지를 갱신하기 위하여 FTP 를 사용할 수 있을 것이다.

7.2. 도메인 네임 서버 (DNS) 의 설정과 관리

우리 회사에서는 리눅스를 DNS 서버로 사용하고 있다. 성능은 상상이상으로 뛰어나다. 이 섹션에서는 레드햇 배포판에서 표준으로 따라오는 BIND 8.x 를 이용하여 도메인 네임 서비스를 할 수 있도록 DNS 테이블을 설정하는 방법을 다룬다.

> 참고**:** 주의: 레드햇 5.1 과 그 이전 버전은 BIND 4.x 패키지를 사용하였는데 여기서 사용되었던 설정 파일은 약간 다른 형식으로 되어있다. BIND 8.x 는 BIND 4.x 보더 더 많은 기능을 제공하고 게다가 BIND 4.x 은 더 이상 개발되지 않기 때문에 BIND 패키지는 최근 버전으로 업그레이드하는 것을 고려하여야 할 것이다. 이럴 때에는 그냥 BIND RPM 패키지를 (RPM 유틸리티에 대한 상세한 내용은 10.1절 을 참조하시요) 설치하고 설정 파일을 새로운 형식으로 바꾸어 주기만 하면 된다.
>
> 다행히도, 기존의 BIND 4.x 설정 파일을 BIND 8.x 에 맞춰 바꾸는 것은 쉽다! BIND 에 딸려서 제공되는 문서 디렉토리 (예를 들어, BIND 버전 8.1.2 의 경우에는 "`/usr/doc/bind-8.1.2/`") 에는 "`named-bootconf.pl`" 라는 실행가능한 펄 프로그램이 있다. 시스템에 펄이 이미 설치되어 있다고 가정한다면, 설정 파일을 변환하기 위하여 그냥 이 프로그램을 이용하면 된다. 이 프로그램을 이용하는 방법은 (루트로서) 다음과 같이 타이핑하면 된다:
>
> ```
> cd /usr/doc/bind-8.1.2
> ./named-bootconf.pl < /etc/named.boot > /etc/named.conf
> mv /etc/named.boot /etc/named.boot-obsolete
> ```
>
> 이제 "`/etc/named.conf`" 라는 파일을 갖게 되었는데 이 파일은 BIND 8.x 와 "아무런 수정없이" 잘 동작할 것이다. DNS 테이블의 형식은 변하지 않았기 때문에, 기존의 DNS 테이블은 새 버전의 BIND 와 현재 그대로 동작한다.

리눅스에서 DNS 서비스를 설정하는데에는 다음과 같은 과정을 밟게 된다:

1. DNS 서비스가 작동하게 하려면 "/etc/host.conf" 파일이 다음과 같이 되어 있어야 한다:

```
# Lookup names via /etc/hosts first, then by DNS query
order hosts, bind
# We don't have machines with multiple addresses
multi on
# Check for IP address spoofing
nospoof on
# Warn us if someone attempts to spoof
alert on
```

   스푸핑 탐지 기능을 추가하면 DNS 참조할 때 성능상의 약간의 (비록 무시할 수 있는 수준이지만) 저하를 가져온다. 따라서, 이런 것을 크게 걱정하지 않는다면 "nospoof" 과 "alert" 항목은 사용하지 않도록 하여도 된다.

2. "/etc/hosts" 파일을 필요에 따라 설정한다. 전형적인 경우에 여기에 많은 항목을 넣을 필요는 없다. 하지만 성능의 향상을 위해서 (로컬 서버와 같이) 자주 엑세스되는 호스트는 DNS 참조를 하지 않도록 항목을 추가할 수 있다.
3. "/etc/named.conf" 파일은 아래의 예에서도 볼 수 있듯이 DNS 테이블을 가리키도록 설정되어야 한다.

   참고**:** (주의: 여기에서 예로 든 IP 주소는 어디까지나 예시일 뿐이며 반드시 당신이 쓰는 주소로 바꾸어주어야 한다!):

```
options {
// DNS tables are located in the /var/named directory
directory "/var/named";
// Forward any unresolved requests to our ISP's name server
// (this is an example IP address only -- do not use!)
forwarders {
123.12.40.17;
};
/*
 * If there is a firewall between you and nameservers you want
 * to talk to, you might need to uncomment the query-source
 * directive below.  Previous versions of BIND always asked
 * questions using port 53, but BIND 8.1 uses an unprivileged
 * port by default.
 */
// query-source address * port 53;
};
// Enable caching and load root server info
zone "named.root" {
type hint;
file "";
};
// All our DNS information is stored in /var/named/mydomain_name.db
// (eg. if mydomain.name = foobar.com then use foobar_com.db)
```

```
zone "mydomain.name" {
type master;
file "mydomain_name.db";
allow-transfer { 123.12.41.40; };
};
// Reverse lookups for 123.12.41.*, .42.*, .43.*, .44.* class C's
// (these are example Class C's only -- do not use!)
zone "12.123.IN-ADDR.ARPA" {
type master;
file "123_12.rev";
allow-transfer { 123.12.41.40; };
};
// Reverse lookups for 126.27.18.*, .19.*, .20.* class C's
// (these are example Class C's only -- do not use!)
zone "27.126.IN-ADDR.ARPA" {
type master;
file "126_27.rev";
allow-transfer { 123.12.41.40; };
};
```

> 작은 정보**:** 팁: 위의 내용 중에서 `allow-transfer` 옵션에 주목하기 바란다. 이는 DNS 영역의 전달을 주어진 IP 주소로 제한한다. 앞의 예에서, 우리는 123.12.41.40 에 있는 (아마 우리 도메인의 보조 DNS 서버) 호스트에게만 영역 전달을 허용하고 있다. 만약 이 옵션을 빼먹는다면, 인터넷에 있는 임의의 사용자가 그러한 전달을 요청할 수 있게된다. 이렇게 해서 제공되는 정보는 스팸 메일을 보내는 사람이나 IP 스푸핑을 하는 사람들에 의하여 악용될 수 있으므로, 영역 전달은 보조 DNS 서버 아니면 루프백 주소 "`127.0.0.1`" 로만 제한할 것을 강력히 추천한다.

4. 이제 "`var/named/`" 디렉토리에 있는 DNS 테이블을 세번째 단계에서 "`/etc/named.conf`" 파일에 설정한 내용에 따라 셋업할 수 있다. DNS 데이터베이스를 처음으로 설정하는 막중한 일로서 이 문서의 범위를 넘어선다. 참조할 만한 가이드라인이 온라인이나 인쇄된 책으로 몇가지 있다. 어쨌든 아래에 몇가지의 예를 제시하였다.

   "`/var/named/mydomain_name.db`" 전달 참조 파일에 들어 있는 항목의 예:

```
; This is the Start of Authority (SOA) record.  Contains contact
; & other information about the name server.  The serial number
; must be changed whenever the file is updated (to inform secondary
; servers that zone information has changed).
    @ IN SOA mydomain.name.  postmaster.mydomain.name. (
19990811 ; Serial number
3600 ; 1 hour refresh
300 ; 5 minutes retry
172800 ; 2 days expiry
43200 ) ; 12 hours minimum
; List the name servers in use.  Unresolved (entries in other zones)
; will go to our ISP's name server isp.domain.name.com
IN NS mydomain.name.
IN NS isp.domain.name.com.
```

```
; This is the mail-exchanger.  You can list more than one (if
; applicable), with the integer field indicating priority (lowest
; being a higher priority)
IN MX mail.mydomain.name.
; Provides optional information on the machine type & operating system
; used for the server
IN HINFO Pentium/350 LINUX
; A list of machine names & addresses
    spock.mydomain.name.    IN A    123.12.41.40   ; OpenVMS Alpha
    mail.mydomain.name.     IN A    123.12.41.41   ; Linux (main server)
    kirk.mydomain.name.     IN A    123.12.41.42   ; Win-
dows NT (blech!)
; Including any in our other class C's
    twixel.mydomain.name.   IN A    126.27.18.161  ; Linux test machine
    foxone.mydomain.name.   IN A    126.27.18.162  ; Linux de-
vel. kernel
; Alias (canonical) names
    gopher IN CNAME mail.mydomain.name.
    ftp IN CNAME mail.mydomain.name.
    www IN CNAME mail.mydomain.name.
```

"/var/named/123_12.rev" 역 참조 파일의 항목 예:

```
; This is the Start of Authority record.  Same as in forward lookup ta-
ble.
    @ IN SOA mydomain.name.  postmaster.mydomain.name. (
19990811 ; Serial number
3600 ; 1 hour refresh
300 ; 5 minutes retry
172800 ; 2 days expiry
43200 ) ; 12 hours minimum
; Name servers listed as in forward lookup table
IN NS mail.mydomain.name.
IN NS isp.domain.name.com.
; A list of machine names & addresses, in reverse.  We are mapping
; more than one class C here, so we need to list the class B portion
; as well.
    40.41 IN PTR    spock.mydomain.name.
    41.41 IN PTR    mail.mydomain.name.
    42.41 IN PTR    kirk.mydomain.name.
; As you can see, we can map our other class C's as long as they are
; under the 123.12.* class B addresses
    24.42 IN PTR    tsingtao.mydomain.name.
    250.42 IN PTR    redstripe.mydomain.name.
    24.43 IN PTR    kirin.mydomain.name.
    66.44 IN PTR    sapporo.mydomain.name.
; No alias (canonical) names should be listed in the reverse lookup
; file (for obvious reasons).
```

다른 B 클래스 (126.27.* 과 같은) 주소를 매핑 시키기 위해서는 추가로 역 참조 파일을 만들 수 있으며 그 내용은 앞에서 본 것과 유사하다.

5. named 데몬이 실행중인지 확인한다. 이 데몬은 대개 시스템이 부팅할 때 "`/etc/rc.d/init.d/named`" 파일에서 시작된다. 수작업으로 데몬은 시작 또는 중지시킬 수도 있다; 이를 위하여는 각기 "`named start`" 와 "`named stop`" 라고 타이핑하면 된다.
6. DNS 테이블에 변화가 생길 때마다, DNS 서버는 "`/etc/rc.d/init.d/named restart`" 라고 타이핑하여 재시작되어야 한다. 새로 추가하거나 변경한 기계에 대하여 제대로 동작하는지 확인하기 위해서는 "`nslookup`" 와 같은 툴을 사용하면 된다.

DNS 서비스 설정에 대한 자세한 정보는 http://metalab.unc.edu/Linux/HOWTO/DNS-HOWTO-5.html 에 있는 "**DNS-HOWTO**" 가이드를 참조하면 된다.

7.3. TACACS 를 이용한 인터넷 사용자 인증

우리 회사에서는 다이얼업으로 접속하는 인터넷 사용자 (우리의 모뎀 풀로 연결하면 이는 다시 두대의 시스코 250x 엑세스 서버로 연결된다) 의 인증을 위하여 Vikas 버전의 "xtacacsd" 를 사용하고 있다.

Vikas 패키지를 (최신판은 ftp://ftp.navya.com/pub/vikas 에서 구할 수 있다; RPM 형식으로는 나와있지 않는 것 같다.) 컴파일하고 설치한 뒤, "`/etc/inetd.conf`" 파일에 다음과 같은 항목을 추가하여서 TACACS 요청이 수신될 때 마다 inetd 데몬에 의하여 데몬이 자동으로 실행되도록 하여야 한다.

```
# TACACS is a user authentication protocol used for Cisco Router products.
tacacs dgram udp wait root /etc/xtacacsd xtacacsd -c /etc/xtacacsd-conf
```

그런 다음, "`/etc/xtacacsd-conf`" 파일을 편집하여 시스템에 맞춰 필요에 따라 변경하여야 한다. (물론 디폴트 셋팅을 그대로 사용할 수도 있을 것이다)

> 참고**:** 주의: 만약 섀도우 패스워드를 (자세한 내용은 6.6절 을 참조하시요) 사용하고 있다면 이 패키지를 사용하는데 문제가 있을 것이다. 불행히도, 레드햇이 사용자 인증을 위하여 사용하는 PAM (Pluggable Authentication Module) 을 이 패키지는 지원하지 않는다. 내가 이 문제를 해결하는 방법은 별도의 "`passwd`" 파일을 "`/usr/local/xtacacs/etc/`" 디렉토리에 두는 것이다. 이 파일은 /etc/ 에 있는 것과 같지만 섀도우를 사용하지 않는다. 이는 물론 약간 혼란스런 방식이다. 만약, 이렇게 하기로 하였다면 그 패스워드 파일이 루트만 읽을 수 있도록 퍼미션을 조정해주어야 한다:
>
> **`chmod a-wr,u+r /usr/local/xtacacs/etc/passwd`**

정말로 섀도우를 사용한다면, "`/etc/xtacacsd-conf`" 파일을 편집하여 섀도우가 아닌 패스워드 파일의 (앞에서 설명한 내가 쓰는 방식을 쓴다면) 위치를 알려주어야 한다.

그 다음 단계는 액세스 서버가 (다이얼업 모뎀과 같이) 원하는 장치로 들어오는 로그인을 TACACS 를 이용하여 인증하도록 설정하는 것이다. 어떻게 설정하는지를 보여주는 세션의 예는 다음과 같다:

```
mail:/tftpboot# telnet xyzrouter
Escape character is '^]'.
User Access Verification
Password: ****
xyzrouter> enable
Password: ****
xyzrouter# config terminal
Enter configuration commands, one per line.  End with CNTL/Z.
xyzrouter(config)# tacacs-server attempts 3
xyzrouter(config)# tacacs-server authenticate connections
xyzrouter(config)# tacacs-server extended
xyzrouter(config)# tacacs-server host 123.12.41.41
xyzrouter(config)# tacacs-server notify connections
xyzrouter(config)# tacacs-server notify enable
xyzrouter(config)# tacacs-server notify logouts
xyzrouter(config)# tacacs-server notify slip
xyzrouter(config)# line 2 10
xyzrouter(config-line)# login tacacs
xyzrouter(config-line)# exit
xyzrouter(config)# exit
xyzrouter# write
Building configuration...
[OK]
xyzrouter# exit

Connection closed by foreign host.
```

모든 TACACS 활동 로그 메시지는 상세한 검사를 위하여 "`/var/log/messages`" 파일에 기록된다.

7.4. 삼바를 이용한 윈도 스타일의 파일과 인쇄 서비스

리눅스는 SMB 서비스 (예를 들어, WfW, 윈95, NT 스타일의 네트웍 파일 & 프린터 공유) 를 삼바 패키지를 이용하여 제공한다. 이 섹션은 어떻게 공유를 설정하며 클라이언트 기계에서 어떻게 접근하여 활용할 수 있는지를 설명할 것이다.

삼바 패키지는 레드햇 배포판에 포함되어있는데 현재 설치가 되어 있는지 그리고 어떤 버전이 설치되어 있는지 알아보려면 다음과 같이 타이핑하면 된다:

```
rpm -q samba
```

만약 설치되어 있지 않다면 RPM 유틸리티를 이용하여 설치하여야 한다. 이에 대한 상세한 내용은 10.1절 을 참조하면 된다.

신경을 써주어야 할 가장 중요한 삼바 파일은:

/etc/smb.conf

공유에 대한 사항과 그 외의 설정 파라메터가 셋업되어 있는 삼바 설정 파일 (자세한 내용은 아래를 보시요)

/var/log/samba/

삼바 로그 파일이 있는 곳

/home/samba/

파일 공유가 셋업될 것으로 예상되는 디렉토리. 물론 저장하려는 파일의 양에 적합한 충분한 공간이 있는 다른 장소를 선택할 수도 있다. 개인적으로 나는 보통 /archive/ 에 큰 파티션을 마운트하고 여기를 공유 공간으로 사용한다.

"/etc/smb.conf" 파일은 파일 & 인쇄 공유에 대한 설정 정보를 갖고 있다. 파일의 처음 몇줄은 전체 설정에 대한 명령어로서 (각 섹션별로 별도로 지정하여 무시하지 않는 한) 모든 공유에 공통적으로 적용되는 것이고 그 다음에는 공유 섹션이 나온다.

삼바 설치 프로그램에는 디폴트 smb.conf 파일이 들어 있는데 상당수의 경우에 사용자의 요구에 적절하도록 되어 있기 때문에 아주 몇 줄만 수정해주면 된다.

다음에 파일의 예를 제시하였다 (이 파일은 중요하고 재미있는 옵션을 보여주기 위하여 심하게 손을 본 것이다):

```
# 모든 공유에 공통인 항목 (공유별로 덮어쓰지 않는 한)
[global]
   # 자원을 소모하는 것을 막기 위하여 활동이 없으면 몇 분 후에 클라인트를
   # 끊을 것인지 지정. 대부분의 클라이언트는 자동으로 재접속을 하기 때문에
   # 이 기능을 사용하는 것이 좋다.
   dead time = 10
   # 사용자가 "루트" 로 접속하는 것을 허용하지 않음, 만약을 위해서.  :-)
   invalid users = root
   # 손님 공유 (연결하는데 패스워드가 필요없는 공유) 에 사용할 계정을
   # 지정한다. 이 사용자 이름은 /etc/passwd 파일에 있는 정상적인 사용자
   # 이름이어야 한다.
   guest account = guest
   # 로그 파일을 기록할 장소를 지정한다.  "%m" 는
   # 로그 파일의 이름이 log.기계이름 형식을 (예. "log.twixel")
   # 사용함을 나타낸다
   log file = /usr/local/samba/logs/log.%m
   # 킬로 바이트 단위로 나타낸 로그 파일의 최대 크기
   max log size = 1000
   # 패스워드 레벨 3이란 패스워드를 입력할 때 대소문자를 구분하지 않음을
   # 나타낸다. 레벨 1이나 2보다는 덜 안전하지만 사용자 편의를 위해서는
   # 이 정도의 양보가 적당한 수준이다.
   password level = 3
   # 모든 공유가 브라우즈 리스트에 나타남을 지정.
   # (공유중에서 리스트에 나타나지 않아야 할 것은 공유별로 지정하여준다)
   browseable = yes
   # 이 기능을 사용하면 "smbstatus" 명령을 이용하여 현재
```

```
   # 사용중인 연결을 볼 수 있다.
   status = yes
   # 로그 파일에 기록되는 디버깅 정보의 레벨을 지정한다. 숫자가 클 수록
   # 더 자세한 정보를 생성한다. (대부분의 경우에 많은 정보가 유용하지는
   # 않다)
   debug level = 2
   # 이 기능은 서버가 받은 모든 윈도 스타일의 "POPUP" 메시지
   # 를 전자메일을 이용하여 포스트마스터에게 전송한다. 그리 유용하지는
   # 않지만, 어떤 것까지 할 수 있는지를 보여주는 좋은 예시이다.
   message command = /bin/mail -
s 'Message from %f on %m' postmaster < %s; rm %s &
   # 이는 캐싱의 일종으로서 사용할 경우 파일 읽기의 성능을 향상시킬 수
   # 있다.
   read prediction = true
   # 자동으로 브라우즈 리스트에 추가되어야 할 서비스의 목록을
   # 지정한다.
   auto services = cdrom
   # 프린터에 대한 정의를 담고 있는 텍스트 파일인
   # "printcap" 파일의 위치를 지정.
   printcap name = /etc/printcap
   # 만약 사용하는 경우에는 /etc/printcap 파일에 들어있는 모든 프린터가
   # 브라우즈 리스트에 들어간다.
   load printers = yes
   # 리눅스에서 프린터에 인쇄작업을 스풀시키는 인쇄 명령을 지정한다.
   print command = lpr -r -P%p %s
   # 인쇄작업 큐 정보 (프린터 상태) 를 가져오는데 사용하는 인쇄 명령을
   # 지정한다.
   lpq command = lpq -P%p
   # 원하지 않는 인쇄 작업을 큐로부터 제거하는데 사용되는 프린트 명령을
   # 지정한다.
   lprm command = lprm -P%p %j
   # 삼바가 자기 자신을 알리는 레벨을 지정한다. 네트웍에 떼거리로 붙어
   # 있는 NT 서버와 공평한 수준에서 "공간" 을 확보하도록
   # 하기 위하여 상당히 높게 하였다.  :-)
   os level = 34
# 여기는 사용자별 개인 공유영역이다.  클라이언트의 사용자 이름이 서버에
# 있는 이름과 일치한다면 (맞는 패스워드만 입력하면) 자기의 홈 디렉토리를
# 액세스할 수 있다.
[homes]
   # 브라우즈 리스트에 나타날 설명문을 지정한다
   comment = Home Directories
   # 이는 클라이언트의 사용자 이름과 공유의 사용자 이름을 짝을 맞춘다.
   # 만약 짝이 맞지 않으면, 브라우즈 리스트에 아무런 공유도 보이지
   # 않거나 연결할 수 있도록 나타난다.
   user = %S
   # 공유에 대한 경로를 지정한다. 예를 들어, "smithj" 는
   # "/home/smithj" 로 매핑된다.
   path = /home/%S
   # 이를 사용하면 공유에 대하여 읽기/쓰기를 액세스를 허용한다.
```

```
   writeable = yes
   # “writeable” 를 그대로 뒤집은 것이다. 물론 둘다를
   # *실제로* 사용할 필요는 없다.  :-)
   read only = no
   # 이 공유를 액세스하기 위해서는 패스워드가 필요하게 하려면 이 기능을
   # no 로 두면 된다.
   public = no
   # 이 공유가 다른 사용자의 브라우즈 리스트에 나타나게 하고 싶지
   # 않음 (즉, 개인용) 을 나타낸다.
   browseable = no
# 이는 모두 사용할 수 있는 공유 프린터 “hp_laser” 에 대한
# 내용이다. 이 프린터는 브라우즈 리스트에 나타나면 패스워드 없이 모든
# 클라이언트가 사용할 수 있다.
[hp_laser]
   # 브라우즈 리스트에 나타날 설명을 지정한다.
   comment = Main office printer (HP Laserjet 400)
   # 이 공유를 액세스할 수 있는 사용자 이름 (guest는 모든 사용자라는 뜻).
   user = guest
   # 생성되는 모든 인쇄 파일은 처음에는 /tmp 디렉토리에 만들어 진다는
   # 것을 지정.
   path = /tmp
   # 프린트 스풀링을 제외하고는 파일을 만드는 것을 허용하지 않는다.
   writeable = no
    # 필요에 따라 퍼미션을 지정한다 -- 인쇄 작업은 루트만이 액세스 가능함.
   create mode = 0700

   # 이 기능을 사용하면 공유를 액세스할 때 패스워드가 필요없다.
   public = yes
   # 이 공유가 프린터 공유임을 나타내기 위하여 이를 yes 로 하여야 한다.
   printable = yes
# 씨디롬 장치에 대한 액세스를 제공하는 서비스를 정의한다.
[cdrom]
   comment = Shared CD-ROM drive on Linux
   user = guest
   path = /cdrom
   writeable = no
   read only = true
   browseable = yes
   public = yes
   guest ok = yes
```

작은 정보**:** 팁: 최근 버전의 삼바, 2.0 이상, 에서는 삼바 설정 과정을 훨씬 편리하게 할 수 있는 “**swat**” 라는 이름의 매우 쓸만한 웹 기반의 설정 유틸리티가 제공된다. 이 유틸리티는 서버의 TCP **901** 번 포트에서 접속을 기다린다. 따라서 이 유틸리티를 사용하려면 웹 브라우저에서 다음과 같이 연결하면 된다:

```
나의도메인.이름:901
```

(물론, SWAT 유틸리티를 사용하려면 아파치와 같은 웹 서버가 실행중이어야 한다. 자세한 내용은 7.1절 을 참조하시요.)

최근 버전의 삼바에는 2.0 이전과 비교하여 상당히 많은 기능을 추가되었다. 따라서 시간을 투자하여 삼바 패키지를 업그레이드할 만한 가치가 있다.

클라이언트는 공유에 연결하기 위하여 TCP/IP 네트웍 스택이 실행중이어야만 한다. 게다가, 브라우징 기능이 동작하려면, TCP/IP 프로토콜은 NETBEUI 와 엮여있어야 한다. 윈도 95 에서는 제어판에 있는 "네트워크" 아이콘을 이용하여 설정할 수 있다.

클라이언트가 제대로 설정되었다면, "네트워크 환경" 에 (윈도 95 나 NT 를 사용하고 있지않다면 그에 해당하는 브라우징 기능에) 서버의 공유가 나타나야 한다. 그러고 나면 네트워크 환경에서 네트웍 드라이브를 매핑하거나 공유의 절대 경로 (예."**\\mail\cdrom**") 를 타이핑하여 이용할 수 있다. 만약 공유 서비스가 패스워드의 입력을 필요로 하도록 되어 있다면, 입력하라는 프롬프트가 나타날 것이다.

삼바에 대한 더 자세한 정보는 http://samba.anu.edu.au/samba/ 에 있는 삼바 홈 페이지에서 얻을 수 있다.

7.5. Netatalk 를 이용한 매킨토시 스타일의 파일과 인쇄 서비스

리눅스는 Netatalk 패키지를 이용하여 애플쉐어 서비스(예를 들어, 매킨토시 스타일의 파일 & 프린터 공유)도 제공할 수 있다. 이 섹션은 어떻게 공유를 설정하고 클라이언트에서는 어떻게 액세스하는지를 설명할 것이다.

Netatalk 를 사용하려면 당신의 리눅스 커널이 애플톡 네트워킹을 지원하도록 해야 한다. 레드햇은 이미 이러한 지원을 모듈의 형태로 제공하고 있으며 그것을 이용하지 않더라도 이를 지원할 수 있도록 커널을 컴파일할 수 있다.

> 참고**:** 주의: 애플톡 지원 기능은 모듈 로 컴파일해야지 커널의 일부로 포함되게 해서는 안된다. (리눅스 커널을 업그레이드하거나 수정하는 방법에 대하여는 10.4절 을 참조하시요) 그렇지 않으면 Netatalk 데몬을 중단시키고 재시작시킬 때 어려움을 겪게될 것이다.

일단, 커널이 애플톡을 지원할 수 있게 되면, Netatalk 패키지를 설치할 필요가 있다. Netatalk 는 레드햇 배포판에는 포함되어 있지 않으므로 먼저 내려받아서 설치하여야 한다. Netatalk 패키지는 레드햇의 "contrib" 사이트인 ftp://ftp.redhat.com/contrib/libc6/i386/ 에 들어있다.

Netatalk 가 설치되고 나면, "`/etc/atalk/`" 에 있는 몇개의 설정 파일을 수정할 필요가 있다. 대부분의 파일은 샘플 설정 예시를 갖고 있으므로 어느 정도 내용을 알아볼 수는 있을 것이다. 파일은 다음과 같다:

config

> 이 파일은 Netatalk 데몬을 튜닝하는데 필요한 설정 정보를 갖고 있다. 이 정보는 환경 변수에 지정되며 서비스가 시작되기 전에 Netatalk 시작 스크립트가 "참조하게" (즉, 읽게)

된다. 동시 접속수, 손님 로그인 기능을 제공할 것인지 아닌지 등을 설정할 수 있다. 이 파일은 대개 필요에 맞춰 수정하여야 한다.

atalk.conf

이 파일에는 사용할 네트웍 인터페이스, 애플톡 라우팅, 이름 등록, 그리고 그 외 관련 정보 등이 포함되어 있다. 이 파일을 수정하지 않아도 될 수 있다; 필요한 네트웍 정보는 Netatalk 서버를 처음 실행할 때 자동으로 탐지되어 이 파일에 추가된다. 물론, 당신 서버의 이름을 추가해주고 싶을 수도 있다.

참고**:** 주의**:** 이 파일에 대한 상세한 정보는 "**man atalkd**" 라고 타이핑하면 얻을 수 있다**.**

afpd.conf

이 파일은 Netatalk 에게 명령행 옵션으로 전달되는 추가 파라메터를 정의할 수 있게 해준다. Netatalk 서버가 어느 포트 또는 어느 IP 주소로 동작할 것인지를 지정하고 접속하는 사용자에게 표시할 로그인 메시지를 추가하는 등의 일을 할 수 있다. 이 파일은 별로 수정할 필요가 없을 것이다.

참고**:** 주의**:** 이 파일에 대한 자세한 정보를 보려면 "**man afpd**" 라고 타이핑하면 된다**.**

papd.conf

이 파일은 맥 사용자가 공유된 네트웍 프린터를 이용하여 인쇄를 할 수 있도록 하는데 필요한 설정 정보를 갖고 있다. 나는 아직 이 파일을 갖고 놀아본 적이 없어서 불행히도 해줄 말이 없다.

참고**:** 주의**:** 이 파일에 대한 상세한 정보는 "**man papd**" 라고 타이핑하여 얻을 수 있다**.**

AppleVolumes.default

이 파일에서는 맥 사용자가 로그인한 후에 볼 수 있는 공유 목록을 갖고 있다. 공유를 사용하려면 디렉토리에 대한 경로명과 그에 대한 문자로된 설명을 입력하면 된다. 예를 들면:

```
~                "Home"
/archive/busdept "Business Department Common Files"
```

(위의 예에서는 맥 사용자가 두개의 공유를 사용할 수 있다: 사용자의 홈 디렉토리와 비즈니스 부서를 위한 공유 영역이 그에 해당한다.)

작은 정보**:** 팁: 여기서 활용할 수 있는 멋진 기법으로는 삼바에서와 같은 경로를 공유시키는 것이다. 이렇게 하면 맥 사용자와 윈도 사용자가 자신이 사용하는 플랫폼에 상관없이 파일을 공유할 수 있게 된다. 상세한 삼바의 사용법에 대하여는 7.4절 을 참조하시요.

AppleVolumes.system

이 파일은 "`AppleVolumes.default`" 파일과 마찬가지로 파일 공유 리스트를 갖고 있다. 차이점은 이 파일에 들어있는 공유는 로그인을 하든 하지않든간에 모든 사용자가 사용할 수 있다는 것이다. 또한 이 파일에는 파일 형식 매핑도 들어있다. 아무나 사용할 수 있는 공유를 정의하려는 경우가 아니라면 이 파일을 수정할 일은 없을 것이다; 그리고 아무나 사용할 수 있는 공유는 대개의 경우 별로 좋은 생각은 아니다.

일단 모든 것을 적절한 설정 정보로 셋업하였다면 Netatalk 서비스를 수작업으로 다음과 같이 시작시킬 수 있다:

```
/etc/rc.d/init.d/atalk start
```

(서비스는 시스템이 부팅될 때 자동으로 시작되어야 한다.)

Netatalk 에 대한 더 자세한 정보는 http://www.umich.edu/~rsug/netatalk/ 에 있는 Netatalk 홈 페이지에서 얻을 수 있다. 그리고, 설정 정보에 대한 매우 유용한 자료를 http://thehamptons.com/anders/netatalk/ 에 있는 리눅스 Netatalk HOWTO 문서에서도 얻을 수 있다.

7.6. 네트웍 파일 시스템(NFS) 서비스

리눅스는 네트웍 파일 시스템 (NFS) 프로토콜을 이용한 파일 시스템 공유에 있어 클라이언트로도 서버로도 역할을 할 수 있다. 네트웍 파일 시스템은 유닉스 시스템간에 파일 시스템의 마운트를 지원하는 사실상의 표준이다.

참고**:** 주의: 당신의 시스템이 NFS 서비스를 제공한다는 것이 보안상의 위험이 될 수 있음에 유의하라. 개인적으로 나는 이 서비스의 사용을 권장하지 않는다.

NFS 를 사용하기 위해서는 커널 또는 커널 모듈로 NFS 지원이 포함되어 있어야 한다. 리눅스 커널을 업그레이드하거나 수정하는 방법에 대하여는 10.4절 을 참조하시요.

NFS 공유를 설정하려면 "`/etc/exports`" 파일을 수정해주면 된다. 아래에 몇가지 사용가능한 옵션을 보여주는 항목의 예를 들었다:

```
/archive spock.mydomain.name(ro)
/archive2 spock.mydomain.name(ro)
/mnt/cdrom other.domain(ro)
/archive2 10.23.14.8(ro,insecure)
```

처음 두 줄은 "spock.mydomain.name" 라는 호스트가 NFS 를 이용하여 "`/archive`" 와 "`/archive2`" 디렉토리를 액세스할 수 있음을 나타낸다. 이들 공유는 "`(ro)`" 옵션을 지정하였기 때문에 읽기전용이 된다. 보안상의 이유로 모든 NFS 공유를 가능하다면 이 옵션으로 해두는 것이 좋다.

세번째 줄은 "domain.name" 도메인에 있는 모든 호스트가 씨디롬을 액세스 할 수 있도록 허용하고 있다. 물론 씨디롬은 미리 "`/mnt/cdrom`" 에 마운트되어 있어야 한다.

참고**:** Note: 이 장치를 "(ro)" 옵션으로 지정하여 읽기전용으로 만드는 것은 약간 불필요한 느낌이 든다. 하지만, 실수로 씨디롬 장치가 마운트되지 않은 상태에서 실제 파일 시스템에 파일을 쓰는 사악한 사람을 막아줄 수 있다.

"/etc/exports" 파일을 수정하고 난 뒤에는 NFS 데몬을 재시작하여야 한다. 이를 위하여는 다음과 같이 타이핑하면 된다:

```
/etc/rc.d/init.d/nfs restart
```

NFS 마운트 위치를 설정하는데는 "Linuxconf" 유틸리티에 포함되어 잇는 "Network Configurator" 도구를 사요할 수도 있다. Linuxconf 유틸리티에 대한 자세한 정보는 7.7절 을 참조하시요.

NFS 에 대한 자세한 정보는 http://metalab.unc.edu/LDP/HOWTO/NFS-HOWTO.html 에 있는 "**NFS-HOWTO**" 에 들어있다. 그 외에도 "nfsd" 와 "exports" 의 매뉴얼 페이지에서도 정보를 얻을 수 있다.

7.7. Linuxconf 를 이용한 설정의 모든 것

상당수의 설정 문제를 쉽게 해결해주는 "linuxconf" 라는 훌륭한 도구가 있다. Linuxconf 는 사용가능한 어떤 화면 환경에서도 잘 돌아간다 -- 콘솔에서 실행할 수도 있고, 텔넷을 통하여 원격지에서도 실행할 수 있고 X 에서는 구이기반의 도구로 실행할 수도 있는데 상황을 자동으로 탐지하여 적절한 방식으로 시작된다.

시스템의 시간을 바꾸고 네트웍 셋팅을 수정하고 파일 시스템을 셋업하고 사용자 관리를 하고 기타의 다양한 관리나 설정 임무를 수행할 때 이 도구를 사용해 볼 수 있다. 딱 한가지의 주의 사항은 이 글을 쓰는 때를 기준으로 구이기반의 도구는 약간 "버그가 있고" 마우스로 클릭을 해도 반응하지 않는 경우가 있다. 하지만, 이 도구는 계속 개선되고 있고 미래의 버전은 꽤 쓸만할 것이다.

8장. 백업과 복구 설차

정기적으로 백업을 수행하는 것은 책임있는 시스템 관리자에게는 최우선의 일로 간주되어야 한다. 비록 리눅스가 극도로 안정적인 운영체제이지만, 장애는 일어날 수 있고, 일어나고 있으며, 아마도 일어날 것이다. 하드웨어상의 장애, 정전 또는 그 외 예측하지 못한 문제로부터 이러한 장애는 발생할 수 있다.

그러나 이러한 요인보다는 사람의 실수에 의하여 문제가 발생하는 경우가 더 많다. 그 결과로 중요한 파일을 원하지 않게 고치거나 심지어는 삭제할 수가 있다. 만약 당신의 시스템을 여러 사용자들이 쓰고 있다면 아마 실수로 지운 파일을 복구해달라는 요구를 틀림없이 받게 될 것이다.

만약 정기적으로 백업을 한다면 (최소한 자주 갱신되는 사용자 파일들이라도) 그것도 일단 위로 한다면 그러한 파일 손실의 가능성을 줄이고 복구 가능성을 높일 수 있다.

백업을 실행하는 가장 안전한 방법은 테이프, 제거가능한 드라이브, 기록가능한 씨디 등 분리된 매체에 기록하여 리눅스 시스템과는 떨어진 위치에 저장해두는 것이다. 어떤 경우에는 이러한 것이 현실성이 없을 수도 있다 -- 아마 백업 테이프를 넣어 둘만한 내화성 금고를 갖고 있지도 않을 것이다! 또는 처음에는 외장 백업 시스템 자체가 없을 수도 있다. 그렇다손 치더라도 백업을 어떻게든 비록 제한된 방법으로나마 수행할 수는 있다.

우리 회사에서 나는 여러대의 리눅스 서버에 대하여 백업을 수행한다. 상황에 따라서 어떤 백업 세트는 테이프에 쓰고 또 어떤 것은 네트웍으로 연결된 다른 서버에 쓰기도 하고 그렇지도 않은 것들은 다른 디스크 파티션에 (예컨대, "`/archive/`" 파일 시스템에) 쓰기도 한다. 이 작업은 자동으로 cron 작업으로 실행된다. (대개 서버는 다른 곳에 있는 경우가 많으므로 테이프 백업을 하기 위하여 여기 저기를 매일 같이 방문한다는 것은 실용적이지 않거나 불가능하다.)

집에서는 외장 백업 시스템이 없을 뿐만 아니라 백업 이미지를 기록해둘만한 여분의 디스크 공간을 충분히 갖고 있지도 않다. 따라서, 따라서, 대신 "`/home/`" 디렉토리에 있는 나의 사용자 파일과 "`/etc/`" 디렉토리에 있는 몇몇 설정 파일만을 별도의 디스크 파티션에 백업하고 있다.

8.1. 서버의 백업 절차

리눅스에서 백업을 수행하는데에는 매우 다양한 방법이 있다. 모든 리눅스 배포판에 포함되어 있는 명령행 도구인 "dd", "dump", "cpio", 나 "tar" 를 사용하는 방법도 이에 속한다. 그 외에도 백업과 복구 절차에 좀 더 사용하기 편리한 인터페이스를 추가하려고 만든 텍스트 기반의 유틸리티로서 "아만다(Amanda)" 와 "테이퍼(Taper)" 가 있다. 구이 기반의 "KDat" 라는 유틸리티도 있다. 마지막으로는 상용 백업 유틸리티로서 "BRU" 와 "PerfectBackup+" 가 있다. 이들 중 어느 것을 사용하더라도 당신의 귀중한 자료를 보호할 수 있다.

사용가능한 도구와 그것을 어디서 얻을 수 있는 지를 정리한 목록은 http://www.xnet.com/~blatura/linapp2.html#back 에 있는 "Linux Applications and Utilities Page" 에서 얻을 수 있다. 백업 솔루션을 결정할 때에는 다음과 같은 사항을 고려할 필요가 있다:

- 이식성 - 백업의 이식성이 (즉, 한 리눅스 또는 유닉스 시스템에서 백업한 것을 다른 시스템에 복구하는 것; 예를 들어, 솔라리스에서 백업한 것을 레드햇 리눅스에다가 복구하는 경우) 중요한가? 만약 그렇다면 명령행 도구(예. “dd”, “dump”, “cpio”, “tar”)를 사용하는 것이 좋을 것이다. 왜냐하면, 그러한 도구는 어떤 리눅스/유닉스 시스템에서도 사용가능하다고 확신할 수 있기 때문이다.
- 자동 백업 - 사람의 개입없이 정기적으로 자동화된 백업이 수행되는 것이 당신에게 중요한가? 만약 그렇다면, 그러한 백업 방식을 지원할 수 있는 도구와 백업 매체를 갖추어야 할 것이다.
- 사용의 편의성 - 편리한 사용자 인터페이스가 중요한가? 그렇다면 텍스트 또는 구이 기반의 인터페이스를 제공하는 도구를 선택하고 싶을 것이다. 상용 유틸리티는 가장 사용하기 쉬운 인터페이스를 제공할 뿐 만 아니라 기술지원도 해줄 수 있다.
- 원격 백업 - 원격지의 기계에서 백업을 시작하거나 복구하는 기능이 중요한가? 그렇다면, (X 세션을 무리없이 실행할 수 있을 정도로 고속 네트웍으로 연결되어 있지 않다면) 구이 기반의 유틸리티보다는 명령행 도구나 텍스트 기반의 유틸리티를 선택하는 것이 좋을 것이다.
- 네트웍 백업 - 네트웍으로 연결된 호스트의 백업과 복구를 실행하는 것이 중요한가? 만약 그렇다면, 백업 장치에 대하여 네트웍 액세스를 지원하는 (“tar” 같은) 명령행 유틸리티나 “아만다(Amanda)” 같은 특별한 유틸리티 또는 상용 유틸리티가 좋을 것이다.
- 매체의 종류 - 백업은 테이프, 추가의 하드 드라이브, 집 드라이브, 다시쓸 수 있는 씨디 등 다양한 매체에 저장될 수 있다. 가격 대 안정성, 저장 용량, 전송 속도 등을 비교하여야 할 것이다.

> **경고**
>
> 특별 주의: 파일 시스템을 백업할 때, 가짜 파일 시스템인 “`/proc`” 를 절대로 포함시키지 마라! /proc 에 있는 파일은 실제 파일이 아니고 파일 모양의 링크로서 커널의 자료구조를 가리키고 있다. 당신의 전체 메모리의 내용을 담고 있는 가짜 파일인 “`/proc/kcore`” 파일을 백업하는 것은 테이프의 엄청난 낭비일 뿐이다! :-) 또한, 씨디롬 장치, 플로피 장치, 네트웍 파일 공유 그리고 그 외 마운트된 장치의 내용을 백업하려는 경우가 아니라면, “`/mnt`” 파일 시스템도 백업하지 않도록 하여야 한다.

당연히, 당신이 어떤 백업 솔루션을 선택하느냐에 따라 백업과 복구를 수행하는 절차는 다르다. 하지만, 이 섹션에서는 내가 가장 자주 사용하는 다음과 같은 두개의 도구를 이용하여 백업을 수행하는 방법을 설명할 것이다: “tar” (이 이름은 “Tape ARchiver” 에서 따낸 말이다)는 명령행 백업 도구로서 유닉스/리눅스 시스템에서 상당히 이식성이 높다. “KDat” 는 구이 기반의 테이프 백업 유틸리티로서 KDE 패키지에 (KDE 에 대하여는 5.6절 을 참조하시요) 포함되어 있다.

마지막으로 덧붙이고 싶은 것은 선택한 백업 솔루션에 따라, 심지어는 그 도구가 자동 백업 기능을 내장하고 있지 않다 하더라도, cron 기능을 이용하여 백업을 자동화할 수 있다는 것이다. cron 을 사용하는 방법과 crontab 스케쥴 파일을 만드는 방법에 대한 상세한 내용은 9.4절 을 참조하시요.

8.1.1. "tar" 를 이용한 백업:

"tar" 를 백업 솔루션으로 사용하기로 결정하였다면 다양한 명령행 옵션을 알기 위하여 시간을 투자할 필요가 있다; 옵션에 대한 설명을 보려면 "`man tar`" 라고 타이핑하면 된다. 그리고 적당한 백업 매체에 액세스하는 방법도 알아야 할 것이다; 비록 유닉스 세계에서는 모든 장치가 파일과 같이 취급되지만, 테이프와 같은 문자 장치에 기록을 할 때에는 "파일" 의 이름이 장치 자체를 나타낸다. (예를 들어, "`/dev/nst0`" 는 스카시 기반의 테이프 드라이브를 나타낸다.)

다음의 명령은 당신 리눅스 시스템 전체를 "`/archive/`" 파일 시스템에 백업할 것이다. 이 백업에서 빠지는 부분은 가짜 파일 시스템인 "`/proc/`", "`/mnt/`" 에 마운트된 파일 시스템, "`/archive/`" 파일 시스템 (백업 셋트 자체를 다시 백업한다는 것은 의미가 없다!), Squid 가 사용하는 상당히 큰 캐시 파일이다. (이들을 백업하는 것은, 내 의견으로는, 백업 매체의 낭비이며 필요하지도 않다):

```
tar -zcvpf /archive/full-backup-`date '+%d-%B-%Y'`.tar.gz \
    --directory / --exclude=mnt --exclude=proc --exclude=var/spool/squid .
```

이 명령의 길이에 겁먹지마라! 우리가 명령을 구성 부분으로 쪼개놓고 나면 이 강력한 유틸리티의 아름다움을 알게될 것이다.

앞의 명령에서 지적한 옵션으로는, "**z**" (압축; 백업 자료를 "gzip" 을 이용하여 압축한다), "**c**" (생성; 새로운 저장 파일을 만든다), "**v**" (수다; 백업이 되고 있는 파일의 목록을 보여준다), "**p**" (퍼미션 보존; 파일 보호를 위한 정보는 복구할 수 있도록 "기억해" 둔다) 가 있다. 그 외에도, "**f**" (파일) 옵션은 그 다음의 인자가 생성할 저장 파일 (또는 장치) 의 이름이라는 것을 나타낸다. 파일 이름에 오늘 날짜를 넣기 위하여 어떻게 했는지 보면, "date" 명령을 두개의 역따옴표로 둘러싸서 만들어내고 있다. 일반적으로 사용되는 이름 규칙은 압축되지 않은 저장 파일에는 "`tar`" 를 뒤에 붙이고, 압축된 저장파일의 경우에는 "`tar.gz`" 를 뒤에 붙이는 것이다.

"**--directory**" 옵션은 백업을 시작하기 전에 옵션 뒤에 지정한 디렉토리 경로로 일단 가라는 것을 나타낸다. (우리의 예에서는, "/" 디렉토리로 가서 백업이 시작된다.) "**--exclude**" 옵션은 지정한 디렉토리나 파일은 백업하지 말라고 지시한다. 마지막으로, "**.**" 글자는 현재 디렉토리에 있는 모든 것을 백업해야 함을 나타낸다.

> 참고**:** 주의: tar 에 사용되는 옵션은 대소문자를 구분한다는 것을 명심하여야 한다! 게다가, 대부분의 옵션은 한 글자짜리 (예. "f") 로 사용될 수도 있고, 좀 더 외기 쉬운 완전한 옵션 이름 (예. "file") 으로도 사용될 수 있다. 한 글자짜리를 쓸 때에는 그 앞에 "-" 문자를 붙여야 하고, 완전한 이름을 사용할 때에는 그 글자를 두개를 써야 한다. 다시한번 말하지만, tar 명령에 대한 "man" 페이지를 꼭 참조하기 바란다.

또 하나의 예로서, (앞의 예에서는 전체 를 백업하면서 그 중에 몇가지를 제외하는 방식을 썼지만) 이번에는 지정한 몇개의 파일 시스템만을 스카시 테이프 드라이브에 기록하는 방법을 살펴 보자:

```
tar -cvpf /dev/nst0 --label="Backup set created on `date '+%d-%B-%Y'`." \
    --directory / --exclude=var/spool/ etc home usr/local var/spool
```

위의 명령에서 "**z**" (압축) 옵션이 사용되지 않았음을 주목하기 바란다. 나는 테이프에 압축된 자료를 기록하는 것을 강력히 반대한다. 왜냐하면, 테이프에 있는 자료의 일부가 깨지면 백업 셋트 전체를 못쓰게 되기 때문이다. 테이프의 일부가 파손되더라도 압축을 사용하지 않고 저장된 경우에는 파손되지 않은 파일은 상당히 많이 복구할 수 있다.

테이프 드라이브는 문자 장치이기 때문에, 실제 파일 이름을 지정하는 것은 불가능하다. 따라서, tar 에서 파일 이름에 해당하는 인자는 장치의 이름 즉 "`/dev/nst0`" (스카시 버스에 연결된 첫번째 테이프 장치) 와 같은 방식으로 지정하여야 한다.

> 참고**:** 주의: "`/dev/nst0`" 장치는 백업 셋트를 다 기록한 다음 자동으로 되감기를 하지 않는다; 따라서 하나의 테이프에 여러개의 셋트를 기록할 수 있다. (같은 장치를 "`/dev/st0`" 라고 지정하게 되면 백업 셋트가 기록되자마자 테이프는 자동으로 감기게 된다.)

백업 셋트에 파일이름을 지정할 수 없기 때문에, "**--label**" 옵션을 사용하여 저장 파일 자체에다가 백업 셋트에 대한 추가 정보를 저장할 수 있다.

마지막으로, "`/etc/`", "`/home/`", "`/usr/local`", "`/var/spool/`" 에 있는 파일들만 (Squid 캐시 자료 파일은 빼고) 테이프에 기록된다.

테이프를 이용할 때에는 테이프를 되감거나 꺼내기 위하여 다음과 같은 명령을 사용하게 된다:

```
mt -f /dev/nst0 rewind
mt -f /dev/nst0 offline
```

> 작은 정보**:** 팁: 저장 파일이 만들어질 때 맨 앞에 있는 "/" (슬래시) 문자는 잘리는 것을 보게 될 것이다. 이는 tar 의 기본적인 동작 방식으로서 복구하는 과정에서 실수로 중요한 파일을 옛날 버전의 파일로 덮어쓰는 것을 막기위한 것이다. 만약 이런 식으로 동작하는 것을 원하지 않는다면 (이것도 하나의 기능 이라는 사실을 기억하라!), tar 명령에다가 "`--absolute-paths`" 옵션을 주면 맨 앞의 슬래시가 보존된다. 물론, 나는 이렇게 하는 것이 위험 하기 때문에 권장하지는 않는다!

8.1.2. "KDat" 를 이용한 백업**:**

만약 KDE 데스크탑 환경을 사용하고 있다면, "KDat" 유틸리티가 강력할 뿐만 아니라 사용하기도 편리하다는 것을 알게될 것이다. 게다가, 한가지 덤으로 따라오는 것은 KDat 가 "tar" 를 백업 엔진으로 사용한다는 점이다. 따라서, KDat 로 기록된 백업 셋트는 KDat 로 읽을 수 있을 뿐만 아니라 tar 를 이용해서도 읽을 수 있다! 따라서 KDat 는 사용의 편리성과 백업의 이식성을 동시에 만족하는 아주 좋은 선택이 되는 것이다.

> 작은 정보**:** 팁: KDE 패키지를 사용하지 않거나 전체를 다 설치하지 않기로 하였다고 하더라도, Qt 라이브러리만 설치한다면 여전히 KDat 를 사용할 수는 있다.

처음으로 KDat 프로그램을 실행시킬 때 백업 프로파일을 만들 필요가 있다. 그러한 프로파일은 KDat 에게 시스템에서 어느 파일을 백업하고 싶은지를 알려주는 역할을 한다. 만약 필요하다면 여러개의 백업 프로파일을 만들 수도 있다. (예를 들어, 시스템 전체를 백업해주는 "전

체 백업" 이라는 프로파일과 사용자 파일만 백업해주는 "빠른 백업" 이라는 프로파일을 각각 만들어 두고 필요에 따라 골라 쓸 수 있다.)

백업 프로파일을 만들려면, 메뉴 바에서 "File" 옵션에서 "Create Backup Profile" 을 선택하거나 또는 "Backup Profiles" 폴더에서 오른쪽 버튼을 누른다음 "Create Backup Profile" 을 선택하면 된다. KDat 창의 오른쪽 편에서는 프로파일의 이름, 저장 파일의 이름, tar 옵션 등 다양한 셋팅을 바꿀 수 있다. 이러한 셋팅이 어디에 쓰이는 지를 알려면 메뉴에서 "Help" 를 선택하여 살펴보면 된다.

백업 프로파일에 포함시킬 파일을 지정하려면, "/" 디렉토리 폴더의 옆에 있는 체크박스를 클릭하면 된다. 이렇게하면 이 디렉토리 아래의 모든 파일을 백업하도록 지정한 것이 된다. 그러고 나서, 폴더의 옆에 있는 조그마한 "+" 표시를 클릭한다. 그러면 폴더가 확장되어 그 안에 들어있는 파일의 목록을 보여준다. 여기서 백업에 포함시키고 싶지 않은 파일을 빼낼 수 있다; 빼고 싶은 파일이나 디렉토리 옆에 있는 체크박스를 클릭해주기만 하면 된다. 예를 들어, 전체 백업은 모든 파일과 디렉토리는 체크 표시를 해주고 단, "/proc" (실행 중인 시스템에 대한 정보를 갖고 있는 가짜 파일 시스템), "/mnt" (씨디롬 드라이브, 플로피, 네트웍 공유 등이 대개 마운트되는 디렉토리), 그리고 Squid 를 사용하고 있다면, "/var/spool/squid" (Squid 의 캐시 데이터 파일) 만 예외로 빼준다. 일단 필요한 파일을 선택하였다면 만들고 있는 백업 프로파일을 클릭하고 "Files >>" 버튼을 클릭하여 선택된 파일 목록을 백업 프로파일로 옮긴다.

> 참고**:** 주의: 자료의 규모가 너무 커서 하나의 테이프에 모두 담을 수 없다면, 여러개의 백업 프로파일을 만들어서 각기 백업할 대상의 일부를 담당하도록 하여야 한다.

실제로 백업을 수행하려면, 드라이브에 테이프를 넣고나서 "File" 메뉴에서 "Mount Tape" 를 선택한다. (또는 테이프같이 생긴 아이콘을 클릭한다) 이렇게 하면 테이프를 "마운트" 하게 된다. (실제로는 테이프 장치가 문자 장치이기 때문에 마운트하는 것이 불가능하다 -- KDat 가 실제로 하는 일은 테이프를 되감고 헤더 정보를 읽으려고 시도하여 만약 읽을 수 있으면 해당되는 테이프 색인을 찾아낸다. 만약, 헤더 정보를 읽을 수 없으면 KDat 는 테이프를 포맷하라고 할 것이다.)

> 참고**:** (주의: 만약 드라이브에 테이프가 확실히 들어있는데도 계속해서 KDat 가 테이프가 드라이브에 들어있지 않다고 투덜대면 preference 에 테이프 장치의 이름이 제대로 되어 있는지 확인하여야 한다; 메뉴 바에서 "Edit" 옵션을 클릭하고 "User Preferences" 를 선택하면 된다.)

일단 KDat 가 테이프를 마운트하고 나면 백업을 시작하기에 앞서 이번 백업에서 어느 백업 프로파일을 사용할 것인지를 선택하여야만 한다. 백업을 시작하려면, 원하는 백업 프로파일을 오른쪽 버튼으로 클릭한 뒤 "Backup" 옵션을 클릭해주면 된다. 그러면 KDat 는 당신이 선택한 백업 프로파일에 대한 상세한 내용을 보여주는 대화상자를 보여준다; 여기에서 백업을 시작하려면 "Ok" 버튼을 클릭하면 된다.

백업이 진행되는 동안, KDat 는 다양한 통계 정보 (작업 시간, 백업의 크기, 백업한 비율, 남은 시간의 예상치, 파일의 수, 기록된 바이트 수 등) 와 백업된 파일의 목록을 나타내는 대화상자를 보여줄 것이다. 수 기가 바이트에 달하는 전체 백업은 끝나는데 까지 몇시간이 걸릴 것이다. 만약 필요하다면 백업이 진행되는 중에 언제라도 "Abort" 버튼을 눌러 백업을 중단시킬 수 있다.

일단 백업이 완전히 끝나고 나면, 메뉴 바에서 “Edit” 를 선택한다음 “Unmount Tape” 를 선택하여 테이프의 마운트를 해제하거나 테이프 모양의 아이콘을 클릭하여 테이프를 되감은 뒤 튀어나오게 할 수 있다.

8.2. 서버 복구 절차

말할 나위도 없이 정기적으로 백업을 하는 것보다 더 중요한 한가지 일은 중요한 파일을 복구할 때가 되었을 때 복구할 수 있게 하는 것이다!

당연히 8.1절 에서 언급한 것과 마찬가지로 당신이 백업 솔루션으로 무엇을 선택했느냐에 따라 복구를 수행하는 방법도 다르다. 이 섹션에서는 “tar” 와 ”KDat” 를 이용하여 백업된 파일을 복구하는 방법을 다룰 것이다.

8.2.1. “tar” 를 이용한 복구:

다음의 명령은 “`full-backup-09-October-1999.tar.gz`” 저장 파일로 부터 모든 파일을 복구한다. (이 파일은 리눅스 시스템의 예제 백업 파일로서 8.1.1절 에서 예시한 명령으로 만들어진 것이다):

```
tar -zxvpf /archive/full-backup-09-October-1999.tar.gz
```

앞의 명령은 압축된 저장 파일에 들어있는 모든 파일을 추출하며 원래 파일의 소유관계와 퍼미션도 그대로 복원한다. “`x`” 옵션은 추출 (extract) 을 의미한다. (다른 옵션은 8.1.1절 에서 이미 설명하였다.)

> **경고**
>
> 특별 주의: tar 저장 파일로부터 파일을 꺼내는 것은 위험한 일일 수 있다. 따라서, 특별히 조심해서 할 필요가 있다. 대개는 파일이 전체 파일 경로를 앞에 달고 저장되지 않기 때문에 현재 디렉토리에 꺼낼 수가 있다. (극소수의 잘못된 또는 잘모르는 개발자들이 자기들 소프트웨어를 전체 경로를 붙여서 tar 파일로 배포하는 경우가 있다.) 하지만 (저장 파일을 만들 때 “`--absolute-paths`” 옵션을 사용함으로써) 파일이 맨 앞에 “`/`” 슬래시를 달고 저장되어 있어서 (당신이 원하지 않더라도) 원래 있던 절대 위치로 복원될 수 있다. 또는 파일이 맨 앞에 “`/`” 슬래시를 달지 않고 저장되어 있기 때문에 (당신이 원하지 않는데도 불구하고) 그냥 현재 디렉토리에 복구되어 버릴 수 있다. 따라서, 나는 우선 “tar” 명령을 “`t`” (표시) 옵션을 사용하여 검사를 해보고나서 명령을 실행하면 어떻게 풀릴 것인지 확실히 알고 있는 상태에서 “`t`” 를 “`x`” (추출) 로 바꿀 것을 강력히 권고한다.

저장 파일에 있는 모든 파일을 복구하려는 것이 아니라면, 복구하려는 파일의 이름을 지정하기 위하여 다음과 같이 할 수 있다:

```
tar -zxvpf /archive/full-backup-09-October-1999.tar.gz \
```

```
etc/profile usr/local/bin/tolower
```

이 명령은 저장 파일에서 "etc/profile" 파일과 "usr/local/bin/tolower" 파일만을 복구한다.

작은 정보: 저장 파일로부터 몇 개의 파일 만들 복구하고자 하는 경우에는 파일이름과 디렉토리를 경로를 저장 파일에 기록된 것과 정확히 똑같게 지정하여야 한다. 다음의 예를 보면 이해를 할 수 있을 것이다:

```
tar -ztvpf /archive/full-backup-09-October-1999.tar.gz \
    | grep -i profile
```

이 예에서는 저장 파일에 들어있는 모든 파일의 이름이 나오게 된다. 그 결과로 나오는 출력은 다시 "grep" 명령으로 연결되고 (대소문자를 구분하지 않도록 하기 위하여 "i" 옵션이 사용되었다) 그 결과로서 파일 이름이나 디렉토리 이름에 "profile" 이 들어있는 모든 파일을 볼 수 있게 된다. 이 상태에서 정말로 복구하고자 하는 파일의 정확한 이름을 확인하고 난 뒤, tar 명령에 추출 옵션과 그 정확한 파일 이름을 지정하면 된다.

8.1절 에서 언급한 것과 마찬가지로 저장 파일을 만들 때 tar 는 파일의 경로 이름에서 맨 앞에 있는 "/" (슬래시) 문자를 잘라낸다. 이는 복구 파일은 원래 그 파일이 백업된 곳과 같은 위치로 되돌아 가지 않는다는 것을 의미한다. 따라서 "/" root 디렉토리로 이동하거나 "--directory /" 옵션을 사용하여야 한다.

참고: 주의: 훨씬 더 안전한 방법은 일단 원하는 파일을 다른 디렉토리 (예를 들어, 당신의 홈 디렉토리) 아래에다가 복구한 다음 원래 위치에 있는 파일과 비교한 뒤 옮기던지 덮어쓰던지 하는 것이다.

8.2.2. "KDat" 를 이용한 복구:

KDat 가 만든 백업 셋트에서 하나 또는 여러 개의 파일을 복구하기 위해서는 백업 테이프를 드라이브에 넣고, "File" 메뉴 옵션에서 "Mount Tape" 를 선택하여야 한다. (또는 테이프같이 생긴 아이콘을 클릭한다.)

KDat 는 테이프로부터 헤더 정보를 읽으려고 시도를 한다. 만약 제대로 읽을 수 있으면 테이프 헤더에서 발견된 고유번호와 맞는 테이프 색인을 찾으려고 한다. 이 테이프 색인은 당신의 하드 드라이브에 저장되어 있으며 KDat 가 백업 테이프를 포맷할 때 마다 만들어지는 고유한 파일로서 백업을 수행할 때 마다 갱신된다.

만약 해당되는 테이프 색인이 없으면 (아마 다른 기계에서 만든 백업 셋트를 가지고 복구하려고 하거나 하드 드라이브에 있던 색인 파일이 삭제 또는 깨진 경우) KDat 는 그 사실을 당신에게 알려주고 테이프를 읽어서 색인을 다시 만들어도 되겠는지를 물어본다. 원하는 파일을 복구할 수 있기 위해서는 색인 파일을 다시 만들어야 하므로 당연히 "Yes" 를 클릭하여야 한다.

참고: (주의: 일단 테이프에 대하여 색인이 새로 만들어지면 이름이 "재색인된 테이프**(Reindexed Tape)**" 로 바뀐다. 따라서 원래의 이름으로 바꾸어 주어야 한다.)

일단 테이프 인덱스를 성공적으로 읽고 나면, 백업 프로파일을 만들 때와 마찬가지의 방식으로 (파일 선택 과정에 대한 상세한 설명은 8.1절 을 참조하시요), 백업 셋트로부터 복구하고자 하는 파일이나 디렉토리를 선택하는데 사용된다.

필요한 파일을 더 선택하고 나면, 메뉴 바의 "File" 옵션에서 "`Restore...`" 를 선택하여 (또는 테이프 복구 아이콘을 클릭하여) 복구 과정을 시작시키면 된다. KDat 는 대화상자를 띄워서 어느 파일을 복구할 것이지를 확인받는다. 게다가 파일이 복구되는 디렉토리를 지정할 수 있는 옵션도 있다. 이 기능을 이용하면 중요한 시스템 파일을 당신의 홈 디렉토리에 일단 꺼내서 내용을 비교한 뒤 나중에 원하는 위치로 옮기거나 갱신할 수 있다. 이는 사실 파일을 복구하는 가장 안전한 방식이다.

복구 과정을 시작시키려면, "`Okay`" 버튼을 클릭하면 된다. 그러면 KDat 는 테이프를 훑어가면서 선택된 파일을 복구한다.

어떤 경우에는 KDat 로 만든 백업 셋트를 KDat 를 사용하지 않고 복구하는 것이 필요하거나 유용할 수가 있다. 아마 구이 기반의 환경이 아니거나 KDat 를 원격지에서 실행시키기에는 연결된 네트웍이 너무 느리거나 하는 경우가 이에 해당할 것이다. 다행히도, KDat 는 백업 셋트를 모든 유닉스/리눅스 시스템에서 사용할 수 있는 명령행 기반의 도구인 "tar" 를 이용하여 만든다.

KDat 로 만든 백업 셋트를 tar 명령으로 복구하여야 한다면 그냥 tar 명령을 써서 만든 임의의 정상적인 백업 셋트를 복구할 때 사용하는 모든 옵션을 그냥 쓰면 된다. 하지만 주의할 사항은 백업 셋트가 압축 포맷으로는 저장되지 않는다는 것이다.

참고: 주의: KDat 백업 셋트를 tar 로 액세스하려고 하면 아마도 에러 메시지나 나오게 될 것이다. 이는 KDat 가 처음 테이프를 포맷할 때 추가한 헤더와 다른 정보들 때문에 나오는 것이다. tar 명령을 몇번 반복하여 실제 tar 저장 파일이 나타날 때까지 건너 뛰면 이 문제는 해결된다.

8.3. 시스코 라우터 설정 백업

우리 회사에서는 WAN 을 이용하여 여러 사무실을 연결하고 있다. 이들 원격지에는 ISDN 을 통하여 시스코 라우터가 연결되어 있거나 어떤 경우에는 인터넷과 WAN 의 연결을 제공하기 위하여 센트렉스 (Centrex) 데이터 회로를 사용한다. 시스코 라우터 제품은 네트웍 서버상의 TFTP ("Trivial File Transfer Protocol") 를 통하여 설정 파일을 읽거나 쓸 수 있도록 해준다. 라우터 설정이 바뀔 때마다, 백업을 관리할 수 있도록 설정 파일을 리눅스 서버에 저장하는 것이 중요하다.

TFTP 는 제대로 설정되지 않으면 실제로 보안상의 구멍이 되므로, 레드햇은 디폴트로 TFTP 서비스를 막아놓았다는 점에 주의하여야 한다. TFTP 데몬은 인증 과정을 거치지도 않고 누구든지 파일을 읽거나 쓸 수 있도록 허용한다. 내가 보통 셋업하는 방법은 "`/tftpboot/`" 디렉토

리를 만들어서 그 소유자를 루트로 하고 "/etc/inetd.conf" 파일에 있는 설정내용을 파일의 위치를 지정하도록 바꾼다:

```
tftpd   dgram   udp    wait   root   /usr/sbin/tcpd  in.tftpd  /tftpboot
```

> 참고: 주의: 줄의 맨 끝에 "/tftpboot" 라는 경로를 추가하는 것은 TFTP 데몬이 액세스할 수 있는 파일의 위치를 지정하는 역할을 한다. 물론 이렇게 지정하지 않고 TFTP 가 시스템의 어디든지 있는 파일을 액세스할 수 있도록 할 수도 있다. 하지만 TFTP 는 보안상 위험을 갖고 있다고 생각되므로 이렇게 하는 것은 좋은 생각이 아니다.

일단 TFTP 서비스를 쓸 수 있도록 하였으면 다음과 같이 타이핑하여야 한다:

```
killall -HUP inetd
```

이 명령은 INETD 데몬을 다시 시작시켜서 inetd.conf 파일의 수정 내용이 적용되게 해준다.

라우터 설정 파일의 백업을 만드는 것은 3단계로 이루어진다: 기존의 파일에 대하여 (없으면 하나를 만들어서) 쓰기를 허용하도록 퍼미션을 지정하고, 백업 파일을 쓰고, 그리고 나서 파일에 대한 액세스를 제한하도록 퍼미션을 리셋한다. 라우터 백업 세션의 예는 다음과 같다:

```
mail:~# cd /tftpboot
mail:/tftpboot# chmod a+w xyzrouter-confg
chmod: xyzrouter-confg: No such file or directory
mail:/tftpboot# touch xyzrouter-confg
mail:/tftpboot# chmod a+w loyola-confg
mail:/tftpboot# telnet xyzrouter
Escape character is '^]'.
User Access Verification
Password: ****
xyzrouter> enable
Password: ****
xyzrouter# write network
Remote host []? 123.12.41.41
Name of configuration file to write [xyzrouter-confg]?
Write file xyzrouter-confg on host 123.12.41.41? [confirm]
Building configuration...
Writing xyzrouter-confg !! [OK]
xyzrouter# exit
Connection closed by foreign host.
mail:/tftpboot# chmod a-wr,u+r xyzrouter-confg
mail:/tftpboot# exit
```

라우터가 (예컨대 천둥 번개로 발생한 서지 전압으로) 말썽을 일으키면 이러한 백업 파일이 라우터 설정을 다시 복구하는데 도움이 된다. 설정 파일을 복구하는 것도 역시 3단계로 이루어진다: 기존 파일의 퍼미션을 셋팅하고, 파일을 올린다음, 파일에 대한 접근을 통제하기 위하여 퍼미션을 리셋한다. 라우터 복구 세션의 예는 다음과 같다.

```
mail:~# cd /tftpboot
```

```
mail:/tftpboot# chmod a+r xyzrouter-confg
mail:/tftpboot# telnet xyzrouter
Escape character is '^]'.
User Access Verification
Password: ****
xyzrouter> enable
Password: ****
xyzrouter# config network
Host or network configuration file [host]?
Address of remote host [255.255.255.255]? 123.12.41.41
Name of configuration file [xyzrouter-confg]?
Configure using loyola-confg from 123.12.41.41? [confirm]
Loading xyzrouter-confg from 123.12.41.41 (via BRI0): !
[OK - 1265/32723 bytes]
xyzrouter# write
xyzrouter# exit
Connection closed by foreign host.
mail:/tftpboot# chmod a-wr,u+r xyzrouter-confg
mail:/tftpboot# exit
```

9장. 잡다한 관리 업무

리눅스는 인터넷 서버로 서비스에 사용한 지난 4년간 극도로 안정적이며 계속 실행시키기 위하여 수작업으로 관리하여야 할 것이 매우 적다는 것이 증명되었다. 가능한한 반복적이거나 지겨운 관리 업무는 crontab 이나 스크립트 파일을 이용하여 자동화하여야 하고 또 할 수 있다. 하지만 리눅스가 문제없이 계속 운영되게 하려면 가끔식 다양한 검사를 해주어야 한다. 여기에 포함되는 것으로는:

9.1. 저장 공간 검사

가끔식 저장 장치에 충분한 빈 공간이 있는지 확인하는 것은 중요하다. "`df`" 명령을 사용하면 남은 공간에 대한 레포트를 얻을 수 있다. 그 내용은 다음과 같이 생겼다 (여기에 나온 내용은 우리 회사에 있는 인터넷 서버에서 나온 것이다):

```
Filesystem         1024-blocks  Used Available Capacity Mounted on
/dev/sda1            1888052  135908  1654551      8%   /
/dev/sdd1            4299828  100084  3977246      2%   /archive
/dev/hda2            3048303  897858  1992794     31%   /archive2
/dev/hda1              11677    1380     9694     12%   /boot
/dev/sdc1            4299828  350310  3727020      9%   /home
/dev/sdb1            4299828  598504  3478826     15%   /usr
/dev/sda2            1888083  700414  1090075     39%   /var
/dev/scd0             593958  593958        0    100%   /cdrom
```

이들 파일 시스템은 상당히 안정적이어서 아주 서서히 크기가 늘어나고 있다. (위의 그림과 아래의 설명은 짝이 맞지 않습니다. 원저자의 오류인 것으로 보입니다:역자)

"/" (즉 root) 파일 시스템은 /dev/hda1 에 마운트되어 있으며 리눅스 커널, 디바이스 드라이버 그리고 다른 디렉토리를 포함한다. 사용자의 메일 메시지가 저장되는 곳 (**/var/spool/mail/**) 과 로그 파일이 저장되는 곳 (**/var/adm/**) 도 있지만 메일 메시지는 사용자들이 받아 가버리고 로그 파일은 순환이 되도록 되어 있어서 사용가능한 공간은 상당히 안정적인 모습을 보인다 (대략 한 달에 1% 정도가 늘어난다). 로그 파일은 순환이 되며 주단위로 자동으로 예전 것을 삭제하므로 대략 한달치 정도의 로그 파일을 갖고 있게 된다.

> 작은 정보**:** 팁: 이 파일 시스템이 급속도로 커지게 되면 /var/spool/mail 디렉토리가 범인일 가능성이 높다 -- 무지 큰 메일박스를 찾아보아야 한다. (예컨대 "`find /var/spool/mail -size +1000k`" 라고 하면 1Mb 이상의 크기를 가지는 메일박스의 목록을 볼 수 있다) 만약 크기가 1Mb 를 훨씬 넘는 파일이 있다면 그 사용자가 메일을 받고 있지 않거나, 많은 양의 메일을 보내는 메일링 리스트에 가입해 있거나, 사용자의 전자메일 패키지가 메일을 읽은 후에 서버에서 메일을 지우지 않도록 설정되어 있기 때문이다. 일단 사용자와 연락을 취하고 나서 메일 파일을 비워준다. 메일 파일을 비우기 위해서는 "`> 메일박스이름`" 이라고 해준다 (예를 들어, "`> smithj`" 라고 하면 Joe Smith 의 메일 박스를 비운다). 또한, "`/tmp/`" 디렉토리도 검사해서 틈나는 대로 비워준다 (중단된 뉴스 일기 세션에서 생성되는 오래된 tim* 파일이나 인쇄 파일등이 여기에 들어있다).

"/usr/" (즉 사용자) 파일 시스템은 /dev/hda2 에 마운트되어 있으며 사용자가 설치한 (여기서 사용자는 시스템 관리자가 사용자로서 설치하였다는 의미이다) 소프트웨어나 당신의 웹 사이트 내용 등이 들어있다. 이는 가장 큰 파일 시스템이며 매우 서서히 증가한다. 웹 페이지에 대한 로그 파일이 여기에 저장되어서 크기가 커질 수 있다; 따라서 정기적으로 확인하여 필요하다면 적당히 잘라 주어야 한다. 내 기계에서는 매달 초에 현재 웹 로그 파일을 월간 요약 로그로 (예를 들어, 11월의 로그는 access_log.11 로) 옮겨버린다. 연말이 되면 이들 로그 파일을 모두 삭제하고 처음부터 다시 시작한다 (따라서 매년 1월 1일에는 상당히 여유공간이 많아지게 된다).

> 작은 정보**:** 팁: 만약 이 파일 시스템이 빠르게 커진다면 "`/usr/local/etc/httpd/logs`" 와 "`/usr/local/squid/logs/`" 디렉토리를 검사해 볼 필요가 있다 (만약 있다면). 여기에는 (만약, 웹 사이트에 방문자가 엄청나게 많다면) 지나치게 커진 로그 파일이 있을 수 있다. 하지만 만약, 내가 하듯이 정기적으로 이전의 파일을 지워준다면 이 파일 시스템에서 공간 문제가 발생하지 않을 것이다 (사실, 로그는 내 사이트에 대한 방문통계를 만드는데 쓰고 있어서 가급적 안지우려고 한다). 그 외 혹시 지워야 할 파일이 있을 만 한 곳으로는 "`/usr/tmp/`" 디렉토리가 있다.

"/home/" (즉, 각 사용자의 홈 디렉토리) 파일 시스템은 /dev/hda3 에 마운트되며 모든 사용자의 디렉토리와 파일이 저장된다. 각 사용자에게 쉘 계정을 주지 않는 한 이 영역의 대부분은 사용자에게 쓸모없거나 접근불가능한 영역이 된다. (각 사용자 계정을 만들 때마다 여기에 디렉토리가 만들어지며 사용자 메일의 전달 등에 활용된다.) 하지만, 쉘 계정을 가지고 있는 사용자나 쉘은 아니라도 웹 페이지 사용자 (예를 들어, 개인 홈 페이지 사용자) 는 자기의 파일을 여기에 저장한다. 게다가 다른 배포판에서는 대개 웹 서비스의 기본 서버 페이지를 /usr 파일 시스템에 저장하는데 반하여, 레드햇에서는 기본 서버 페이지가 여기에 있는 /home/httpd 디렉토리 아래에 저장된다. (이에 대한 자세한 정보는 7.1절 을 참조하시요.)

이 파일 시스템은 쉘 계정을 아주 많이 만들어주지 않는 한, 아마 가장 느리게 크기가 커질 것이다.

> 작은 정보**:** 팁: 이 파일 시스템의 크기가 갑자기 커진다면 사용자들 중 누군가가 자기 디렉토리에 웹 페이지나 바이너리 파일을 왕창 넣었기 때문일 가능성이 크다. 사용자가 어떤 웹 페이지를 추가하였는지를 보여주는 "`/var/adm/xferlog.*`" 로그 파일을 검사하여 최근에 추가된 내용을 살펴보면 된다.

내 시스템에는 "/archive/" (즉, 저장용) 파일 시스템이 /dev/hdb1 에 마운트되어 있으며 그 크기는 1.02 Gb 로서 어떤 파일이건 (예를 들어, 데이터 파일이든 소프트웨어 키트 건 간에) 저장하는데 사용한다. 나는 (약 70%에 달하는) 상당히 많은 영역을 현재 시스템에 대한 디스크-대-디스크 완전 백업을 위한 용도로 사용한다. 간단히 말하자면 필요에 따라서 장치를 추가 마운트하여 이런 식으로 활용하면 된다.

씨디롬 드라이브는 "/mnt/cdrom/" 라는 이름으로 /dev/scd0 에 마운트 된다. 이 장치는 24배속 씨디롬으로서 ISO9660 포맷의 어떤 씨디라도 읽을 수 있다. 이 장치는 기본적으로 소프트웨어 설치를 위하여 사용되지만 도스/윈도용 씨디를 마운트하여 삼바 서비스를 통하여 윈도 3.x/95/NT 네트웍 공유로 사용하기도 한다. (이에 대한 상세한 내용은 7.4절 을 참조하시요.)

"`rm`" 명령은 파일을 삭제한다. 사용법은 "`rm 파일이름`" 이다. 확인을 해가며 파일을 지우려면 "`-i`" 옵션을 사용한다 (예. "`rm -i *`"). 이렇게 하면 각 파일에 대하여 지울 것인지를 물어

볼 것이다.

> 참고: (주의: 이는 보통의 쉘 사용자에게는 디폴트이지만 주의할 것은 -- 루트 계정은 "-i" 옵션을 지정하지 않는 한 파일을 지울 때 확인을 하지 않는다는 점이다!)

이 명령을 사용할 때에는 오타를 하지 않도록 특별히 주의하여야 한다 -- 특히 "루트" 로 로그인하였을 때에는 -- 왜냐하면 엉뚱한 파일을 지워서 후회하게 될 것이기 때문이다.

9.2. 프로세스 관리

때때로, 리눅스에서 현재 실행중인 프로세스들을 보고 싶을 때가 있을 것이다. 이러한 프로세스 리스트를 보려면, "`ps -aux`" 라고 타이핑하면 되는데 그 결과는 다음과 같은 식으로 나온다:

```
USER       PID %CPU %MEM  SIZE   RSS TTY STAT START   TIME COMMAND
bin         69  0.0  1.0   788   320  ?  S   Nov 30   0:00 /usr/sbin/rpc.portmap
frampton 10273  0.0  2.1  1136   664  p0 S    14:12   0:00 -bash
frampton 10744  0.0  1.1   820   360  p0 R    17:25   0:00 ps -aux
frampton 10745  0.0  0.8   788   264  p0 S    17:25   0:00 more
nobody   10132  0.0  1.8  1016   588  ?  S    13:36   0:00 httpd
nobody   10133  0.0  1.8   988   568  ?  S    13:36   0:00 httpd
nobody   10413  0.0  1.8  1012   580  ?  S    14:56   0:00 httpd
nobody   10416  0.0  1.8  1012   580  ?  S    14:56   0:00 httpd
nobody   10418  0.0  1.8  1012   588  ?  S    14:57   0:00 httpd
nobody   10488  0.0  1.7   976   556  ?  S    15:34   0:00 httpd
nobody   10564  0.0  1.8   988   564  ?  S    16:06   0:00 httpd
nobody   10600  0.0  1.8   988   564  ?  S    16:15   0:00 httpd
nobody   10670  0.0  1.8   988   568  ?  S    16:45   0:00 httpd
nobody   10704  0.0  1.7   976   552  ?  S    17:03   0:00 httpd
root         1  0.0  1.0   776   312  ?  S   Nov 30   1:13 init [3]
root         2  0.0  0.0     0     0  ?  SW  Nov 30   0:00 (kflushd)
root         3  0.0  0.0     0     0  ?  SW  Nov 30   0:00 (kswapd)
```

리스트에는 프로세스의 소유주 (웹 서버와 같은 특별한 서비스의 경우 소유주가 "nobody" 로 되어 있다), 프로세스 고유 번호, 프로세스가 현재 사용하고 있는 CPU 시간의 점유 비율, 프로세스가 사용하고 있는 메모리의 비율, 프로세스가 무엇을 하고 있는 지에 대한 설명 그리고 그 외 관련 정보가 나온다.

어떤 프로세스에 대한 더 자세한 정보를 얻으려면 "`ps pid`" (여기서 "`pid`" 는 프로세스 고유 번호 - the process identification number - 이다). 위의 예와 같은 상태에서 "`ps 10704`" 라고 하면 다음과 같이 나올 것이다:

```
10704  ?  S      0:00 /usr/local/etc/httpd/httpd
```

이 결과는 이 특정 프로세스가 웹 서버 (아파치 웹 서버는 프로세스 리스트에 여러번 나타나게 되는데 그 이유에 대하여는 7.1절 을 참조하면 된다) 라는 것을 나타낸다.

어떤 서비스가 제대로 운영되고 있지 않다는 것을 알게되면 "`kill -HUP pid`" (여기서 "`pid`"는 프로세스의 고유 번호로서 "ps" 명령에 의하여 나오는 프로세스 리스트에 있는 번호이다) 이라는 명령을 사용할 수 있다. 예를 들어, 인터넷 서비스 (inetd 라고 불리는 프로세스로서 우리의 예에서는 123번 프로세스이다) 가 원하는대로 동작하고 있지 않다면, "`kill -HUP 123`" (또는 더 안전하게 하고 싶다면 프로세스의 이름을 지정하는 명령인 "`killall`" 을 사용하여 "`killall -HUP inetd`" 라고 할 수도 있다) 라고 하여 프로세스를 재시작 시킬 수 있다. kill 명령에서 `-HUP` 옵션은 "전화를 끊음 - hang up" 을 나타낸다; 프로세스는 이를 스스로 재시작하라는 뜻으로 이해한다.

그래도 문제가 해결되지 않는다면 시스템을 내렸다가 다시 부팅하는 방법을 사용하여야 한다. (이에 대한 자세한 정보는 6.7절 을 참조하시요).

가끔은 어떤 프로세스를 잠시 중지시켰다가 나중에 계속 실행시켜야 할 때가 있다. 예를 들어, 현재 CPU 를 많이 사용하는 어떤 작업을 돌리고 있는데 동시에 IDE 기반의 CD 롬 라이터로 씨디를 구우려고 한다고 생각해보자. IDE 기반의 장치는 입출력을 위하여 CPU 에 상당히 많이 의존하므로 CPU 가 너무 바쁘다면 버퍼 처리를 제대로 할 수 없게 되고 그 결과로서 제대로 구워진 씨디가 아니라 쓸모없는 원반하나를 얻는 수가 생긴다! 다음 두개의 명령은 어떤 프로세스를 잠시 중지시키는 일과 재시작 시키는데 각기 사용된다:

```
kill -STOP 945
kill -CONT 945
```

레드햇에서는 프로세스를 시작시키고 중단시키는데 더 좋은 방법이 있는데 그 내용은 아래에 나오는 9.3절 에서 다루고 있다.

9.3. 프로세스의 시작과 중단

리눅스의 레드햇 배포판은 프로세스를 관리하는데 있어 약간 더 조직적인 방법을 제공한다. 프로세스 표에서 프로세스의 번호를 찾아 죽이는 대신에 "`/etc/rc.d/init.d`" 디렉토리에 스크립트를 모아놓고 필요에 따라 프로세스를 시작시키고 중단시킬 수 있게 해준다.

예를 들어, "`httpd`" (아파치 웹 서버) 를 중단시키려면 httpd 스크립트를 다음과 같이 실행시키면 된다:

```
/etc/rc.d/init.d/httpd stop
```

마찬가지로 "`start`" 옵션을 써서 서비스를 시작시킬 수 있다. 혹은, 설정 파일에 어떤 변경이 있어서 서비스를 재시작시켜서 그 변화를 반영하고 싶다면 "`restart`" 옵션을 사용하면 된다.

> 참고**:** (주의: 참으로 이상하게도 어떤 서비스에서는 "`restart`" 옵션이 제대로 지원되지 않는 것 같다.)

9.4. Cron 과 Crontab 파일을 이용하여 작업을 자동화하기

당신도 대부분의 리눅스 사용자와 마찬가지로, 반복적으로 어떤 특정한 시간에 어떤 작업을 시작하도록 예약할 필요를 느낄 것이다. 그러한 작업은 분 단위로 반복되는 것일 수도 있고 1년에 한번 반복되는 것일 수도 있다. 이러한 예약 기능은 "cron" 기능을 이용하여 구현할 수 있다.

리눅스의 cron 기능은 다른 유닉스와 거의 같게 구현되어 있다. 하지만, 레드햇은 작업의 스케쥴을 잡는데 있어서 다른 리눅스 배포판과는 약간 다른 방식을 채택하고 있다. 다른 배포판과 마찬가지로 스케쥴링 정보는 시스템내의 ("/etc/" 디렉토리의) "crontab" 파일에 다음과 같은 형식으로 저장된다:

```
분 시 일 월 연 명령
```

각 항목은 정수로 (예를 들어, 1월부터 12월을 나타내기 위하여 1부터 12사이의 숫자를 사용) 지정할 수도 있고 몇개의 항목은 와일드카드 문자로 인식되는 "*" 문자로 (예를 들어, 월에 해당하는 항목에 * 표시가 있으면 매월 지정된 날짜, 지정된 시간에 명령이 실행된다는 것을 의미한다) 표시할 수도 있다. 몇가지 예를 들어보면:

```
# 매년 6월 15일 오후 4시 30분에 시스템 로그 파일을 메일로 발송한다
30 16 15 06 * for x in /var/log/*; do cat ${x} | mail postmaster; done
# 관리자에게 계절이 바뀌었음을 자정에 알려준다
00 00 20 04 * echo 'Woohoo, spring is here!'
00 00 20 06 * echo 'Yeah, summer has arrived, time to hit the beach!'
00 00 20 10 * echo 'Fall has arrived.  Get those jackets out.  :-('
00 00 20 12 * echo 'Time for 5 months of misery.  ;-('
```

표준출력 (즉, 터미널) 으로 출력이 나오는 명령, 예를 들어 위의 예에서 사용한 "echo" 명령, 의 출력은 "루트" 계정에게 메일로 전송된다. 이렇게 메일이 가지 않도록 하려면 다음의 예처럼 출력을 널 장치 (null device) 로 파이프 연결하면 된다:

```
00 06 * * * echo 'I bug the system administra-
tor daily at 6:00am!' >/dev/null
```

표준인 "crontab" 외에도 레드햇은 몇개의 디렉토리를 추가하였다:

```
/etc/cron.hourly/
/etc/cron.daily/
/etc/cron.weekly/
```

이름에서 알 수 있듯이, 이들 디렉토리의 아무 곳에나 실행가능한 파일을 둘 수 있으며, 이들 파일은 각기 매시, 매일, 매주 단위로 실행된다. 이 방식은 빈번한 작업을 지정할 때에는 시간을 절약해줄 수 있다; 그냥 실행가능한 스크립트나 프로그램을 적절한 디렉토리에 가져다 놓고 (또는 다른 곳에 있는 파일에 대하여 심볼릭 링크를 걸어놓고) 그 일은 신경쓰지 않아도 된다.

10장. 리눅스와 응용 프로그램을 업그레이드하기

리눅스 시스템을 최대한 활용하기 위해서는 즉, 기능을 추가하거나 잠재적인 버그를 제거하거나 보안상의 구멍이 없도록 하기 위해서는 -- 리눅스 커널, 모듈, 사용자 응용 프로그램을 포함하여 -- 서버를 계속 업그레이드 해주는 것이 좋다. 때때로는 더 큰 하드 드라이브를 달아준다든지하는 하드웨어의 업그레이도 필요하다. 이 장에서는 이러한 이슈를 다룬다.

10.1. 레드햇 패지지 관리자 (RPM) 사용하기

리눅스의 레드햇 배포판은 커널을 포함하여 라이브러리, 응용 프로그램이 모두 RPM 파일로 제공된다. RPM 파일은 "패키지" 라고 부르기도 하는데 소프트웨어를 쉽게 설치, 업그레이드, 조회, 제거할 수 있도록 소프트웨어를 배포하는 방법이다. RPM 파일은 패키지의 이름, 버전, (필요한 경우) 다른 파일의 의존성 정보, (인텔버전인지 알파버전인지 등을 나타내는) 플랫폼, 파일을 설치하는 디폴트 위치 등의 정보를 포함하고 있다.

RPM 유틸리티는 레드햇에 의하여 처음 개발되었으며 리눅스 커뮤니티 전체에 오픈 소스 제품으로 제공되었다. 다른 개발자들이 이를 이용하여 추가의 기능을 넣기도 하였다. 파일을 RPM 방식으로 패키징하는 것은 아주 널리 퍼졌으며 이제는 레드햇뿐 만 아니라 다른 배포판에서 사용하기도 한다.

인기있는 리눅스 응용 프로그램은 거의 항상 곧바로 RPM 파일로 공개된다. 하지만 유닉스 세계에서 패키지 배포의 사실상 표준은 여전히 소위 "타르볼(tarball)" 이다. 타르볼은 "`tar`" 유틸리티로 읽을 수 있는 단순한 파일이다. 타르를 이용하여 프로그램을 설치하는 것은 대개의 경우 RPM 을 이용하는 것보더 훨씬 더 지겹다. 그렇다면 왜 사람들은 그렇게들 하고 있는가? 불행히도 최신판의 패키지를 RPM 으로 변환하는데 몇주씩 걸리는 경우가 있다 (많은 개발자들은 최초의 배포판은 타르볼로 내놓는 경우가 많다).

만약 타르를 이용하여 시스템이나 응용 프로그램을 설치하거나 업그레이드하기 시작하면 당신의 RPM 데이터베이스는 낡은 것이 되고 일관성을 잃게 된다. 이게 뭐 엄청난 일은 아니지만 (내가 슬랙웨어를 쓸 당시에는 -- 다른 선택의 여지도 없었지만 -- 대단한 불편없이 타르만을 사용하였다) 가능한한 나는 RPM 이 나올 때까지 인내심을 갖고 기다리거나 패키지의 개발자에게 아주 정중하게 요청을 한다. (당신이 스스로 RPM 파일을 만들어 다른 사람들에게 배포할 수도 있다. 이는 그런 파일을 만들 능력이 없거나 시간이 없는 개발자들에게 도움이 될 수 있다.)

어떤 소프트웨어가 RPM 형태로 있는지를 확인하는 정말로 좋은 장소는 http://rufus.w3.org/linux/RPM/ 에 있는 RPM 창고이다. 이 창고에는 어떤 RPM 파일을 찾는데 도움이 되도록 색인으로 꾸며진 범주를 제공하며 수천가지의 파일에 대한 링크를 갖고 있다.

어떤 패키지를 조회하려면, "`rpm -q 패키지이름`" (예. "`rpm -q pine`") 을 사용하면 된다. RPM 은 현재 어떤 버전이 설치되어 있는지 아니면 설치되지 않았는지를 알려준다.

그 패키지가 이미 설치되어 있고 이전 버전이라서 새로 다운로드 받은 (꼭 미리 받아두어야 함) 패키지로 갱신하고 싶다면 "`rpm -Uvh 패키지이름`" 이라고 하여 갱신을 적용할 수 있다. 모든 것이 순조롭게 실행되면 패키지는 자동으로 설치되고 곧바로 사용할 수 있는 상태가 된다. 만약 문제가 있다면 RPM 은 꽤 상세한 이유를 알려줄 것이다 (예를 들어, 다른 지원 패키

지를 먼저 설치하여야 한다든가 하는 식으로). 이런 경우 아주 조금이나마 생각을 해보아야 하겠지만 문제를 알아내는 과정은 아주 단순하다.

한편으로 패키지가 한번도 설치되지 않았고 그래서 설치하기로 했다면 "`rpm -ivh 패키지이름`" 이라고 타이핑하면 된다. 만약 다른 지원 패키지가 필요하다면 RPM 이 알려줄 것이다.

가끔 소스 포맷으로만 구할 수 있는 패키지를 설치하여야 하는 경우가 있다. 사실 믿을만한 곳 (예컨대 레드햇 FTP 사이트) 에서 가져온 패키지를 설치하는 것이 아닌한 바이너리 프로그램은 트로이의 목마나 고약한 것이 들어있을 수 있으므로 (물론 소스 파일에도 그런 것이 들어있을 수도 있지만 다른 개발자에 의하여 곧바로 들통이 나기 때문에 그렇게 하지 않을 것이라고 봐야 한다) 반드시 소스 프로그램으로부터 설치를 하여야 한다.

소스로부터 패키지를 설치하는 방법은 RPM 유틸리티에 "`다시 만들기(rebuild)`" 스위치를 지정하는 것이다. 예를 들어:

```
rpm -ivh --rebuild foo.src.rpm
```

위의 명령은 "foo" 패키지를 설정하여 컴파일하여 (인텔 플랫폼에서 리눅스를 쓰는 경우에는) "`/usr/src/redhat/RPMS/i386/`" 디렉토리에 바이너리 RPM 파일을 만들어낸다. 그런 다음 보통의 패키지를 설치하듯이 설치하면 된다.

마지막으로, 만약 소스 패키지를 컴파일하는데 문제가 발생한다면 (아마도 makefile 을 수정하거나 설정 옵션을 바꾸거나 한다면) 소스를 컴파일하고 새로운 바이너리 패키지를 만들고 바이너리 패키지로부터 설치하기 위하여 다음과 같은 과정을 밟아야 한다 (다시 한번 상상의 "foo" 패키지를 예로 들었다):

```
rpm -ivh foo.src.rpm
cd /usr/src/redhat/SPECS
pico -w foo.spec
```

무엇이든 간에 ".spec" 파일에 수정할 것이 있으면 수정한 뒤 다음과 같이 타이핑한다:

```
rpm -ba foo.spec
```

이 명령은 ".spec" 에 수정한 내용에 따라 패키지를 다시 만든다. 위에서와 마찬가지로 결과로서 만들어지는 바이너리 RPM 파일은 "`/usr/src/redhat/RPMS/i386/`" 에 들어 있게 되고 나머지 설치과정은 정상적인 RPM 파일처럼 하면 된다.

RPM 에 대한 더 자세한 정보는 레드햇의 문서를 참조하여야 한다. 이 도구는 자세하게 배워둘만한 가치가 있는 극도로 강력한 도구이다. RPM 에 대한 정보의 최고의 원천은 "Maximum RPM" 인데 이는 포스트스크립트 포맷으로 http://www.rpm.org/maximum-rpm.ps.gz 에서 구할 수 있다. (만약 이 파일을 인쇄하기로 결정했다면 많은 종이가 필요할 것이라는 점을 명심하라!) 좀 간단한 안내서로는 http://www.rpm.org/support/RPM-HOWTO.html 에 있는 "`RPM-HOWTO`" 가 있다.

10.2. RPM 없이 설치하고 업그레이드 하기

가끔은 RPM 패키지가 없는 상황에서 어떤 응용 프로그램을 설치하거나 업그레이드해야 하는 경우가 있다. 물론, 그렇게 하는 것도 가능은 하지만 (그리고 "실제" 유닉스 세계에서는 그렇게 하는 것이 "사실상의 표준"이긴 하지만) 절대적으로 필요한 경우가 아니라면 그러지 않도록 추천한다 (그 이유에 대하여는 10.1절를 참조하시요).

어떤 것을 타르볼로부터 설치해야만 한다면, 시스템 전체가 사용할 소프트웨어의 설치 장소에 대한 일반적인 법칙은 "`/usr/local/`" 파일 시스템에다가 갖다 두는 것이다. 따라서, 소스 타르볼은 "`/usr/local/src/`" 에 풀어야 하고 이를 컴파일해서 만든 결과는 "`/usr/local/bin`" 에 들어가야 하며 설정 파일은 "`/usr/local/etc/`" 에 둔다. 이러한 규칙을 따르는 것이 시스템 관리를 조금이나마 편하게 해 줄 것이다. (비록, RPM 만 쓰는 시스템 만큼은 편하지는 않겠지만)

마지막으로 타르볼로부터 개인적인 용도로 소프트웨어를 설치하고자 하는 일반 사용자는 자기의 홈 디렉토리 아래에 설치하여야 한다.

믿을만한 소프트웨어 제공 사이트에서 타르볼을 다운로드한 다음 적당한 상위 수준의 디렉토리로 가서 다음의 예에 나온 것 처럼 (필요하다면 루트로서) 다음과 같은 명령을 타이핑하여 파일을 푼다:

```
tar zxvpf cardgame.tar.gz
```

위의 명령은 "`cardgame.tar.gz`" 라는 이름의 압축된 파일로부터 모든 파일을 꺼낸다. "`z`" 옵션은 저장 파일이 gzip 을 이용하여 압축되었다는 것을 의미한다 (따라서 타르볼 파일이 압축되어 있지 않다면 이 옵션을 빼주어야 한다); "`x`" 옵션은 저장 파일로부터 모든 파일을 다 꺼내겠다는 것을 나타낸다. "`v`" 옵션은 자세한 설명을 보여달라는 것이며 따라서 파일을 꺼내면서 파일 이름의 목록을 보여준다. "`p`" 옵션은 저장 파일이 만들어질 당시의 원래 퍼미션을 유지하라는 의미이다. 마지막으로, "`f`" 옵션은 그 다음의 인자가 저장 파일의 이름이라는 것을 의미한다. tar 명령의 옵션은 대소문자를 구분한다는 것을 꼭 명심하여야 한다.

> **경고**
>
> 특별 주의: 8.2.1절 에서도 언급하였듯이 실제로 파일을 꺼내기 전에, 처음에는 "**t**" 옵션을 사용하여 저장 파일의 내용을 점검해보아야 한다. 그렇게 함으로써 원하지 않는 위치에 파일을 꺼내는 또는 심지어는 기존의 파일을 덮어써버리는 위험을 피하는데 도움이 된다.

일단 타르볼이 적절한 디렉토리에 설치되면 새로이 설치된 파일에 포함되어 있는 "`README`" 또는 "`INSTALL`" 파일을 찾을 수 있을 것이다. 이들 파일에는 소프트웨어 패키지를 사용하기 위하여 어떻게 해야하는지에 대한 자세한 설명이 들어 있다. 대개의 경우에는 다음의 예시와 같은 명령을 타이핑해야할 것이다:

```
./configure
make
make install
```

위의 명령은 당신의 시스템에서 소프트웨어가 성공적으로 컴파일될 수 있는 기능과 라이브러리가 갖추어져 있는지를 확인하여 설정하고, 패키지를 컴파일하여 모든 소스 파일을 실행 가능한 바이너리 파일로 바꾸고, 바이너리 파일과 필요한 보조 파일을 적절한 위치에 설치한

다. 당신이 실제로 밟아야할 절차는 소프트웨어 패키지가 다양한 만큼 다를 수 있다. 따라서 포함된 문서를 완전히 읽어보아만 한다.

다시 한번 말하지만 절대적으로 필요한 경우가 아니라면 타르볼을 피하고 RPM 에 매달릴 것을 권한다.

10.3. 시스템을 최신으로 유지하는 전략

가끔은 여기 저기에서 리눅스 커널이나 사용자 응용 프로그램에 중요한 업그레이드가 있다는 얘기를 듣게 될 것이다. 이런 소식은 잡지, 뉴스그룹, 웹 페이지 등에서 듣게 된다.

아마도 리눅스 관리자가 꼭 -- 아니야, 반드시 -- 살펴보아야 할 최고이자 유일한 온라인 자료는 http://freshmeat.net/ 웹 사이트이다. 이 사이트는 새로운 오픈 소스 응용 프로그램과 프로젝트, 문서, 그외 리눅스 커뮤니티에 관심을 끌만한 발표 등에 대한 설명을 담고 있다.

그 외에 새로운 응용 프로그램 발표를 추적할 수 있는 곳은 comp.os.linux.announce (news:comp.os.linux.announce) 뉴스 그룹이다. 이 뉴스그룹은 리눅스와 관련된 새로운 응용 프로그램, 커널이나 응용 프로그램의 업그레이드, 웹 페이지 등을 다루고 있다. 이 뉴스그룹은 관리되는 뉴스그룹이기 때문에 높은 "S/N (잡음대 신호 signal to noise)" 비를 나타낸다.

모든 제품이 comp.os.linux.announce 를 통하여 발표되는 것은 아니다. 따라서, 당신이 사용하고 있는 응용 프로그램에 대한 웹 페이지나 FTP 사이트를 방문하는 것도 좋은 생각이다.

10.4. 리눅스 커널 업그레이드

가끔은 리눅스 커널을 업그레이드해 주는 것이 좋다. 이렇게 함으로써 새로운 기능의 추가나 버그의 해결이 이루어진 최신판을 유지할 수 있다. 그 외에도 새로 나왔거나 특이한 하드웨어에서 리눅스를 돌리기 위해서 또는 어떤 특별한 기능을 활용하기 위해서는 맞춤 커널이 필요하게 된다.

이 섹션에서는 새로운 커널로 업그레이드 하거나 맞춤 커널을 만드는 방법에 대하여 설명한다. 이는 당신이 생각하는 만큼 어렵지는 않다!

새로운 커널 버전에 대한 발표는 여러 곳에서 얻을 수 있는데 그 중에는 comp.os.linux.announce (news:comp.os.linux.announce) 뉴스그룹이나 http://freshmeat.net/ 그리고 http://slashdot.org/ 와 같은 웹 사이트가 있다.

현재 커널 개발에는 두 "줄기" 의 흐름이 있음을 알아야 한다 -- 한 줄기는 "안정" 판이고 또 다른 줄기는 "개발" 판이다. 인터넷 서버와 같은 중요한 업무에는 개발판 커널을 멀리하고 안정판을 사용할 것을 강력히 권장한다.

두 줄기의 차이는 개발판 커널에서는 새롭고 아직 검사되지 않은 하드웨어 드라이버, 파일시스템, 그리고 그 외 "첨단" 개발 내용이 정기적으로 추가된다는 점이다. 이런 커널은 해커들만 사용하는 것이다 -- 이들은 시스템을 리부팅하던지 커널에 엄청난 버그가 숨어있던지 개의치 않는 사람들이다.

안정판 커널에는 새로운 기능이나 드라이버는 완전히 검사된 뒤에 포함시킨다. 이 줄기에서 마이너 릴리즈는 그래도 남아있는 버그를 제거하는 역할을 한다.

두 줄기는 서로 다른 버전 번호를 사용하여 구분하기 쉽도록 되어 있다. 안전판 커널은 두번째 숫자가 짝수이고 (예. 2.0.35, 2.0.36, 2.2.4) 반면에 개발판 커널은 두번째 숫자가 홀수이다 (예. 2.1.120, 2.1.121, 2.3.0).

최신의 안정판 커널은 ftp://ftp.redhat.com/redhat/updates/ FTP 사이트에 소스와 미리 컴파일된 바이너리 형태로 항상 올려져있다. 당신이 사용할 플랫폼과 버전에 맞는 커널 패키지를 찾아서 다운로드한다 (예를 들어, 인텔 플랫폼에 돌아가는 6.1 버전은 "`/6.1/i386/`" 디렉토리에 있는 "`kernel-*.i386.rpm`" 파일을 다운로드 하면 된다).

> 참고**:** 주의: 스스로 맞춤 커널을 만들려고 하는 것이 아니라면 커널 소스 파일을 다운로드할 필요는 없다. (맞춤 커널을 만드는 방법에 대하여는 10.6절 을 참조하시요)

때때로, 아직 RPM 형태로 만들어져 있지 않은 커널을 사용할 필요가 있는 경우가 있다. 이런 경우에는 최신의 커널을 ftp://ftp.kernel.org FTP 사이트의 /pub/linux/kernel/ (ftp://ftp.kernel.org/pub/linux/kernel/) 디렉토리에서 찾을 수 있다. 적절한 메이저 버전 서브디렉토리로 가면 (예. "`v2.0`") 그곳에 최신판의 커널이 들어있다. 원하는 커널 패키지를 (예를 들어, 인텔용 압축된 2.0.36 타르볼의 이름은 "`linux-2.0.36.tar.gz`" 이다) 찾아서 "`/usr/src`" 디렉토리에 푼다.

> 참고**:** 주의: RPM 을 통하지 않고 설치되는 모든 사용자 응용 프로그램은 "`/usr/local/src/`" 디렉토리 아래에 풀어야 하지만 이것은 커널 트리이므로 예외이다. :-)

타르볼을 다운로드하여 커널을 업그레이드하기로 결정하였다면 당신 스스로 설정, 컴파일, 설치를 하여야 한다. 아주 최근의 개발판 커널을 요하는 특별한 경우가 아니라면 나는 레드햇이 제공하는 RPM 파일을 사용하여 커널을 업그레이드할 것을 강력히 추천한다 -- 이들 RPM 파일은 미리 설정하여 미리 컴파일되어 있지만 필요하다면 RPM 파일로부터 맞춤 커널을 컴파일 할 수도 있다.

10.5. 레드햇 제공 커널로 업그레이드하기

커널을 업그레이드하는 가장 쉬운 방법은 레드햇이 제공하는 기성 커널 RPM 을 이용하는 것이다. 이들 RPM 파일은 미리 컴파일된 바이너리 커널 코드를 포함하고 있고 다양한 하드웨어를 지원하며 인기있는 기능을 제공한다.

기성 커널을 설치하는 것은 쉽고 별다른 위험이 따르지 않는다. 루트로서 그냥 다음과 같은 명령을 주욱 타이핑만 해주면 된다:

```
rpm -Uvh kernel-2.0.36.i386.rpm
cd /boot
ls
```

새로운 커널의 이름은 앞의 내용중에서 "ls" 명령으로 알 수 있다. 눈길을 끄는 파일은 "vmlinuz" 파일이다; 예를 들어, 2.0.36 버전 커널의 세번째 RPM 판의 이름은 "vmlinuz-2.0.36-3" 이 된다.

이제 LILO 설정 파일을 수정하기 위하여 편집기를 띄운 다음 (예를 들어, "pico -w /etc/lilo.conf" 라고 타이핑한다) "image=/boot/..." 라고 되어 있는 줄을 수정하여 새로운 커널을 가리키도록 한다. 그런 다음 "/sbin/lilo" 라고 타이핑한다. 만약, LILO 가 에러 메시지를 내면 "lilo.conf" 파일에 들어있는 커널의 이름과 "/boot/" 디렉토리에 있는 커널을 이름을 점검해보아야 한다.

경고

특별 주의: 이 단계를 빼먹으면 안되요!

(위의 명령은 당신이 인텔 플랫폼에서 LILO 를 이용하여 부팅한다는 것을 전제로 하고 있다. LILO 부트 로더에 대한 상세한 내용은 4.8절 을 참조하시요).

커널을 업그레이드하고 부트 로더 정보를 갱신한 뒤 시스템을 내린다음 새로운 커널로 재부팅할 수 있다 (시스템을 내리는 방법에 대하여는 6.7절 을 참조하시요).

10.6. 맞춤 커널 만들기

기성 커널에서는 지원되지 않는 하드웨어나 기능을 사용하고 싶거나 시스템 메모리를 더 효과적으로 활용하기 위하여 커널 메모리 공간을 줄이고 싶다면 새로운 맞춤 커널을 만들 필요가 있다.

커널을 업그레이드하는 과정은 필요로하는 모듈을 설정하고, 커널과 모듈을 컴파일하고 마지막으로 커널 이미지를 설치하는 과정으로 구성된다. 그런 다음 새로운 커널을 올리기 위하여 (두손 모아 빌면서!) 시스템을 재부팅한다. 이러한 과정에 대하여는 각 커널 패키지에 따라오는 "README" 파일에 들어있다. 더 자세한 자료는 "Documentation/" 디렉토리 아래에 들어있다. 거기에 들어있는 파일 중에서 특별히 도움이 될만한 파일로는 "Configure.help" 파일이 있는데 이 파일에는 사용가능한 커널 컴파일 옵션과 모듈에 대한 상세한 정보가 들어있다.

다음의 예시 세션은 인텔 플랫폼에서 2.0.36 버전의 맞춤 커널을 만드는 과정을 보여준다. 대개 맞춤 커널을 만드는 것은 그냥 설정하고 컴파일하고 설치하는 과정에 지나지 않는다. 가끔은 (대개 새로운 하드웨어를 사용하려는 경우) 컴파일 하려는 커널이 지원하지 않는 하드웨어를 지원하기 위하여 추가의 드라이버 소프트웨어를 다운로드 받아야 한다.

맞춤 커널을 만드는 첫번째 단계는 RPM (이게 더 낫다) 또는 타르볼을 다운로드하여 커널 소스를 설치하는 것이다. 적절한 파일을 찾는 방법에 대하여는 10.4절 을 참조하시요.

그 다음은 "rpm" 유틸리티 (또는 상황에 따라서는 "tar") 를 사용하여 커널 소스와 헤더 파일을 설치한다. 예를 들어 2.0.36-3 커널 RPM 파일을 설치하려면:

```
rpm -Uvh kernel-source-2.0.36-3.i386.rpm kernel-headers-2.0.36-3.i386.rpm
rpm -Uvh kernel-ibcs-2.0.36-3.i386.rpm
```

(만약 노트북에서 리눅스를 실행시킨다면 전원 관리 기능을 제공하는 "kernel-pcmcia-cs-2.0.36-3.i386.rpm" 파일도 설치하여야 할 것이다.)

커널 파일을 설치하고 나면 새로운 소스 트리를 "/usr/src/linux/" 디렉토리에서 볼 수 있을 것이다.

그 다음 단계는 (만약 필요하다면) 추가의 드라이버 파일을 다운로드하고 새로운 커널 소스 트리에 설치한다. 예를 들어, Mylex DAC960 하드웨어 RAID 컨트롤러를 지원하는 기능을 추가하려면, http://www.dandelion.com/ (http://www.dandelion.com/) 웹 사이트에서 드라이버 소프트웨어를 다운로드 한다. 불행히도 그런 드라이버 소프트웨어는 대개 타르볼로만 제공되므로 "tar" 유틸리티를 사용하여 설치하여야 한다. 예를 들면:

```
cd /usr/src/
tar zxvpf DAC960-2.0.0-Beta4.tar.gz
```

추가 드라이버 소프트웨에 따라오는 문서가 있다면 꼭 읽어보아야 한다. 예를 들어, DAC960 드라이버에는 "README" 파일이 있는데 이 파일에는 새로 다운로드 받은 파일이 어디에 놓여야 하며 어떻게 커널 패치를 적용하는지에 대한 설명이 들어있다:

```
mv README.DAC960 DAC960.[ch] /usr/src/linux/drivers/block
patch -p0 < DAC960.patch
```

다음 단계는 시스템의 심볼릭 링크가 새로운 커널 트리를 제대로 가리키고 있는지 확인하는 것이다. 사실 이 단계는 딱 한번 해주면 된다. 따라서 다음의 내용은 이전에 한번도 맞춤 커널을 컴파일 한 적이 없는 경우에만 해주면 된다:

```
mail:/usr/src# cd /usr/include
mail:/usr/include# rm -rf asm linux scsi
mail:/usr/include# ln -s /usr/src/linux/include/asm-i386 asm
mail:/usr/include# ln -s /usr/src/linux/include/linux linux
mail:/usr/include# ln -s /usr/src/linux/include/scsi scsi
```

참고: 주의: 위의 단계는 2.2.x 나 그 이상의 커널 버전에서는 이제 필요없어졌다.

다음 단계는 커널 셋팅을 설정하는 것이다. 이는 맞춤 커널을 만드는데 있어 가장 중요한 단계이다. 어떤 셋팅을 실수로 안쓰도록 하면 어떤 원하는 기능이나 필요로 하는 하드웨어를 지원하지 못하게 될 수 있다. 한편 필요없는 셋팅을 사용하도록 하면 쓸데없이 커널의 크기가 커져서 소중한 시스템 메모리를 낭비하게 된다 (물론 앞의 경우보다는 뒤의 경우가 차라리 낫긴하다).

커널을 제대로 컴파일하는 것인지를 확실히 하는 가장 좋은 방법은 어떤 기능을 필요로 하며 시스템에서 지원을 필요로 하는 하드웨어가 무엇인지를 정확히 아는 것이다. 맞춤 커널을 만드는 경험을 몇 번하고 나면 이 과정은 이제 "뻔할 뻔자"가 되어 별로 두렵지 않게 될 것이다!

다음과 같이 타이핑하면 설정 과정이 시작된다:

```
mail:/usr/include# cd /usr/src/linux
mail:/usr/src/linux# make mrproper
mail:/usr/src/linux# make menuconfig
```

(만약 X 윈도우 시스템을 사용하고 있다면 "`make menuconfig`" 대신에 "`make xconfig`" 라고 타이핑할 수 있다; X 윈도우 시스템이 어떻게 동작하는 지에 대한 자세한 정보는 5장 을 참조하시요.)

커널을 설정하려면 다양한 셋팅 중에서 필요로 하는 것들을 (쓸 수 있도록) 선택하고 필요로 하지 않는 것들을 (쓰지 않도록) 선택을 꺼주면 된다. 지원기능을 커널에 만들어 넣을 수도 있고 모듈로 만들어서 필요에 따라 커널이 올렸다 내렸다하게 할 수도 있다. (스카시 드라이버처럼 시스템을 부팅하는데 필요한 기능을 모듈로 컴파일한다면 램디스크 이미지를 만들어야 한다. 그렇게 하지 않으면 아예 부팅이 되지 않는다. 이를 위해서는 "`mkinitrd`" 명령을 사용하여야 한다; 이 과정에 대하여는 아래에서 다루었다.)

설정을 하는 도중에 `<Help>` 를 선택하면 각 커널 옵션이 어디에 사용되는지에 대한 설명을 볼 수 있다.

커널 셋팅을 설정하고 나면 다음과 같은 명령을 타이핑하여 커널을 컴파일 한다:

```
mail:/usr/src/linux# make dep ; make clean
mail:/usr/src/linux# make bzImage
mail:/usr/src/linux# make modules
```

만약 이전에 갖고 있던 것과 같은 커널을 (이 예에서는 2.0.36-3) 다시 컴파일 하는 것이라서 기존의 모듈을 백업 디렉토리로 옮겨두고 싶으면 다음과 같은 명령을 타이핑하면 된다:

```
mail:/usr/src/linux# mv /lib/modules/2.0.36-3 /lib/modules/2.0.36-3-backup
```

새 모듈을 실제로 설치하기 위해서는 다음과 같이 타이핑하면 된다:

```
mail:/usr/src/linux# make modules_install
```

그 다음 단계는 커널을 "`/boot/`" 디렉토리로 복사하고 LILO 를 사용하여 부트 기록을 갱신함으써 새로운 커널을 인식시키는 것이다. 다음의 명령은 기존 커널의 백업본을 만들고 새로운 커널을 덮어서 복사하고 LILO 부트 레코드를 갱신하는 역할을 한다:

```
mail:/usr/src/linux# cd /boot
mail:/boot# cp vmlinuz vmlinuz.OLD
mail:/boot# cp /usr/src/linux/arch/i386/boot/bzImage vmlinuz-2.0.36
mail:/boot# /sbin/lilo
```

마지막으로, "`/etc/lilo.conf`" 파일을 편집하여 "이미지 (image)" 가 새로 만들어진 커널을 확실히 가리키도록 해준다. 그리고 백업 커널을 카리키는 섹션, 예컨대 "OldLinux" 를 만들어 준다. 예시 파일은 다음과 같다:

```
boot=/dev/hda
map=/boot/map
install=/boot/boot.b
prompt
timeout=50
image=/boot/vmlinuz
label=Linux
root=/dev/hdb1
```

```
read-only
image=/boot/vmlinuz.OLD
label=OldLinux
        read-only
```

백업 커널 정보를 이렇게 추가해두면 새로운 커널이 (장치가 제대로 인식되지 않는다거나 데몬이 제대로 시작되지 않는다거나 하는 식으로) 제대로 부팅되지 않을 경우 그냥 "`OldLinux`" 라고 타이핑하여 이전의 커널로 부팅한다음 문제를 조사해 볼 수 있다.

> 참고**:** 주의: 앞에서도 언급했듯이 부팅에 필요한 기능을 모듈로 컴파일하였다면 시스템을 부팅하기 위하여 초기 램디스크 이미지를 만들어야 한다. (커널을 그러한 초기 부트 이미지를 지원하도록 컴파일 해두어야 한다.)

초기 램디스크 이미지를 만들고 사용하는 과정은 다음과 같다:

- "`/etc/lilo.conf`" 파일에 초기 램디스크 이미지로부터 부팅하도록 항목을 추가한다; 앞에서 예시한 설정 파일에 추가하면 다음과 같이 된다:

  ```
  image=/boot/vmlinuz
          label=Linux
          root=/dev/hdb1
          initrd=/boot/initrd-2.2.4-4.img
          read-only
  ```

- mkinitrd 명령을 사용할 수 있으려면 루프백 장치가 로드되어 있어야 한다. 루프백 장치 모듈이 로드되어 있는지 확인하려면:

  `/sbin/insmod loop`

  (루프백 모듈을 로드할 수 없다는 에러 메시지가 나온다면 시스템이 실행중인 현재 커널에 대한 전체 경로를 - 예를 들어, "`/lib/modules/2.0.35/loop`" - 지정해 줄 필요가 있다.)

- "`mkinitrd`" 명령을 사용하여 이미지를 실제로 만든다:

  `/sbin/mkinitrd /boot/initrd-2.0.36-3.img 2.0.36-3`

- 부트 로더를 갱신하기 위하여 "`/sbin/lilo`" 를 실행한다.

이제 시스템을 내렸다가 새로운 커널로 부팅한다!

```
mail:/boot# /sbin/shutdown -r now
```

커널이 한번에 재대로 부팅되지 않더라도 허둥댈 필요는 없다. 리눅스를 설치할 때 만든 부트 디스크로 부터 부트를 하면 된다. 만약 이 디스클 갖고 있지 않다면 레드햇 씨디로부터 하나를 새로 만들 수도 있다. 부트 디스켓을 드라이브에 넣고 컴퓨터를 재부팅한 뒤 "boot:" 가 나타나면 다음과 같이 타이핑한다:

```
mount root=/dev/hda1
```

위의 명령은 당신의 "/" (root) 파티션이 /dev/hda1 에 있다는 것을 가정하고 있다.

그러면 리눅스는 정상적으로 부팅이 된다 (부트 디스크에 있는 커널을 사용하고 있으므로 이번 세션에서는 모든 서비스나 장치가 제대로 동작하지는 않지만). 그런다음 예전의 커널로 복구하고 LILO 부트 로더 정보를 다시 설정한 뒤에 (즉, "`mv /vmlinuz.old /vmlinuz ; /sbin/lilo`" 라고 한 뒤) 시스템을 내리고/재시작 시킨다. 그리고 나서 다른 옵션을 선택하여 커널을 다시 컴파일 한 뒤 재도전 한다.

10.7. 리눅스 2.2.x 커널로 옮아가기

리눅스 커널 2.2.0 은 1999년 1월 25일에 발표되었다. 이 버전은 수많은 새로운 기능과 성능상의 개선, 하드웨어 지원을 추가하였다. 기존의 어떤 리눅스 시스템이라도 이들 새로운 커널로 10.4절 에서 설명한 방법에 따라 똑같이 (약간 주의를 할 필요는 있지만) 업그레이드 할 수 있다.

이 섹션은 레드햇 시스템을 어떻게 새로운 커널로 업그레이드하는지 설명할 것이다. 레드햇 6.0 (과 그 이상의) 버전에서는 이미 디폴트로 새로운 커널과 지원 패키지를 담고 있으므로 이 섹션은 아직까지도 예전 버전, 예컨대 5.2, 을 사용하는 사용자에게만 유용할 것이다. 일단 대부분의 사용자가 6.0 또는 그 이상의 버전으로 옮겨갔다고 확신하게 되면 나는 앞으로 이 문서에 이 부분을 삭제할 생각이다.

> **주의**
>
> 경고! 새로운 커널을 지원하기 위하여 오래된 시스템을 업그레이드하기로 결정하였다면 그 과정에 수많은 패키지를 업그레이드할 것이기 때문에 뭔가 완전히 잘못되어 버릴 수도 있다. 항상 그렇듯이 뭔가 잘못될 경우를 대비하여 최근의 백업을 보관하여야 한다. RPM 을 이용하여 파일을 업그레이드하는 것과 커널 컴파일 경험이 없다면 레드햇 6.1로 업그레이드하는 것이 좋다.

레드햇이 제공하는 기성 커널을 이용하여 업그레이드 할 수도 있고 맞춤 커널을 컴파일하여 업그레이드 하는 방법도 있다. 나는 일단 기성 커널을 이용하여 돌아가게 해 놓은 다음 필요하다면 맞춤 커널을 만들어 올리도록 권장한다. (자세한 내용은 10.5절 을 참조하시요.)

최신의 커널을 쓰고 싶으면 우선 최신의 유틸리티와 라이브러리로 업그레이드를 해야 한다. 레드햇은 최신의 커널을 지원하기 위하여 어느 패키지를 업그레이드할 필요가 있는지 정리하여 자신들의 FTP 사이트인 (i386 플랫폼상의 5.2 사용자의 경우) ftp://ftp.redhat.com/redhat/updates/5.2/kernel-2.2/i386/ (ftp://ftp.redhat.com/redhat/updates/5.2/kernel-2.2/i386/) 에 필요한 RPM 파일을 올려두었다.

2.2.x 로 옮겨가는데 필요한 시스템 도구에 대해 상세한 내용을 담고 있는 매우 좋은 웹 페이지가 http://www-stu.calvin.edu/~clug/users/jnieho38/goto22.html 에 있다; 아래에 정보를 요약하였다 (맨 앞에 "**" 로 시작하는 항목은 레드햇 5.2 에서 업그레이드하려고 하면 꼭 필요한 항목을 표시한 것이다; 그런 표시가 없는 항목은 아마도 괜찮겠지만 확인을 해볼만 한 것들이다).

- ** **initscripts-3.78-2.4** 또는 그 이상 (현재 설치된 버전을 확인하려면 "`rpm -q initscripts`" 라고 타이핑한다)

- ** **modutils-2.1.121** 또는 그 이상 (현재 설치된 버전을 확인하려면 "`rpm -q modutils`" 라고 타이핑한다)
- ** **mount-2.9-0** 또는 그 이상 (현재 설치된 버전을 확인하려면 "`rpm -q mount`" 라고 타이핑한다)
- **gcc-2.7.2.3** 또는 그 이상 ("`rpm -q gcc`")
- **binutils-2.8.1.0.23** 또는 그 이상 ("`rpm -q binutils`")
- **libc-5.4.46** 또는 그 이상 (레드햇에서는 더 새로운 버전인 "**glibc**" 를 사용한다. 필요없음.)
- **glibc-2.0.7-6** 또는 그 이상 ("`rpm -q glibc`")
- **ld.so 1.9.9** 또는 그 이상 ("`ls -l /lib/ld.so.*`")
- **libg++-2.7.2.8** 또는 그 이상 ("`rpm -q libg++`")
- **procps-1.2.9** 또는 그 이상 ("`rpm -q procps`")
- ** **procinfo-15** 또는 그 이상 ("`rpm -q procinfo`")
- **psmisc-17** 또는 그 이상 ("`rpm -q psmisc`")
- ** **net-tools-1.50** 또는 그 이상 ("`rpm -q net-tools`")
- **loadlin-1.6** 또는 그 이상 (도스에서 Loadlin 을 이용하여 리눅스를 부팅하는 경우에만 필요함. 버전을 확인하는 방법이 알려져 있지 않음; 확실히 하기 위하여 무조건 최신 버전을 다운로드하여야 함.)
- **sh-utils-1.16** 또는 그 이상 ("`rpm -q sh-utils`")
- **autofs-3.1.1** 또는 그 이상 ("`rpm -q autofs`")
- **nfs-server2.2beta37** 또는 그 이상 ("`rpm -q nfs-server`"; NFS 파일 공유 서비스를 제공하는 경우에만 필요함.)
- **bash-1.14.7** 또는 그 이상 ("`rpm -q bash`")
- **ncpfs-2.2.0** 또는 그 이상 ("`rpm -q ncpfs`"; 노벨 파일 시스템을 마운트하는 경우에만 필요함.)
- **kernel-pcmcia-cs-3.0.6** 또는 그 이상 ("`rpm -q kernel-pcmcia-cs`"; PCMCIA 카드 지원이 필요한 랩톱의 경우에만 필요함.)
- **ppp-2.3.5** 또는 그 이상 ("`rpm -q ppp`"; 인터넷을 모뎀과 PPP 를 이용하여 접속하는 경우에만 필요함.)
- **dhcpcd-1.3.16-0** 또는 그 이상 ("`rpm -q dhcpcd`"; 케이블 모뎀처럼 인터넷을 연결하는데 DHCP 클라이언트가 필요한 경우에만 필요함).
- ** **util-linux-2.9.0** ("`rpm -q util-linux`")
- **setserial-2.1** 또는 그 이상 ("`rpm -q setserial`")
- **ipfwadmin/ipchains** (IP 방화벽 기능을 하는 경우에만 필요함; http://isunix.it.iltu.edu/resources/ldp/HOWTO/IPCHAINS-HOWTO.html 에 있는 "**IPCHAINS-HOWTO**" 가이드를 참조하시요.)

필요한 패키지를 다운로드 받고 RPM 을 이용하여 업그레이드하여야 한다 (RPM 의 사용법에 대한 자세한 내용은 10.1절 을 참조하시요).

경고

특별 주의: 새로운 "`modutils`" 패키지로 업그레이드를 하면 모듈들이 예전의 2.0.x 커널에서는 동작하지 않는다! 따라서 새로운 커널을 "`/usr/src/linux`" 에 설치하기 전까지는 이 패키지를 업그레이드해서는 안된다.

시스템 도구를 최신판으로 만든 뒤에 커널 소스를 설치할 수 있다. 이것도 레드햇의 FTP 사이트에서 찾을 수 있다; 나는 레드햇 6.1 수정판으로 제공되는 것을 ftp://ftp.redhat.com/redhat/updates/6.1/i386/ 에서 다운로드 받도록 권장한다. 그렇게 하려면 다음과 같이 타이핑한다:

```
rpm -Uvh kernel-source*.rpm kernel-headers*.rpm
```

새로운 커널 소스가 설치되었으므로 modutils 패키지를 업그레이드하더라도 안전하다. 물론 새로운 커널은 커널 모듈을 필요에 따라 올리기 위하여 더이상 "`kerneld`" 모듈을 사용하지 않는다. 따라서 modutils 를 갱신하기 전에 이 모듈을 사용하지 않도록 막아야 한다. kerneld 를 동작하지 않도록 막고 modutils 패키지를 업그레이드 하려면 "루트" 로서 다음과 같이 타이핑하여야 한다:

```
/sbin/chkconfig kerneld off
/etc/rc.d/init.d/kerneld stop
rpm -Uvh modutils*.rpm
```

이제 보통의 경우처럼 2.2 커널을 설정, 컴파일, 설치할 수 있게 되었다. (자세한 내용은 10.6절 을 참조하시요) 새로운 설정 셋팅이 아찔할 정도로 많은 것을 보고 놀라지 말기 바란다. 친숙하지 않은 옵션에 대하여 도움말을 시간내서 읽어보기 바란다!

운이 따른다면 다음 번에 시스템을 부팅하면 최신의 그리고 최강의 리눅스 커널 버전이 실행될 것이다!

이 과정에 대한 더 자세한 내용은 레드햇 웹 사이트인 http://www.redhat.com/corp/support/docs/kernel-2.2/kernel2.2-upgrade.html 에 있다.

10.8. 아파치 웹 서버 설정하기

우리 회사에서는 웹 서비스를 제공하기 위하여 아파치 패키지를 사용한다. 아파치는 HTTP 1.1 표준을 완전히 지원하고 프록시 캐싱, 패스워드 인증 웹 페이지 등 많은 기능을 제공하는 모든 기능을 갖춘 웹 서버이다. 아파치는 가장 널리 사용되는 웹 서버 중 하나이며 (넷크래프트의 최근 조사에 따르면 인터넷 상의 모든 웹 사이트 중 54% 가 아파치 또는 그의 변종을 쓰고 있다고 한다) 상용 서버만큼 또는 그 이상의 성능을 낸다.

(공사중. :-p)

아파치에 추가된 기능과 버그 수정을 활용하려면 가끔식 업그레이드를 해주는 것이 좋을 것이다. 아파치의 웹 사이트는 http://www.apache.org/ 에 있으며 최신판에 대한 정보가 들어 있다.

10.9. Squid HTTP 캐싱 프록시 데몬 설정하기

우리 회사에서는 웹 페이지 캐싱 기능을 제공하기 위하여 Squid 패키지를 사용하고 있다. Squid 는 웹 클라이언트는 물론이고 FTP, 고퍼, HTTP 요청에 대한 고성능의 캐싱을 제공한다.

게다가 페이지 캐싱을 엮어서 할 수 있도록 다른 Squid 기반의 프록시 서버와 계층구조로 연결하는 것이 가능하다.

현재 사용가능한 Squid 는 두가지 버전이 있다. 하나는 "일반" 버전으로서 램이 많이 있는 기계에서 잘 돌아가는 것으로 보인다. 두번째 버전은 "**SquidNOVM**" 버전으로서 이는 상대적으로 적은 램을 가진 (64 MB 이하의 메모리를 갖고 있는 경우에는 이 버전을 쓰는 것이 좋다) 기계에 적당하다. 기본적으로 "NOVM" 버전은 메모리를 적게 쓰는 대신 파일 디스크립터 (descriptor) 를 많이 사용한다. 이 버전을 나는 쓰고 있는데 상당히 잘 동작한다.

(공사중 :-p)

새로운 기능 추가와 버그 수정을 활용하기 위해서는 가끔 Squid 서버를 업그레이드하는 것이 좋다. 자세한 정보는 http://squid.nlanr.net/Squid/ 에 있는 Squid 의 홈페지이를 참고하기 바란다.

10.10. Sendmail 전자우편 데몬 설정하기

전자메일 서비스를 제공하기 위해서 나는 Sendmail 패키지를 사용하고 있다. Sendmail 은 가장 확실한 메일 핸들러이다; 사실 인터넷을 통하여 전송되는 전자메일의 80% 이상이 한쪽 또는 양쪽에서 이 프로그램을 사용한다고 추정된다. 이 프로그램이야말로 해야할 일을 제대로 하는 프로그램이며 이 프로그램없이 인터넷 서버를 운용한다는 것은 상상조차 할 수 없다 (Qmail 이라고 불리는 또다른 전자메일 서버도 약간 인기가 있는 모양이다 -- 하지만 나로서는 그 프로그램 사용을 시도할 이유를 찾지 못하고 있다).

새로운 기능과 버그 수정을 활용하고 더 중요하게는 보안상의 이유로 Sendmail 을 가끔 업그레이드하는 것이 바람직하다. 게다가 최신판의 Sendmail 은 강력한 스팸방지 기능이 포함되어 있어 허락받지 않은 사용자가 메일 서버를 남용하는 것을 방지할 수 있다.

이 섹션에서는 Sendmail 을 받는 메일 서버로 사용하려고 할 때 해야할 일에 대하여 약간 다루려고 한다. 그리고 주로 서버 시스템에서 어떻게 하는 지에 대한 내용이 될 것이다. 한편 만약에 받는 메일을 처리할 필요는 없고 보내는 메일 기능만 사용하려고 하는 것이라면 ((여기에 내용을 추가해야함)) 를 해야한다.

이 섹션에서는 최신 버전의 Sendmail (이 글을 쓰는 시점에서는 8.9.3) 을 이미 설치하여 사용하고 있다고 가정하고 있다.

레드햇 배포판에 들어있는 것을 보면 Sendmail 은 대부분의 서버에서 제대로 돌아갈 수 있도록 적절한 설정 정보를 포함하고 있다. 그럼에도 불구하고 "`/etc/sendmail.cf`" 파일을 편집하고 필요에 따라 몇가지 셋팅을 맞춰주어야 한다. 이는 물론 이 문서의 수준을 넘는 내용이다.

하지만, 내 생각에 도움이 될만한 것으로는 스패머들을 차단하기 위하여 설정 파일을 약간 수정하는 것이다. 여기에 포함되는 내용은:

```
O PrivacyOptions=authwarnings
바꾼 뒤:
O PrivacyOptions=authwarnings,noexpn,novrfy
O SmtpGreetingMessage=$j Sendmail $v/$Z; $b
바꾼 뒤:
```

```
O SmtpGreetingMessage=$j Sendmail $v/$Z; $b  NO UCE  C=xx L=xx
```

첫번째 변경내용은 스패머가 "EXPN" 과 "VRFY" 명령을 사용하지 못하도록 하는 것이다. 나는 이들 명령이 너무나도 자주 비도덕적인 사람들에 의해 악용된다는 것을 발견하였다. 두번째 변경내용은 Sendmail 이 연결을 받을 때 표시하는 배너의 내용을 바꾸는 것이다. 내용 중에서 "C=xx L=xx" 에 들어있는 "**xx**" 는 국가 코드와 지역 코드로 바꾸어 주어야 한다. 예를 들어, 내 경우에는 캐나다 온타리오이므로 "C=CA L=ON" 라고 쓰고 있다. (두번째 변경은 실제로 어떤 영향을 미치지는 않지만 news.admin.net-abuse.email (news:news.admin.net-abuse.email) 뉴스그룹에 있는 친구들이 이를 법적인 예방조치로서 해두는 것이 좋다고 권장하고 있다.)

만약 당신의 메일 서버가 실제로 실행되는 실제 기계의 호스트 이름과 다르다면 "/etc/sendmail.cw" 파일에 한두개의 별명을 덧붙여 주어야 한다. 예를 들어, 당신 시스템의 이름이 "**kirk.mydomain.name**" 인데 이 시스템을 mydomain.name 도메인의 우편 교환기로 사용하고 들어오는 메일의 주소가 "**user@mydomain.name**" 형식으로 된 메일이 "**kirk**" 에게 배달되게 하려면 이 별명을 다음과 같이 추가해주면 된다:

```
mydomain.name
```

마지막으로 당신의 sendmail 서비스에 연결할 수 있는 도메인 (또는 부도메인 subdomain) 을 제한하려면 "/etc/mail/access" 파일을 편집하여 도메인에 대한 정보와 제한의 종류를 입력해주면 된다. 예를 들어:

```
some.domain              REJECT
hax0r.another.domain     550 관리자에게 전화하세요 (555) 555-1234.
```

위의 예는 "**some.domain**" 사이트에서 오는 모든 전자우편 연결을 거부하고 "**hax0r.another.domain**" 기계에서 오는 연결에 대하여는 거부를 하면서 그 다음에 있는 메시지를 내보낸다.

이 파일을 변경한 다음 "access.db" 파일을 갱신해주고 난 다음 sendmail 을 다음과 같이 재시작시켜준다:

```
/usr/sbin/makemap hash /etc/mail/access.db < /etc/mail/access
/etc/rc.d/init.d/sendmail restart
```

> 작은 정보: 팁: 만약 전자메일 악용을 우려한다면 그러한 악용을 다루는 "메일 악용 예방 시스템 Mail Abuse Prevention System" (MAPS) 프로젝트에서 매우 유용한 정보를 얻을 수 있다; http://www.mail-abuse.org/ 에 있는 웹 페이지를 참조하시요
>
> 만약 Sendmail 8.9 버전 또는 그 이상을 사용하고 있다면 RBL 지원기능이 내장되어 있다. 하지만 이 기능을 디폴트로 사용하지 않도록 되어 있다. 이 기능을 사용하려면 sendmail.mc 파일에 다음과 같은 내용을 추가하면 된다:
>
> ```
> FEATURE(rbl)
> ```
>
> 그리고 나서, Sendmail 데몬을 다시 설정하고 재시작시키면 된다.
>
> 다른 메일 전송 에이전트를 위한 설정 정보 등 더 자세한 정보를 보려면 http://www.mail-abuse.org/rbl/usage.html 를 보면 된다.

가끔은 계속 통신을 해야 하는 어떤 도메인이 RBL 리스트에 들어있을 수 있다. 어쩌면 블랙 리스트에 오른 도메인에 속한 어떤 사람과 꼭 통신을 해야할 수도 있다. 이런 경우에는 Sendmail 에서 (블랙 리스트에 들어있더라도) 특정 도메인에서는 전자우편을 수신할 수 있도록 지정할 수 있다. "`/etc/mail/access`" 파일을 편집하여 적절한 도메인 정보를 넣어주면 된다. 예를 들어:

```
블랙리스트의.어떤도메인          OK
```

(앞에서 설명했듯이) access.db 파일을 다시 만드는 것을 잊으면 안된다!

RBL 에 가입하기로 결정했다면 가급적 메일 사용자들에게 알려주는 것이 현명한 처사이다. 만약 사용자들이 이에 반발한다면 다른 서비스를 준비해두어야 하기 때문이다.

Sendmail 에 대한 자세한 정보는 http://www.sendmail.org/faq/ 에 있는 FAQ 문서를 참조하시요.

11장. 리눅스를 기업차원에서 쓰기

안정성과 신뢰성에서 견실한 명성을 얻게 됨에 따라 기업이나 과학 분야에서 점점 더 결정적인 응용 분야로 활용되고 있다.

이 장에서는 기업에서 리눅스를 사용하는 것과 관련한 이슈들 예컨대 많은 부하가 걸리는 상태에서 더 높은 성능을 내도록 서버를 튜닝하고 RAID 기술을 사용하여 자료를 더 안전하게 보존하고 다른 서버로 업무를 이관하는 절차 등을 다룬다.

11.1. 성능 튜닝

(공사중. **:-p**)

11.2. RAID 를 이용한 고가용성 구현

저장공간에 대한 수요가 늘어감에 따라 추가의 드라이브를 연결할 필요가 생긴다. 하지만 역설적이게도 확률의 법칙에 따르면 저장 장치의 갯수가 늘어남에 따라 장치의 고장 가능성도 늘어나게 된다. 따라서, 하드 드라이브가 하나 밖에 없는 시스템은 네개가 달린 시스템에 비하여 25% 정도의 고장 가능성 밖에 없다. [물론, 이론적으로 그렇다는 얘기다 :-)]

다행히도 "레이드 Redundant Array of Inexpensive Disks" (**RAID**) 라고 불리는 기술을 사용함으로써 그러한 고장을 매끄럽게 그리고 더 중요하게는 다운시키지 않고 대처할 수 있다. 이 기술은 자료를 여러개의 디스크에 여유있게 분산시키는 여러가지 방법 중 하나를 사용한다. 이러한 여유(redundancy)는 한 장치가 고장나더라도 자동으로 복구할 수 있게 해준다.

이 장에서는 Mylex AcceleRAID DAC960 컨트롤러를 사용하여 RAID 디스크 어레이를 설치, 설정, 셋업하는 방법을 설명한다. 나는 컨트롤러 그 자체의 성능과 신뢰도에서도 그랬지만 Mylex 사의 기술지원 -- 그들은 매우 친절하다! -- 에서는 더더욱 깊은 인상을 받았다. (물론, 리눅스에서 사용할 수 있는 하드웨어 RAID 솔루션도 다양하게 있고 리눅스 커널 그 자체에도 RAID 기능을 구현해서 넣을 수도 있다.) RAID 의 구현 방식 중에서 가장 유용한 것은 아마도 RAID 레벨 5 인것 같다.

RAID 컨트롤러를 리눅스에서 사용하는 첫번째 단계는 하드웨어를 지원하는 드라이버가 내장되도록 맞춤 커널을 만드는 것이다. Mylex DAC960 용 드라이버는 Dandelion 디지털 리눅스 페이지인 http://www.dandelion.com/Linux/DAC960-2.0.tar.gz 에서 구할 수 있다.

리눅스에서 RAID 어레이를 사용하도록 하기 위한 마지막 단계는 "`fdisk`" 유틸리티를 사용하여 제대로 된 파티션을 만드는 것이다. 이는 IDE 나 보통의 스카시 드라이브를 사용할 때와 완전히 같은 방법으로 하면 된다. 어떻게 파티션 정보를 셋업하는지에 대한 자세한 정보는 4.3절 을 참조하기 바란다.

> 참고**:** 주의: DAC960 드라이버는 논리 드라이브 하나 당 최대 7개까지의 파티션을 지원한다. 더 많은 파티션이 필요하면 RAID 설정 유틸리티에서 여러개의 논리 드라이브를 정의하여야 한다. (시스템이 부팅할 때 <**Alt**>-<**R**> 를 누르면 셋업 유틸리티로 들어갈 수 있다).

일단 RAID 어레이를 볼 수 있게되면, 정의하고 싶은 대로 스왑 영역이나 파일 시스템을 초기화할 수 있다. 다음의 예는 두번째 드라이브의 세번재 파티션에 스왑 영역을 초기화하고 첫번째 드라이브의 첫번째 파티션에 ext2 형식의 파일 시스템을 초기화하는 과정을 보여준다:

```
/sbin/mkswap -c /dev/rd/c0d1p3
/sbin/swapon /dev/rd/c0d1p3
/sbin/mkfs.ext2 -c /dev/rd/c0d0p1
```

> 참고: 주의: 위의 내용 중에 "`-c`" 옵션이 "`mkswap`" 과 "`mkfs.ext2`" 명령에서 사용되었는데 이는 스왑/파일 시스템을 만들때 불량 블럭에 대한 검사를 하라는 것을 지정한다. 이는 그 명령을 수행하는데 상당한 시간이 더 걸리게 한다. 하지만 검사를 하는 것이 매우 바람직하다.

새로이 스왑 영역을 정의하였다면 "`/etc/fstab`" 파일에 항목을 추가하여 그 스왑 영역이 다음번 부팅때부터는 실제로 사용되도록 하여야 한다. 위의 예를 기준으로 한다면 다음과 같은 줄을 추가하여야 한다:

```
/dev/rd/c0d1p3  swap       swap  defaults   0 0
```

마지막으로 일단 파일 시스템의 초기화가 끝났으면 거기에다가 마운트 포인트를 만들고 필요에 따라 큰 파일 시스템을 어레이로 옮길 수 있다. 실제 업무에 적용하기에 앞서서 며칠 정도 시험을 해보는 것이 바람직하다.

Mylex AcceleRAID 컨트롤러에 대하여 더 자세한 정보는 http://www.mylex.com/ 에 있는 Mylex 사의 웹 사이트나 Dandelion Digital DAC960 드라이버 페이지인 http://www.dandelion.com/Linux/DAC960.html 을 참조하면 된다. RAID 에 대한 일반적인 정보는 (소프웨어 솔루션이건 하드웨어 솔루션이건 간에) 리눅스 고가용성 웹 사이트인 http://linas.org/linux/raid.html 를 참조하기 바란다.

11.3. 서버 이관과 확장성

사용가능한 하드웨어의 다양함과 증명된 성능과 신뢰성 때문에 리눅스는 점점 더 높은 요구수준을 맞추어야하는 도전에 직면해있다. 여기에 포함되는 것으로는 높은 처리 요구에 대응하기 위한 SMP (대칭형 다중 프로세싱 **Symmetric Multi Processing**) 설정, (소프트웨어 또는 하드웨어 방식의) RAID 레벨 0 부터 5 지원 등이 있다.

가끔은 현재 가동되고 있는 하드웨어에 비하여 리눅스 서버가 너무 커버렸다고 느끼거나 전혀 새로운 리눅스 버전으로 업그레이드하고 싶거나 전혀 다른 리눅스 배포판으로 바꾸고 싶을 때가 있다. 이를 해내는 방법에는 물론 두가지가 있다. 첫번째는 기존의 하드웨어 그대로 또는 업그레이드해서 가는 방법이고 (이 경우에는 서비스를 내리고 데이터를 백업한 다음 필요한 부분을 수정하고 데이터를 복구해주기만 하면 된다) 또 한가지 방법은 새로운 하드웨어로 이전해가는 것이다.

이 섹션에서는 두번째 경우에 집중할 것이다. 이 경우에는 기존의 서버에서 다양한 서비스를 새로운 서비스로 옮겨가야 한다. 서버를 이관하는데에는 여러가지 전략이 있을 수 있지만

이 섹션에서는 대략의 가이드라인을 제시함으로써 이를 따르기만 하면 사용자들에게 최소한의 충격을 주면서 이관할 수 있도록 하려고 한다.

- 필요에 맞추어 새로운 서버를 준비한다; 새로운 하드웨어 장치를 지원할 수 있도록 하고 데몬과 커널 기반의 기능들 (예를 들어 방화벽) 을 돌아가도록 리눅스를 설치하고 설정한다. 이에 관한 자세한 내용은 4장 과 10.6절 을 참조하시요.
- 기본의 서비스를 (예컨대 아파치 웹 서버, 삼바 또는 Netatalk 파일 & 인쇄 서비스 등) 셋업하고 테스트용 자료를 이용하여 며칠간 사용해봄으로써 모든 것이 제대로 동작하는지 확인한다. 이에 대한 자세한 사항은 7.4절 과 7.5절 을 참조하시요. 이전의 서버에서 "`/etc/`" 디렉토리에서 "`/etc/rc.d/`" 등에 수정한 부분이나 따로 넣은 스크립트 등을 새로운 서버에도 필요에 따라 적용시켜 주어야 한다. 그리고, "`/etc/passwd`", "`/etc/group`", 와 만약 섀도우 패스워드를 사용하고 있다면 "`/etc/shadow`" 에 들어있는 사용자 계정 정보를 옮기는 것이 특별히 중요하다!
- 예전 서버에서 서비스를 내려서 파일 시스템에 변화가 최소한으로 나타나게 한다. 당연히 자료를 새 서버에 복구하는 중에 예전 서버에 사용자들이 웹 페이지를 올리거나 전자메일을 받아가도록 하면 안된다! 대부분의 서비스를 내리기 위해서는 루트로서 다음과 같이 명령을 내리면 된다:

  ```
  killall httpd atalkd smbd nmbd squid sendmail ftpd
  ```

  위의 명령은 웹 서버, 파일 & 인쇄 서비스, 전자메일 서버, FTP 서비스를 중단시킨다. (내가 위에서 열거한 서비스 보다 더 적거나 많은 서비스를 돌리고 있을 수 있다. 프로세스 목록을 살펴보고 필요하다고 느껴지는 것이 있으며 그것도 중단시켜야 한다; 자세한 내용은 9.2절 을 참조하시요.)

  그리고 예전 서버에서 "`/etc/inetd.conf`" 파일을 편집하여 "`#`" 글자를 사용하여 파일 시스템에 변화를 일으킬 수 있는 모든 서비스 (예컨대 FTP, IMAP, POP3 서비스) 를 주석으로 막아버려야 한다. 그리고 나서 다시 루트로서 다음과 같이 타이핑한다:

  ```
  killall -HUP inetd
  ```

  위의 명령은 (인터넷 서비스를 위한 보안 래퍼인) TCP 래퍼를 다시 올려서 "`/etc/inet.conf`" 파일에서 막아둔 서비스가 더이상 연결되지 않도록 한다.

- 이제 한 시스템에서 다른쪽으로 자료를 옮길 수 있다. 새로운 서버는 필요한 소프트웨어 중에서 레드햇 배포판에 포함되어 있지 않은 소프트웨어까지 설치되어 필요한 기능을 모두 할 수 있도록 이미 준비되어 있어야 한다. 이제, "`/home`", "`/var/spool`", 그리고 만약 있다면 선택적인 파일 시스템인 "`/archive`" 에 저장된 자료를 백업하여 새로운 시스템에 복구하여야 한다. 다음의 예는 "`tar`" 유틸리티를 사용하여 압축된 백업 파일을 만드는 과정을 보여준다:

  ```
  cd /
  tar zcvpf /tmp/backup_data.tar.gz --exclude=var/spool/squid \
            home archive var/spool
  ```

  위의 명령은 "`/archive`", "`/home`", 과 "`/var/spool`" 파일 시스템 (또는 시스템의 셋업 상태에 따라서는 서브디렉토리) 의 백업본을 "`/tmp/backup_data.tar.gz`" 라는 이름의 파일에

압축된 타르 형식으로 기록한다. 백업 파일을 만들 수 있는 충분한 공간이 있는지 확인해야 한다!

> 작은 정보**:** 팁: 필요한 공간을 미리 알기 위해서는 "du" 유틸리티를 사용하면 도움이 된다. 예를 들어, "/archive/" 와 "/home/" 디렉토리 트리가 얼마나 되는지 알아보려면 다음과 같이 타이핑 하면 된다:
>
> **du -h -s /archive /home**
>
> 위의 명령은 자료의 실제 크기를 보여주는 반면에 백업 파일을 만들때에는 크기를 줄이기 위하여 타르 명령에서 "z" 옵션을 (앞의 예에서 처럼) 사용하기 때문에 실제 필요한 공간은 훨씬 적을 수 있다는 점을 명심하여야 한다. 따라서 "du" 명령의 결과값은 최악의 경우에 해당하는 추정치라고 생각하면 된다.

- 이제 타르 파일로부터 백업한 자료를 새로운 서버에 복구하면 된다. NFS 를 (NFS 설정에 대한 상세한 정보는 7.6절 을 참조하시요) 이용하여 직접 복구할 수도 있고 일단 FTP 를 이용하여 타르 파일을 옮긴 다음 새 서버에서 타르를 풀 수도 있다. 다음은 앞에서의 예시에서 만든 파일을 복구하는 과정을 보여준다:

  cd /
  tar zxvpf /tmp/backup_data.tar.gz

- 그 다음은, 필요하다면, IP 주소를 변경하여 새로운 서버가 예전의 서버 주소에서 보이도록 해준다.
- 마지막으로 서버를 내렸다가 다시 시작시켜서 아무런 에러 메시지가 나오지 않는지 확인하여야 한다. 서버를 내리고 재시작하는 과정에 대한 자세한 설명은 6.7절 을 참조하시요.

일단 끝나고 나면 모든 것이 기대한대로 동작하는지 확인하여야 한다! 만약 그렇지 않으면 항상 예전 서버에서 중단시켜두었던 서비스를 다시 가동시켜서 새로운 서버에서 나타난 문제가 해결될 때까지 사용자들이 계속 서비스를 이용할 수 있도록 하여야 한다. (하지만 이렇게 하기로 하였다면 앞에서 얘기한 모든 단계를 몽땅 다시 하여야 한다는 점을 명심하라)

12장. 서버를 안전하게 지키기 위한 전략

리눅스는 다른 회사에서 나온 운영체제만큼 -- 또는 더 -- 안전하다고 생각된다. 리눅스가 점점 더 인기가 있어짐에 따라 크래커들이 공격 노력을 집중할 만한 매우 매력적인 표적이 되어 가고 있다. 가끔은 헛점이 발견되기도 하지만 리눅스가 갖고 있는 개방성때문에 그러한 헛점은 빨리 패치가 되고 임시로 어떻게 대처할 것인지 또는 갱신된 소프트웨어는 어디서 구할 수 있는지 등을 담은 보안관련 안내문이 널리 배포된다.

나는 보안분야에서 전문가인척 하려는 것은 아니다. 하지만 나는 이러한 문제에 관심을 갖고 있으며 어떤 시스템을 최대한 안전하게 만드는 전투에서 중요한 부분이라고 믿고 있다. 보안관련 최신 뉴스에 관심을 갖고 부지런히 대처한다고 해서 시스템의 보안이 무너지지 않는다고 보장할 수는 없지만 침입의 가능성은 상당히 낮추어 줄 수 있다.

비록 크래커가 시스템에 침입하기 위하여 악용할 수 있는 외부 서비스에서도 보안상 취약성이 발견되기도 하지만 (예를 들어, IMAP 데몬 취약성) 내 생각으로는 어떤 크래커가 시스템을 내부로부터 뚫고 들어올 가능성이 훨씬 높다고 본다. 외부 세계와 통신하는데 사용되는 서비스는 한 줌 밖에 안되는데 비하여 쉘에서 사용할 수 있는 명령과 유틸리티는 수천개가 넘는다. 이들 중 어떤 것은 버그가 있어서 보안을 무너뜨릴 가능성이 있다. (물론, 내가 관리하는 서버 중 하나가 최근에 외부 서비스를 통하여 탈취당한 적이 있긴하다)

이런 이유로 나는 사용자들에게 절대적으로 필요한 경우가 아니라면 쉘 계정을 주는 것을 피하도록 권고한다. 비록 사용자들이 전적으로 믿을만 하고 쉘을 제공하는데 전혀 마음에 꺼림칙한 구석이 없다고 하더라도 사용자중에 간단한 패스워드를 갖고 있는 사람이 하나만 있어도 모든 것은 허사가 된다. 외부의 크래커는 바로 이 간단한 패스워드를 이용하여 들어올 수 있고 이를 발판으로 안에서 작업을 하면서 그 외의 약점을 노리게 된다.

리눅스 시스템의 보안성을 엄청나게 끌어올리는 방법이 다행스럽게도 있다. 보안관련 사항에 대한 자세한 설명은 이 문서의 범위를 넘는 것이므로 보안수준을 높이기 위하여 하여야 할 몇가지 사항에 대한 체크 리스트를 제시하였다:

- 시스템 도구, 응용 프로그램, 커널의 업그레이드: 현재까지 시스템 침입의 가장 공통된 이유는 서버를 최신 상태로 유지할만큼 부지런하지 않아서이다. 시스템 커널, 도구와 유틸리티에 대해서 정기적으로 업그레이드를 수행하면 취약성이 알려진 오래된 항목이 시스템에 많지 않게 된다. 시스템을 최신의 상태로 유지하는 방법에 대하여는 4.9절 과 10.3절 을 참조하시요.
- 섀도우 패스워드: 반드시 섀도우 패스워드를 사용하여야만 한다; 이 패스워드 형식으로 바꾸는 것은 쉽다! 자세한 내용은 6.6절 을 참조하시요.
- 안전한 패스워드 관리: 패스워드는, 특히 쉘 액세스를 제공하는 사용사에 대한 패스워드는, 강력하고 자주 바꾸어야 한다. 그리고 만약 여러 시스템을 사용하고 있다면 모든 시스템에서 같은 패스워드를 사용하고 싶은 욕망을 뿌리쳐야 한다 (그렇지 않으면, 어떤 크래커가 알아낸 패스워드를 이용하여 침입해온 경우 모든 시스템에 침입할 수 있게 된다).
- 보안 쉘(ssh)을 사용하라: “ssh” 를 “telnet” 대신에 사용하라. 텔넷은 두가지 이유에서 안전하지 않다: 첫번째는 세션 자체가 암호화되지 않는다. 따라서 사용자이름과 패스워드를 포함하여 모든 내용이 그냥 텍스트로 전송된다. 두번째 이유는 열려있는 텔넷 포트는 크래커가 연결을 시작하는 첫번째 장소이기 때문이다.

ssh 는 암호화되고 압축된 연결을 제공하며 텔넷 연결보다는 훨씬 많은 보안을 제공한다. 리눅스에서는 ssh 서버와 (이는 안전하게 접속해 들어오는 것을 가능하게한다) 클라이언트 (다른 서버로 안전하게 접속하도록 해주는 것) 를 모두 제공한다. 바이너리 RPM 은 ftp://ftp.replay.com/pub/replay/redhat/i386/ 에서 구할 수 있다. 다음과 같은 파일이 필요하다 (이 글을 읽을 때 쯤이면 더 새로운 버전이 나와있을 것이다):

- ssh-1.2.27-5i.i386.rpm (ftp://ftp.replay.com/pub/replay/redhat/i386/ssh-1.2.27-5i.i386.rpm) 기본 패키지.
- ssh-clients-1.2.27-5i.i386.rpm (ftp://ftp.replay.com/pub/replay/redhat/i386/ssh-clients-1.2.27-5i.i386.rpm) 클라이언트.
- ssh-extras-1.2.27-5i.i386.rpm (ftp://ftp.replay.com/pub/replay/redhat/i386/ssh-extras-1.2.27-5i.i386.rpm) 펄로 만든 몇가지 유용한 스크립트.
- ssh-server-1.2.27-5i.i386.rpm (ftp://ftp.replay.com/pub/replay/redhat/i386/ssh-server-1.2.27-5i.i386.rpm) 서버.

> 참고**:** 주의: SSH RPM 파일은 국제용 버전이다. 만약 당신이 미국이나 캐나다에 거주하고 있다면 미국용 패키지(여기서는 더욱 강력한 암호화 알고리즘을 사용하고 있다)를 다운로드할 수 있다; 이들 패키지에는 "i" 대신에 "us" 가 버전 번호 뒤에 붙어있다. 미국의 법에 의하여 미국이나 캐나다 바깥으로 강력한 암호화 제품을 수출하는 것은 불법으로 되어 있다. 언젠가는 미국 법무부에 있는 얼간이들이 마침에 빛을 보게 되어 이러한 멍청한 제한을 제거해주길 바란다 (레드햇은 바로 이런 이유 때문에 자기들의 배포판에 SSH 를 포함시키지 않고 있으며 이로 인해 우리들 모두 가 고통 받고 있다).

만약 윈도 사용자들이 더이상 당신 시스템에 접속을 할 수 없어서 흥분하고 있더라도 다음과 같은 윈도용 공짜 ssh 클라이언트가 있다는 사실을 알게 되면 행복해할 것이다:

"TeraTerm Pro" 클라이언트 소프트웨어

http://hp.vector.co.jp/authors/VA002416/teraterm.html

"TTSSH" 클라이언트 소프트웨어

http://www.zip.com.au/~roca/download.html

"Cryptlib" 클라이언트 소프트웨어

http://www.doc.ic.ac.uk/~ci2/ssh

"Putty" 클라이언트 소프트웨어

http://www.chiark.greenend.org.uk/~sgtatham/putty.html

> 참고**:** 주의: 만약 ssh 을 쓰기로 결정했다면 모든 서버에 설치하고 사용해야한다. 만약 다섯 대는 안전한 서버를 쓰고 한 대는 그렇지 않다면 이는 시간 낭비다. 특히 만약 당신이 여러대의 서버에서 같은 패스워드를 쓸 정도의 바보라면.

- 외부 서비스에 대한 액세스를 금지:그 다음에는 "/etc/hosts.allow" 파일과 "/etc/hosts.deny" 파일을 편집하여 외부 호스트에 대한 서비스 액세스를 제한한다. 다음은 telnet 과 ftp 액세스를 제한하는 예이다. 먼저 "/etc/hosts.allow" 파일은:

  ```
  # hosts.allow
  in.telnetd: 123.12.41., 126.27.18., .mydomain.name, .another.name
  in.ftpd: 123.12.41., 126.27.18., .mydomain.name, .another.name
  ```

  위의 예는 IP C 클래스 123.12.41.* 과 126.27.18.* 에 속하는 모든 호스트와 mydomain.name 과 another.name 도메인에 속하는 모든 호스트에서는 telnet 과 ftp 연결을 할 수 있도록 허용해주고 있다.

  다음은 "/etc/hosts.deny" 파일이다:

  ```
  # hosts.deny
  in.telnetd: ALL
  in.ftpd: ALL
  ```

- 필요없는 서비스는 꺼두거나 제거한다: "/etc/inetd.conf" 파일을 편집하여 필요하지 않은 서비스는 (예를 들어, 만약 앞에서 권고한대로 ssh 를 사용하기로 하였다면 "telnet" 서비스는 꺼두고 싶을 것이다) 사용하지 않도록 막아준다 (즉, "#" 문자를 사용하여 주석으로 만들어 버린다). 이렇게 한 다음에는 루트로서 "/etc/rc.d/init.d/inet restart" 라고 타이핑하여 변경된 내용을 가지고 inetd 데몬를 다시 시작시킨다.
- 보안 탐지 시스템을 설치한다: 보안 프로그램을 설치할 것을 고려한다. 예를 들어 "Tripwire" (http://www.tripwiresecurity.com/ 참조) 는 침입을 탐지할 수 있으며 "Abacus Sentry" (http://www.psionic.com/abacus/ 참조) 는 침입을 막는데 도움이 된다.
- 주의 의무: 시스템에 대하여 항상 감시하고 무작위적으로 보안 감사를 하는 것은 (패스워드 파일에 이상한 항목이 있지는 않은지 확인하고, 프로세스 리스트를 검사하고, 로그 파일에 이상한 흔적이 있지 않은지 검사하는 등 간단한 절차로도 가능하다) 안전한 시스템을 향한 머나먼 여정의 출발점이다. 그리고 어떤 침입 시도라도 발견되면 관계 기관에 보고하여야 한다 -- 이렇게 하는 것이 성가실 수도 있다. 특히, 어떤 짧은 기간에 여러 번의 공격이 있었다면 특히나 그럴 것이다. 하지만 그러한 보고는 처벌에 대한 두려움때문에 크래커가 포기를 하게 할 수도 있고 다른 시스템은 (이미 그들도 탈취당해 있을 수 있다) 안전한지 확인할 수 있는 계기를 제공하기 때문이다.
- 만약 시스템 도구와 응용 프로그램을 "RPM" 유틸리티로 설치하거나 업그레이드 한다면 설치된 패키지의 무결성을 다음과 같은 명령으로 감사해 볼 수 있다:

  rpm --verify -a > /tmp/rpm-audit.txt

  위의 명령은 당신 시스템의 RPM 데이터베이스를 이용하여 관련된 모든 파일을 검사한다. 만약 변경된 파일이 있으면 '5' 표시하여 나타내준다. 다음은 그러한 감사의 결과 예이다.:

  ```
  S.5....T   /bin/ls
  S.5....T   /usr/bin/du
  ......G.   /dev/tty5
  .....U..   /dev/vcs5
  .....U..   /dev/vcsa5
  S.5....T c /etc/lynx.cfg
  S.5....T c /etc/sendmail.cf
  ```

위의 예시에서 목록에는 총 일곱개의 파일이 있고 그 중에서 네개가 수정되었다는 것을 알 수 있다. 시스템을 필요에 따라 커스터마이징하였다면 명백히 수정된 파일이 몇개, 사실은 상당히 많이, 있을 것이다. `/etc/lynx.cfg` 과 `/etc/sendmail.cf` 를 눈으로 살펴보거나 백업본과 비교해 봄으로서 정상적인 설정을 위한 변경이었는지를 알 수 있게 된다.

하지만 위의 예에서 수정된 파일 중에서 두개는 실행 가능한 이진 파일이다? 이들 두 바이너리, "ls" 명령과 "du" 명령, 는 아마 실제로는 시스템 크래커가 사악한 용도로 심어둔 트로이의 목마일 것이다. ("`diff`" 명령을 이용하여 수정된 바이너리와 백업본 또는 RPM 에 들어있는 것을 비교해보면 크기 나 그 외 어떤 부분에서 심각한 차이가 있음을 알 수 있게 될 것이다; 이는 트로이 목마임을 드러내는 추가의 증거가 된다.)

("`RPM`" 에 대한 자세한 정보는 10.1 절 을 참조하시요.)

보안 관련 이슈에 대한 더 자세한 정보는 http://redhat-security.ens.utulsa.edu/ 에 있는 훌륭한 자원인 "Securing RedHat 5.x" 문서를 참조하기 바란다. 리눅스에서의 암호화나 관련 소프트웨어에 대한 훌륭한 정보는 http://replay.com/redhat/ (http://replay.com/redhat) 에 있다.

13장. 사람 살려! 천국에 문제가 생겼어요!

리눅스는 세계적으로 성능과 안정성이라는 측면에서 명성을 얻고 있다. 하지만, 어떤 시스템도 완벽할 수는 없다. 살다보면 문제점에 봉착하는 경우가 생긴다. 다행히도 중단없이 운영되는 기간은 월단위로 측정할 정도이니까 (NT의 경우에는 단위자체가 일 단위이나 주 단위인데 반하여) 그러한 장애는 드물고 띄엄띄엄 나타난다고 할 수 있다.

13.1. 리눅스를 새로운, 지원되지 않는 하드웨어에 설치하기

(공사중 **:-p**)

13.2. 정전이나 시스템의 파괴에 따른 파일 시스템의 손상

비록 리눅스가 안정적인 운영체제이긴 하지만, (아마도 커널의 버그나 정전으로 인하여) 예상치않게 시스템이 파괴된다면, 당신의 파일 시스템은 마운트가 해제되지 않고 리눅스 시스템이 재시작될 때 자동으로 에러를 검사하게 될 것이다.

대부분의 경우에 파일 시스템의 문제점은 파일 버퍼의 내용이 디스크에 기록되지 않음으로써 나타나는 -- 예를 들어 이미 지워진 inode 가 여전히 사용중인 것 처럼 보이는 식의 -- 사소한 것들이다. 대부분의 경우에 파일 시스템 검사는 그러한 문제점을 자동으로 탐지하여 수리할 수 있을 것이며 수리가 끝나면 리눅스의 부팅 과정이 정상적으로 진행되게 된다.

만약 파일 시스템의 문제가 그 이상 심각한 것이라면 (이런 문제는 결함이 있는 하드웨어 예컨대 불량 하드 드라이브나 메모리 칩 때문에 나타나는 경향이 있다; 만약 파일 시스템 문제가 자주 나타난다면 이러한 사실을 명심할 필요가 있을 것이다) 파일 시스템 검사가 자동으로 문제를 수리하지 못할 수 있다. 이런 경우는, 항상 그렇지는 않지만, 대개 root 파일 시스템 자체가 깨진 경우이다. 레드햇의 부팅과정에 에러 메시지가 나오면서 쉘로 떨어지게 되며 관리자가 수작업으로 시스템을 수리해보아야 한다.

복구 쉘은 모든 파일 시스템의 마운트를 해제하고 root 파일 시스템을 “읽기 전용” 으로 마운트해주기 때문에 관리자는 적절한 유틸리티를 사용하여 완전한 파일 시스템 검사를 실행할 수 있다. 아마, 찾아낸 모든 문제를 e2fsck 를 돌려서 해결할 수 있게 되기를 기대하여야 할 것이다.

만약에 (기대한대로) 어떤 파일 시스템의 문제를 해결하였다면 쉘을 빠져나가서 리눅스는 재부팅 시킴으로써 다시 시작해 보아야 한다.

당연히, 복구할 수 없는 파일 시스템의 문제를 대비하기 위하여 다음에 있는 것 중 한가지 이상을 준비해 두어야 한다:

- 부트/root 응급복구 디스크 셋트, 그리고/또는
- LILO 응급복구 디스크, 그리고
- 중요한 파일에 대한 최근의 백업본 -- 만약의 경우를 대비해서!

13.3. 어디에서 도움을 받을까 ?

리눅스가 인터넷 커뮤니티의 회원들에 의하여 만들어졌기 때문에 도움을 받을 수 있는 최상의 장소는 아마도 다음과 같은 뉴스 그룹에 글을 올리는 것이라 할 수 있다:

다른 그룹에서 다루지 않는 잡다한 글들

comp.os.linux.misc (news:comp.os.linux.misc)

리눅스에서 네트워킹과 관련된 이슈들

comp.os.linux.networking (news:comp.os.linux.networking)

리눅스에서의 보안과 관련된 이슈들

comp.os.linux.security (news:comp.os.linux.security)

리눅스의 설치 & 시스템 관리

comp.os.linux.setup (news:comp.os.linux.setup)

누구든지 의견을 낼 자격이 있다 :-p

alt.linux.sux (news:alt.linux.sux)

리눅스 고유의 주제가 아닌 경우에는 필요에 따라 comp.* 아래로 많은 그룹이 있다. 그 중에 일부만 뽑아보면 다음과 같다:

시스코 라우터/액세스 서버 제품

comp.dcom.sys.cisco (news:comp.dcom.sys.cisco)

웹 서버와 관련된 잡다한 질문들

comp.infosystems.www.servers.misc (news:comp.infosystems.www.servers.misc)

(리눅스에 국한되지 않은)일반 유닉스에 관한 질문들

comp.os.unix (news:comp.os.unix)

SMB 프로토콜 (WfW/95/NT 스타일의 파일, 인쇄 서비스)

comp.protocols.smb (news:comp.protocols.smb)

웹에도 유용한 자료들이 많이 있다. “Linux” 라는 단어를 이용해서 검색을 해보던지 아니면 다음과 같은 사이트를 가볼 수도 있다:

리눅스 리소스(Linux Resources)

http://www.linuxresources.com/

리눅스 문서화 프로젝트

http://metalab.unc.edu/LDP/

RPM 창고

http://rufus.w3.org/linux/RPM/

리눅스 소프트웨어 지도

http://www.boutell.com/lsm

리눅스 응용 프로그램 & 유틸리티 가이드

http://www.xnet.com/~blatura/linapps.shtml

LinuxHardware.net: 하드웨어 드라이버 지원

http://www.linuxhardware.net/

리눅스 사용자 지원 팀

http://www.ch4549.org/lust

리눅스 v2 정보 본부(The Linux v2 Information Headquarters)

http://www.linuxhq.com/

삼바 홈 페이지 (WfW/95/NT 스타일의 파일/인쇄 서비스)

http://samba.anu.edu.au/samba/

아파치 웹 서버

http://www.apache.org/

Squid HTTP 프락시 캐싱 데몬

http://squid.nlanr.net/Squid/

물음에 대하여 답을 해줄 수 있는 유용한 메일링 리스트도 엄청나게 많이 있다. 이러한 메일링 리스트는 간단히 웹에서 검색해보면 쉽게 찾을 수 있다. (예를 들어, **"linux raid mailing list"** 라고 검색해보면 리눅스에서 RAID 사용에 관련된 메일링 리스트들을 찾을 수 있다.) 내가 추천할 만한 몇몇을 아래에 열거하였다; 이들 리스트에 가입하려면 나와있는 가입주소로 메시지 본문에 "**subscribe**" 라는 내용을 담아서 전자메일을 보내면 된다:

레드햇 메일링 리스트

사용가능한 레드햇 리스트에 대한 설명: http://www.redhat.com/ (http://archive.redhat.com/)

그놈 메일링 리스트

사용가능한 그놈 리스트에 대한 설명: http://www.gnome.org/mailing-lists/index.shtml

KDE 메일링 리스트

사용가능한 KDE 리스트에 대한 설명: http://www.kde.org/contact.html

리눅스 스카시 메일링 리스트

가입 주소: linux-scsi-request@vger.rutgers.edu (mailto:linux-scsi-request@vger.rutgers.edu)

리눅스 RAID 메일링 리스트

가입 주소: linux-raid-request@vger.rutgers.edu (mailto:linux-raid-request@vger.rutgers.edu)

마지막으로, 다음의 두 사이트를 살펴보는 것이 재미있을 것이다. 이 두 사이트는 모두 내가 "매일 꼭 읽는" 사이트이다. 슬래시도트 (SlashDot) 는 최신 기술 뉴스를 다루고 있는데 확실히 리눅스 편향을 가지고 있다. 한편, 프레시미트 (FreshMeat) 는 오픈 소스 응용 프로그램 발표의 최신 리스트를 제공한다.

슬래시도트 (SlashDot: News For Nerds)

http://slashdot.org/

프레시미트 (FreshMeat: Open Source Applications Announcements)

http://freshmeat.net/

13.4. 추가 문서에 대한 포인터

리눅스와 그 응용 프로그램에 대하여는 믿을 수 없을 정도로 많은 문서가 있다. 이들 중 대부분은 웹이나 서점에서 구할 수 있다. 하지만, 유용한 문서중 상당수가 이미 당신에게 있다는 것을 알아야 한다. 이들은 레드햇 리눅스를 설치하는 과정에서 설치된 것이다.

man 페이지는 어떤 명령이 어떻게 동작하는지 알기 위해서 꼭 보아야 하는 것이다. 예를 들어, "tar" 유틸리티를 어떻게 사용하는지 알고 싶다면, "`man tar`" 라고 타이핑하여 명령 옵션을 포함한 매우 자세한 정보를 볼 수 있다.

좀 더 일반적인 정보는 "`/usr/doc/`" 디렉토리에 들어 있다. 여기에는 서브디렉토리별로 유틸리티와 명령에 대한 문서, FAQ (Frequently Asked Questions) 문서, 다양한 주제에 대하여 좋은 지침이 되는 -- 예를 들어, 어떻게 네트웍을 셋업하나, 어떻게 일본어를 지원할 수 있나와 같은 -- HOWTO 문서 등이 들어있다.

그리고 유틸리티, 라이브러리, emacs와 같은 응용 프로그램에 대한 입문서가 들어있는 "`/usr/info/`" 디렉토리도 살펴볼 필요가 있다.

마지막으로, http://www.pobox.com/~aturner/RedHat-FAQ/ 에 가면 리눅스의 레드햇 배포판에 관련한 수많은 정보를 담은 레드햇의 FAQ (Frequently Asked Questions) 문서를 볼 수 있다.