The Linux Kernel HOWTO in Korean

Brian Ward, bri@cs.uchicago.edu v1.0, 5 June 1999

Translated by 맹지찬, max0125@nownuri.nowcom.co.kr, jcmaeng@nuri.net Updated by 강상우, swk@usc.edu

이 문서는 커널 구성과 컴파일, 그리고 업그레이드에 관한 자세한 안내서이다

목 차

제 1 절 소 개

꼭 이 문서를 읽어야만 하는가? 자, 만일 여러분이 아래 내용중에 어느 하나라도 해당된다면 읽어라:

- "이런!! 이 wizzo-46.5.6 패키지는 커널 2.8.193이 필요한데 나는 아직도 1.0.9 잖아!"
- 여러분이 필요한 장치 드라이버가 새로 가져올 커널에 있다
- 여러분이 어떻게 커널을 컴파일하는지 모른다
- "이 README 안에 있는 내용이 다야?"
- 여러분이 시도했지만 작동하지 않는다
- 당신에게 자기 커널을 설치 해 달라고 부탁하는 분께 알려주게

1.1 가장 먼저 읽을것!

이 문서안에 있는 몇몇 예시들은 여러분이 GNU `tar`와 `find`, 그리고 `xargs`를 가지고 있다고 가정한다. 이것들은 표준적으로 보급되므로 문제가 되지 않는다. 또한 여러분이 자신의 시스템의 파일 시스템 구조를 알고 있다고 가정한다. 알지 못한다면, 평상시 시스템 작동시의 `mount` 명령 결과 사본(만약 읽을 수 있다면, `/etc/fstab`의 목록이라도)을 가지고 있는 것이 매우 중요하다. 이 정보는 중요하다. 이것은 여러분이 여러분의 디스크를 재파티션하거나, 새것을 추가하거나, 여러분의 시스템을 다시 설치하거나, 아니면 그와 비슷한 일을 하지 않는 한 바뀌지 않는다.

이 글이 쓰여지는 동안의 "안정" 커널 버전은 2.2.9이었다. 이 말은 글의 내용과 예시들이 이 버전에 적용된 것이라는 뜻이다. 나는 가능한한 이 글을 버전과 는 독립된 문서로 만들려고 했지만, 커널은 지속적으로 개발중에 있고, 만일 여러분이 새로운 버전을 가지게 된다면, 약간의 차이가 생기는 것을 어쩔 수가 없다. 다시 말하자면, 이것은 큰 문제를 일으키지는 않는다. 그러나, 약간의 혼란을 일으킬지도 모른다.

리눅스 커널 소스에는 두가지 버전, 즉 "안정" 과 "개발" 버전이 있다. 안정버전은 1.0.x 버전과 함께 시작됐고 현재는 짝수 버전으로 매겨지고 있다. 1.0.x 는 안정 버전이었고, 1.2.x도 안정버전이다. 2.0.x이나 2.2.x처럼 말

이다. 이 커널들은 그 버전대에서는 가장 안정하고, 버그가 없는 것으로 여겨진다. 개발 버전(1.1.x, 1.3.x, 2.3x, 등등)은 시험용 커널들로써, 사람들이 새롭고, 가능한 것들을 시험해 볼 수 있는 매우 버그가 많은 커널이다. 주의하기 바란다.

1.2 이 글의 표현법에 대해 한마디

`이처럼 쓰여진 것들은` 화면이나 화일명, 또는 직접 입력할수 있는 것이나 명령어의 옵션이다 (만약 이 문서를 일반 텍스트 화일처럼 본다면 아무런 차이가 없을것이다). 명령어나 입력은 종종 인용되었는데 (' '로), 이것은 전형적인 다음과 같은 구두법 문제를 일으킨다: 만약 인용문이 줄 끝에 있으면, 사람들은 '.'(마침표)를 명령어와 같이 적어 넣는다 - 왜냐하면 미국식 인용양식은 마침표를 따옴표 안에 넣기 때문이다. 상식적으로 (불행히도, 이것은 미국식의 인용법에 익숙한 사람의 상식을 의미한다) 구두를 먼저 제거해야 하지만, 많은사람들이 이를 단순히 기억하지 못하므로 나는 이런 일이 있을때마다 마침표를 따옴표 밖에다 적을 것이다. 즉, "`make config`"을 치라고 할때 나는 '`make config`'이라고 쓸 거지, '`make config.`' 이라고 쓰지 않을 것이다.

제 2 절 중요한 질문과 대답들

2.1 커널이 하는 일은 무엇입니까

유닉스 커널은 여러분의 프로그램들과 하드웨어 사이에서 조정자 역할을 한다. 첫째로, 실행되는 모든 프로그램들(프로세스들)을 위해 메모리 관리(배열)를 하 고, 프로세서 사이클을 공정히(여러분이 원한다면 그렇지 않게 할수도 있다) 나 누어 가지도록 해준다. 또한, 커널은 프로그램들이 여러분의 하드웨어와 대화하 기 위한 아주 훌륭한 이식성 좋은 인터페이스를 제공한다.

커널의 임무에 대한 더 많은 것들이 있지만, 이 기본적인 활동들은 알아두어야 할 가장 중요한 것이다.

2.2 왜 나는 커널을 업그레이드 하려고 합니까?

새로운 커널은 일반적으로 더 많은 종류의 하드웨어를 지원하고(이 말은 그들 더 많은 장치 드라이버들을 가지고 있다는 말이다.), 더 나은 프로세스 관리를 할 수 있으며, 구버전에 비해 더 빠르게 실행할 수 있다. 또한, 구버전보다 더 안정하고 그들이 가지고 있던 버그들이 수정된 것이다. 대부분의 사람들은 장 치 드라이버들과 버그 수정들 때문에 커널을 업그레이드한다.

2.3 새로운 커널들이 지원하는 하드웨어의 종류는 어떤게 있습니까?

Hardware-HOWTO를 보아라. 아니면, 리눅스 소스나 '`make config`'하는 중에 찾 을 수 있는 '`config.in`'이라는 파일을 볼 수 있다. 이것은 리눅스가 지원하는 모든 것이 아니라, 표준 커널 배포본에서 지원하는 모든 하드웨어를 보여준다. 많은 일반적인 장치 드라이버들(예를 들면 PCMCIA나 몇몇 테잎 드라이버 같은 것들)은 적재 가능한 모듈들로 따로 관리되고 제공된다.

2.4 어떤 버전의 gcc와 libc가 필요합니까?

Linus는 리눅스 소스에 포함된 `README` 파일에서 필요한 gcc의 버전을 이야기 하고 있다. 만일 여러분이 이 버전을 가지고 있지 않다면, 요구되는 gcc 버전 의 문서는 여러분에게 여러분이 libc를 업그레이드 해야만하는지를 알려줄 것 이다. 이것은 어려운 작업이 아니지만, 이후의 내용에서는 매우 중요하다.

2.5 적재 가능한 모듈이 무엇입니까?

이것은 커널에 직접적으로 연결되지 않은(포함되지 않은) 커널 코드의 부분들 이다. 그들을 나누어서 컴파일하고, 거의 아무때나 실행중인 커널에 집어넣거 나 제거할 수 있다. 모듈의 유연성 때문에, 이것은 특정 커널 부분의 코드화 하는 좋은 방법이다. PCMCIA나 QIC-80/40 테잎 드라이버와 같은 많은 인기있 는 장치 드라이버들은 적재 가능한 모듈들이다.

2.6 얼마나 많은 디스크 용량이 필요합니까?

그것은 여러분의 실제 시스템 구성에따라 달라진다. 우선은, 압축된 리눅스 소스는 버전 2.2.9의 경우에 거의 14메가바이트 정도 된다. 대부분의 장소에 는 풀어논 상태로 가지고 있다. 압축을 풀고, 일반적인 구성으로 커널을 컴파일 할 경우 또다른 67MB 정도가 쓰인다.

2.7 얼마나 오래 걸립니까?

새 기종에서는 컴파일 시간은 오래된 기종보다 훨신 시간이 적게 들것이다; AMD K6-2/300에 빠른 하드디스크를 가진 것은 2.2.x 커넬을 4분 정도에 만들수 있다. 옛날 펜티움, 486, 386등은 시간이 걸릴것이다 – 몇시간부터 몇 일까지도....

만약 이것이 문제이면, 근처에 빠른 기종이 있으면 빠른 기종에서 커넬을 만든 다음 (물론 바른 옵션을 지정하고 모든 유틸리티 프로그램이 최신것으로 갱신된 상태에서) 거기서 만들어진 kernel image를 느린 기종으로 옮기면 된다.

제 3 절 실제로 커널을 설정하는 방법

3.1 소스 구하기

여러분은 소스를 anonymous ftp인 `ftp.kernel.org` 의 디렉토리인 `/pub/linux/kernel/vX.Y`에서 구할수 있다 (여기서 끝에 `X.Y`는 버젼이다 - 예: 2.2). 아까 말한것처럼 짝수로 끝나는 것은 안정버젼이고 (2.0, 2.2, ...), 홀수로 끝나는 것은 안정적이지 못할수 있는 개발버젼이다 (1.3, 2.3, ...). 커넬은 보통 `linux-x.y.z.tar.gz` (여기서 `x.y.z`는 버젼)이다. 보통 사이트들은 `.bz2`로 끝나는 것들도 가지고 있는데 이는 bzip2로 압축된 것이다 (이들은 크기가 조금 작아서 조금 빨리 받을 수 있다)

가장 좋은 곳은 `ftp.xx.kernel.org` - 여기서 xx는 당신이 있는 국가 약칭이다. 예를 들면, `ftp.at.kernel.org`는 오스트리아, `ftp.us.kernel.org`는 미국, `ftp.kr.kernel.org`는 한국이다.

3.2 소스 압축 풀기

루트로 로그인 하거나 `su`를 사용하여 루트가 된 후 `/usr/src` 로 `cd`하라. 만약 여러분이 처음 리눅스를 설치했을 때 커널 소스를 설치했다면, 이전의 전체 소스를 포함하는 `linux`라는 디렉토리가 있을 것이다. 여러분이 디스크 공간이 충분히 있고 안전하게 사용하기를 원한다면, 그 디렉토리는 가지고 있는 것이 좋다. 현재 여러분의 시스템에서 실행되고 있는 버전을 알기 위해서는 그 디렉토리의 이름을 알맞게 바꾸는 것이 좋다. `uname -r`명령은 현재 커널 버전을 표시해 준다. 그러므로 `uname -r`의 결과가 `1.0.9`라면, `linux`디렉토리를 `linux-1.0.9`로 (`mv`를 사용해서) 이름을 바꿔라. 만약 별로 개의치 않는다면 그 디렉토리 전체를 지워라. 어떠한 경우든지, 전체 소스 코드를 풀기 전에 `/usr/src` 디렉토리 안에 `linux`라는 것이 없음을 꼭 확인해야 한다.

이제, `/usr/src`에서, '`tar zxpvf linux-x.y.z.tar.gz`'명령으로 소스를 풀자. (만일 여러분이 끝이 `.gz` 이 아닌 `.tar`로 된 파일을 가지고 있다면 '`tar xpvf linux-x.y.z.tar`'를 사용해도 된다.). 소스 안의 내용이 빠르게 지나갈 것이다. 모두 풀리면, `/usr/src` 안에 새로운 '`linux`' 디렉토리가 생겼을 것이다. `linux` 로 들어가서 `README` 파일을 읽어보아라. '`INSTALLING the kernel`' 이라는 제목이 붙은 부분이 있을 것이다. 그곳에 쓰

여진대로 실행하라. 심볼릭 링크가 제자리 에 올바로 되어 있는지 확인하고, 쓸모없는 .o 파일들을 삭제한다든지 하는 등등의 것들을 적절히 행하라.

만약 .bz2로 된 파일과 bzip2라는 프로그램이 있다면 다음을 하라 (이것에 관해서는 http://www.muraroa.demon.co.uk/에서 더 읽을 수 있다):

```
bz2cat linux-x.y.z.tar.bz2 | tar xvf -
```

3.3 커널 설정

이 글중의 약간은 Linus의 README 파일내의 비슷한 장의 반복/해설 이다

/usr/src/linux 내에서의 'make config'명령은 여러분에게 수많은 질문을 하는 설정 스크립트를 실행한다. 이것은 bash가 필요하므로, /bin/bash나 /bin/sh, 또닫녈뉜뉜논녘높놈농뇨$냥냠넬넓를 확인하라.

'make config'외에 더 편한 방법들도 있다. 아마 'make menuconfig' 이 가장 많이 쓰이는 것일 것이다. 당신이 무엇을 선택하든 그 방법과 친해지는 것이 중요하다 - 왜냐하면 조만간 그것을 쓰고 또 쓸것 이기 때문이다. X를 쓰고 있고, Tk가 설치 되어 있으면 'make xconfig' 을 쓸 수 있다. 'make menuconfig'은 (n)curses가 설치 되어 있거나, 택스트로 된 매뉴를 쓰고 싶을 때 쓸 수 있다. 이들은 한가지 명확한 장점이 있다 : 만약 실수로 잘못된 값을 입력했을때, 언제라도 고칠수가 있다.

'make menuconfig' 과 'make xconfig' 에서 설정 옵션은 계급형태로 나타난다 (좀 큰 기능을 고르면 새로운 작은 기능들을 고를수 있다).

여러분이 질문에 답할 준비가 되면, 보통 'y' (yes) 또는 'n' (no) 로 대답한다. 장치 드라이버들은 보통 'm'옵션을 가지고 있다. 이것은 "module"을 뜻하며, 시스템이 컴파일할 때 직접 커널에 집어넣지 않고 적재 가능 모듈로 만드는 것을 말한다. 그것을 좀더 우습게 설명하자면, "maybe" 라고 할 수 있다. 여기서는 더 명확하고 필요하지 않은 몇몇 옵션들에 대해서는 설명하지 않는다. 다른 것들에 대해서는 "다른 구성 선택 사항들" 을 읽어보기 바란다. 'make menuconfig'은 스페이스 키로 기능을 선택한다.

2.0.x나 그 이후에서는, '?'옵션이 있다. 이 옵션을 쓰면 구성 파라매터에 대한 정확한 설명을 보여준다. 이 정보는 최신의 것일 것이다. 여기에는 중요한 기능, 이것이 무엇의 일부분인지, 그리고 간단한 설명이 들어있다.

3.3.1 Kernel math emulation (Processor type and features)

만일 여러분이 수치 연산 보조 프로세서를 가지고 있지 않다면 (여러분이 386하 나만이나 486SX를 가지고 있다면) 여러분은 'y'라고 해야 한다. 여러분이 보조 프로세서를 가지고 있는데 'y'라고 했더라도 너무 걱정하지 말라. 이 경우에는 보조 프로세서가 사용되고 에뮬레이션은 무시된다. 단지 중요한 것은 커널이 더 커진다는 것이다(RAM을 소비한다). 나는 수학 에뮬레이션이 느리다고 말한적이 있다. 비록 이것이 이 장에서는 별로 상관없을지라도, 느린 X 윈도우 시스템 실행을 할 때에는 꼭 염두에 두어야 할 것이다.

3.3.2 Enhanced (MFM/RLL) disk and IDE disk/cdrom support (Block Devices)

여러분은 아마 이것을 지원해야 할 것이다. 이것은 커널이 대부분의 사람들이 가지고 있는 표준 PC 하드 디스크를 지원한다는 것이다. 이 드라이버는 SCSI 드라이브는 포함하지 않는다. 그것은 구성의 나중에 나온다.

여러분은 "old disk-only"와 "new IDE" 드라이버에 해서 질문을 받을 것이다. 여러분이 그들중 하나를 고르고자 한다면, 그 둘사이의 주된 차이점은 구 드라이버는 하나의 인터페이스에 오직 두개의 디스크만을 지원하는데 비해, 새것은 두번째 인터페이스와 IDE/ATAPI 시디롬 드라이브를 지원한다. 새 드라이버는 그 이전것 보다 4k 더 크고 또한 "개선된", 즉 가지고 있는 버그의 수가 다른 것을 뜻한다. 이것은 여러분의 디스크의 실행을 ,특히 여러분이 새 (EIDE 타입) 하드웨어를 가지고 있다면, 개선시켜 줄 것이다.

3.3.3 Networking support (General Setup)

여러분의 머신이 인터넷과 같은 네트워크와 연결되어 있거나, 전화를 걸어서 인터넷에 접근하기 위해 SLIP, PPP, 터미날 등을 사용하고자 한다면 'y'라고 대답해야 한다. 그러나, 많은 패키지들(X 윈도우 시스템 같은)이 여러분의 머신이 진짜 네트워크에 연결되어 있지 않더라도 네트워크 지원을 요구하므로 여러분은 'y'라고 답해야만 한다. 나중에 TCP/IP 네트워킹을 지원하기를 원하는지 물을 것이다. 다시 말하지만, 여러분이 정말로 확신하지 않는다면, 여기에 'y'라고 답하라.

3.3.4 System V IPC (General Setup)

IPC(Interprocess Communication: 내부 프로세스간 통신)에 대한 가장 좋은 정의 중의 하나가 Perl 책의 용어 풀이에 있다. 놀랄 것도 없이, 몇몇 Perl 프로그래머들은 이것을 다른 패키지들처럼(가장 주목할 만하게도, DOOM같은 것) 프로세스들이 서로 대화하게 하는데 사용한다. 그러므로 여러분이 무엇을 하는지 정확히 알지 못하면 'n'라고 답하는 것은 좋은 생각이 아니다.

3.3.5 Processor family (Processor type and features)

(이전의 커널에서는 486에 대한 최적화를 위해서 -m486 플래그를 사용한다.)

예전부터, 이것은 특정 프로세서를 최적화하여 컴파일하였다. 커널은 다른 칩 들에서도 잘 작동하지만, 커널은 약간 커졌다. 그러나 새 커널에서는, 더이상 사실이 아니다. 그래서 여러분은 커널을 컴파일하는 프로세서를 입력해야 한다. "386" 커널은 모든 머신에서 작동한다.

3.3.6 SCSI support

만약 여러분이 SCSI 장치를 가지고 있다면 'y'라고 답하라. 여러분은 더 많은 정보들을 보게 될 것이다. 시디롬, 디스크, 그리고 여러분이 가진 SCSI 아답터가 무엇인지 또 지원하는지 같은것들 말이다. 더 자세한 것을 알기 위해서 는 SCSI-HOWTO를 보기 바란다.

3.3.7 Network device support

여러분이 네트워크 카드를 가지고 있거나, 인터넷에 접속하기 위해 SLIP, PPP, 또는 패러렐 포트 아답터를 사용하고자 한다면 'y'로 답하라. 설정 스크립트는 여러분이 가지고 있는 카드가 어느 것인지, 어떤 프로토콜을 사용할 것인지를 보여줄 것이다.

3.3.8 Filesystems

그리고 나서, 설정 스크립트는 여러분에게 다음의 파일 시스템을 지원하기를 원하는지에 대해서 물어볼 것이다.

Standard (minix) - 새로운 배포판에서는 미닉스 파일 시스템을 만들지 않고 또 많은 사람들이 그것을 사용하지 않는다. 그러나 아직은 넣는 것이 좋다. 몇몇 "구조 디스크" 프로그램들이 그것을 사용하고, 플로피에 사용하기에는 미닉스 파일 시스템이 덜 나쁘기 때문에, 여전히 많은 플로피들이 미닉스 파일 시스템을 사용한다.

Second extended - 이것은 새 배포판에서 널리 쓰이고 있다. 여러분은 아마도 이중에 하나를 가지고 있을 것이므로 'y'라고 답해야 한다.

msdos - 만일 여러분이 MS-DOS 파티션을 사용하고자 한다면, 또는 MS-DOS로 포맷된 플로피 디스크를 마운트하고자 한다면 'y'이다.

그 외에 다양한 외부 운영체계에 존재하는 파일 시스템도 쓸 수 있다.

/proc - (아무래도 내 추측에는, 벨 연구소에서 온 것 같다). 아무도 디스크에 proc 파일 시스템을 만들 수 없다. 이것은 커널과 프로세스들을 위한 파일 시스템 인터페이스이다. 많은 프로세스 목록기들('ps' 같은)이 이것을 사용한다. 언젠가 'cat /proc/meminfo'나 'cat /proc/devices'를 시도해 보아라. 몇몇 쉘들은 (특히 rc)는 입출력을 위해서 /proc/self/fd (다른 시스템들에서는 /dev/fd 로 알려진) 를 사용한다. 여러분은 여기에 거의 확실히 'y'라고 답해야 한다. 많은 중요한 리눅스 도구들이 이것에 의존하고 있다.

NFS - 만일 여러분의 머신이 네트워크에 연결되어 있고 NFS로 다른 시스템에 존재하는 파일 시스템을 사용하기 위해서는 'y'라고 답하라.

ISO9660 - 대부분의 시디롬들에 있다. 여러분이 시디롬 드라이브를 가지고 있고 리눅스하에서 사용하고자 한다면, 'y'이다.

하지만 나는 나한테 필요한 파일 시스템이 어떤건지 모르는데!! 좋다.그럼 'mount'라고 쳐보라. 그 결과는 다음과 비슷할 것이다.

```
blah# mount
/dev/hda1 on / type ext2 (defaults)
/dev/hda3 on /usr type ext2 (defaults)
none on /proc type proc (defaults)
/dev/fd0 on /mnt type msdos (defaults)
```

각 라인을 보라. 'type' 다음에 오는 단어가 파일 시스템 타입이다. 이 예 에서는, 내 / 와 /usr 의 파일 시스템은 second extended 이고, 나는 /proc를 사용하고 있다. 그리고 플로피 디스크를 msdos 파일 시스템으로 마운트하여 사용하고 있다.

여러분이 /proc 를 가지고 있고 현재 사용중이라면, 'cat /proc/filesystems' 를 해 볼 수 있다. 그것은 여러분의 현재 커널의 파일 시스템 목록이다.

거의 쓰지 않는, 필요하지 않은 파일 시스템의 구성은 커널을 부풀리게 할 수 있다. 이것을 피할 수 있는 방법으로 모듈에 대한 섹션을 읽어보기 바란다. 그리고 "함정" 섹션에서 왜 부풀려진 커널이 좋지 않은지 보아라.

3.3.9 Character devices

여러분은 여러분의 프린터(패러렐 프린터를 말함)나 버스 마우스, PS/2 마우스 (많은 노트북들에서는 장착된 트랙볼을 위해서 PS/2 마우스 프로토콜을 사용하고 있다.), 몇몇 테이프 드라이브들, 그리고 다른 "특정" 장치들을 위해 드라이버들을 사용할 수 있다. 적절한 곳에 'y'를 하라.

알아둘 것 : gpm이란 프로그램은 가상 콘솔에서 마우스로 cut & paste 를 할 수 있게 한다. X (X 윈도 시스템) 가 있어도 문제없이 쓸 수 있어, 마우스가 있는 사람에게는 괜찮은 것이다. 하지만 X 가 아닌 다른것에서는 특별한 꾀가 필요하다.

3.3.10 Sound

만약 여러분이 다양한 소리를 듣고 싶다면 'y'이다. 그러면 또 여러분에게 여러분의 사운드 카드에 대한 모든것을 물어보고 컴파일할 것이다. (사운드 카드 구성에서 알아둘 것: 만약 풀버전의 드라이버를 설치할 것이냐고 물어오면, 'n'라고 답함으로써 여러분이 정말로 필요한 부분만을 커널에 집어 넣고 메모리를 절약할 수 있다.) 나는 여러분이 사운드 카드를 가지고 있다면 사운드 지원에 대한 더 자세한 것을 알기 위해서 꼭 Sound-HOWTO 를 읽어보기를 권한다.

만약 특정 사운드 카드가 지원되는지를 알고싶으면 http://www.linux.org.uk/OSS/에서 무료 드라이버 를 보던지 Open Sound System <http://www.opensound.com/>에서 상업용 드라이버를 봐라.

3.3.11 다른 구성 선택사항들

여기에 모든 구성 선택사항들이 있는 것은 아니다. 왜냐하면 그들은 너무 자주 바뀌거나 아니면 자명한 것들이기 때문이다.(예를 들면, 3Com 3C509 지원은 이 특정 이더넷 카드를 사용하기 위해 장치 드라이버를 컴파일해야 한다).

온라인 help에는 Axel Boldt(`boldt@math.ucsb.edu`)씨가 시작하고 관리하고 있는데 여기에는 만든 모든 선택사항(그들을 `Configure` 스크립트에 넣는 방법까지도)에 대한 매우 포괄적인 목록들이 있다. 이것은 또 하나의 큰 파일로써 `Documentation/Configure.help` 라는 이름으로 Linux 소스 버전 2.0부터 존재한다.

3.3.12 Kernel hacking

>Linus의 README에서:

"kernel hacking" 구성은 보통 커널이 더 크거나 더 느려지는(또는 둘다) 결과를 자세히 설명해주고, 어떤 루틴들을 넣어서 커널의 문제점(kmalloc())이 되는 잘못된 코드를 찾아 멈추게 하려고 하기 때문에 커널을 덜 안정하게 만들 수도 있다. 그러므로 여러분은 아마도 "안정" 커널에서는 질문에 'n'라고 답해야 할 것이다.

3.4 이제 무얼하지?(The Makefile)

make config를 한 후에, 여러분은 여러분의 커널 설정이 끝났으므로 "추가적인 구성을 위해서 최상위의 `Makefile`을 확인해 보라" 고 하는 등의 메세지를 만날 것이다.

이제 `Makefile`을 보자. 여러분은 아마도 고칠 필요가 없을 것이다. 하지만 본다고 상하지 않으니까 한번 보자. 여러분은 또한 새 커널을 설치했을때만 한번 '`rdev`' 명령을 사용함으로써 선택사항들을 바꿀 수 있다.

제 4 절 Compiling the kernel

4.1 Cleaning and depending (청소와 의존성 검사)

설절 스크립트가 끝났을 때, '`make dep`'와 '`make clean`'을 하라는 메세지를 보았을 것이다. 그러므로, '`make dep`'를 하라. 이것은 모든 의존성, 그러한 포함된 파 일들, 이 제대로 되어 있는지를 확인한다. 이것은 여러분의 컴퓨터가 너무 느리 게 시작되지 않는다면 그리 오래 걸리지 않는다. 이것이 끝나면 '`make clean`'을 하라. 이것은 모든 오브젝트 파일과 구버전이 남겨논것을 제거하는 것이다. 절대로 이 단계를 잊지 말기 바란다.

4.2 컴파일할 시간

cleaning 과 depending 한 후에, 여러분은 '`make bzImage`'나 '`make bzdisk`'를 해야한다.(이부분이 가장 시간이 오래 걸린다.). '`make bzImage`'는 커널을 컴파일 하고, '`bzImage`'라는 파일을 `arch/i386/boot` 에 남긴다. 이것은 새로 압축된 커널이다. '`make bzdisk`'도 같은 것인데, 이것은 새 bzImage를 "A:" 드라이브의 플로피 디스크에 넣는다. '`zdisk`'는 새 커널을 시험해 보는데 좋다. 만약 그것이 폭탄(제대로 작동하지 않는것)이라서, 플로피를 제거하고 여러분의 구버전 커널로 부팅해야한다면 말이다. 또 여러분이 사고로 여러분의 커널을 지웠다면, (아니면 이와 비슷하게 치명적인 일이 생긴다면) 이것으로 부팅할 수도 있다. 그리고 여러분이 한 디스크에 있는 내용을 다른 쪽으로 옮겨서 새 시스템에 설치하고자 할 때도 사용할 수 있다.("이것이 다가 아니다! 얼마나 값어치가 있는가!")

모든, 최근것이라고 보기에는 불충분한 것들까지도, 커널들은 압축되어 있다. 이 때문에 이름앞의 'bz'가 나왔다. 압축된 커널은 실행될 때 자동적으로 자기 자신이 압축을 푼다.

오래된 커넬은 `bzImage`라는 것으로 커넬을 못 만들수 있다; 그때는 간단히 `zImage` 였다. 이 옵션은 현재 존재하지만 현재 커넬 크기를 볼때 이 옵션은 큰 크기의 커넬을 못 쓰므로 `bzImage`를 만드는것이 기본처럼 되어가고 있다.

4.3 다른 "make"할 수 있는 것들

'make mrproper'는 더욱 확장된 'clean'ing을 한다. 이것은 때때로 필요하다. 여러분은 매 패치때마다 이것을 할 수 있다. 'make mrproper'는 또한 여러분의 구성 파일을 지우기 때문에, 그것이 필요하다고 생각한다면 사본을 만들어야 할 것이다.

'make oldconfig'는 이전의 설정 파일로부터 커널 설정을 시도한다. 이것은 'make config'를 넘어가게 된다. 하지만 여러분이 이전에 커널 컴파일을 한적이 없다거나 이전의 구성 파일이 없다면 여러분은 이것을 할 수 없고, 기본 구성을 원하는 대로 바꾸어야 할 것이다.

'make modules'의 설명에 대해서는 모듈에 대한 섹션을 보기 바란다.

4.4 커널의 설치

여러분이 하고자 하는 일을 할 수 있는 새 커널을 가진 후에는, 설치를 해야한다. 대부분의 사람들은 이것을 위해서 LILO(Linux Loader)를 사용한다. 'make zlilo'는 커널을 설치하고, 그것으로 리로를 실해시킨다.그리고 만일 리로가 다음과 같은 방법으로 여러분의 시스템에 설치되어 있다면, 여러분이 시작할 준비를 하게 해준다: 커널은 /vmlinuz이고, 리로는 /sbin에 있으며, 이것이 여러분의 리로 설정(/etc/lilo.conf)과 같아야한다.

그렇지 않다면, 여러분이 직접 리로를 실행해야한다. 이것은 매우 설치하기 쉽고 잘 작동하는 패키지이다. 그러나 사람들이 구성 파일과 혼동하는 경향이 있다. 설정 파일을 보아라.(구버전의 /etc/lilo/config 나 새버전의 /etc/lilo.conf). 그리고 현재의 설정이 어떠한지 보아라. 그 설정 파일은 다음과 같다.

```
image = /vmlinuz
    label = Linux
    root = /dev/hda1
    ...
```

'image ='는 현재 설치된 커널을 나타낸다. 대부분의 사람들은 /vmlinuz을 사 용한다. 'label'은 리로가 어느 커널이나 운영체제로 부팅할 것인지를 결정하는데 사용한다. 그리고 'root'는 특정 운영체제의 / 이다. 여러분의 이전 커널의 사본을 만들고, 여러분이 막 만들어낸 bzImage를 제위치에 복사한다(여러분이 '/vmlinuz' 를 사용한다면, 'cp bzImage /vmlinuz'이라고 쳐야한다.). 그리고 나서, 리로를 새로운 시스템에서 재실행 시킨다. 여러분은 단지 'lilo'라고 치기만 하면 된다. 그러나 구버전에서는 /etc/lilo/install 이나 /etc/lilo/lilo -C /etc/lilo/config 라고 해야할 것이다.

만일 여러분이 리로의 구성에 대하여 더많이 알고자 하거나 리로를 가지고 있지 않다면, 여러분이 좋아하는 ftp 사이트에 가서 최신 버전을 가져와서 설명에 따르기 바란다.

정상적이지 않은(커널이 제대로 작동하지 않는:역자주) 하드디스크에서 여러분 의 이전 버전의 커널들중에 하나로 부팅하기 위해서는(이러한 경우에 여러분을 구하는 다른 방법은 새 커널을 설치하는 것이다.), 리로 설정 파일의 아래부분 에 있는 'image = xxx'를 포함한 부분을 복사한다. 그리고 'image = xxx' 를 'image = yyy'로 고친다. 여기서 'yyy' 는 여러분의 백업커널이 저장되어 있는 파일의 전체 경로명이다. 그리고 나서 'label = zzz' 을 'label = linux-backup' 으로 고치고 리로를 다시 실행한다. 여러분은 설정파일에다가 'delay=x'라고 써넣어 주어야 할 것이다. x는 10분의 1초 단위의 양으로 이것은 리로가 부팅하기 전에 잠시 기다리도록 하는 것이다. 그러면 여러분은 그것을 정지시킬 수 있고(예를 들면, 시프트 키를 사용해서), 백업 부트 이미지의 라벨을 써 넣으면 된다 (좋지 않은 일이 생길 경우에).

제 5 절 커널 패치하기

5.1 패치의 적용

커널의 늘어난 양은 패치로 배포된다. 예를 들어, 여러분이 1.1.45버전을 가지고 있고 다른 어딘가에 'patch46.gz'이 있다고 하자. 이것은 여러분이 패치의 적용을 통해서 1.1.46버전으로 업그레이드 할 수 있다

는 뜻이다. 여러분은 가장 먼저 소스 구조의 백업을 만들어야 할 것이다.('make clean' 을 하고 'cd /usr/src; tar zcvf old-tree.tar.gz linux'라고 하면 여러분은 tar 압축 파일을 만들 수 있다).

위의 예에서 계속하면, 여러분이 'patch46.gz'을 /usr/src 에 가지고 있다고 가정하자. /usr/src로 들어가서 'zcat patch46.gz | patch -p0' 라고 한다. (또는 패치가 압축되어있지 않다면 'patch -p0 < patch46'이라고 해도 된다). 그 것이 성공하든 안하든간에, 내용을 적용하고 있다는 소리를 들으며 윙하고 날 아가는 것(만약 여러분의 시스템이 느리다면 천천히 내려오는 것을)을 보게 될 것이다. 보통, 이 작업은 여러분이 읽기에는 너무 빨리 지나가서 그것이 제대 로 작동되고 있는건지 아닌지 확인할 수 없다. 그래서 패치할 때에 -s 플래그를 사용함으로써 단지 에러 메세지만을 나오게 할 수 있다(여러분은 "내 컴퓨 터가 실제로 무언가 바꾸고 있구나!"라는 느낌밖에는 가질 수 없을 것이다. 하 지만 여러분이 원한다면...). 부드럽게 넘어가지 않는 부분들을 보기 위해서는 /usr/src/linux로 가서 .rej 확장자를 가진 파일을 보아라. 어떤 버전의 패치 는(구버전일수록 하급의 파일 시스템에서 컴파일되었다.) # 확장자를 가진 파 일에 거부된(실패한) 사항을 남겼다. 이들을 찾기 위해 'find' 명령어를 사용할 수도 있다.

```
find .  -name '*.rej' -print
```

표준 출력으로 현재 디렉토리와 그 아래의 모든 서브 디렉토리에 있는 .rej 확장자를 가진 모든 파일들을 출력한다.

모든것이 제대로 되면, 3장과 4장에서 설명한 'make clean', 'config', 'dep' 를 실행한다.

패치 명령에는 약간의 옵션이 있다. 위에서 말한 것처럼, patch -s 는 에러 메세지 이외의 모든 메세지는 나오지 않게 한다. 만일 여러분이 여러분의 커널 소스를 /usr/src/linux 보다는 다른 장소에 가지고 있으려한다면, patch -p1 (그 디렉토리에서)는 패치한 것들을 말끔히 할 것이다. 다른 패치 옵션들은 메 뉴얼 페이지에 잘 설명되어 있다.

5.2 만일 무언가 잘못된다면

(알아둘 것 : 아래는 오래 된 커넬에게만 적용된다)

제기되는 가장 중요한 문제는 제대로 되지 않는 'config.in' 파일이 언제 생성되었나 하는 것이다. 이것은 여러분이 그 선택사항들을 여러분의 머신에 맞게 고쳐야하기 때문이다. 이것은 조심해서 해야 한다. 하지만 여전히 한가지가 이전 버전과 충돌한다. config.in.rej 파일을 보면서 고치고 원래의 패치에는 무엇이 남아 있는지를 보아라. 고쳐진것은 보통 그 행의 처음에 '+'와 '-'로 표시 가 된다. 그 행 주변을 보고 그들이 'y'나 'n' 어떤 것으로 되어 있는지 기억하라. 이제, config.in을 편집해서, 'y'는 'n'로, 'n'는 'y'로 적절히 바꾸기바란다.

```
patch -p0 < config.in.rej
```

를 실행해서 제대로 되었다고 나오면(실패가 없으면), 구성과 컴파일을 계속하고, 남아있는 config.in.rej 파일은 지워도 된다.

만일 'previously applied patch detected: Assume -R?'이라는 말이 나오면, 여러분은 아마 현재 버전 번호 아래의 것으로 패치를 적용하려고 하는 것일 것이다. 'y'라고 답하면, 여러분의 소스를 낮은 버전으로 되돌리려고 하는 것이므로 대부분은 실패할 것이다. 이렇게 되면, 여러분은 새로운 소스 구조 전체를 가져야만 한다.(처음에는 그렇게 나쁜 생각만은 아니다.).

패치를 되돌리려면(적용한 것을 되돌리려면), 원래 패치에서 'patch -R'을 사용하라.

패치가 정말 잘못되었을 때 가장 좋은 방법은 모두 끝내고 clean으로 다시 하는 것이다. 소스 구조의 테두리를 벗어나서(예를 들면 linux-x.y.z.tar.gz 파일들 중에 하나에서부터) 처음부터 다시 시작한다.

5.3 .orig 파일들을 없애기

몇번의 패치 후에는, .orig 파일들이 쌓이기 시작할 것이다. 예를 들어, 내가 가지고 있던 한 1.1.51 트리는 1.1.48이었을 때 한번 청소했을 뿐이다. 반 메가 정도 저장되어 있는 .orig 파일들을 없애자.

```
find . -name '*.orig' -exec rm -f {} ';'
```

주의해서 하기 바란다. 거부사항을 위해서 #를 사용하는 패치 버전은 .orig대신에 (tilde)를 사용하라.

.orig 파일들을 없애는 더 좋은 방법은 GNU Xargs를 이용하는 것이다.

```
find . -name '*.orig' | xargs rm
```

또는 "아주 확실하지만 약간 좀 화면출력이 많은" 방법이 있다.

```
find . -name '*.orig' -print0 | xargs --null rm --
```

5.4 다른 패치들

Linus의 배포물들 이외에도 다른 패치들이 있다.(나는 그들을 "비표준"이라고 부른다.) 만일 여러분이 이것들을 적용한다면, Linus의 패치들은 제대로 작동하지 않을지도 모르고, 또 여러분은 이것을 되돌리기 위해서, 소스나 패치를 수정하거나, 새로운 소스 트리를 설치하거나, 아니면 위의 작업들을 섞어가며 해야 할 것이다. 이것은 아주 쓸모없는 일이 될 수 있다. 그래서 만약 여러분이 소스를 수정하기를 (아주 나쁜 결과가 나올 가능성을 가지고) 원하지 않는다면, Linus의 패치를 적용하기 전에 비표준 패치들을 되돌려 놓던가, 아니면 새 소스를 설치해야 한다. 그리고 나면, 비표준 패치들이 아직도 동작하는지 볼 수 있을 것이다. 만일 그들이 동작하지 않으면, 여러분은 이전 커널에 머물러 있거나, 하고자 하는 일을 할 수 있는 소스나 패치를 가지고 놀든가, 아니면 새로운 버전의 패치가 나올때까지 기다릴 수 있다.

표준 배포판이 아닌경우에 패치는 얼마나 일상적인 것일까? 여러분은 아마도 이 것에 대해 들어보았을 것이다. 나는 깜빡이는 커서를 정말 싫어하기 때문에 내 가상 콘솔을 위해서 깜빡임없는 패치를 사용한 적이 있다.(이 패치는 새 커널 버전에서는 빈번히 업데이트된다.(적어도 전에는 그랬다.)). 가장 새로운 장치 드라이버들은 적재가능 모듈로 개발되고 있는 것과 동시에 빈번한 "비표준" 패 치도 확실히 줄어들고 있다.

제 6 절 추가적인 패키지들

여러분의 리눅스 커널은 커널 소스 그 자신이 설명하고 있는 것들이 안닌 다른 많은 모습들을 가지고 있다. 이 모습들은 보통 외부 패키지들이 사용한다. 가장 보편적인 것 몇가지의 목록이 여기에 있다.

6.1 kbd

리눅스 콘솔은 그것이 나타내는 것보다 더 많은 모습들을 가지고 있다. 이들 사이에는 폰트를 바꾸고, 여러분의 키보드를 재배치하고, 비디오 모드를 바꾸는(새 커널들에서) 등의 능력을 가지고 있다. kbd 패키지에는 사용자가 이러한 모든 일을 할 수 있게하는 프로그램과 많은 폰트들, 거의 모든 키보드들의 배치도, 가 들어 있고 커널을 가져온 사이트들에서 사용할 수 있다.

6.2 util-linux

Rik Faith (`faith@cs.unc.edu`)는, 우연히도, util-linux라고 부르는 리눅스의 유용한 프로그램들의 모음을 가져다 놓았다. 이것들은 현재 Andries Brouwer (`util-linux@math.uio.no`)에 의해서 운영되고 있다. sunsite.unc.edu의 익명 ftp내의 `/pub/Linux/system/misc`를 이용할 수 있고, 이것은 커널과 관련된 `setterm`, `rdev`, `ctrlaltdel`과 같은 프로그램들을 포함하고 있다. Rik의 말에 따르면, 생각없이 설치하지 말라. 여러분은 패키지안의 모든것을 다 설치할 필요는 없다. 만약 그럴 경우에는 심각한 문제를 야기시킬 수도 있다.

6.3 hdparm

많은 패키지들에서 이것은 커널 패치와 지원 프로그램들이었다. 그 패치들은 정식 커널 안에서 그것을 만들며, 여러분의 하드 디스크를 최적화시키고 작동시키는 프로그램은 따로 구분해서 배포된다.

6.4 gpm

gpm은 일반적인 용도의 마우스(general purpose mouse)를 나타낸다. 이 프로그램은 가상 콘솔들 사이에서 문서의 자르기와 붙이기를 할 수 있게 해주고, 마우스 타입에 따라 매우 다양한 다른 기능들을 사용하게 해준다.

제 7 절 몇몇 함정들

7.1 make clean

만일 여러분의 커널이 커널 업그레이드 과정을 거친 후에 정말 이상하게 되었다면, 새 커널을 컴파일하기 전에 `make clean` 하는 것을 잊은 것이다. 여러분의 시스템이 잘못되어가는 증상은, 이상하고 낯선 I/O 문제 등의 어떠한 것이든 될 수 있다. `make dep` 를 하는 것도 반드시 확인하기 바란다.

7.2 너무 크거나 느린 커널들

만일 커널이 너무 커서 많은 메모리를 차지하고/거나, 여러분이 새로운 콰드바질리움-III/4400 (해석: 빠른 CPU) 에서 사용할 때조차 컴파일하는데 너무나 많은 시간이 걸린다면,여러분은 아마도 필요하지 않은 것(장치 드라이버들이나, 파일 시스템들 등등) 들을 너무많이 구성하여 집어넣었을 것이다. 여러분이 사용하지 않는 것이라면, 넣지 마라. 그것은 메모리를 차지한다. 부풀린 커널의 가장 명확한 증상은 스와핑이 극에 달해서 디스크에 메모리가 부족해지는 것이다.만일 여러분의 디스크가 너무 소리가 많이 나고, 전원을 끌때 제트기가 뜨는 것 같은 소리가 나는 구형 Fujitsu Eagle(하드 디스크 모델인것 같다:역자주)이 아니라면, 여러분의 커널 구성을 조사해 보기 바란다.

여러분은 여러분 머신의 총메모리 양에서 차지하는 것과 `/proc/meminfo` 내용중에 "`total mem`"의 양에서 빼거나 '`free`'명령의 결과를 통해서 커널이 얼마나 많은 양의 메모리를 사용하는지 알아낼 수 있을 것이다.

7.3 병렬포트나 프린터가 작동이 안된다

PC에서는 구성 목록중 'General Setup' 에서 'Parallel port support' 과 'PC-style hardware' 를 선택한다. 그 후에 'Character devices' 에서 'Parallel printer support'를 선택한다.

그 후에는 파일 이름 문제가 있다. Linux 2.2는 이전 버젼과 다른 프린터 이름을 쓴다. 결론적으로는 이전 버젼의 리눅스에 프린터가 `lp1`였다면 새 커넬에서는 `lp0`일 것이다. 'dmesg'나 `/var/log`에 나오는 로그를 보고 어느것인자 알아봐라.

7.4 커널이 컴파일되지 않는다

만약 컴파일되지 않는다면, 패치가 실패했거나, 아니면 여러분의 소스가 어떻게해서든 방해를 받았다는 것이다. 여러분의 gcc 버전이 맞지 않다거나, 그렇지 않아도 역시 중지될 수 있다(예를 들면, 포함하는 파일들이 에러가 있다면). Linus가 `README`에서 설명한대로 심볼릭 링크가 제대로 되어 있는지 확인하라. 보통, 표준 커널이 컴파일되지 않으면, 시스템에 심각한 문제들이 있으므로, 특정한 도구들은 재설치를 할 필요가 있을 것이다.

어떤 경우에는, 하드웨어의 문제 때문에 gcc가 잘못될 경우도 있다. 그 에러 메세지는 "xxx exited with signal 15"같은 것인데, 보통 매우 색다르게 보일것이다. 이것은 말하기 싫지만, 나에게도 이런일이 한번 일어난 적이 있다. 나는 약간 좋지 않은 캐쉬 메모리를 가지고 있었는데, 컴파일러가 갑자기 에러 메세지를 내보내면서 제

대로 작동하지 않았다. 만일 여러분이 문제에 닥치면 제일 먼저 gcc를 재설치하라. 여러분의 커널이 RAM 양을 줄이고 외부 캐쉬를 끌때에만 제대로 컴파일된다면, 한번 의심해보기 바란다.

기계에 문제가 있다고 누가 그러면 골치가 아파질 것이다. `http://www.bitwizard.nl/sig11/` 에 FAQ까지있으니 믿을 만한 이야기다.

7.5 새 버전의 커널이 부팅되지 않는것 같다

여러분이 LILO를 실행하지 않았거나, 아니면 설정을 제대로 하지 않은 것이다. 내가 "겪었던" 것들중의 하나는 설정 파일내의 문제였다. 그것은 바로 '`boot = /dev/hda`' 대신에 '`boot = /dev/hda1`' 라고 한 것이다.(이것은 처음에는 정말로 화가날 수 있는 것이지만, 제대로 작동하는 설정 파일을 가지고 있을 때는, 그것을 바꿀 필요가 없다.).

7.6 여러분이 리로를 실행하는 것을 잊거나, 시스템이 전혀 부팅되지 않는다

저런! 여기에서 여러분이 할 수 있는 가장 최선의 방법은 플로피 디스크나 CDROM으로 부팅하고 또다른 부팅가능한 플로피를 준비하는것이다('`make zdisk`' 같은 것으로 할 수 있다). 여러분은 여러분의 루트(/ 파일 시스템이 어디에 있고, 어떤 타입(예. second extended, minix)인지를 알아야만 한다. 아래의 예에서는, 또한 여러분의 `/usr/src/linux` 소스 트리가 어떤 타입의 파일 시스템에 있고, 보통 어디에 마운트되어 있는지 알아야만 한다.

다음 예에서는 / 는 `/dev/hda1`이고 `/usr/src/linux`가 있는 곳은 `/dev/hda3`이며, 보통 `/usr`에 마운트되어 있다. 둘다 second extended 파일시스템으로 되어 있다. 사용하는 커널은 `/usr/src/linux/arch/i386/boot`에 있는 `zImage`이다.

제대로 작동하는 `bzImage`가 있다면, 새 플로피를 사용하는 것도 가능하다. 또 다른 방법에 대해서는, 그것이 더 좋을 수도 있고 아닐 수도 있지만, 이 예 다음에 논하기로 한다.(이것은 여러분이 여러분의 시스템을 혼란에 빠뜨린 방법에 따라 다르다.)

우선, 부트/루트나 복구용 디스크로 부팅한다. 그리고 커널 이미지가 있는 파일 시스템을 마운트한다.

```
mkdir /mnt
mount -t ext2 /dev/hda3 /mnt
```

만일 `mkdir`이 디렉토리가 이미 존재한다고 하면, 무시하라. 자, 커널 이미지가 있는 디렉토리로 들어가자.

```
/mnt + /usr/src/linux/arch/i386/boot - /usr = /mnt/src/linux/arch/i386/boot
```

포맷된 디스크를 "A:" 드라이브에 넣고(부트나 루트 디스크가 아니다!), 이미지를 디스크로 복사한다. 그리고 여러분의 루트 파일 시스템에 그것을 설정한다.

```
cd /mnt/src/linux/arch/i386/boot
dd if=bzImage of=/dev/fd0
rdev /dev/fd0 /dev/hda1
```

/ 로 들어가서 `/usr` 파일 시스템을 언마운트시킨다.

```
cd /
umount /mnt
```

이제 여러분은 이 플로피로 보통때처럼 여러분의 시스템을 재부팅할 수 있을 것이다. 재부팅후에 리로(또는 여러분이 잘못한 것이 무엇이든)를 실행하는 것을 잊지 말라!

위에서 말한것처럼, 일상적인 다른 방법이 있다. 만일 여러분이 / 에 커널이미지를 가지고서(예로 /vmlinuz) 이런 상황이 일어났다면, 이것을 부트디스크로 사용할 수 있다.위의 상황을 모두 가정하고, 내 커널 이미지가 /vmlinuz 일때, 위의 예를 다음과 같이 하라. /dev/hda3를 /dev/hda1(/ 파일 시스템)으로 바꾸고, /mnt/src/linux 를 /mnt 로 바꾼다. 그리고 if=bzImage 를 if=vmlinuz으로 바꾼다. 어떻게 /mnt/src/linux가 무시되는지 확실하게 알아두어라.

큰 용량(1024 실린더 이상의)의 드라이브들에서 리로를 사용하는 것은 문제를 일으킬 수도 있다. 그것에 대한 것은 도움말 문서나 LILO mini-HOWTO를 보아라.

7.7 'warning: bdflush not running'이라고 말한다

이것은 큰 문제일 수 있다. 커넬 1.0부터는 (1994년 4월 20일 정도) 'update' 이라는 정기적으로 파일 시스템의 버퍼을 비워주는 프로그램이 바뀌었다. 해결책은 'bdflush'라는 프로그램의 소스를 구해서 (커넬 소스를 구한 곳에서 구할 수 있을 것이다) 설치하라 (이때 아마도 옛 커넬 안에서 하는 것이 좋을 것이다). 이 프로그램은 자신을 'update'라는 이름으로 설치할 것이고, 재부팅 후에는 더이상 에러를 안 낼 것이다.

7.8 'warning: bdflush not running'라고 말한다

이것은 심각한 문제가 될 수 있다. 정기적으로 파일 시스템 버퍼를 모두 소비시키는 'update'라고 불리는 프로그램은 커널 버전 1.0(1994년 4월 20일 경) 이후에서 시작해서 업그레이드/대체 되었다. 'bdflush'에 커널 소스를 가져와서 (여러분의 커널 소스를 어디서 가져왔는지는 여러분이 찾아야한다), 그것을 설치하라(여러분은 여러분의 시스템이 이것을 하는 동안은 이전 커널에서 실행되기를 원할 것이다). 그것은 그 자신이 'update®'로 설치되고 재부팅한 후에는 더이상 불평하지 않을 것이다.

7.9 내 IDE/ATAPI CD-ROM이 작동하지 않는다

이상하게도, 많은 사람들의 ATAPI 드라이브들이 작동하지 않는데, 이것은 아마도 잘못될 수 있는 것들이 많기 때문일 것이다.

만일 여러분의 CD-ROM 드리이브가 단지 특정 IDE 인터페이스에 있는 장치라면, 틀림없이 점퍼가 "master" 나 "single"로 설정되어 있을 것이다. 이것은 가장 일상적인 에러이다.

예로, Creative Labs 는 IDE 인터페이스를 그들의 사운드 카드에 장착하고 있다. 그러나, 이것은 몇몇 사람들은 단지 하나의 인터페이스를 가지고 있는 반면에 많은 사람들은 그들의 마더보드에 두개의 IDE 인터페이스(보통 IRQ 15에)를 내장하고 있어서 아주 흥미로운 문제를 야기시킨다. 그래서 보통은 사운드 블러스터의 인터페이스를 세번째 IDE 포트(내가 들은 바로는 IRQ11)로 만든다.

이것은 1.2.x 버전의 리눅스가 새번째 IDE 인터페이스를 지원하지 않음으로 해서 문제를 발생시킨다.(1.3.x대의 어디에선가 이것을 지원한다. 하지만 기억해 둘것은 그것은 개발중이므로 자동으로 찾아주지는 않는다). 이에 대해서는, 몇가지 선택이 있다.

여러분이 이미 두번째 IDE 포트를 가지고 있다면, 그것을 사용하지 않아서 아직 두개의 장치를 가지지 않은 것이다. ATAPI 드라이브를 사운드 카드에서 꺼내서 두번째 인터페이스에 연결한다. 그리고 나서는 사운드 카드의 인터페이스를 사용할 수 없게 함으로써 어떤식으로든 IRQ를 아낀다.

여러분이 두번째 인터페이스를 가지고 있지 않다면, 사운드 카드의 인터페이스의 점퍼 (사운드 카드의 사운드 부분이 아니다)를 IRQ15, 즉 두번째 인터페이스로 설정한다. 이제 작동할 것이다.

7.10 쓸모없는 라우팅을 요구하는 것들에 대해 말한다

새버전의 route® 프로그램과 라우트 조작을 하는 다른 프로그램들을 가져온다. /usr/include/linux/route.h(실제로 /usr/src/linux에 있는 파일)이 바뀌었다.

7.11 1.2.0 에서 firewalling 이 작동하지 않는다

적어도 1.2.1 버전으로 업그레이드 하라.

7.12 "Not a compressed kernel Image file"

부트 이미지로 /usr/src/linux에 생성된 vmlinux 파일을 사용하지 말라. [..]/arch/i386/boot/zImage 가 옳은 것이다.

7.13 1.3.x로 업그레이드한 후에 콘솔 터미날에 문제가 있다.

/etc/termcap 의 콘솔 termcap 내용중에서 단어 dumb 를 linux로 바꾸라. 또한 여러분은 terminfo 내용을 만들어야 할 것이다.

7.14 커널 업그레이드 후에 컴파일이 잘 안된다.

리눅스 커널 소스는 /usr/include에 있는 표준들이 참고로하는 많은 include 파일들(끝이 .h로 끝나는 것)을 포함하고 있다. 그들은 보통 다음과 같이 참조되었다. (xyzzy.h는 /usr/include/linux에 있는 것이다.)

```
#include <linux/xyzzy.h>
```

보통, /usr/include에는 linux라는 여러분의 커널 소스내의 include/linux 디렉토리로의 링크가 있다(전통적인 시스템에서는 /usr/src/linux/include/linux). 만일 이 링크가 없거나 잘못된 곳을 가리키고 있을며 대부분은 전혀 컴파일되지 않을 것이다. 만일 여러분이 커널 소스가 디스크를 너무 많이 차지해서 지우기로 했다면, 이것은 명백히 문제가 될 것이다. 그것이 잘못될 수 있는 또다른 방법은 파일 퍼미션(허가)에 있다. 만일 여러분의 루트가 기본설정에 의해서 다른 사용자들이 파일들을 볼 수 없게하는 umask를 가지고 있고, 여러분이 p 옵션(보존 파일모드) 없이 커널 소스를 풀었다면, 그 사용자들은 C 컴파일러를 사용하지 못할 것이다. 여러분이 이것을 고치기 위해서 chmod 명령을 쓴다고해도, 아마 include 파일들을 다시 푸는 것이 더 쉬울 것이다. 여러분은 단지 아규먼트를 추가함으로써, 처음 시작때에 전체 소스를 가지고 했던것과 같은 방법으로 할 수 있다.

```
blah# tar zxvpf linux.x.y.z.tar.gz linux/include
```

Note: "make config"은 /usr/src/linux로의 소프트 링크를 를 필요한대로 만들것이다.

7.15 용량 늘이기

다음은 커널에 속해 있는 용량을 늘이는 방법의 예이다.

```
echo 4096 > /proc/sys/kernel/file-max
echo 12288 > /proc/sys/kernel/inode-max
echo 300 400 500 > /proc/sys/vm/freepages
```

제 8 절 2.0.x, 2.2.x 버전으로 업그레이드하기 위해 알아둘 것

커널 버전 2.0.x/2.2.x는 커널 설치가 아주 많이 바뀌었다고 말하고 있다. 2.0.x 소스 트리안의 Documentation/Changes 파일은 버전 2.0.x로 업그레이드 할 때, 여러분이 알아야만하는 것의 정보를 가지고 있다. 여러분은 이를 위해 대부분 몇몇의 key 패키지들과 gcc, libc, 그리고 SysVInit, 약간의 시스템 파일들을 업그레이드해야 할 것이다. 당황하지 말라.

제 9 절 Modules

적재가능한 커널 모듈들은 구성하기 쉽고 메모리를 절약할 수 있다. 모듈의 범위는 파일 시스템들, 이더넷 카드 드라이버들, 테이프 드라이버들, 프린터 드라이버들과 더욱 많은 것들을 포함해 가고 있다.

9.1 모듈 유틸리티 설치

모듈 유틸리티는 여러분이 커널 소스를 가져온 곳이면 어디든지 modules-x.y.z.tar.gz 을 가져오는 것이 가능하다. 가장 높은 패치로 여러분의 현재 커널보다 낮거나 같은 x.y.z를 선택한다. 'tar zxvf modules-x.y.z.tar.gz' 으로 풀고, 그것이 만든 디렉토리(modules-x.y.z)로 들어가서 README 파일을 읽고, 그 설치 설명(보통 make install 처럼 매우 간단한)대로 따른다. 여러분 은 이제 /sbin에 insmod, rmmod, ksyms, lsmod, genksyms, modprobe, 그리고 depmod 라는 프로그램들을 가지게 되었을 것이다. 여러분이 원한다면, insmod 에 있는 "hw"라는 예제 드라이버로 유틸리티를 테스트해 볼 수 있다. 자세한 것은 그 서브디렉토리에 있는 INSTALL이라는 파일을 보아라.

insmod는 모듈을 현재 실행중인 커널안에 삽입하는 것이다. 모듈들은 보통 .o 확장자를 갖는다. 위에서 언급한 예제 드라이버는 drv_hello.o 이므로, 그것을 삽입하기 위해서는, 'insmod drv_hello.o'라고 해야한다. 커널이 현재 사용중인 모듈을 보기 위해서는 lsmod를 사용한다. 그 결과는 다음과 같다.

```
blah# lsmod
Module:        #pages:  Used by:
drv_hello         1
```

'drv_hello'는 모듈의 이름이고, 메모리의 한 페이지(4k)를 사용하고 있다. 그 리고 그 순간에는 다른 커널 모듈은 없다. 이 모듈을 제거하기 위해서는 'rmmod drv_hello'를 사용한다. rmmod 다음에는 파일 이름이 아니라 모듈이름 을 사용함을 주의하라. 여러분의 이것을 lsmod의 목록으로부터 얻을 수 있다. 다른 모듈 유틸리티의 목적은 그들의 매뉴얼 페이지에 적혀있다.

9.2 커넬과 같이 배포되는 모듈들

버전 2.0.30의 예를 들자면, 많은 파일 시스템들과, 약간의 SCSI 드라이버들, 몇개의 이더넷 아답터 드라이버들, 그리고 나머지 다른것들은 모듈로서 적재할 수 있는 것이다. 그들을 사용하기 위해서는, 우선 우선 그들을 현재 커널에 구 성하여 넣지 않았는지 확인하라. 이것은 'make config'하는 동안에 y라고 하지 않은 것을 말한다. 새 커널을 컴파일하고 재부팅하라. 그리고 나서, 다시 /usr/src/linux로 들어가서, 'make modules'를 친다. 이것은 여러분이 커널안 에 구성하여 집어넣지 않은 모듈들을 모두 컴파일하고,/usr/src/linux/modules 안에 링크시키는 것이다. 여러분은 그 디렉토리에서 직접 사용할 수 도 있고, 'make modules_install'을 실행하여 /lib/modules/x.y.z에 설치할 수 있다. 여기서 x.y.z는 커널 버전 번호이다.

이것은 특히 파일 시스템들에 알맞다. 여러분은 아마 minix나 msdos 파일 시스 템은 자주 사용하지 않을 것이다. 예를 들면, 만약 내가 msdos 플로피를 사용하 게 된다면, 나는 /usr/src/linux/modules/msdos.o를 커널로 집어넣고(insmod), 끝나면 rmmod msdos하면 된다. 이 과정은 보통때 커널의 RAM 사용량을 50k정도 줄여준다. minix 파일 시스템을 사용할 때 알아두어야 할 것은, 이것을 "긴급 복구" 디스크에 사용하기 위해서는 항상 커널에 직접 구성하여 넣야 한다.

제 10 절 여러가지 팁들

10.1 make 또는 패치 명령의 결과 출력 방향 바꾸기(리다이렉션)

여러분이 'make'나 'patch' 명령이 한 것이 무엇인지 그 내용에 대한 기록을 원한다면, 결과 출력을 파일로 바꿀 수 있다. 우선, 여러분이 사용중이 쉘이 무엇인지 알아내라. 'grep root /etc/passwd' 와 '/bin/csh'같은

것들을 통해서 알 수 있다.

여러분이 sh 나 bash를 쓴다면,

```
(명령) 2>&1 | tee (결과 파일)
```

은 '(결과 파일)'에 (명령)의 결과를 복사한다.

csh나 tcsh 사용자는

```
(명령) |& tee (결과 파일)
```

rc(주의: 여러분은 아마 rc를 사용하지 않을 것이다.)에서는

```
(명령) >[2=1] | tee (결과 파일)
```

10.2 조건부 커널 설치

이전의 커널을 건드리지 않고 새 커널을 시험해 보는 방법은, 플로피 디스크를 사용하는 방법이외에도 몇가지가 있다. 많은 유닉스 애호가들은 좋아하지 않지만, 리로(LILO)는 디스크의 어느 곳에서든지 커널은 부팅할 수 있다.(만일 여러분이 대용량의 (500MB이상의) 디스크를 가지고 있다면, 어떻게 이것이 문제를 일으킬 수 있는지 LILO 문서를 읽어보아라). 그러므로 여러분이 다음과 비슷한 행을 LILO 설정 파일의 맨 마지막에 추가한다면, 여러분은 여러분의 이전 /vmlinuz을 건드리지 않고 새로 컴파일한 커널을 실행할 수 있다(물론 'lilo'를 실행한 후에).

```
image = /usr/src/linux/arch/i386/boot/bzImage
    label = new_kernel
```

LILO에게 새 커널은 부팅하도록 말해주는 가장 쉬운 방법은 부팅할 때에 프롬 프트를 나타나게 하기위해, 시프트 키를 누르는 것이다. 새 커널을 부팅하기 위해서는 여기에 'new_kernel'을 써넣어주면 된다.

만일 여러분이 여러분의 시스템에 동시에 몇개의 서로다른 커널 소스 트리를 가지고 있으려면(이것은 많은 디스크 용량이 필요하므로, 주의하라), 그들을 각각 /usr/src/linux-x.y.z라고 이름 붙이는 것이 가장 일반적이다. 여기서 x.y.z은 커널 버전이다. 그리고 나서는 소스 트리를 심볼릭 링크함으로써 선 택할 수 있다. 예를 들면, 'ln -sf linux-1.2.2 /usr/src/linux'는 버전 1.2.2를 현재것으로 만들어준다. 이와 같이 심볼릭 링크를 만들기 전에, ln 의 마지막 아규먼트가 실제 존재하는 디렉토리(이전의 심볼릭 링크는 괜찮다) 가 아닌지 확인하라. 존재한다면 그 명령의 결과는 여러분이 기대하던 것과 다를것이다.

10.3 커널 업데이트

Russell Nelson(nelson@crynwr.com)은 새 커널 버전의 변한 점을 요약한다. 이것은 짧지만, 여러분이 업그레이드하기 전에 보면 좋다. 이것은 익명 ftp인 ftp.emlist.com 의 pub/kchanges나 다음의 URL을 통해서 사용할 수 있다.

```
http://www.crynwr.com/kchanges
```

제 11 절 다른 읽을거리

- Sound-HOWTO: 사운드 카드와 유틸리티들
- SCSI-HOWTO: 모든 SCSI 콘트롤러와 장치들에 대해서

- NET-2-HOWTO: 네트워킹
- PPP-HOWTO: 특히 PPP 네트워킹에 대해서
- PCMCIA-HOWTO: 여러분의 노트북을 위한 드라이버들에 대해서
- ELF-HOWTO: ELF: 파일 시스템 변환하기
- Hardware-HOWTO: 지원되는 하드웨어에 대한 전반적인 것
- Module-HOWTO: 모듈에 관해서 더 자세하개
- Kerneld mini-HOWTO: kerneld에 관해서
- BogoMips mini-HOWTO: 만약 궁금하다면

제 12 절 잡동사니

12.1 저자

리눅스-하우투의 저자이자 관리자는 Brian Ward(`bri@cs.uchicago.edu`). 이다. 어떠한 말이나, 추가할 내용이나, 수정할 사항은 나에게 보내주기 바란 다(특히 수정할 사항은 나에게 가장 중요하다.)

다음의 URL중에서 여러분은 내 '홈 페이지'를 볼 수 있다.

```
http://www.math.psu.edu/bri/
http://blah.math.tu-graz.ac.at/~bri/
```

나는 메일에 가능한한 많은 정성을 들이려 하지만, 매일 너무 많은 메일을 받기 때문에, 여러분에게 돌아가는 시간이 매우 적음을 기억해 주기 바란다. 특히 나에게 전자우편으로 질문을 할 때는, 특별히 여러분의 메세지를 명확하 고 자세하게 설명하는데 노력해 주기 바란다. 만약 여러분의 편지가 작동하지 않는 하드웨어(또는 그와 비슷한)라면, 나는 여러분의 하드웨어 구성이 어떠 한지 알아야만 한다. 만일 여러분이 에러에 대해 보고하려 한다면, "나는 해 봤지만, 에러가 났다." 라고만 얘기하지 말라. 나는 에러가 무엇이었는지를 알아야 한다. 나는 여러분이 간단한 질문을 하는지에 대해서는 신경쓰지 않는 다. 기억하라! 여러분이 묻지 않는다면, 여러분은 결코 답을 얻을 수 없다! 나에게 의견을 보내 주는 모든 분들께 감사한다.

만약 질문이 커넬과 관련이 없거나 모르는 언어로 적혀 있으면 답을 안 할 수도 있다.

여러분이 나에게 메일을 보냈는데 어느정도의 시간(3주 이상) 지난 후에도 답장을 받지 못했다면, 내가 우연히 여러분의 메세지나 그런 것을 지웠을 수도 있으니(죄송), 다시 보내주기 바란다.

나는 실제로 하드웨어의 문제나 그러한 주제에 대한 메일들을 많이 받는다. 그건 괜찮다. 하지만 나는 세상에 있는 모든 하드웨어에 대해서 다 알지 못하 고 또, 어떻게 도울 수 있는지도 모른다는걸 염두에 두기 바란다. 나는 개인적으로 AMD, Adaptec, Sybios SCSI 컨트롤러, 그리고 IBM SCSI 디스크를 쓴다.

버전 -0.1은 1994년 10월 3일에 쓰여졌다. 이 문서는 SGML, 포스트 스크립트 , TeX, roff, 그리고 보통의 텍스트 형식으로도 사용할 수 있다.

12.2 할 것들(To do)

"여러가지 팁들(Tips and tricks)" 섹션이 약간 작다. 나는 다른이들 의견 으로 그 내용을 확장하기를 바란다.

"추가 패키지들(Additional pakages)" 도 마찬 가지다.

더 많은 디버깅/파손 복구에 대한 정보가 필요하다.

12.3 도움들

Linus의 README 의 몇몇 부분(kernel hacking options)이 포함되었다. (Thanks, Linus!)

`uc@brian.lunetix.de` (Ulrich Callmeier): patch -s and xargs.

`quinlan@yggdrasil.com` (Daniel Quinlan): 많은 부분에서 수정과 추가를 해주었다.

`nat@nat@nataa.fr.eu.org` (Nat Makarevitch): mrproper, tar -p, 이외 많은 것

`boldt@math.ucsb.edu` (Axel Boldt): 통신상으로 커널 구성 선택 사항에 대한 설명들을 수집했고, 그 목록을 제공했다.

`lembark@wrkhors.psyber.com` (Steve Lembark): 다중 부팅을 제안

`kbriggs@earwax.pd.uwa.edu.au` (Keith Briggs): 약간의 수정과 의견

`rmcguire@freenet.columbus.oh.us` (Ryan McGuire): 'make'할 수 있는 것 추가

`dumas@excalibur.ibp.fr` (Eric Dumas): 프랑스어로 번역

`simazaki@ab11.yamanashi.ac.jp` (Yasutada Shimazaki): 일본어로 번역

`jjamor@lml.ls.fi.upm.es` (Juan Jose Amor Iglesias): 스페인어로 번역

`mva@sbbs.se` (Martin Wahlen): 스웨덴어로 번역

`jzp1218@stud.u-szeged.hu` (Zoltan Vamosi): 헝가리어로 번역

`bart@mat.uni.torun.pl` (Bartosz Maruszewski): 폴랜드어로 번역

`donahue@tiber.nist.gov` (Michael J Donahue): 철자 수정. "얇게 썬 빵 대회" 우승자

`rms@gnu.ai.mit.edu` (Richard Stallman): 무료 문서 개념/배포 공고

`dak@Pool.Informatik.RWTH-Aachen.DE` (David Kastrup): NFS 에 관한 것

`esr@snark.thyrsus.com` (Eric Raymond): 다양한 조그마한 것들

나에게 질문들과 문제들의 메일을 보내준 사람들도 많은 도움이 되었다.

12.4 저작권, 라이센스, 그리고 다른 모든 것들