KERNELD MINI-HOWTO

Version 1.7, last updated: July 19, 1997 by Henrik Storner (*storner@osiris.ping.dk* <mailto:storner@osiris.ping.dk>)

번역: 1998년 10월 3일, 정재영(*cychong@metro.telecom.samsung.co.kr* <mailto:cychong@metro.telecom.samsung.co.kr>)

이 문서의 가장 최근 버전은 http://eolicom.olicom.dk/ storner /kerneld-mini-HOWTO.html 에서 구할 수 있다. http://eolicom.olicom.dk/ storner/kern.html를 참조하면 여러가지 mini-HOWTO 사이의 변경된 내용의 목록을 발견할 수 있을 것이다. Credits 만일 이 문서에서 틀린 점을 발견하면 나에게 알려주길 바란다. 다음의 사람들이 이 mini-HOWTO에 기고한 사람들이다. * Bjorn Ekwall < *bj0rn@blox.se* <mailto:bj0rn@blox.se>> * Ben Galliart < *bgallia@luc.edu* <mailto:bgallia@luc.edu>> * Cedric Tefft < *cedric@earthling.net* <mailto:cedric@earthling.net>> * Brian Miller <*bmiller@netspace.net.au* <mailto:bmiller@netspace.net.au>> * James C. Tsiao <*jtsiao@madoka.jpl.nasa.gov* <mailto:jtsiao@madoka.jpl.nasa.gov>> 이 mini-HOWTO를 읽은 사람들이 보내주는 격려와 제안에 매우 감사한다.

목 차

제 1 절 kerneld 란 무엇인가??

kerneld는 Ejorn Ekwall에 의해 커널 개발 버전 1.3에서 소개된 기능이다. 이것은 2.0과 2.1대의 커널에 모두 포함되어 있다. 이 kerneld는 커널의 "모듈" - 다시 말하면 디바이스 드라이버, 네트워크 드라이버, 파일시스템 등등 - 을 사용자가 직접 modprobe나 insmod를 사용하지 않고, 사용자가 해당 모듈을 액세스할 때 자동적으로 로드될 수 있도록 한다. 그리고 아직 표준 커널과 통합되지는 않았지만 몇 가지 놀라운 기능들을 가지고 있다.

- 기본 스크린 브랭커 대신 사용자 프로그램을 동작시킬 수 있어, 다른 스크린 세이버를 사용할 수 있다.

- 스크린 블랭커 지원과 유사하게 표준 콘솔의 벨 소리를 전혀 다른 것으로 바꿀 수 있다. kerneld는 두 개의 독립된 엔티티로 구성된다.
- 리눅스 커널이 어떤 작업에 필요한 데몬에게 신호(request)를 보낼 수 있게 한다.
- 커널로부터의 신호를 만족시키기 위해 어떤 프로그램들이 로드되어야 하는지 알아내는 사용자 수준의 데몬.

이 두개 부분이 모두 kerneld가 동작하는데 반드시 필요한 기능들이다. 즉 이 중 어느 하나가 제대로 설정되어 않으면 동작하지 않는다.

제 2 절 왜 kerneld 를 사용하길 원하는가?

kerneld를 사용하는데는 몇 가지 이유가 있다. 여기서 언급하는 것들은 나의 경우를 말한 것이다. 다른 사람들은 또 다른 이유를 가지고 있을 것이다.

- 만일 조금씩 다른 사양을 갖는 여러 가지 시스템에서 동작할 커널을 만들어야 한다면 - 예를 들면 다른 종류의 네트워크 카드 - 사용자는 매 시스템마다 다른 커널을 만들 필요없이 하나의 커널을 만들고 여러 가지 모듈을 만들 수 있다.
- 모듈은 개발자들이 좀 더 쉽게 테스트할 수 있다. 개발자는 특정 드라이버 를 로드하거나 언로드하기 위해 시스템을 리부트할 필요가 없다.(이것은 단순히 kerneld에서 로드된 것 뿐만 아니라 모든 모듈에 적용된다)
- kerneld는 커널의 메모리 사용량을 줄여준다. 즉 다른 어플리케이션을 돌리는데 좀 더 많은 메모리를 사용할 수 있다. 커널이 사용하는 메모리는
- 절대로* 스웝 아웃(Swap Out, 역자 주: 사용자에게 돌려준다는 의미)되지 않는다. 그러므로 만일 커널을 컴파일 할 때 사용하지 않는 드라이버에게 100kb의 메모리를 할당했다면 이것은 단지 RAM을 낭비한 셈이다.
- 내가 사용하는 어떤 것들은 - 예를 들면 ftape 플로피 테이프 드라이버나 iBCS - 오직 모듈의 형태로만 이용할 수 있다. 그러나 나는 매번 이 모듈 이 필요할 때마다 로딩하는 번거로움을 겪고 싶지 않다.
- 리눅스 배포본을 만드는 사람들은 284가지의 다른 부트 이미지를 만들고 싶어하지 않는다. 대신 사용자 자신이 가진 하드웨어에 필요한 드라이버를 로드하면 된다. 이러한 인스톨 방법은 레드햇 4.0에서 사용된다.

물론 사용자가 이러한 기능을 사용하고 싶은 않은 이유도 있을 것이다. 예를 들어 모든 드라이버가 포함된 하나의 커널 이미지가 필요할 지도 모른다. 이런 경우에는 이 문서를 읽는 것은 시간 낭비일 것이다.

제 3 절 필요한 것들은 어떻게 구하는가?

커널에서의 kerneld 지원은 리눅스 커널 1.3.57에서 도입되었다. 만일 사용자가 이것보다 이전 버전의 리눅스 커널을 가지고 있는데 kerneld를 사용하고 싶다면 커널의 버전을 높여야 한다. 모든 리눅스 사이트들은 커널 소스를 가지고 있다 - 가능하면 가장 최근의 안정 버전인 2.0으로 업그레이드 하길 권한다.

sunsite.unc.edu/pub/Linux/kernel/v2.0/linux-2.0.29.tar.gz `<ftp://sunsite.unc.edu/pub/Linux/kernel/v2.0/linux-2.0.29.tar.gz>`

tsx-11.mit.edu/pub/linux/sources/system/v2.0/linux-2.0.29.tar.gz `<ftp://tsx-11.mit.edu/pub/linux/sources/system/v2.0/linux-2.0.29.tar.gz>`

funet.fi/pub/Linux/PEOPLE/Linus/v2.0/linux-2.0.29.tar.gz <ftp://funet.fi/pub/Linux/PEOPLE/Linus/v2.0/linux-2.0.29.tar.gz>

사용자 수준의 데몬은 modules-1.2.8 과 새로운 modules-2.0 패키지에 포함 되어 있다. 이것들 역시 커널 소스와 같은 장소에서 구할 수 있지만 아래가 공식적인 장소이다.

sunsite.unc.edu/pub/Linux/kernel/v2.0/modules-2.0.0.tar.gz <ftp://sunsite.unc.edu/pub/Linux/kernel/v2.0/modules-2.0.0.tar.gz>

tsx-11.mit.edu/pub/linux/sources/sbin/modules-2.0.0.tar.gz <ftp://tsx-11.mit.edu/pub/linux/sources/sbin/modules-2.0.0.tar.gz>

funet.fi/pub/Linux/tools/modules-2.0.0.tar.gz <ftp://funet.fi/pub/Linux/tools/modules-2.0.0.tar.gz>

주의: 만일 사용자가 최근의 2.1 개발자 버전에서 모듈 로드 기능을 이용하고 싶으면 최근의 modutils- (modules- 이 아니라) 패키지를 사용해야 한다. 그러나 아래에 있는 2.1대의 커널과 모듈을 사용할 때 발생할 수 있는 문제점에 대해 확인해야 한다.

제 4 절 설정은 어떻게 하는가?

먼저 필요한 것들을 구한다: 적당한 버전의 커널과 최신 버전의 modules-utilities. 그리고 modules-utilities를 인스톨해야 한다. 이 작업은 매우 간단하다. 소스를 푼 다음 make install을 실행한다. 이것은 genksysm, insmod, lsmod, modprobe, depmod, kerneld 들을 컴파일하여 /sbin에 인스톨한다. 그리고 사용자가 리눅스를 부팅할때 필요한 작업을 이루어지도록 스타트업 스크립트에 필요한 내용을 추가하는 것이 좋다. 다음과 같은 내용을 /etc/rc.d/rc.S파일 (슬랙웨어의 경우) 이나 /etc/rc.d/rc.sysinit (데비안, 레드햇, 칼데라와 같은 SysVinit를 사용하는 경우)에 추가한다.

```
# Start kerneld - this should happen very early in the
# boot process, certainly BEFORE you run fsck on filesystems
# that might need to have disk drivers autoloaded
if [ -x /sbin/kerneld ]
then
        /sbin/kerneld
fi
# Your standard fsck commands go here
# And you mount command to mount the root fs read-write
# Update kernel-module dependencies file
# Your root-fs MUST be mounted read-write by now
if [ -x /sbin/depmod ]
then
        /sbin/depmod -a
fi
```

첫번째 부분은 kerneld 자신을 실행시키는 것이다. 두번째 부분은 시작할때 'depmod -a'를 호출한다. depmod 프로그램은 사용 가능한 모든 모듈의 리스트를 만들고 그것들 간의 의존성을 분석하여 한 모듈이 로드되기 전에 어떤 모듈이 로드되어야 하는지 알게 해준다.

주의 : 최근 버전의 kerneld는 GNU dbm 라이브러리, libgdbm과 링크하는 것을 옵션으로 갖는다. 만일 module-utilites를 컴파일할때 이 기능을 활성화시킨 경우 libgdbm이 없으면 kerneld는 동작하지 않는다. 예를 들면 /usr가 다른 파티션 에 존재하고 /usr이 마운트되기 전에 kerneld가 동작하는 경우이다. 이런 경우 /usr/lib의 libgdbm을 /lib로 옮기거나 kerneld를 정적으로 링크할 것을 권장한다.

다음, 커널 소스를 풀고 configure를 하고 커널을 만든다. 만일 이러한 과정을 이전에 한 적이 없다면 Linux 커널 소스의 루트 디렉토리에 있는 README를 반드시 읽어봐야 한다. make config를 할 때는 다음과 같은 질문에 대해 관심을 기울여야 한다.

Enable loadable module support (CONFIG_MODULES) [Y/n/?] Y Loadable module support를 선택하지 않으면 kerneld가 로드할 모듈이 없게 된다. 그러므로 Yes를 선택해야 한다.

Kernel daemon support (CONFIG_KERNELD) [Y/n/?] Y 이것 역시 반드시 필요한 것이다. 이제는 커널에서 모듈로 설정할 수 있는 다양한 것들이 제시될 것이다. 예를 들면

Normal floppy disk support (CONFIG_BLK_DEV_FD) [M/n/y/?] 여기서 M을 선택하면 해당 기능이 모듈로 컴파일된다. 일반적인 시스템을 부트할때 필요한 디바이스 - 하드디스크 드라이버, 루트 파일 시스템을 위한 드라이버만 커널에 포함하면 된다. 나머지는 모드 모듈로 사용할 수 있다.

'make config'를 끝내면 'make dep', 'make clean', 'make zImage' 혹은 'make zlilo', 'make modules' 그리고 'make modules_install'를 차례로 실행한다. 'make zImage'는 새로운 커널 이미지를 arch/i386/boot/zImage 디렉토리에 둘 것이다. 이것을 부트 이미지를 두는 곳이나 LILO에서 설치하는 곳에 복사 하면 된다.

커널을 설정하고 빌드하는 방법에 대해 좀 덜 알고 싶으면 *comp.os.linux.answers* `<http://comp.os.linux.answers>`에 주기적으로 포스트되고 sunsite.unc.edu의 /pub/Linux/docs/HOWTO의 Kernel-HOWTO를 참고한다.

```
<bf/kerneld 시험해 보기/
   이제 새로운 커널로 리부트한다. 시스템이 부트되면 'ps -ax'를 사용하여
   다음과 같은 내용이 있는 지 확인해 본다.

    PID TTY STAT  TIME COMMAND
     59  ?  S     0:01 /sbin/kerneld

   kerneld를 사용해서 좋은 점 한가지는 한번 사용자가 커널을 만들고 데몬을
   설치한 다음에는 환경설정하는데 별다른 작업이 필요하지 않는다는 것이다.
   시험삼아 모듈로 빌드하려고 하는 드라이버중 하나를 사용해 본다. 아마도
   추가 설정 없이도 잘 동작할 것이다.  플로피 드라이버를 모듈로 빌드하고
   DOS 플로피 디스크를 드라이브에 넣었다.

   osiris:~ $ mdir a:
   Volume in drive A has no label
   Volume Serial Number is 2E2B-1102
   Directory for A:/
   binuti~1 gz       1942 02-14-1996  11:35a binutils-2.6.0.6-2.6.0.7.diff.gz
   libc-5~1 gz      24747 02-14-1996  11:35a libc-5.3.4-5.3.5.diff.gz
          2 file(s)        26689 bytes

   플로피 드라이버가 동작한다. 이것은 매번 플로피 디스크를 사용할 때마다
   kerneld에 의해 자동으로 로드된다.
   실제로 플로피 모듈이 로드되었는지 알아보기 위해 /sbin/lsmod를 실행해보면
   현재 로드된 모듈들을 볼 수 있다.

   osiris:~ $ /sbin/lsmod
   Module:        #pages:  Used by:
   floppy            11    0 (autoclean)

   "(autoclean)"은 모듈이 일 분이상 사용되지 않으면 자동적으로 kerneld에
   의해 제거된다는 것을 의미한다. 그러므로 11 페이지만큼의 메모리(한
```

```
페이지가 4kB이므로 총 44kB)가 오직 내가 플로피 드라이브를 사용할때만
사용된다는 것을 의미한다. 만일 내가 일 분이상 플로피를 사용하지 않으면
플로피 드라이브 모듈에 할당된 메모리는 해제된다. 이러한 기능은 사용자
어플리케이션이 메모리 부족으로 고생할 때 매우 유용하다.
```

제 5 절 어떤 모듈이 로드될 지 kerneld는 어떻게 하는가?

흔한 형태의 모듈에 대해서는 kerneld가 정보를 가지고 있겠지만 때로는 kerneld가 커널이 요청한 것을 어떻게 처리해야하는 지 모를 수 있다. 한가지 이상의 모듈이 존재하는 CD-ROM 드라이버나 네트워크 드라이버가 이러한 경우 이다.

커널로부터 kerneld 데몬에게 전달되는 요구사항은 다음 중 하나이다.

- 블럭 디바이스 드라이버(a block-device driver)
- 문자 디바이스 드라이버(a character-device driver)
- 바이너리 포맷(a binary format)
- tty line discipline
- 파일 시스템(a filesystem)
- 네트워크 디바이스(a network device)
- 네트워크 서비스(a network service (e.g. rarp))
- 네트워크 프로토콜(a network protocol (e.g. IPX))

kerneld는 어떤 모듈이 로드되어야 하는지를 /etc/conf.modules 파일을 검사 해서 결정한다. 여기에는 두가지 항목이 있는데 경로명(로드된 모듈 파일이 있는 곳)과 별명(alias: 로드될 모듈의 이름)이 그것이다. 만일 이 파일이 존재하지 않는다면 다음과 같이 만들 수 있다.

/sbin/modprobe -c | grep -v '^path' >/etc/conf.modules 만일 기본 패스에 다른 패스를 추가하려면, 추가하려는 패스 뿐만 아니라 모든 기본 패스도 추가해야 한다. 이것은 /etc/conf.modules의 패스가 modprobe가 알고 있는 기본 패스를 완전히 대체하기 때문이다. 대개 사용자는 이러한 경로를 직접 손으로 입력하고 싶어하지 않고 내부적으로 이러한 작업이 이루어지길 바랄 것이다.

alias나 다른 옵션을 직접 추가하고 싶다면 /etc/conf.modules에 추가한 내용은 modprobe가 이미 알고 있는 것에 추가될 것이다. 만일 사용자가 alias나 옵션을 재정의하였다면, /etc/conf.modules의 내용이 기본 설정을 대신할 것이다.

블럭 디바이스(Block devices)

```
'/sbin/modprobe -c'를 실행하면 kerneld가 알고 있는 모듈들의 목록과 어떤
신호들이 이것과 연관되어 있는지 표시된다. 예를 들어 플로피 드라이버를
로드하도록 하는 신호는 메이저 번호 2를 갖는 블럭 디바이스이다.
osiris:~ $ /sbin/modprobe -c | grep floppy
alias block-major-2 floppy
왜 블럭 메이저 번호가 2일까? 그 이유는 /dev/fd* 플로피 디바이스들이 메이저
번호 2를 사용하고 블럭 디바이스들이기 때문이다.
osiris:~ $ ls -l /dev/fd0 /dev/fd1
brw-rw-rw-   1 root     root       2,   0 Mar  3  1995 /dev/fd0
brw-r--r--   1 root     root       2,   1 Mar  3  1995 /dev/fd1
```

캐릭터 디바이스(Character devices)

캐릭터 디바이스도 비슷한 방법으로 처리된다. 예를 들어 ftape 플로피 드라이버는 메이저 번호 27을 갖는다.

```
osiris:~ $ ls -lL /dev/ftape
crw-rw----   1 root     disk      27,   0 Jul 18  1994 /dev/ftape
```

그러나 기본적으로 kerneld는 ftape 드라이버를 알지 못한다. 이것은 '/sbin/modprobe -c'의 결과 목록에 들어있지 않다.

그러므로 kerneld가 ftape 드라이버를 로드하도록 하려면 /etc/conf.modules에 다음과 같은 내용을 추가해야 한다.

```
alias char-major-27 ftape
```

네트워크 디바이스(Network devices)

사용자는 'char-major-xxx' 나 'block-major-yyy'형태의 설정 대신 디바이스의 이름을 사용할 수 있다. 이같은 방법은 네트워크 디바이스에 특히 유용하다. 예를 들어 eth0로 동작하는 ne2000 네트워크 카드는 다음과 같은 방법으로 로드할 수 있다.

```
alias eth0 ne
```

만일 드라비어에 옵션을 인자로 넘기려면 - 예를 들면 네트워크 카드가 사용할 IRQ 등 - 'options'를 추가한다.

```
options ne irq=5
```

이것은 kerneld가 NE2000 드라이버를 다음과 같이 로드하도록 한다.

```
/sbin/modprobe ne irq=5
```

물론 실제 옵션은 사용자가 로드하려는 모듈에게만 사용된다.

바이너리 포맷(Binary formats)

바이너리 포맷 역시 유사한 방법으로 처리된다. 사용자가 kerneld가 어떻게 로드할 지 모르는 프로그램을 실행할 때마다 kerneld는 'binfmt-xxx'와 같이 요구한다. 이때 xxx는 바이너리 포맷의 첫번째 몇개 바이트로부터 결정되는 숫자이다. 그러므로 ZMAGIC (a.out) 바이너리 포맷을 위한 binfmt_aout을 로드하기 위한 kerneld 설정은

```
alias binfmt-267 binfmt_aout
```

ZMAGIC 파일에 대한 매직 넘버(/etc/magic 참조)는 267이다.(/etc/magic 파일을 보면 숫자가 0413으로 표시되어 있을 것이다. 그러나 이것은 kerneld가 8진수를 사용하기 때문이고, 8진수 413은 10진수 267이다. a.out 실행 파일에 대해서는 조금씩 다른 3가지 변종이 존재한다. 그러므로 이것들 모두를 지원하는 binfmt_aout을 로드하려면

```
alias binfmt-264 binfmt_aout  # pure executable (NMAGIC)
alias binfmt-267 binfmt_aout  # demand-paged executable (ZMAGIC)
```

alias binfmt-204 binfmt_aout # demand-paged executable (QMAGIC)
a.out, Java 그리고 iBCS 이진 포맷은 kerneld에 의해 자동적으로 인식되므로 특별한 설정은 필요없다.

Line disciplines (slip, cslip and ppp)

Line disciplines은 "tty-ldisc-x"와 같이 요청되는데 이때 'x'는 대개 1 (SLIP) 이거나 3 (PPP)이다. 이 두가지 모두 kerneld에 의해 자동적으로 인식된다.

PPP의 경우 ppp를 위해 bsd_comp 데이터 압축 모듈을 로드하고 싶으면 다음과 같은 두 줄을 /etc/conf.modules에 추가해야 한다.
alias tty-ldisc-3 bsd_comp
alias ppp0 bsd_comp

네트워크 프로토콜 패밀리 (IPX, AppleTalk, AX.25)

몇몇 네트워크 프로토콜 역시 모듈로 로드될 수 있다. 커널은 "net-pf-X"와 같은 형식으로 kerneld에게 프로토콜 패밀리에 대해 요청한다. 이때 X는 사용하고자 하는 패밀리를 표시하는 번호이다. 예를 들면 net-pf-3은 AX.25 이고, net-pf-4는 IPX 그리고 net-pf-5는 AppleTalk이다.(이 번호들은 리눅스 소스 파일 include/linux/socket.h 에 정의된 AF_AX25, AF_IPX 값에 의해 결정된다. (역자 주: 인터넷에서 흔히 사용되는 IP 프로토콜값은 2 이다) 그러므로 IPX 모듈을 자동으로 로드하려면 다음과 같은 내용을 /etc/conf.modules에 입력해야 한다.

alias net-pf-4 ipx

! 확인 !
이제 아래에서 볼 내용은 부팅할때 정의되지 않은 프로토콜 패밀리와 관련된 에러 메시지가 발생할 때 어떻게 대처하는지에 대해 알아본다.

파일 시스템

파일 시스템에 대한 kerneld 신호는 파일 시스템 타입의 이름이다. 흔히 "iso9660" 파일 시스템 타입을 갖는 CD-ROM 파일 시스템에 대한 isofs 모듈을 로드하려면 아래와 같이 한다.

alias iso9660 isofs

특별한 설정을 필요로하는 디바이스들

몇몇 디바이스들은 alias만을 사용해서는 디바이스를 모듈로 사용할 수 없다.
<itemize>
<item> major 10번을 갖는 캐릭터 디바이스들
<item> SCSI 디바이스들
<item>특별한 초기화를 필요로하는 디바이스들
<itemize>

char-major-10 : Mice, watchdogs and randomness 하드웨어 디바이스는 대개 메이저 디바이스 번호를 통해 구별된다. 예를 들면 ftape은 char-major-27 이다. 그러나 /dev의 내용을 보면 캐릭터 메이저 번호 10을 갖는 항목을 볼 수 있는데 이것은 매우 다양한 디바이스에 의해 사용된다.

- 다양한 종류의 마우스(bus mice, PS/2 mice)
- Watchdog devices
- The kernel 'random' device
- APM (Advanced Power Management) interface

분명히 이러한 디바이스들은 하나가 아닌 서로 다른 모듈에 의해 제어된다. 그러므로 이것들에 대한 kerneld 설정은 메이저 번호와 마이너 번호를 사용해야 한다.

alias char-major-10-1 psaux # For PS/2 mouse alias char-major-10-130 wdt # For WDT watchdog

이것을 사용하기 위해서는 커널의 버전이 1.3.82이상이어야 한다. 이전 버전은 kerneld 에게 마이너 번호를 넘겨주지 않아 kerneld가 어떤 디바이스 모듈을 로드할지 알지 못한다. **드라이버 로드하기: scsi_hostadpater 항목**

```
SCSI 디바이스를 위한 드라이버는 SCSI 호스트 어댑터(예를 들면 Adaptec 1542)
와 사용자가 사용하는 SCSI 디바이스 타입 - 예를 들면 하드 디스크, CD-ROM
그리고 테이프 드라이브 - 에 대한 드라이버로 구성된다. 이 모든 것들은 모듈로
로드될 수 있다. 그러나 예를 들어 Adaptec card에 연결된 CD-ROM을 엑세스
할 때 커널과 kerneld는 SCSI CD-ROM을 지원하기 위해 sr_mod 모듈을 로드해야
한다는 것만을 알고 CD-ROM이 연결된 SCSI 컨트롤러에 대해서는 모른다. 그러므로
SCSI 컨트롤러를 지원하는 어떤 모듈을 로드해야 할 지는 모른다.
이러한 문제점을 해결하려면 /etc/conf.modules의 SCSI 드라이버 모듈항목에서
다양한 SCSI 컨트롤러 모듈중에 어떤 것을 kerneld가 로드해야 할지를 지정한다.

    alias scd0 sr_mod               # sr_mod for SCSI CD-ROM's ...
    alias scsi_hostadapter aha1542  # ... need the Adaptec driver

이러한 기능은 커널 1.3.82 이상의 버전에서만 가능하다.
위와 같은 작업은 오직 하나의 SCSI 컨트롤러만 있을때 제대로 동작한다. 만일
하나 이상의 컨트롤러가 있다면 좀 더 복잡한 작업을 해야 한다.
일반적으로 다른 호스트 어댑터용 드라비어가 이미 설치되어 있다면 kerneld는
SCSI 호스트 어댑터에 대한 드라이버를 로드할 수 없다. - 반드시 두개의 드라
이버를 커널에 포함시키던가(모듈이 아니라) 직접 모듈을 로드해야 한다.
그러나 kerneld가 여러개의 SCSI 드라이버를 로드하도록 하는 방법이 있다.
아래의 내용은 James Tsiao씨의 아이디어이다.:

kerneld가 두번째 SCSI 드라이버를 로드하도록 하려면 modules.dep를 직접
수정하여 의존성을 만들면 된다. 단지 아래와 같이 하면 된다.
  /lib/modules/2.0.30/scsi/st.o: /lib/modules/2.0.30/scsi/aha1542.o
kerneld는 st.o를 로드하기 전에 aha1542.o를 로드할 것이다. 집에 있는
내 컴퓨터에 위와 같은 설정이 있고 이것은 테이프와 CD-ROM 그리고 일반적인
SCSI 디바이스들(generic scsi devices)에서 잘 동작한다. 이때 단점은
'depmod -a'가 제대로  의존성을 검사하지 못하므로 사용자가 직접 추가하고,
부팅할때 'depmod -a'를 사용해서는 안된다는 거이다. 그러나 한번 설정이 되면
kerneld는 자동으로 aha1542.o를 로드한다.
```

이 때 주의해야 할 것은 이러한 방법은 서로 다른 두 개의 컨트롤러의 다른
SCSI 디바이스에만 제대로 동작한다는 것이다. 예를 들면 한개의 컨트롤러에
하드 디스크가 있고, 다른 컨트롤러에 CD-ROM 드라이브나 테이프 혹은
일반적인 SCSI 디바이스가 있는 경우이다.

모듈을 로드하는 것만으로 충분하지 않을때 : 설치 후 항목(The 'post-install' entry)

때대로 모듈을 로드하는 것만으로도 충분하지 않을 때가 있다. 예를 들어
모듈로 컴파일된 사운드 카드가 있을때, 초기값을 소리값을 지정하는 것이
편한다. 이때 문제점은 설정값들이 다음에 모듈이 로드될때 사라진다는 것이다.
여기 Ben Galliart(<url url="mailto: bgallia@luc.edu" name="bgallia@luc.edu" >)씨가 알려준 해결법이다:
마지막 방법은 setmix-0.1 패키지 (<url url="http://sunsite.unc.edu/pub/Linux/apps
/sound/mixers/setmix-0.1.tar.gz" name="sunsite.unc.edu/pub/Linux/apps
/sound/mixers/setmix-0.1.tar.gz" >) 를 설치하는 것이다.

그리고 다음과 같은 내용을 /etc/conf.modules에 추가한다:

```
post-install sound /usr/local/bin/setmix -f /etc/volume.conf
```

이것은 사운드 모듈이 로드된 후 kerneld가 "post-install sound"라고 쓰여진
명령어를 실행시킨다. 그러므로 사운드 모듈은 '/usr/local/bin/setmix -f
/etc/volume.conf'의 명령으로부터 소리 볼륨값을 갖게 된다.

이러한 방법은 tunelp 프로그램으로 환경 설정을 해야하는 lp 모듈과 같은
다른 모듈에도 유용한다.

```
post-install lp tunelp <options>
```

kerneld가 이러한 옵션을 인식하려면 1.3.69f 나 그 이후 버전의 kerneld가
필요하다.

주의: 이전 버전의 mini-HOWTO 는 kerneld가 모듈을 제거하기 전에 실행시키는 "pre-remove" 라는 명령어에 대해 언급하고 있다. 그러나 이것은 제대로 동작 하지 않아, 다음 kerneld 릴리즈에서는 제거될 것으로 예상된다. 모듈 설정과 관련된 모든 이슈들은 현재 진행중이고 이 글을 읽게되는 시점에는 이미 많은 변화가 일어났을 것이다.

제 6 절 kerneld의 작업 엿보기

만일 모든 방법을 시도해보았는데도 커널이 kerneld에게 요구하는 작업이 무엇인지 모를때는, kerneld가 받는 신호를 확인해 보는 방법이 있다. 그 결과 /etc/conf.modules에 어떤 내용을 넣어야 하는 알 수 있게 된다. 이때 사용하는 것이 kdstat 유틸리티이다.

이 멋진 프로그램은 모듈 패키지에 포함되어 있지만 기본적으로 컴파일되거나 설치되지 않는다. 이것을 빌드하려면

cd /usr/src/modules-2.0.0/kerneld make kdstat

이제 kerneld 가 어떤 작업이 이루어지고 있는지 화면에 표시하도록 하려면 kdstat debug를 실행시킨다. 그러면 kerneld는 콘솔에 현재 어떤 일이 일어나고 있는지 메시지를 표시한다. 이때 사용자가 사용하고 싶어하는 명령어를 실행시키면 kerneld request들을 볼 수 있을 것이다. 이제 이것을 /etc/conf.modules에 추가하고 모듈에 대한 alias를 설정한다. 디버깅을 끝내려면 '/sbin/kdstat nodebug'을 실행시킨다.

제 7 절 kerneld의 특별한 사용

어떤 사람들은 스크린세이버 모듈을 설정하는 방법을 알고 싶어할 것이다. 모듈 패키지의 'kerneld/GOODIES' 디렉토리에는 스크린세이버를 위한 커널 패치를 포함하고 있다. 이것들은 공식적인 커널의 일부는 아니다. 그러므로 사용자가 직접 커널을 패치하고 커널을 빌드해야 한다.

패치를 설치하려면 "patch" 명령어를 사용한다.

cd /usr/src/linux patch -s -p1 </usr/src/modules-2.0.0/kerneld/GOODIES/blanker_patch

그리고 새로운 커널을 다시 빌드한 후 설치한다. 스크린 세이버 모두로 바뀌면 kerneld는 "/sbin/screenblanker"를 실행시킨다. 이 파일은 사용자가 좋아하는 스크린 세이버를 동작시키는 스크립트이다. 커널이 스크린 세이버 기능을 멈추려면 /sbin/screenblanker를 실행중인 프로세스에게 SIGQUIT 시그널을 보낸다. 사용자가 만든 쉘 스크립트나 스크린세이버는 이것을 포착한 후 종료해야 한다. 이때 원래 텍스트 모드로 복원해야 한다는 것을 명심해야 한다.

제 8 절 흔히 발생하는 문제점들

왜 ifconfig를 실행시켰을때 "Cannot locate module for net-pf-X"라는 메시지가 표시되는가?

```
 커널 1.3.80에서 네트워킹 코드는 프로토콜 패밀리(예를 들면 AX.25나
 AppleTalk)를 모듈로 로드할 수 있도록 변경되었다. 이를 위해 새로운 kerneld
 request: net-pf-X를 추가되었다.(이때 X는 프로토콜을 가리키는 번호, 이 번호에
 대해 알려면 /usr/src/linux/include/linux/socket.h를 확인해 본다)
 불행하게도 ifconfig는 때때로 이 메시지를 트리거시켜 많은 사람들이 시스템을
 부팅할때나 loopback 디바이스를 설정하기 위해 ifconfig를 실행시킬때 이러한
 메시지를 보게 된다. 물론 이 메시지는 결코 위험한 것이 아니고 단지 다음과
 같은 내용을 /etc/conf.modules에 추가하여 메시지가 나타나지 않게 할 수 있다.

      alias net-pf-3 off      # Forget AX.25
      alias net-pf-4 off      # Forget IPX
      alias net-pf-5 off      # Forget AppleTalk

 물론 IPX를 모듈로 사용하고자 한다면 위와 같은 내용을 추가해서는 안된다.
kerneld를 실행한 후 ppp 연결을 활성화시키면 컴퓨터가 갑자기 느려진다.
```

이러한 내용에 대한 보고가 몇 번 있었다. 이는 kerneld와 tkPPP 스크립트 사이에 부적절한 동작이 일어나는 것으로 추측된다. tkPPP 스크립트는 몇몇 시스템에서 PPP 연결을 설정하고 모니터링하는데 사용되는데 이 스크립트는 ifconfig가 동작하는 동안 루프를 돈다. 이것은 kerneld로 하여금 net-pf-X 모듈을 찾게 하고, 시스템 로드를 높게 유지시켜 시스템로그에 "Cannot locate module for net-pf-X"라는 메시지를 쏟아내게 한다. 아직까지는 tkPPP를 사용하지 않거나 다른 방법으로 PPP 연결을 모니터링하도록 하는 방법 밖에 없다.

kerneld가 SCSI 드라이버를 로드하지 않는다.

```
 /etc/conf.modules에 SCSI 호스트어댑터를 추가한다. 이 글의 위쪽에 있는
 scsi_hostadapter에 대한 내용을 확인해 본다.


modprobe가 "gcc2_compiled"가 정의되어 있지 않다고 투덜거린다.
```

이것은 2.6.0.9 이후 버전의 binutils에서만 보여지는 module-utilities의 버그 이다. 그리고 이러한 내용은 binutils 의 releasenote에 문서화되어 있다. 그 글을 읽어보는 것이 좋다. 혹은 modules-2.0.0과 같이 이 버그를 고친 module-utilities로 업그레이드한다.

사운드 드라이버가 볼륨과 같은 설정값을 자꾸 잊어버린다.

```
모듈에 대한 설정값은 모듈이 로드될때 모듈 자체에 저장된다. 그러므로 kerneld
가 모듈을 자동으로 언로드하면 사용자가 지정한 설정값은 날아가 버리고
다음에 모듈을 로드하면 기본값을 복원된다.
이러한 문제점을 해결하기 위해 모듈이 자동으로 로드된 후 프로그램을 실행시켜
모듈의 설정값을 바꾸도록 할 수 있다. 이 글의 "post-install" 항목을 읽어본다.
```

```
DOSEMU가 몇몇 모듈을 필요로 하는데 kerneld가 이것들을 어떻게 로드하도록
하는가?
```

이것은 불가능하다. 어떠한 dosemu 버전도 - 공식 버전이건 개발버전이건- kerneld를 통한 dosemu 모듈을 로드하는 것을 지원하지 않는다. 그러나 만일 2.0.26이나 그 이후의 커널을 사용하고 있다면 더 이상 어떠한 dosemu 모듈이 필요하지 않다. 단지 dosemu를 0.66.1로 업그레이드하면 된다.

왜 "Ouch, kerneld timed out, message failed"와 같은 메시지가 출력되는가?

```
커널이 kerneld 에게 신호(request)를 보내면 1초 내에 응답 신호
(Acknowledgement)가 올 것으로 예상한다. 만일 kerneld가 이 응답 신호를
보내지 못한다면 이 메시지는 로그파일에 기록된다. 커널은 신호(request)를
다시 보낼 것이고 결국은 해결 될 것이다.

이것은 대개 시스템에 로드가 심할 때 일어난다. 이것은 kerneld가 사용자 모드
프로세스이기 때문에 시스템의 다른 프로세스처럼 스케쥴링된다. 그러므로 로드가
심할때는 kerneld 타임 아웃이 되기 전에 커널에게 응답 신호(acknowledgement)를
제 때 보내지 못할 수 있다.

만일 로드가 심하지 않을 때 이런 일이 발생하면 kerneld를 다시 실행시켜 본다.
(kerneld 프로세스를 종료시키고 /usr/sbin/kerneld 명령을 다시 실행시킨다)
만일 문제가 계속 발생한다면 <url url="mailto: linux-kernel@vger.rutgers.edu" name=" linux-kernel@vger.rutgers.edu" >에게 메일을 보낸다.
이때 문제점을 포스팅하기 전에 kernel과 kerneld가 최신 버전인지 확인해야
한다.
```

```
kerneld가 파일 시스템 모듈을 로드할때까지 mount가 기다리지 않는다.
```

---

mount(8) 명령어가 kerneld가 파일시스템 모듈을 로드할 때까지 기다리지 않는 다는 보고가 있었다. lsmod는 kerneld가 모듈을 로드했다고 할 때, mount를 반복하면 마운트를 할 수 있을 것이다. 이것은 module-utilities 1.3.69f 버전 의 버그로 보이는데 이것은 Debian 사용자에게 영향을 미친다. 최신 버전의 module-utilities를 사용하면 문제점을 해결할 수 있다.

kerneld가 ncpfs 모듈을 로드하지 못한다.

```
ncpfs 유틸리티를 컴파일할때 반드시 -DHAVE_KERNELD옵션을 주어야 한다.
ncpfs Makefile을 살펴본다.
```

```
kerneld가 smbfs 모듈을 로드하지 못한다.
```

아마도 smbmount 유틸리티의 버전이 오래된 것일 것이다. 최신 버전(0.10이나 그 후)을 가져와 사용한다.

tsx-11.mit.edu/pub/linux/filesystems/smbfs/ `<ftp://tsx-11.mit.edu/pub/linux/filesystems/smbfs/>`

나는 모든 것을 모듈로 빌드했는데 시스템이 부팅되지 않는다. kerneld가 루프 파일 시스템 모듈을 로드하지 못한다. 모든 것을 모듈로 만들 수는 없다. 커널은 루트 파일 시스템을 마운트할 수 있도록 충분한 드라이버를 가지고 있어야 하고 kerneld를 실행시키는데 필요한 프로그램들을 가지고 있어야 한다. 그러므로 다음과 같은 것들은 모듈로 만들 수 없다.

- 루트 파일 시스템이 있는 하드디스크용 드라이버
- 루프 파일 시스템 드라이버
- nit를 위한 binary foramt loader, kerneld 그리고 다른 프로그램들

[실은 위에서 말한 내용이 사실은 아니다. 1.3.x 이후 버전과 2.x 대의 커널은 initial ram-disk를 LILO라 LOADLIN에 의해 로드할 수 있도록 하고, 부트 과정에서 모듈들을 이 "디스크"에서 로드할 수 있다. 자세한 정보는 커널 소스의 Documentation/initrd.txt 파일에서 얻을 수 있다.]

kerneld가 부트할때 로드되지 않고 libgdbm에 대해 불평한다.

새로운 버전의 kerneld는 실행하는데 GNU dbm 라이브러리 libgdbm.so를 요구한다. 대개 /usr/lib에 이 파일이 설치되지만 혹시나 /usr 파일 시스템을 로드하기 전에 kerneld를 실행시킬 수도 있다. 이 경우 kerneld는 부트할때 실행되지는 않지만, 시스템이 부팅된 후 사용자가 직접 kerneld를 실행시켜도 괜찮다. 해결 방법은 /usr 이 마운트된 후 kerneld 스타트 업을 옮기거나 gdbm 라이브러리 를 루트 파일 시스템 - 예를 들면 /lib - 로 옮긴다.

"Cannot load module xxx"라는 메시지가 표시되지만, 커널을 빌드할때 xxx를 포함시키지 않았다.

```
슬랙웨어는(혹은 다른 배포본도) 다양한 모듈에 대해 /etc/rc.d/rc.modules에서
modprobe를 실행시킨다. 이때 어떤 모듈을 modprobe할 것인가는 원래 커널 설정에
따른다. 그러므로 커널을 재설정하면서 rc.modules에 포함되지 않은 모듈을
설치하였다면 에러를 발생시키게 된다. 그러므로 rc.modules에서 사용하지 않는
모듈은 주석으로 바꾸고, 새로 추가한 모듈의 주석 표시를 제거하여 kerneld가
필요한 모듈이 어떤 것인지 제대로 알 수 있도록 한다.
```

```
커널과 모듈을 다시 빌드하였는데 여전히 부팅할때 unresolved symbols라는
메시지가 표시된다.
```

아마도 사용자는 커널을 재설정하고 다시 빌드하여 몇몇 모듈을 제외시켰을 것이다. 그 결과 더 이상 사용되지 않는 모듈이 /lib/modules 디렉토리에 남아 있게 된다. 이런 문제를 해결하는 가장 쉬운 방법은 /lib/modules/x.y.z 디렉토리를 지우고 커널 소스 디렉토리에서 'make modules_install'을 다시 하는 것이다. 이때 버전을 바꾸지 않고 커널을 재설정에만 문제가 발생한다. 만일 새로운 커널 버전으로 바꾸었을때 이러한 에러를 만난다면, 또 다른 이유에서 문제가 발생한 것이다.

리눅스 커널 2.1을 설치했는데 모듈이 로드되지 않는다.

```
리눅스 커널 2.1이 현재 개발 커널 버전이다. 그러므로 많은 것들이 수시로
변할 것이다. 가장 크게 바뀐 중 하나는 모듈을 처리하는 방법과 커널과
```

```
모듈이 메모리에서 로드되는 곳이다. 현재 Richard Henderson은 모듈 커널
개발을 책임지고 있다.
요약해서 말하자면, 만일 커널 2.1 에서 모듈을 사용하고 싶다면 반드시

  * 시스템에서 업그레이드해야할 패키지를 설명한 Documentation/Changes 파일
    을 읽는다.

  * 최신 modutils 패키지를 사용한다. 아래의 곳에서 구할 수 있다.

    <url url="ftp://ftp.redhat.com/pub/alphabits/ or the mirror site at" name="ftp.redhat.com/pub/alphabits/ or the mirror site at" >
    <url url="ftp://tsx-11.mit.edu/pub/linux/packages/alphabits/" name="tsx-11.mit.edu/pub/linux/packages/alphabits/" >

만일 커널 2.1 대의 커널에서 모듈을 사용하고 싶다면 최소한 2.1.19 버전의
커널을 사용할 것을 권한다.


Dial-on-demand 네트워킹은 ?
```

kerneld는 원래 on demand 다이얼 업 네트워크를 지원하도록 되어 있다: 연결되지 않은 상황에서 네트워크로 패킷을 보내면 kerneld가 PPP나 SLIP 연결을 만들기 위해 /sbin/request_route 스크립트를 실행시킨다. 이러한 방법은 그리 좋은 아이디어가 아니라고 밝혀졌다. 리눅스 네트워킹을 담당하고 있는 Alan Cox는 리눅스 커널 메일링 리스트에 다음과 같이 썼다. Request-route 기능은 진부한 것이고 이미 망가진 것이어서 커널 2.1.x 대에 서 제거되었다.

request-route 스크립트와 kerneld를 사용하는 대신 *www.dna.lth.se/˜erics/diald.html* `<http://www.dna.lth.se/~erics/diald.html>` 에서 구할 수 있는 Eric Schenk's의 diald 패키지를 설치할 것을 충고한다.

Copyright Messages