리눅스 커널 QnA

이번 호에서는 리눅스의 핵심이라 할 수 있는 커널에 대해 리눅서들이 주로 궁금해하는 부분을 중심으로 QnA 형식으로 엮어 보았다. 이 글을 읽고 커널이 결코 해커만의 전유물이 아니라 리눅서라면 누구든 쉽게 친해질 수 있다는 것을 알 수 있는 계기가 되었으면 한다.

오늘과내일 넷센터 홍석범(antihong@tt.co.kr)

문) 커널과 리눅스는 어떤 관계입니까?

답) 가끔 리눅스를 사용하시는 분께 운영하는 리눅스 서버의 버전이 어떻게 되냐고 물으면 "6.2" 또는 "7.1"이라고 말씀하십니다. 그러나 엄격히 말한다면 이는 틀린 대답입니다. 왜냐하면 이는 단지 일부 배포판 업체나 기관에서 커널을 기반으로 GNU 유틸리티와 각 벤더의 유틸리티를 포함하여 패키징을 한 배포판에 붙인 임의의 버전일 뿐이며 실제로 리누스가 관리하며 발표하는 리눅스(커널) 와는 직접적으로 관계가 없기 때문입니다. 리눅스는 그 자체로, 일반적으로 이야기하는 넓은 범위의 리눅스라는 운영 체제의 핵심 부분이라 할 수 있는 "커널" 을 뜻하는 것이며, 따라서 정확하게 이야기한다면 리눅스란 커널을 뜻하는 것입니다. 그러나 일반적으로는 커널 및 각종 패키지를 포함한 각 벤더의 배포판을 그냥 리눅스라 칭하기도 합니다.

문) 왜 리눅스 커널은 C 와 어셈블리로 만들어졌나요?

답) 여러가지가 이유가 있습니다.

첫번째는 리눅스 커널의 창시자인 리누스가 Linux 를 쓰기 시작할 때 처음 사용한 것이 386 의 Minix OS 와 gcc 이었기 때문입니다. 그리고 이론적인 측면에서 보면 어떠한 OS 든 전체 혹은 부분적으로 커널은 항상 어셈블리어로 쓰여졌기 때문입니다. 이는 어셈블리가 하드웨어에 가장 가깝고 또한 밀접한 언어이기 때문입니다. 또 다른 한편으로 OS 디자이너들은 전통적으로 UNIX 를 포함하여 많은 OS 커널루틴을 C 로 만들어 왔기 때문이기도 합니다.

문) /usr/src 에서 리눅스 커널 소스를 압축 해제하면 linux-x.y.z 대신 linux 로 풀려서 linux 를 linux-x.y.z 로 rename 한 후 linux 로 링크를 하는데, 이렇게 하는 이유가 무엇인가요? 그리고 아예 처음부터 linux-x.y.z 로 풀리지 않는 이유가 있나요?

답) 물론 꼭 이렇게 할 필요는 없습니다. 단지 /usr/src 디렉토리에 여러 버전의 리눅스 커널 소스를 가지고 설치를 할 때 각 소스간에 버전의 구별을 쉽게 하기 위해서입니다. 그래서 linux 를 현재 사용중인 linux-버전으로 링크를 하여 사용하는 것이지요.

그리고 압축 해제시 처음으로 풀리는 디렉토리 이름이 linux 인 것은 리누스가 이 방식을 선호하기 때문입니다. 이와 같이 하였을 경우 패치를 할 때마다 매번 디렉토리 이름을

rename 할 필요가 없고 더 쉽고 간단히 패치등의 작업을 할 수 있기 때문이지요.

문) 커널 컴파일을 잘못 하면 부팅이 아예 안 될 수도 있다던데요. 꼭 해야 합니까?
답) 그럴 수도 있고 그렇지 않을 수도 있습니다. 어떤 분은 커널 컴파일을 해 보았냐 해 보지 않았느냐에 따라 리눅스 초급과 중/고급을 나누는 기준이 될 정도로 커널 컴파일은 리눅스 중급자로 가기 위한 필수 조건이라고 생각합니다. 다시 원론으로 돌아와서 이야기하면 커널 컴파일 과정은 결론적으로 새로운 하나의 커널 이미지를 생성하는 것인데, 부팅시 여러 커널 이미지중 어떤 커널 이미지로 부팅할 것인지 선택할 수 있습니다. 따라서 만약 새롭게 생성한 커널 이미지로 부팅이 되지 않는다면 이전의 커널 이미지를 선택하여 재부팅을 하시면 되는 것입니다.(물론 lilo 설정에서 복수개의 이미지로 부팅할 수 있도록 설정해야 합니다.) 부팅시 LILO: 상태에서 부팅할 커널 이미지를 선택하는 부분을 상기해 보시면 이해가 가실 겁니다. 따라서 커널 컴파일을 한다고 부팅이 아예 안 될 수도 있다는 것은 틀린 말입니다. 그러니 걱정하지 마시고 마음껏 커널 컴파일에 도전해 보세요.
그리고 커널 컴파일 및 업그레이드를 하는 이유는 크게 “보안” 과 “성능 향상” 이라는 두 가지로 설명할 수 있습니다.

(1) 시스템 보안 (Security) 을 위해 .
아래는 얼마전 실제로 필자가 운영하는 리눅스 서버에서 테스트한 결과입니다.

```
[user1 @www user1]$ id
uid=500(user1) gid=500(user1) groups=500(user1),10(wheel)
[user1 @www user1]$ gcc exploit.c -o exploit
[user1 @www user1]$ ./exploit
bug exploited successfully.
enjoy!
bash# id
uid=0(root) gid=0(root) groups=500(user1),10(wheel)
bash# uname -a
Linux kernel.tt.co.kr 2.2.18 #1 SMP Tue Nov 21 01:22:41 KST 2000 i686 unknown
bash# exit
```

이 버그는 리눅스 배포판과 관계없이 커널 버전 2.2.19 이전의 모든 시스템에 통용되는 버그로서 통상적으로 가장 많이 사용하는 레드햇 6.x 계열을 설치 후 커널 업그레이드를 하지 않았다면 초기 배포판 버전대로 커널이 2.2.16 이므로 위 버그가 바로 적용이 되어, 일반 유저 권한만 있으면 위 소스를 gcc로 컴파일 후 실행하여 바로 root 권한을 획득할 수 있게 되는 치명적인 문제가 있습니다. 위와 같이 리눅스 커널에서 치명적인 보안 문제가

발생하는 일은 그리 흔한 일은 아니지만 이러한 경우가 가끔 발생하므로 시스템의 보안을 위해 지속적으로 커널 패치 또는 업그레이드를 하시는 것이 좋습니다.

(2) 확장된 기능과 최적화된 성능을 위해

리눅스를 서버로 사용하거나 특정한 목적으로 사용한다면 서버의 성능을 높이기 위해 CPU를 Dual 로 한다거나 메모리를 추가로 장착하기도 합니다. 그러나 실제 물리적인 성능을 높이기 보다는 시스템의 핵심인 커널과 각종 데몬의 설정을 최적화함으로써 더욱 가시적인 향상을 기대할 수 있는 경우가 많이 있습니다. 일반적인 리눅스 배포판의 커널은 배포판을 이용하는 유저들이 어떤 하드웨어를 사용하는지 그리고 어떤 목적으로 사용할 것인지 알 수 없으므로 여러 다른 종류의 하드웨어와 상당수의 설정들을 지원 하도록 설정되어 배포되므로 커널 이미지의 크기가 당연히 커질 수 밖에 없고, 사용하지 않는 여타 기능들이 많이 추가 되었으므로 시스템에 최적화되지 못해 당연히 성능이 떨어지게 돕니다. 혹 이전에 커널 컴파일을 한 적이 없다면 /usr/src/linux 로 이동한 후 make menuconfig 를 입력후 Network device support ---> Ethernet (10 or 100Mbit) ---> 를 선택해 보시기 바랍니다. 실제 사용하지도 않는 모든 이더넷 카드가 모듈로 선택되어 있고 이것들이 현재 시스템의 커널모듈에 올라가 있습니다. 이렇듯 불 필요한 기능을 커널에 포함시키거나 모듈로 설정하였을 경우 시스템이 전반적으로 무거워지고 많은 메모리를 소모하게 되지요. 또한 정작 자신에게 필요한 기능은 커널에 포함되지 않는 경우가 있어 특수한 기능을 이용하거나 자신에게 필요한 기능만을 선택하여 사용할 수 있도록 하기 위해 커널 컴파일이 필요하기도 합니다. 이를테면 리눅스를 이용해 라우터로 이용하거나 클러스터링을 구성 하려면 반드시 커널 패치 및 컴파일이 필요하게 되지요.

문) 커널 컴파일 하는 방법을 알려주세요?

답) 각각의 절차에 대한 자세한 설명은 http://kernel.pe.kr/ 이나 http://kldp.org/리눅스_커널/커널_컴파일/ 를 참고하세요.

```
[root@work /usr/src]# rm -f linux
[root@work /usr/src]# tar zxvfp linux-2.4.8.tar.gz
[root@work /usr/src]# mv linux linux-2.4.8
[root@work /usr/src]# ln -s linux-2.4.8 linux
[root@work /usr/src]# cd /usr/include/
[root@work /usr/include]# rm -rf asm linux scsi
[root@work /usr/include]# ln -s /usr/src/linux/include/asm-i386    asm
[root@work /usr/include]# ln -s /usr/src/linux/include/linux    linux
[root@work /usr/include]# ln -s /usr/src/linux/include/scsi    scsi
[root@work /usr/include]# cd /usr/src/
```

[root@work /usr/src/linux]# **make menuconfig**

[root@work /usr/src/linux]# **make dep; make clean; make bzlilo; make modules; make modules_install**

/etc/lilo.conf 설정 변경후

[root@work /boot]# **/sbin/lilo**

문) 커널 컴파일은 하지 않고 그냥 편하게 rpm 으로 커널 업그레이드만 하는 방법을 알려 주세요.

답) 이 방법은 그리 권장하는 방법은 아니므로 가급적이면 커널 소스를 가지고 와서 직접 커널 컴파일을 하시기 바랍니다. 하지만 커널 컴파일이 익숙하지 않은 상태에서 단순히 커널 버전만을 업그레이드하기를 원할 경우 이 방법을 사용하시면 됩니다.

먼저 레드햇의 경우 ftp://updates.redhat.com/ 에서 각 버전에 맞는 최신 버전의 Kernel RPM 을 가져옵니다. (레드햇에서 패키징하는 Kernel RPM 은 배포한 이전 커널 버전에 치명적인 문제가 있을 경우에만 업데이트된 RPM 을 패키징하므로 일반적으로 소스로 배포되는 최신 버전보다는 버전이 낮습니다.) 여기에서는 레드햇 7.X 에서 Kernel-2.4.3-12 버전을 기준으로 하여 설명하도록 하겠습니다.

먼저 다운로드 받은 kernel 을 rpm -Uvh 옵션을 이용하여 업그레이드 합니다.

만약 의존성 에러가 날 경우에는 rpm - Uvh -nodeps 옵션을 주어 설치를 하면 됩니다.

```
# rpm -Uvh kernel-source-2.4.3-12.i386.rpm
Preparing...           ########################################### [100%]
1:kernel-source        ########################################### [100%]
# rpm -Uvh kernel-headers-2.4.3-12.i386.rpm
Preparing...           ########################################### [100%]
1:kernel-headers       ########################################### [100%]
# rpm -Uvh kernel-2.4.3-12.i386.rpm
Preparing...           ########################################### [100%]
1:kernel               ########################################### [100%]
```

이렇게 하고 나면 /boot/ 에 vmlinuz 및 System.map 등 몇 가지 링크가 자동으로 생성됩니다. 그리고 이제는 mkinitrd 명령어를 통해 RAMDISK 파일을 생성할 차례입니다.

```
# mkinitrd /boot/initrd-2.4.3-12.img 2.4.3-12
```

mkinitrd 실행시 형식은 /boot/initrd-(커널버젼).img 커널버젼 입니다.

그리고 부팅시 업데이트된 새로운 커널로 부팅할 수 있도록 LILO 에 등록하면 됩니다.

/etc/lilo.conf 를 읽어 이전 버전의 내용을 그대로 복사한 다음, initrd 부분과 vmlinuz 부분의 설정을 업그레이드 한 버전으로 변경합니다. 적절히 수정한 후 /sbin/lilo

명령어를 실행하고 재부팅을 하면 됩니다. 다시 한번 말씀드리지만 rpm 의 커널은 많은 옵션들이 모듈로 선택되어 있기에 커널 이미지의 크기가 크고(커널 컴파일후 /boot 디렉토리에 있는 양 커널 이미지의 사이즈를 서로 비교해 보세요.) 커널은 메모리에 올라가 작동하므로 커널의 크기가 큰 만큼 메모리도 많이 소모하게 됩니다. 따라서 rpm 으로 설치하셨다 하더라도 가급적 커널 컴파일을 하여 시스템에 최적화 하실 것을 권장합니다.

문) 커널 업그레이드를 하려고 하는데, 어떤 커널 버전을 받아야 하나요?

현재 대부분의 유저들이 사용하는 커널 버전은 2.2.X 버전이거나 일부 2.4.X 일 것입니다. 2.2.18 을 포함한 이전 버전의 경우 위에서 설명한 바와 같이 심각한 보안 버그가 있으므로 반드시 2.2.19 이상 버전으로 업그레이드를 하시기 바라며, 아울러 잘 아시다시피 2.4.X 버전부터는 엔터프라이즈급에 맞도록 커널코드가 상당히 개선되었으므로 가능하시다면 2.4.X 대의 커널을 사용하시는 것이 좋습니다.

그리고 커널 버전은 두 번째 번호가 짝수인가 홀수인가에 따라 안정 버전과 개발 버전으로 나뉘어 지는데, 개발 버전의 경우 테스트 단계이므로 일반 유저들은 특별한 경우가 아니라면 2.짝수.X 형태의 안정 버전을 사용하셔야 합니다. 아울러 2.4.X 일부 버전에도 리눅스가 작동중 갑자기 멈추어 버리거나 특정한 프로세스가 작동하지 않는등의 버그가 있으므로 가급적이면 이러한 문제를 해결한 가장 최신 버전을 받으시는 것이 좋습니다.

현재 최신 버전은

```
# finger @finger.kernel.org
  [zeus.kernel.org]
  The latest stable version of the Linux kernel is:     2.4.8
  The latest prepatch (alpha) version *appears* to be:  2.4.9-pre2
```

로 확인 가능하며 8 월 14 일 현재 2.4.8 이 최신 안정 버전입니다. 최신 버전의 커널은 ftp://ftp.kernel.org/ 나 한국의 미러 사이트인 ftp://ftp.kr.kernel.org/ 에서 다운로드 가능합니다.

문) 커널 컴파일을 하기 전에 /usr/include 에서 asm, linux, scsi 등을 링크하라고 하던데, 링크를 하는 이유는 무엇인가요?

답) /usr/include 디렉토리는 표준 C 라이브러리 헤더 파일이 있는 디렉토리 입니다.따라서 /usr/include 디렉토리에서 압축을 해제한 커널 소스에 있는 디렉토리로 링크를 하는 이유는 소스내에 #include 를 하는 파일들은 기본적으로 현재 소스의 위치와는 관계없이 /usr/include 디렉토리에서 해당 링크 파일을 찾기 때문입니다. 예를 들어 커널 소스내에 #include <linux/config.h> 와 같이 파일을 참조하고 있다면 /usr/include/linux 디렉토리에 있는 config.h 라는 헤더 파일을 찾게 되는 것이지요. 실제로 커널 소스의 헤더 파일들은 커널 소스(/usr/src/linux/)의 include 디렉토리 아래에 있습니다.

문) 커널 컴파일 옵션 선택시 모듈로 하는 것이 좋나요? 아니면 정적으로(Static 하게) 포함시키는 것이 좋나요?

답) 이는 어떤 옵션을 선택하고 어떤 목적으로 시스템을 사용하느냐에 따라 다릅니다.
모듈 기능은 매우 유용하기는 하지만 그렇다고 해서 모든 옵션에 모듈을 적용하는 것은 바람직하지 않습니다. 사용하지 않는 기능은 과감히 선택하지 말고 꼭 필요한 기능만을 선택하되 자주 사용하는 기능은 모듈보다는 정적으로 선택하여 커널에 포함하는 것이 좋습니다. 왜냐하면 모듈은 커널에서 직접 지원되는 것보다는 느리기 때문이지요. 이를테면 이더넷 카드(랜카드)의 경우 모듈로 선택하는 것보다 커널에 정적으로 포함시키는 것이 약 10-15%정도 더 나은 성능을 발휘하는 것으로 알려져 있습니다.

문) 커널 컴파일시 특히 주의하여야 하는 옵션에 대해 알려주세요.

답) 커널 컴파일시 각각의 옵션에 대해 잘 알고 자신의 하드웨어 사양에 따라 꼭 필요한 기능만 선택하는 것이 중요하지만 특히 아래의 설정을 잘못 했을 경우 성능을 크게 좌우하고 심할 경우에는 정상적으로 부팅이 안 될 수도 있으므로 주의하셔야 합니다.

(1) 적절한 CPU 타입 선택

Pentium 3, AMD K6, Cyrix, Pentium 4, Intel 386, DEC Alpha, PowerPC등 시스템의 CPU 에 맞는 타입을 선택하셔야 합니다. 만약 cpu 선택 메뉴에서 386을 사용하면 모든 CPU 에서 사용 가능하나 Pentium-III 를 선택하면 펜티엄이나 486의 경우에는 사용할 수 없게 되니 주의하시기 바랍니다. 즉, 자신의 현재 사양보다 낮은 타입을 선택할 수는 있으나 현재 사양보다 높은 타입은 선택할 수 없습니다. 현재의 시스템에서 어떤 CPU 를 사용하는지 잘 모르겠으면 cat /proc/cpuinfo 로 확인하면 됩니다.

"Pentium-Classic" -- 인텔 펜티엄의 경우 선택
"Pentium-MMX" -- 인텔 펜티엄 MMX의 경우 선택
"Pentium-Pro" -- 인텔 펜티엄 프로/셀러론/펜티엄II의 경우 선택
"Pentium-III" -- 인텔 펜티엄 III 의 경우 선택.

(2) SMP 지원 여부

현재 시스템이 Single CPU 인지 아니면 Dual CPU 인지에 따라 선택 사항이 다소 달라집니다. Single CPU 인데 SMP 를 선택하거나, Dual CPU 인데 SMP 를 선택하지 않는 것은 시스템의 안정성이나 성능상으로 좋지 않습니다.

(3) 파일 시스템

리눅스에서 사용할 파일 시스템을 적절히 선택합니다. 이때 파일 시스템을 잘못 선택하거나 모듈로 설정할 경우 부팅이 되지 않을 수도 있으니 주의하셔야 합니다.

(4) 커널모듈을 로드할 수 있도록 하기

Loadable module support ---> 에서 Enable loadable module support 을 선택합니다. 이 옵션으로 현재 작동하고 있는 리눅스 시스템에서도 insmod 와 rmmod 등을 이용하여 운영체제가 작동하고 있는 상태에서도 동적으로 모듈을 올리거나 내릴 수 있습니다.

문) 레드햇 리눅스를 사용하고 있는데, 커널 컴파일을 하려고 보니 /usr/src/redhat 만 있고 /usr/src/linux 는 없네요. 뭐가 문제죠?

답) /usr/src/redhat 는 RPM 패키징를 하기 위해 필요한 디렉토리이며 커널과는 직접적으로 관계가 없습니다. /usr/src 에 linux 디렉토리가 없는 경우라면 CD 를 이용하여 리눅스 설치시 커널 소스를 선택하지 않고 설치하셨기 때문이므로 단순히 CD에서 kernel-source rpm 을 설치하시거나 커널 소스를 다운로드 받아 설치하시면 linux 디렉토리에 커널 소스사 설치됩니다.

문) 방화벽을 설정하기 위해 iptables 를 사용하려고 합니다.
커널을 어떻게 설정해 주어야 사용가능하나요?

답) iptables 는 커널 레벨에서 작동하는 패킷 필터링툴 이므로 커널에서 지원이 되어야 사용가능합니다. Iptables 를 사용하기 위해서는

Networking options ---> [*] Network packet filtering (replaces ipchains) 를 선택하면 하단에 IP: Netfilter Configuration ---> 메뉴가 생깁니다. 이 부분을 선택한 후

<*> Connection tracking (required for masq/NAT)
<*> FTP protocol support
<*> IP tables support (required for filtering/masq/NAT)
<*> Packet filtering
<*> Full NAT

를 선택하면 됩니다. 그러나 보다 확장된 기능의 iptables 를 사용하려면 모든 옵션을 다 선택하시면 됩니다.

그리고 iptables 실행파일은 별도로 설치하여야 하는데,

http://netfilter.samba.org/ 에서 iptables 소스 파일을 다운로드받아 압축해제후 make ; make install 로 설치를 해도 되고 http://rpmfind.net/ 에서 각자의 버전에 맞는 rpm 형태의 iptables 를 다운로드 받아 설치하여도 됩니다.

문) 정식 커널과 알랜콕스의 - ac 패치 버전과는 어떤 차이가 있나요?

답) 알랜콕스는 리누스와 함께 핵심적인 커널 개발자중의 한 사람으로 커널 개발자 사이에서는 리눅스 커널의 어머니라고도 칭합니다.

커널 메일링 리스트에서도 알랜콕스의 제안과 글을 많이 볼 수 있는데, 일반적으로

알랜콕스의 패치를 리누스의 커널에 대한 테스트 버전정도로 생각하면 됩니다.
리누스는 커널에 대해 다소 보수적(?)이고 조심스러워 수많은 테스트를 거쳐 완전히 안정하다고 확신이 되었을 경우에만 커널에 포함시키는 반면 알랜은 다소 새로운 개념과 새로운 기능의 드라이버, 더욱 공격적인 코드를 포함시키는 성향이 있습니다.
알랜콕스의 패치를 통해 안정하다고 판단되었을 경우 알랜은 리누스에게 이 패치를 보내어 정식 커널에 포함시키도록 합니다. 알랜콕스를 포함하여 여러 커널 개발자들의 패치를 적용하려면 http://www.kernel.org/pub/linux/kernel/people/ 사이트를 참고하면 됩니다.
그리고 리누스에 의해 발표되는 Pre 패치도 있는데, 이는 다음 버전의 공식 커널 발표전에 실제로 잘 동작하는지 테스트해 보려는 기능들에 대한 패치입니다.
이를테면 8 월 14 일 현재 안정버전은 2.4.8 인데, 프리패치는 2.4.9-pre2 입니다.

문) 패치 버전을 이용하여 커널을 패치 하는 방법을 알려주세요?
답) 커널을 업그레이드하고자 할 때 현재의 커널 버전에서 완전히 새로운 커널 소스를 가지고 와 새롭게 컴파일하여 설치하는 방법도 있지만 20 여메가가 넘은 압축 소스를 일일이 다운로드하여 커널을 재설치하는 것이 부담이 될 때에는 간단히 패치를 하는 방법도 있습니다.
커널 패치를 하는 방법에 대해서는 각각 리눅스커널 패치, 알랜콕스의 패치,
프리 패치에 대해 설명을 드리겠습니다.

현재의 커널 버전이 2.4.5 인데 2.4.7 로 업그레이드를 하고자 할 때에는
먼저 커널 2.4.5 소스가 있는 /usr/src/linux 로 이동하여 패치 파일의 압축을 해제한 후
[root@work /usr/src/linux]# **patch -p1 < patch-2.4.6**
로 2.4.5 을 2.4.6 로 패치합니다. 그리고
[root@work /usr/src/linux]# **patch -p1 < patch-2.4.7**
로 2.4.6 을 2.4.7 로 추가 패치합니다.
그리고 현재의 2.4.7 에 알랜콕스의 2.4.7-ac8 를 패치하고자 한다면
[root@work /usr/src/linux]# **patch -p1 < patch-2.4.7-ac8**
로 2.4.7 를 2.4.7-ac8 로 패치하면 됩니다.

일반적인 커널 패치는 이전 버전까지의 패치가 모두 들어있다는 가정에서 만들어진 패치이므로 2.4.5 에서 2.4.7 로 패치를 하려면 2.4.5 에서 바로 2.4.7 로 패치할 수 없고
2.4.5->2.4.6->2.4.7 순으로 순서대로 패치 하여야 합니다.
반면에 알랜콕스의 패치는 이와 달리 상위 버전의 패치가 하위 버전의 패치를 포함하고 있으므로 한번만 패치를 하시면 됩니다. 따라서 2.4.7-ac8 에서 2.4.7-ac9 로 패치를 하려면 이미 2.4.7-ac9 에는 ac8 의 패치를 포함하고 있으므로 바로 패치할 수 없고 먼저 역패치를 한 후 다시 패치하여야 합니다.
[root@work /usr/src/linux]# **patch -p1 -R < patch-2.4.7-ac8**

위와 같이 -R 옵션을 주어 역패치를 하면 전혀 패치가 가해지지 않은 2.4.7 원본 커널 소스로 만들어 줍니다.

[root@work /usr/src/linux]# **patch -p1 < patch-2.4.7-ac9**

이후 위와 같이 깨끗한 상태의 2.4.7 에서 2.4.7-ac9 로 패치하시면 됩니다.

그리고 2.4.8 에서 2.4.9-pre1 으로 패치하려면

[root@work /usr/src/linux]# **gzip -d patch-2.4.9-pre1.gz**

와 같이 patch-2.4.9-pre1.gz 를 압축 해제한 후

[root@work /usr/src/linux]# **patch -p1 < patch-2.4.9-pre1** 를 적용하면 됩니다.

문) lilo.conf 에 보면 initrd 라는 것이 있는데, initrd 는 무엇인가요?
없는 경우도 있던데, 꼭 필요한가요?

답) initrd 란 부팅시 초기화에 필요한 루트 디스크의 이미지를 나타내는 것으로 특히 SCSI 하드 디스크를 사용할 때 중요합니다. 만약 SCSI 하드를 사용할 경우 SCSI 부분을 커널 옵션에서 모듈로 설정했다면, 부팅시 램디스크 이미지를 생성하지 않았을 경우 SCSI 하드를 인식할 수 없게 되므로 결국은 부팅을 할 수가 없게 됩니다. 이러한 이유로 램디스크의 이미지를 만드는 것이지요. CD 를 이용하여 리눅스 운영체제를 처음 설치할 때는 이러한 경우를 대비하여 lilo.conf 에 initrd 부분이 추가되는 것입니다. 물론 SCSI 가 없거나 SCSI 부분을 커널 옵션에서 모듈로 설정하지 않고 정적으로 포함하였을 경우에는(즉 메뉴에서 * 로 선택하였을 경우) 굳이 initrd 설정이 필요하지 않으므로 이 부분을 삭제하셔도 됩니다. 가급적이면 커널 컴파일시 SCSI 부분을 모듈로 선택하지 말고 정적으로 포함시키기 바랍니다.

문) System.map 은 어떤 기능을 하는 것입니까? 꼭 필요한 파일인가요?

답) 만약 커널 패닉(Kernel Panic) 이 일어나거나 기타 커널과 관련된 문제가 발생할 경우에는 화면에 여러 가지 레지스터들과 그에 해당하는 16 진수와 관련된 페이지의 정보가 출력되게 됩니다. /boot/System.map 파일이 있을 경우에는 그 주소가 나타내는 함수 이름으로 변환해 주어 어떤 위치에서 어떤 원인으로 커널이 문제를 일으켰는지 쉽게 판단할 수 있도록 디버깅해 주는 역할을 합니다. 그러나 System.map 파일이 없다면 c01000cb 와 같이 전혀 이해할 수 없는 16 진수 주소들만을 보게 될 것입니다.

따라서 System.map 파일이 없이도 부팅하는 데에는 문제가 없습니다만 만일에 발생하는 문제에 대비하여 반드시 설정하실 것을 권장합니다.

문) 커널 튜닝에는 어떤 방법이 있습니까?

답) 시스템을 최적화 하거나 자신의 필요에 따라 커널 설정을 수정하여야 할 필요가 있는 경우가 있습니다. 이 때는 커널 컴파일을 하기 전에 커널 소스를 직접 수정한 후 커널 컴파일을 할 수 있는데, 이를 하드레벨 튜닝이라 하며 이를 통해 파일 오픈 개수나 처리할 수

있는 프로세스 개수등의 제한을 변경할 수 있습니다. 반면에 커널이 제공하는 파라미터값을 /proc 파일 시스템을 이용해서 부팅이 완료된 시점 후에 변경할 수 있는데, 여기서는 주로 파일 시스템과 네트웍 자원에 관련된 내용에 대해서 튜닝을 합니다. 이러한 방법을 소프트 레벨 튜닝이라 하는데, 이 방법으로 설정되는 값들은 시스템이 부팅되면서 원래의 값으로 초기화되어 부팅시 스크립트를 통해 설정되어야 되기 때문에 /etc/rc.d/rc.local 파일이나 /etc/sysctl.conf 파일에 설정하여야 합니다. 소프트 레벨 튜닝으로 설정 가능한 값은 sysctl - a 로 확인할 수 있습니다. 커널 튜닝은 시스템에 끼치는 영향이 크므로 조심스럽게 설정하셔야 합니다.

문) 잘 운영되다가 "Kernel Panic" 이라며 시스템이 멈추어 버리는 경우가 있는데, 이런 경우에는 어떻게 하여야 하나요?

답) 커널 패닉은 매우 많은 경우에 발생할 수 있습니다.

시스템의 과부하로 인한 문제일 수도 있고 CPU 나 메모리등 하드웨어의 불량으로 발생할 수도 있습니다. 따라서 커널 패닉이 발생할 경우에는 콘솔이나 로그 파일에 패닉에 대한 메시지가 남으므로 로그메시지를 근거로 아래의 사이트를 참고하여 커널 패닉의 이유를 찾아 보시기 바랍니다.(이를 위해서는 일정정도의 C 언어의 해석 능력이 있어야 합니다.)

http://linux.fh-heilbronn.de/doku/Linux/linux-2.4.7/R/panic.htmlz

예를 들어 설명하면 필자가 운영하는 한 시스템의(커널 2.4.7) 경우

```
[root@www log]# cat messages|grep panic
Jul 22 14:57:45 www kernel: Kernel panic: CPU context corrupt
Jul 25 19:25:30 www kernel: Kernel panic: CPU context corrupt
```

와 같이 2-3 일에 한번씩 "CPU context corrupt" 라는 메시지를 내면서 커널 패닉이 발생한 적이 있었는데, 이는 하드웨어와 관련하여 주로 다음과 같은 경우에 발생한다고 알려져 있습니다.

(1) CPU 를 overclocking 하였다.
(2) CPU 가 불량이다.
(3) 전압이 문제가 있거나 전원상태가 좋지 않다.
(4) 주위 온도가 높다.

커널 패닉의 구체적인 원인과 대처법에 대해서는 아래에서 설명할 커널 메일링 리스트(http://www.uwsg.indiana.edu/hypermail/linux/kernel/index.html) 에서 검색하면 도움을 받으실 수 있습니다.

문) 새로운 기능을 시험해 보려고 커널 컴파일을 하기 위해 옵션을 살펴보니 몇 가지 옵션이 보이지 않습니다. 커널 소스를 다시 받아야 하나요?

답) 커널 옵션중

Code maturity level options ---> Prompt for development and/or incomplete code/drivers 부분을 선택하셨나 확인해 보시기 바랍니다.
네트워크 드라이브나 파일 시스템등 리눅스 커널에서 지원하는 많은 기능중 일부는 아직 안정화 단계가 아닌 개발 단계에 있어서 기타 기능이나 안정성 측면에서 안정적이지 않을 경우가 않습니다. 이러한 상태를 "알파테스트 상태"라고 하는데, 위 옵션을 선택하지 않으면 이러한 단계에 있는 메뉴들이 보이지 않게 됩니다. 따라서 만약 선택하려는 메뉴가 보이지 않을 경우 이 옵션을 선택하시면 보이실 것입니다.

문) 커널 컴파일후 부팅을 하려고 하니 "Uncompressing .. Ok, booting the kernel" 메시지까지만 나오고 멈추어서 부팅이 되지 않습니다.

답) 새롭게 컴파일한 커널로 적용시 부팅이 되지 않는다면 커널 옵션에서 어딘가 실수를 하셨기 때문입니다. 위와 같은 메시지만 나오는 경우는 대부분 커널 컴파일 옵션중
Processor type and features ---> 에서 CPU Type 을 잘못 선택하셨기 때문이므로 이 부분을 살펴보시고 자신의 CPU 타입에 맞는 CPU를 선택한 후 재컴파일해 보세요.

문) 커널 컴파일후 모듈 컴파일까지 했는데, 생각해 보니 옵션을 하나 잘못 선택한 것이 있습니다. 이러한 경우 처음부터 다시 해야 하나요?

답) 그러실 필요는 없습니다. 단지 커널 옵션 수정후 make bzlilo(또는 make bzImage) 만 실행해 주신 후 루트 파티션(/) 에 새로운 커널 이미지인 vmlinuz 와 System.map 파일이 생기므로 이 파일들만 이전의 설정 파일에 덮어쓰시면 됩니다.

문) uname - r 을 실행하면 커널 버전이 나오는데, 실제 커널 버전이 아니라 커널 버전을 제 마음대로 설정하고 싶습니다. 어떻게 하면 되나요?

답) /usr/src/linux/ 디렉토리에 Makefile 이 있는데, 이 파일의 제일 윗 줄에 있는

```
VERSION = 2
PATCHLEVEL = 4
SUBLEVEL = 8
EXTRAVERSION =-3
```

부분을 원하시는대로 수정해 준 후 재컴파일을 하시면 됩니다.
참고로 위의 경우에는 커널 버전이 2.4.8-3 임을 알 수 있습니다.

문) 커널 컴파일을 하여 업그레이드를 하였는데, 새로운 버전이 아니라 예전 버전으로 부팅됩니다. 어디가 잘못 되었나요?

답) 이러한 경우는 /etc/lilo.conf 를 변경 후 /sbin/lilo 를 실행하여 변경된 내용을 LILO 에 적용하지 않고 재부팅을 하였기 때문입니다. /sbin/lilo 로 정보를 갱신해 주었는지 확인하고 아울러 /etc/lilo.conf 설정에서 새로운 커널 이미지로 설정하였는지 확인해

보시기 바랍니다.

문) 부팅시 Kernel Panic: VFS: Unable to mount fs 라는 메시지가 나오고 부팅이 되지 않습니다. 어떻게 하여야 하나요?

답) 커널 컴파일을 한 후 재부팅시 Kernel Panic: VFS: Unable to mount fs 와 같은 메세지가 나면서 부팅이 되지 않는다면 부팅 과정에서 루트 파일 시스템을 제대로 마운트 하지 못했기 때문에 발생하는 현상입니다. 이러한 경우에는 대부분이 두 가지 이유인데, 일단 부팅이 가능한 기존의 커널 이미지(linux등) 로 다시 부팅한 후 lilo.conf 에서 지정한 root= 부분이 실제 시스템의 루트 파티션과 맞는지, 그리고 사용하는 시스템의 하드 디스크(IDE 또는 SCSI) 를 커널 컴파일 옵션에서 제대로 선택하였는지 그리고 파일시스템(ext2등)도 제대로 선택하였는지 확인해 보시기 바랍니다.
특히 SCSI 를 사용한다면 SCSI 어댑터를 모듈로 설정하지 말고 반드시 * 로 설정하여 커널에 정적으로(Static 하게) 포함 시키시기 바랍니다. 부팅시 파일 시스템이 마운트 되지 않은 상태에서는 어떠한 모듈도 로드할 수가 없기 때문입니다.

문) 커널 2.2.16 에서 아파치 웹서버를 운영하고 있는데, 동시 접속자를 설정하는 MaxClients 가 256 이상으로 설정되지 않습니다. 어떤 분은 1024까지 설정하여 사용중이던데, 알아보니 커널 설정을 변경해야 한다더군요. 아파치와 커널이 어떤 관계입니까?

답) 아파치 웹서버에서 동시 접속자 수를 제한하는 MaxClients 는 기본적으로 256이 설정 가능한 최대치입니다. 이는 두 가지 이유로 제한이 되는데, 아파치 소스 자체에서 MaxClients 가 가질 수 있는 값을 256으로 제한해둔 이유도 있지만 2.2.X 대의 리눅스 커널에서도 한 유저가 생성할 수 있는 프로세스가 256개로 제한되어 있기 때문입니다. 그래서 이 제한을 변경하려면 두 가지 작업을 함께 해 주셔야 하는 것입니다.
먼저 아파치 소스인 src/include/httpd.h 에서 #define HARD_SERVER_LIMIT 256 수치를 1024 와 같이 설정을 원하는 값으로 적절히 변경한 후 아파치를 재컴파일 하여야 하고 두번째로 커널 소스인 /usr/src/linux/include/linux/tasks.h 에서 NR_TASKS와 MAX_TASKS_PER_USER 변수를 역시 재설정후 커널 컴파일을 다시 해 주셔야 합니다.
그러나 커널 2.4. 에서는 이 설정 제한이 없어졌으므로 2.4.X 에서는 아파치 소스만 변경하신 후 아파치를 재컴파일 하시면 됩니다.

문) 랜카드를 자주 변경해야 할 이유가 있어 자주 사용하는 랜카드를 모듈로 설정 하였습니다. 모듈 명령어를 이용하여 랜카드를 잡는 방법을 알려주세요.

답) lsmod , rmmod , insmod 또는 modprobe 명령어만 아시면 됩니다.
참고로 현재 로딩된 모듈을 보려면 lsmod 로 확인 가능하며 현재의 로딩된 모듈을 삭제하려면 rmmod 를, 추가하려면 insmod 를 하시면 됩니다.

현재 3Com 이더넷 카드를 사용하고 있다면 3c59x 가 로딩된 모듈이 보일 것입니다.
또는 /etc/conf.modules(커널 2.4.X 라면 /etc/modules.conf) 에 보시면 어떤 모듈이 설정되어 있는지 확인 가능합니다.
3Com 랜카드를 빼고 인텔의 EtherExpress 100 으로 변경하였다면 두 가지 방법이 있습니다.
단순히 /etc/conf.modules(커널 2.4.X 라면 /etc/modules.conf) 에 alias eth0 3c59x 대신 alias eth0 eepro100 과 같이 변경후 부팅하시면 자동으로 모듈이 로딩되어 인식하는 방법이 있습니다.
또 다른 방법은 rmmod 3c59x 로 기존의 모듈을 삭제한 후 insmod eeprp100 으로 새로운 모듈을 추가하면 모듈이 바로 로드됩니다.
또는 사용할 이더넷 카드들을 커널에 정적으로 포함하였을 때에는(즉 3Com 과 eepro100 등 사용할 이더넷 카드를 모두 선택하였을 때에는) 별도의 설정없이 부팅시 바로 인식됩니다.

문).관리하는 리눅스 서버가 50대가 넘습니다.
모든 서버에서 일일이 커널 컴파일을 해 주어야 하나요?
답) 만약 커널 컴파일을 하여야 할 시스템이 1 대가 아니라 여러 대라면 일일이 복잡한 과정을 거치는 것이 여간 번거롭지 않을 수 없습니다. 이러한 경우, 시스템의 하드웨어 구성이 완전히 똑같을 때는 커널 컴파일 과정없이 단지 vmlinuz 와 System.map 파일만 그대로 복사해서 /etc/lilo.conf 에 설정하여 부팅해도 사용이 가능합니다.
어차피 커널 컴파일이라는 복잡한 과정이 결국은 vmlinuz 라는 커널 이미지 파일을 생성하는 단계이기 때문이죠. 만약 하드웨어나 기타 설정이 다소 차이가 있을 경우에는 아래와 같이 해 주시면 시간과 노력을 절약하실 수 있습니다.
make menuconfig 에서 설정한 커널 옵션의 설정 내용은 /usr/src/linux 에 .config 라는 파일에 저장됩니다. 따라서 하드웨어 구성이 다른 시스템이 있다면 다른 리눅스 시스템의 커널 컴파일시 기존의 서버에서 이 파일을 복사하여 /usr/src/linux 디렉토리에 복사하여 make menuconfig 를 하면 기존의 선택 사항이 그대로 적용되므로 일일이 옵션을 다시 선택할 필요 없이 새롭게 커널을 컴파일 할 시스템에 해당하는 몇 가지 옵션만 변경하여 그대로 사용하는 방법을 이용할 수도 있습니다. 만약 새로운 커널 버전을 다운로드 받아 설치할 때도 이전버전의 .config 를 복사하여 사용하셔도 됩니다.

문) 커널 컴파일을 하려면 make dep 을 실행하고 이 명령어가 끝나면 make clean 이후 make bzlilo → make modules --> make modules_install 등을 순서대로 하여야 하는데, 너무 불편합니다. 한 번에 실행할 수 있는 방법이 없나요?
답) 위 명령어를 한번에 이어서 쓰시면 됩니다.
즉 # make dep; make clean; make bzlilo; make modules; make modules_install
과 같이 각각의 명령어에 세미콜론(;) 을 붙이거나 또는
make dep clean bzlilo modules modules_install 와 같이 make 뒤에 한번에 붙여서 사용하는 방법도 같은 방법입니다.

문) 커널 컴파일시 시간이 너무 오래 걸립니다. 좀 더 빨리 할 수 있는 방법은 없나요?
답) nice 를 이용하면 됩니다. nice 는 실행되는 프로세스에 우선 순위를 주는 명령어로서 -20부터 19까지 설정해 줄 수 있습니다. (-20이 가장 우선순위가 높은 것이고 19가 가장 낮은 것입니다.) 아무런 옵션을 주지 않았을 때 기본적인 프로세스 선호도는 10인데, 가장 우선 순위를 높게 하기 위해서는 - 20 을 설정해 주면 됩니다.
nice 는 비단 커널 컴파일뿐만 아니라 어떠한 프로그램을 컴파일 할 때라도 모두 적용이 가능합니다. 따라서 커널 컴파일시 다른 프로세스보다 우선 순위를 높게 하려면
nice - 20 make dep clean bzlilo 와 같이 실행하면 됩니다.
또한 멀티 CPU 일 경우에는 CPU 를 동시에 할당하기 위해 make 에 - j 옵션을 쓸 수 있습니다. 즉 CPU가 2개일 경우에는 make - j 3 dep clean bzlilo 와 같이 할 수 있습니다.
실제 커널 컴파일시 얼마나 소요가 되는지는 time 명령어를 이용하여
time make - j 3 dep clean bzlilo 와 같이 확인할 수 있습니다.

문) 커널 컴파일이 끝난후 /sbin/lilo 를 실행해 주니 error Kernel vmlinuz_2.4.4 is too big 과 같은 메시지가 뜨면서 에러가 납니다. 왜 그런가요?
답) 에러 메시지대로 커널 이미지 사이즈가 너무 크기 때문에 발생하는 에러입니다.
커널 컴파일시 bzlilo 나 bzImage 로 압축을 하여 커널 이미지의 크기를 작게 만드시기 바랍니다. 또는 커널 옵션에서 메뉴 선택시 불필요한 내용은 최대한 선택하지 말고 아울러 선택해야 하는 내용중 모듈로 설정할 수 있는 부분이 있다면 모듈로 설정한 후 재컴파일해 보시기 바랍니다. 커널 이미지를 압축해야 하는 이유에 대해서는 http://kernel.pe.kr/ 운영자이신 정원영님께서 잘 설명하신 글이 있습니다. 아래의 내용을 참고하시기 바랍니다.

우리는 커널 컴파일을 할 때 make zImage, make bzImage 이런 식으로 커널 이미지를 압축한다.(make zlilo, zdisk, bzlilo 모두 마찬가지...) 아무런 의문 없이 무의식적으로 당연히 이렇게 쓰고 있다.

여기에는 약간의 배경지식이 필요한데 간략히 설명하겠다.

우리가 흔히 아는 매킨토시의 M68 계열 processor 는 8bit 환경에서 32bit 환경으로 발전되었지만 인텔의 8086 계열은 DOS 의 대중성 때문에 바로 32bit 환경으로 가지 못하고 16bit 환경을 가지게 되었다. 대중적으로 많이 쓰이고 있는 DOS 를 계속 쓸 수 있도록 하기 위해서였다. 이러한 이유 때문에 리얼모드, 보호모드, 가상 86 모드가 생겼다.

리얼모드, 보호모드, 가상 86 모드 모두 세그먼트 레지스터와 오프셋 레지스터를 이용하여 주소를 지정하는데 이들 레지스터의 사용방법이 다르다.

리얼모드는 세그먼트 x10h + 오프셋으로 주소를 만드는데 16bit 에서 세그먼트와 오프셋의 최대 값은 FFFF 이다. 그러므로 최대로 지정할 수 있는 주소는 FFFF0 + FFFF 가 된다. 이걸 계산하면 1MB + 64KB 가 된다.

보호모드에선 주소지정 방식이 리얼모드와는 많이 다르며 페이징 등을 이용하여 32bit 모두 주소 값을 만들 수 있으므로 이론적으론 4GB 의 메모리를 이용할 수 있으며 가상 86 모드 또한 선형주소를 만들어내는 과정만 다르므로 기본 매커니즘은 보호모드와 같다.
(보호모드와 가상 86 모드의 주소 생성 법은 생략한다.)

프로그램은 code 부분과 data 부분으로 나누어지는데 리얼모드에서 프로그램의 code 부분은 반드시 위 최대 주소 지정영역 내에 있어야 한다는 것이다. 리얼모드에서는 여러 개의 프로그램이 메모리에 올라와서 수행될 수 없으며 (만약 여러 개의 프로그램이 메모리에서 수행된다면 다른 프로그램 영역을 침범 할 수 있기 때문이다.) interrupt 를 이용하는 RAM 상주 프로그램만이 메모리를 같이 차지할 수 있다.

위에서 설명한 리얼모드의 약 1MB 정도의 영역 중 Coventional Memory 가 640KB 를 차지하고 나머지는 비디오램이나 기타 디바이스가 차지한다.

그러므로 커널 이미지는 Coventional Memory 즉 640KB 내에 들어가야 하므로 커널 이미지의 크기가 640KB 보나 작아야 한다. 커널 이미지의 압축으로 이러한 제약을 부분적으로 극복하였으나 640KB 의 일부는 여러 가지 버퍼(DMA buffer)나 특정 주소가 시스템에 예약되어 있으므로 640KB 보다 더 작은 크기의 커널이 요구된다. 이러한 문제의 대안으로 커널을 Extended Memory 에 적재하는 방법을 생각해 볼 수 있다. Extended Memory 영역에 자유로이 적재하기 위해서는 보호모드를 사용해야 하는데 보호모드에서는 BIOS 와 같이 시스템이 완전히 준비되기 전의 기본적인 기능들을 사용할 수 없다. 이럴 경우 디스크를 액세스하는 자체적인 함수를 준비하여 커널을 Extended Memory 에 적재하거나 아니면 커널을 줄이는 수밖에 없는 것이다.

문) 2.2.X 커널에서는 한 파일 사이즈에 2G 제한이 있는데, 2.4.X 에서는 이 제한이 없어졌다고 알고 있습니다. 이에 관련된 설명을 부탁드립니다.

답) 이에 대해서는 이호님께서 자세하게 설명한 글이 도움이 되실 것입니다.
아래의 글을 참고하시기 바랍니다.

x86에서 2.2.x 버전의 커널까지는 파일 크기가 2GB를 넘을 수 없다는 제한이 있습니다. 이는 리눅스에서 사용하는 EXT2 파일시스템의 문제가 아니라 리눅스에서 파일 시스템을 추상화한 계층인 VFS(Virtual File System)와 glibc의 문제입니다. EXT2 파일시스템에서는 이미 파일 크기를 나타내는데 64비트를 사용하고 있기 때문에 문제가 되지 않습니다. 그러나 VFS에서는 그냥 long을 사용하고 있기 때문에 32비트 시스템에서는 2GB(32비트로는

4G까지 표시할 수 있지만, signed이기 때문에 2G가 됩니다)의 제한이 생깁니다. 그렇지만 알파같은 시스템에서는 long이 64비트이기 때문에 이런 제한은 없습니다.
커널 외에도 C Library인 glibc의 문제도 있습니다. glibc에서는 파일 크기와 offset을 나타내는데 모두 long을 쓰기 때문에 여기서도 2GB의 제한이 발생합니다.
2.4.x 커널부터는 파일 크기와 offset을 나타내는데 long long을 사용합니다. 따라서 커널 자체에서의 제한은 사라졌습니다. 프로그램에서 2GB를 넘는 파일을 사용하려면 glibc를 2.1.3 버전 이상을 사용해야 합니다. 이 버전부터 64비트 offset을 지원합니다.
2.2.x 버전에서도 커널 패치를 하면 2G를 넘는 파일을 사용할 수 있습니다. 이 패치는 LFS(Large File System)이라고 부르며 http://www.scyld.com/software/lfs.html에서 구할 수 있습니다. 여기서는 LFS와 관련된 이슈들을 다루고 있습니다.

문) 커널에 대한 가장 빠른 소식이나 정보를 알고 싶습니다.
어떤 방법이 좋을까요?

답) 가장 빠르고 확실한 방법은 커널 개발자 메일링 리스트에 가입하시는 방법입니다.
이 메일링 리스트를 통해 새로운 커널 개발에 대한 정보나 의견 및 토론, 버그리포트, 새로운 커널 발표등에 대한 정보를 빠르게 받아보실 수 있습니다. 이 메일링 리스트에 가입하는 방법은 메시지 본문에 subscribe linux-kernel your_email@your_Domain 라고 기입하여 majordomo@vger.kernel.org 로 메일을 보내시면 됩니다.
이 메일링 리스트를 통해 각국의 커널 개발자는 물론 알랜콕스나 리누스도 자주 만나실 수 있습니다. 그러나 이 메일링 리스트에 가입을 하게 되면 매일 엄청난 양의 메일을 받게 되므로 메일링 리스트 가입시 신중하게 판단하신후 가입하셔야 합니다. 메일링 리스트에서도 커널 개발에 아주 관심이 많아서 적어도 1주일에 한번씩 커널 패치를 하지 않을 것이라면 가입하지 말라고 경고(?)하고 있습니다.
해지하는 방법은 메시지 본문에 unsubscribe linux-kernel 라고 기입 후 majordomo@vger.kernel.org 로 메일을 발송하시면 됩니다.
이와 관련하여 더 자세한 안내는 http://vger.kernel.org/majordomo-info.html 를 참고하시기 바랍니다. 또한 http://www.uwsg.indiana.edu/hypermail/linux/kernel/index.html 에 접속하시면 95년부터 현재까지의 모든 메일링 리스트의 내용을 검색하거나 살펴볼 수 있으며 커널 트래픽 홈페이지인 http://kt.zork.net/ 에서는 메일링 리스트에서 논의되고 있는 내용을 매주 정리하여 제공하고 있으니 굳이 메일링 리스트에 가입하지 않아도 많은 정보를 검색하실 수 있습니다.

문) 리눅스 커널에 대해 공부하려고 합니다. 좋은 자료나 사이트 있으면 알려주세요.

답) 본 문서를 작성하면서 도움을 많이 받은 사이트들이기도 합니다.
특히 http://kernel.pe.kr/ 은 큰 도움이 되실 것입니다.

http://kernel.pe.kr/

http://kldp.org/리눅스_커널/

http://www.tux.org/lkml/

http://www.linuxhq.com/kernel/

http://members.aa.net/~swear/pedia/kernel.html