Linux + XFS HOWTO

Linux on Steroids

IngramRussel

ringram (at) gargoylecc.com

도운이

서성용

pooh (at) kldp.org

이 문서는 SGI XFS 저널링 파일시스템에서 돌아가는 리눅스 시스템을 구축하는법에 대해서 설명한다.

고친 과정

고침 v1.022001-10-08고친이 ri

약간의 주석, blockquote 그리고 ulink 태그를 추가함. "마치기" 의 명령 부분에서 에러를 수정함. libuuid.a 를 참조하여 e2fsprogs-devel 에 대한 주석을 변경.

1. 소개

1.1. 리눅스를 위한 XFS 소개

이 문서는 SGI XFS 저널링 파일시스템에서 돌아가는 리눅스 시스템을 구축하는법에 대해서 설명한다. XFS FAQ에 의하면: "XFS 는 SGI 에 의해 개발된 저널링 파일시스템이고 SGI 의 IRIX 운영 체제에서 사용된다. 지금은 GPL 하에 리눅스에서 사용할 수 있다. 그것은 매우 scalable한데, btree 를 광범위하게 사용하여 거대한 희소(sparse) 파일들과 극도로 거대한 디렉토리를 지원한다. 저널링 기능은 더이상 fsck 를 기다리거나 meta-data 손상을 걱정할 필요가 없음을 의미한다. " 본질적으로 XFS 는 SGI 가 자사의 하이엔드 서버 시스템을 위하여 고안한 파일시스템이므로, 이 문서의 부제는 "Linux on Steroids" 이다. :-)

1.2. 서문, 피드백 그리고 저자 소개

Irix 시스템 관리자 공동체의 비교적 신입 회원으로써, Irix 를 지원하기 위해 개발된 파일시

스템(물론 XFS)의 강력함에 사랑에 빠졌다. 그래서 말할 필요도 없이 나는 한동안 리눅스로의 포팅 노력을 따라왔으며 나의 리눅스 시스템을 처음부터 모두 XFS 파일시스템의 위에서 운영하는 것을 꿈꾸어왔다. 그 프로젝트는 이것이 정말 현실이 될 수 있는 지점에 이르렀다(이 글을 쓰기 전에 거의 1년 전). 그러나, 다른 프로그래밍/포팅 프로젝트의 많은 경우가 그러하듯이, 그러한 작업에 대한 문서화가 항상 바로 읽을수 있다거나 누구나 기대하듯이 따라하기가 쉽지는 않다. 이 문서는 그러한 상황을 구제하고자 하는 시도에서 작성되었다.

이 문서에 포함된 정보는 XFS 개발 메일링 리스트의 Jason Walker 와 Russell Cattelan 의 메시지, SGI Open Source Development 웹 사이트에 있는 다양한 곳에서 수집한 정보, 그리고 XFS 시스템을 직접 설치하면서 겪은 내 자신의 경험에 기반을 두고 있다.

수정할 점이나 어떠한 정보, URL이나 기타 사항이 누락되어 있다면 자유롭게 나에게 `<ringram@gargoylecc.com>`로 이메일을 보내주기 바란다. 이 HOWTO 에서 내가 보다 많은 피드백을 얻을수록 모두에게 보다 도움이 될 것이다.

이 문서의 최신 버전은 Gargoyle Computer Consulting (http://www.gargoylecc.com/Linux+XFS-HOWTO/t1.html) 에서 찾을 수 있다.

1.3. 저작권과 권리 포기

이 문서에 포함된 정보는 Russel 의 지식 한도내에선 정확하게 최선을 다해서 작성되었다. 그러나 XFS Linux 포트는 사람에 의해 쓰여졌으므로 때에 따라선 실수나 버그나 기타 문제가 발생할 가능성이 있다.

어떤 사람이나 단체도 당신의 컴퓨터에 입은 손상과 이 문서에 있는 정보를 사용함으로써 생긴 다른 손실들에 대해 책임지지 않는다.

저자는 이 문서에 있는 정보에 기반하여 취해진 행동에 의해 초래된 어떤 손상에 대해서도 책임을 지지 않는다.

2. XFS 설치를 위한 준비

2.1. Linux 2.4.x-XFS 커널 소스 내려받기

현재 XFS 를 사용할 수 있는 리눅스 커널을 얻을 수 있는 유일한 방법은 CVS 를 이용해 SGI 의 Open Source Development 사이트에서 직접 받는 것이다.

Note: 두개의 다른 트리를 이용할 수 있다:

- linux-2.4-xfs: 빠르게 이동중인 개발 트리

- linux-2.4-xfs-beta: 안정적이며 버그 수정만 되는 트리

내 경험은 개발 트리에 기반을 두고 있으나, 베타 트리에서도 모든것이 똑같이(아마도 보다 안정적으로) 동작하리라고 예상한다. 다음은 커널 소스 트리를 내려받기 위한 과정이다:

A. 보통 리눅스 커널 소스는 /usr/src 디렉토리에 설치되므로, 그 디렉토리로 이동하는 것으로 시작한다.

```
$ cd /usr/src
```

B. 다음으로, CVSROOT 환경 변수를 설정해서 그것이 적절한 cvs 서버를 가리키도록 한다.

- 만약 당신이 sh, bash, ksh 나 기타 쉘을 사용중이라면:

```
$ export CVSROOT=':pserver:cvs@oss.sgi.com:/cvs'
```

- csh나 tcsh 를 사용중이라면

```
$ setenv CVSROOT  :pserver:cvs@oss.sgi.com:/cvs
```

커널을 자주 업데이트할 계획이라면(최신의 변화를 따라잡기 위해서), 이것을 당신의 로그인 스크립트에 넣어둘 수도 있다.

C. 그리고나서 cvs 서버에 로그인하라.

```
$ cvs login  (암호는 "cvs")
```

이것은 한번만 하면 되고, cvs 에 접속할때마다 할 필요는 없다.

D. linux-2.4-xfs 를 가져온다. 첫번째로 다음처럼 한다:

```
$ cvs -z3 co linux-2.4-xfs
```

그 코드를 체크아웃 한 후에, 다음의 명령을 이용하여:

```
$ cvs -z3 update linux-2.4-xfs
```

CVS 서버에서 최신의 버전으로 당신의 사본을 업데이트 할 수 있다.

2.2. XFS 지원을 모듈로 할 것인가 커널에 컴파일해서 넣을 것인가?

리눅스 커널에 대해 XFS 지원을 모듈로 빌드하는 옵션이 이용가능하고 초기화 RAM 디스크와 lilo 설정에 대한 몇개의 추가사항으로 동작할 것이다(혹은 그렇게 나는 들었다). 나는 (아직) 이것을 시도해보지는 않았으므로, 개발 메일링 리스트의 Russell Cattelan 의 메시지를 인용하는 것 이외에 이것이 어떻게 이루어지는지에 대한 문서를 포함시키지는 않겠다.

CattelanRussell

사실 xfs 를 모듈로 실행하는 것은 매우 어려운 것은 아니다. cmd/xfs/misc 디렉토리에는 수정된 mkinitrd 가 있는데, 이것은 항상 pagebuf xfs_support 와 xfs 를 지원하는 램 디스크를 생성할 것이다.

이렇게 하고 나면 lilo.conf 파일에 initrd 행과 다음의 행을 추가한다.

```
append = "ramdisk_size=25000"
```

기본 크기는 4096 인데 xfs 를 넣기엔 작다.

이것은 내 랩탑에서의 결과이다.

```
punch[12:57am]-=>mount
/dev/ide/host0/bus0/target0/lun0/part8 on / type xfs (rw,noatime)
none on /proc type proc (rw)
/dev/ide/host0/bus0/target0/lun0/part6 on /boot type ext2 (rw,noatime)
none on /dev/pts type devpts (rw,mode=0620)
/dev/ide/host0/bus0/target0/lun0/part1 on /mnt/windows type vfat (rw,nosuid,nodev,umask=0)
/dev/ide/host0/bus0/target0/lun0/part9 on /blam type xfs (rw)

punch[12:57am]-=>lsmod
Module                  Size  Used by
autofs                 13180   1 (autoclean)
usb-uhci               24918   0 (unused)
usbcore                35339   0 [usb-uhci]
3c59x                  25149   1 (autoclean)
maestro                29757   0 (unused)
soundcore               6085   2 [maestro]
vfat                   13075   1 (autoclean)
fat                    37733   0 (autoclean) [vfat]
xfs                   447888   2
xfs_support            13954   0 [xfs]
pagebuf                39935   2 [xfs]
```

```
image=/boot/vmlinuz-2.4.0-XFS-test13-pre4
label=t13p4
root=/dev/hda8
initrd=/boot/initrd-2.4.0-XFS-test13p4.img
append="ramdisk_size=25000"
read-only
```

나에겐 XFS 지원을 커널에 집어넣어 컴파일하는 것이 훨씬 간단해 보이므로, 이 문서에서는 그 방법을 다룰 것이다. 나는 나중에 모듈로 그것을 시도해보고 그때 보다 자세한 설명을 추가하겠다. 만약 내가 이렇게 하기 전에 누군가가 이 방법을 문서화한다면, 부디 나에게 이메일을 보내주길 바란다. 그리고 나는 적절한 크레디트쪽에 추가를 하겠다. :-)

3. 커널 설정과 설치

3.1. 커널이 XFS를 지원하도록 설정하기

> 주의: 만약 당신이 새로운 리눅스 커널을 설정하고 컴파일해본적이 없다면, 이 과정을 행하기 전에 리눅스 커널 HOWTO 를 읽어보라. 그것은 Linux Documentation Project (LDP) (http://www.linuxdoc.org/HOWTO/Kernel-HOWTO.html) 이나 그 미러에서 찾을 수 있다.

cvs 소스 트리를 내려받은 후에, 실제 커널 소스는 /usr/src/linux-2.4-xfs(-beta)/linux 에 있을 것이므로, 당신이 필요한 것들을 선택하기 위한 make config 를 실행하기 전에 그 디렉토리로 이동해야 한다. XFS 지원을 제공하기 위해서 커널에 포함되어야 하는 주요한 것들은 "Page Buffer support" 와 "SGI XFS filesystem support " 이다. 두 옵션 모두 커널 설정의 "File systems" 섹션에서 이용가능하다. 그 옵션들을 사용가능하게 하려면 "Code maturity level options" 안에 있는 "Prompt for development and/or incomplete code/drivers" 를 선택해야 한다. 선택적으로 "SGI XFS filesystem support"아래에 있는 "Enable XFS Debug mode" 과 "Enable XFS Vnode Tracing" 를 선택하고 싶을지도 모른다. 이러한 옵션들은 당신의 XFS 구현을 다소 느리게 할 수 있으나, 문제가 발생한다면 문제의 원인을 추적하는데 유용할 것이다.

3.2. 커널과 모듈을 빌드하기

다른 커널들을 빌드하는 것처럼, 실제로 새 커널을 빌드하기 위해서 다음의 명령을 실행해야 한다.:

```
$ make dep
$ make bzImage
$ make modules
```

3.3. 새 커널과 모듈 설치하기

이것 역시 다른 커널을 빌드할때와 마찬가지로 표준적인 작업이다:

```
$ make modules_install
```

```
$ cp arch/i386/boot/bzImage /boot/vmlinuz-2.4.0-XFS
```

3.4. lilo 설정에 새로운 항목을 추가하고 lilo 를 재설치하기

```
$ vi /etc/lilo.conf
```

당신의 lilo.conf 파일에 다음과 비슷한 새로운 이미지 섹션을 추가한다:

```
image=/boot/vmlinuz-2.4.0-XFS label=xfs  read-only  root=/dev/hda2
```

"root=" 행은 당신의 lilo.conf 파일에 기존에 있던 이미지 섹션의 "root=" 행과 같아야 한다. lilo.conf 파일의 수정이 끝났을때 lilo 를 실행해서 변경을 유효하게 하는 것을 잊지 말도록 하라.

3.5. XFS 유틸리티를 빌드하고 설치하기

XFS 파일시스템에 딸려오는 여러개의 툴은 당신의 XFS 파일시스템을 빌드하고 관리할수 있게 해주는데, 역시 빌드되어야 한다. 이러한 툴은 /usr/src/linux-2.4-xfs(-beta)/cmd/xfsprogs 디렉토리에 있다.

> 주의: 이러한 툴은 /usr/lib/libuuid.a 공유 라이브러리에 의존한다. 만약 이 라이브러리를 설치하지 않았다면, XFS 유틸리티를 컴파일하기 위해서 그것이 필요할 것이다. 당신의 리눅스 버전에 대한 rpm 패키지를 Rpmfind.net (http://www.rpmfind.net) 에서 "/usr/lib/libuuid.a" 로 검색하여 찾을 수 있다. libuuid를 포함하는 데비안 패키지는 uuid-dev 이다. 이 라이브러리의 다른 배포판용 패키지가 다른 장소에 있으리라는 것은 의심의 여지가 없다. 그러한 배포판을 위한 정확한 패키지를 찾는 좋은 방법은 Google Linux (http://www.google.com/linux) 검색 엔진에서 검색하는 것이다.

그 디렉토리로 이동한다:

```
$ cd ../cmd/xfsprogs
```

xfs 유틸리티를 빌드하고 설치한다:

```
$ make install
```

3.6. 새로운 커널로 부트하기

```
$ reboot
```

주의: lilo.conf 파일에서 기본 부트 라벨을 변경하지 않는 한은 새로운 커널 이미지로 부팅하기 위해서 "LILO Boot:" 프롬프트에서 "xfs" 를 입력해야 한다.

4. 파일시스템 이동(migration)

전체 과정에서의 마지막 부분은 데이터가 손실될 가능성이 있는 한 아마도 실제로 가장 까다롭고 위험한 과정일 것이다. 나는 XFS 로의 이동을 하기 전에 시스템의 완전한 백업 (혹은 적어도 중요한 자료 모두)을 하기를 추천한다. 이부분은 역시 설명하기에도 가장 어려운데, 당신의 파일시스템 설정에 따라서 이동을 할 수 있는 방법이 수백가지가 될 것이기 때문이다. 나는 여러분에게 새로운 파일시스템을 만드는 기본 명령을 알려줄 것이고, 파일시스템 문제를 해결하는 방법에 어떻게 착수할지에 대한 몇가지 요점들을 알려주려고 노력할것이고, 전체적으로는 나 자신의 파일시스템을 이동하는데에 사용했던 방법을 여러분에게 알려줄 것이다.

4.1. / 파일시스템 이동하기

완전한 XFS 시스템을 만드는 것의 가장 어려운 부분은 아마도 / 파일시스템을 이동시키는 것일텐데, / 가 시스템의 나머지 전체를 지탱하는 것이고 그것은 시스템 운영중에는 실제로 언마운트 될 수 없기 때문이다. 만약 / 로 마운트될 수 있는 여분의 파티션이 있다면 이와 비슷한 방법으로 할 수 있다(예제에서는 /dev/hda4 를 여분의 파티션으로, /dev/hda2를 / 로 사용한다).

```
$ mkfs -t ext2 /dev/hda4
$ mkdir /mnt/temp
$ mount -t ext2 /dev/hda4 /mnt/temp
$ cd /
$ tar lcf - .|(cd /mnt/temp; tar xpvf - )
```

내가 / 파일시스템에서 여분의 파일시스템으로 파일들을 복사하기 위해 여기서 tar 을 사용함을 주목하라. 대신 cp -dpR 을 사용할수도 있지만, 내가 여기서 보인것과 같이 -l 플래그를 붙여 tar 을 사용하면 오직 / 파일시스템 안에 있는 파일만을 복사할 것이다(다시 말하면, 당신이 /usr 로 마운트해둔 또다른 파티션이 있다면 그러한 것은 복사되지 않을 것이라는 말이다).

다음 단계는 /etc/fstab 와 /etc/lilo.conf 에 있는 /dev/hda4 에 대한 모든 참조를 /dev/hda2 로 바꾸고 lilo 를 실행한다. 그리고 나서 시스템을 다시 부팅해야 한다.

시스템을 재부팅 후에는 /dev/hda4 가 /로 마운트될 것이고 원래의 / 파일시스템 (/dev/hda2)는 마운트되지 않을 것이다. 이제는 /dev/hda2 에 새로운 XFS 파일시스템을 만들 수 있다.

```
$ mkfs -t xfs /dev/hda2
```

그리고 나서 새로운 xfs 파일시스템을 마운트한다:

```
$ mount -t xfs /dev/hda2 /mnt/temp
```

그리고 원래의 / 파일시스템을 그것의 원래 위치로 돌려놓는다:

```
$ cd /
$ tar lcf - .|(cd /mnt/temp; tar xpvf -)
```

다시 한번 /etc/fstab 와 /etc/lilo.conf 에 있는 /dev/hda4 의 모든 객체를 바꾸고 lilo 를 실행해야 한다. 또한 /etc/fstab 에 있는 / 의 파일시스템 유형을 바꿔주어야 한다. 이제는 이와 비슷해 보일 것이다:

```
/dev/hda2               /                       xfs     defaults 1 1
```

> 주의**:** 몇몇 리눅스 배포판에서 위의 예제 상자 이외의 fstab 에 주어진 옵션은 단지 "defaults" 이상의 깊이있는 것일수도 있다. 예를 들어 데비안 시스템은 "defaults,errors=remount-ro"를 사용한다. "defaults" 키워드를 제외하고는 마운트 옵션이 모든 파일시스템마다 다르다. 당신이 사용하고자 하는 특정한 XFS 마운트 옵션을 알고 있지 않는 한은 디폴트 옵션을 사용하라. 주어진 데비안 예제에서는, errors 옵션은 XFS 에서는 사용이 불가능하며 당신의 파일시스템을 마운트되지 못하게 할 것이다.

이제 새로운 xfs / 파일시스템으로 시스템을 리부트한다.

물론 루트 파일 시스템을 이동하는 방법에는 여러가지 다른 방법이 있고 만약 당신이 좋은 생각을 갖고 있다면 나는 정말 그것을 알고 싶고 여기서의 방법보다 더 간단해보인다면 이 문서에 그것을 넣을 것이다. 나 자신은 작업할 여분의 파티션이 없었으나 CD 레코더가 있었으므로 내 루트 파일시스템을 CD 로 구워서 새로운 xfs / 를 만드는동안 루트로 마운트했다. 하지만, 모든 경우에도 새로운 파일시스템을 생성하고 마운트하는 기본적인 명령어는 모두 같을 것이다.

4.2. 마치기

과정의 마지막은 매우 간단하고 새로운 파일 시스템을 만들면서 / 에 대해서 행한 파티션 바꾸는 과정과 본질적으로 같은 과정이다. 나는 이 과정의 나머지를 싱글 유저 모드에서 작업하여 / 이외의 모든 것을 언마운트할 수 있도록 하여 여러번 재부트할필요 없이 모든 교체를 할 수 있기를 추천한다. 싱글 유저 모드로 부팅하기 위해서는 다음과 같이 init 프로세스에 런레벨 변경 명령을 내리던가:

```
$ telinit 1
```

혹은 재부팅해서 LILO 프롬프트에서 싱글 유저 모드로 부팅하도록 요청할 수 있다:

```
LILO Boot: xfs single
```

이렇게 하면 시스템이 부트되고 나서 바로 루트 쉘로 들어가서 외부의 접속이나 가상터미널이 없으므로 다른 사용자들이나 프로세스들에 의해 파일시스템이 사용될 기회가 없다(파일시스템을 사용중이면 당신이 그것을 언마운트 할수 없게 만든다). 이제 당신은 여분의 파티션을 마운트할 수 있고, 예전처럼, 이동될 파티션중의 하나를 그곳에 복사하라(아마도 / 에서 왼쪽으로부터 존재하는 내용을 지워야 할 것이다), 예전의 파일시스템을 언마운트하고, xfs 파일시스템을 그 위에 만들고, 그것을 xfs 로 다시 마운트하고, 예전의 파일시스템의 내용을 그곳에 복사하라. 당신이 /dev/hda3 파티션을 /usr 로 마운트했다고 가정하자. 이 과정은 다음과 유사할 것이다:

```
$ mount -t ext2 /dev/hda4 /mnt/temp
$ cd /usr
$ tar lcf - .|(cd /mnt/temp; tar xpvf - )
$ umount /usr
$ mkfs -t xfs /dev/hda3
$ mount -t xfs /dev/hda3 /usr
$ tar lcf - .|(cd /usr; tar xpvf - )
```

/etc/fstab 에서 /usr 의 파일시스템 유형을 xfs 로 바꾸는 것을 잊지 말자.

이것이 그것에 대한 전부다. 나머지 이동될 파일시스템은 같은 방식으로 작동하며, 그 후에는 완전한 멀티유저 모드로 재부팅해서 당신 자신의 "Linux on Steroids!" 를 갖는 것이다.