완벽한 리눅스 서버 구축

이만용 geoman@nownuri.net, 이준하 1998년 10월 17일

SGML편집 및 교정: 유 성태 alloying@nownuri.net 1998년 11월 7일

이 글에서는 텔넷 서비스, FTP 서비스, 메일 서비스, 웹 서비스, NFS 서비스를 어떻게 구현할 것인지 그리고 구현할 때 주의할 점은 무엇인지를 알아보기로 한다.

목 차

제 1 절 시작하는 이야기

리눅스의 세계로 발을 들여놓은 여러분을 언제나 환영한다. 여러분이 리눅스로부터 무엇을 원하는가? 리눅스로부터 원하는 것은 가지각색이고 많기도 하면서 적을 수도 있다. 리눅스를 게임기로 사용하거나 이런저런 장난감을 시험해보는데 사용하는 사람도 있고 윈도95보다 안정적인 웹 서핑을 바라고 리눅스를 쓰는 사람도 있을 것이다. 만족스럽지는 않지만 리눅스에서 네트스케이프 웹 서핑은 괜찮은 수준이다. 필자의 경우 수많은 리눅스 리소스를 찾아다니기 위해 8개 이상의 브라우저 화면을 써도 안심하고 사용할 수 있기에 웹 서핑 클라이언트로 리눅스를 높이 평가한다. 또는 리눅스를 프로그래밍 환경으로 사용하는 사람도 있을 것이다. 네트워크 프로그래밍, X 윈도 프로그래밍에 아주 좋은 환경이라고 말할 수 있다. 리눅스에서 POSIX 규격으로 프로그래밍한 소스는 다른 플랫폼에서 얼마든지 소스 코드의 변경없이 컴파일하여 사용할 수 있다. 특히 X 윈도는 표준적이기 때문에 포팅은 정말 쉽다. 많은 기대가 있기는 하지만 그러한 기대들 중에서 전세계적으로 리눅서들이 리눅스에게서 원하는 것 중 하나는 "막강한 네트워크 서버", "다양한 기능의 네트워크 서버" 로서의 역할일 것이다. 규모로는 아직 썬마이크로시스템즈사의 스팍 머신들을 따라갈 수 없지만 인터네트에 퍼져있는 개인 사용자들의 사이트는 여지없이 인텔 머신 위에서 리눅스 서버로 운영되고 있다는 것을 아는지 모르겠다. 우리 나라의 경우 비정상적인 컴퓨터 활용으로 인해 안정성을 의심받고 있는 윈도95같은 것을 가지고 서버까지 해결하지 않으면 안심이 안되는 분위기와는 정말 다르다. 리눅스의 진가가 발휘되는 곳은 바로 네트워크 서버이다.

많은 대학생과 연구원들이 자신들의 네트워크 서버를 구축하고 싶다는 생각을 많이 할 것이다. 여러분에게 서버 운영체계로서 많은 선택권이 있지는 않다. 제일 많이 사용되는 클라이언트 운영체계 윈도95같은 것으로 서버를 한다는 것은 좀 우스운 일이고 대부분의 선택권은 리눅스/윈도NT/FreeBSD 정도가 될 것이다. 더 이상 배우는 것이 귀찮거나 남이 알아서 해주길 바라는 사람의 경우에는 유닉스 계열처럼 다양한 기능을 기본으로 제공하는 것은 아니지만 버전 4.0이 되고 나서 쓸만해졌다는 평가도 있는 윈도즈NT 서버 버전을 설치하는 것

도 좋을 것이다(이미 지난 달 서버 버전이 프로그램세계 CD부록으로 나간 것으로 안다. 워크스테이션 버전과 혼동하지 말기 바란다. 물론 베타버전임 또한 잊지 말 것). IIS 또는 FrontPage같은 것에 대한 내용은 해당 지식을 가진 사람이나 동호회에서 해결하기 바란다. 나머지 선택권은 리눅스/FreeBSD 와 같은 자유 운영체계들인데 어떤 것을 선택하든 여러분 마음이다. 자기에게 가장 익숙한 것을 사용하기 바란다. FreeBSD 경우에 버클리 유닉스의 전통적인 강점인 네트워킹에서 강력한 기능을 발휘한다는 사실을 아는 사람은 이미 알고 있는 바이다. 하지만 결국 선택은 리눅스에게로 돌아온다고 생각한다. FreeBSD는 UC 버클리측에서만 중앙집중적으로 개발하고 있다. 이점이 리눅스와 많이 다르다. '사용자 = 개발자' 인 독특한 강력함을 지니고 있는 리눅스의 경우 바로 FreeBSD 등의 모든 표준적인 모델들을 보면서 네트워킹 데몬들을 포팅하였기 때문에 BSD 계열의 강력함도 가지고 있는 운영체계이기 때문이다.

처리 능력으로 따지자면 둘 사이에 커다란 차이는 없다. 만약 리눅스가 뒤진다고 한다면 며칠 안으로 분명히 패치가 나올 것이다. 전세계 해커들의 자존심이 걸린 운영체계 리눅스는 충분히 믿을 만한 운영체계라는 것을 강조하고 싶다. 하지만 구슬이 서말이라도 꿰어야 보배라고 여러분이 어떻게 사용하는가에 따라 크게 그 성능이 좌우된다는 것도 잊지 말아야 한다. 리눅스는 그야말로 운영의 묘미를 발휘할 수 있는 젊은 관리자들에 따라 그 성패가 달렸다. 열정을 가지고 달려들면 여러분 주위의 어떤 서버들도 따라올 수 없는 강력한 서버가 되며 그렇지 않으면 리눅스에 대해 실망하고 리눅스의 명예를 떨어뜨릴 수도 있다. 항상 옆에 여러분과 같은 리눅서들이 있다는 것을 잊지 말고 어려운 것이 있으면 서로 물어보라. 물론 리눅서로서 모든 일의 책임은 자기가 진다는 사실도 잊지 말자. 끝내 해내고야 말겠다는 생각을 가지면 리눅스는 여러분에게 초강력 서버로 다가올 것을 약속하며 글을 시작한다.

제 2 절 어떤 서비스를 제공할 것인가?

여러분이 알고 있는 네트워크 서비스를 생각해보도록 하자. 텔넷 서버, FTP 서버, 메일 서버, 웹 서버 등은 가장 기본적인 서버의 기능이라고 말할 수 있다. 그외에도 NFS 서버 등의 기능 또한 해낼 수 있고 네트워킹에 대한 충분한 지식을 가지고 있는 DNS 서버, NIS 서버 그리고 다양한 플랫폼의 운영체계에서 돌아가는 클라이언트들에 대한 파일 서버, 프린트 서버 기능을 충분히 해낼 수 있다.

1. 텔넷 서비스
2. FTP 서비스
3. 메일 서비스
4. NFS 서비스
5. 웹 서비스

이제 위에서 말한 5 대 서비스를 어떻게 구현할 것인지, 그리고 구현할 때 주의할 점은 무엇인지를 알아볼 것이다.

제 3 절 텔넷 서비스(Telnet)

유닉스 계열 고유의 멋진 기능이라고 할 수 있다. 원격 호스트로부터 서버로의 접속을 허락하고 여러분 서버의 시스템 자원을 사용할 수 있도록 허용해주는 것이다. telnet <호스트명 또는 호스트 IP 주소> 를 입력하고 들어감으로써 접속한 사람이 마치 그 서버 콘솔 앞에 앉아 있는 것처럼 서버의 프로그램을 사용할 수 있다.

클라이언트/서버 모델을 모르고서는 네트워크를 사용한다는 것이 불가능하다고 말하고 싶다. 하지만 이 이슈는 클라이언트 사용자에게 필요한 내용이다. 사용자는 서버의 자원을 사용하고 있으며 그 결과만을 자신의 터미널로 출력 받는다는 사실을 잘 알고 있어야 한다.

3.1 사전 준비

텔넷 등의 데몬에 약간의 버그가 있다고 알려져 있다. 꼭 엄청난 버그, 보안에 관계된 버그는 아니더라도 네트워크 데몬에 대해서는 정기적으로 최신 버전으로 교체할 필요성이 있다. 그것 말고 네트워크 관리자가 할 일이 또 무엇이겠는가? 여러분이 받아와야 할 파일은 netkit-base-0.10.tar.gz라는 파일이다(물론 이보다 상위 버전이 있다면 그것을 받아오라). 주 사이트는 `ftp.uk.linux.org`이며 부 사이트는 유명한 `sunsite.unc.edu`이다. 물론 국내에도 가 있다. 파일을 받을 때는 꼭 .tar.gz대신 .txt 또는 .lsm (Linux Software Map)이라는 이름으로 끝나는 파일을 받아서 확인해보기 바란다. 자료를 풀고 그냥 make하는 것만으로도 원하는 파일을 컴파일할 수 있도록 되어 있다. 필자는 알짜 슬랙웨어 3.1과 libc 5.4.7에서 에러 하나 없이 무사히 컴파일 하였다. 설치는 루트 권한으로 make install이다. 텔넷 말고도 여러 가지 네트워크 데몬들을 설치해준다. biff, comsat, finger, fingerd, inetd, ping, rlogin, rlogind, rpc, rpcgen, rpc.usersd, rpc.rwalld, rpcinfo, rsh, rshd, rusers, rwall, rwho, rwhod, bsdslattach, talk, talkd, timed 등이 바로 그것이다. 뒤에 d로 끝나는 파일은 모두 데몬을 뜻한다.

3.2 사용자 계정 만들기

일단 사용자의 요청을 받아서 서버의 하드디스크에 계정을 만들어야 한다. 보통 많은 배포판에서는 adduser와 같은 스크립트 또는 실행파일(스크립트는 해킹하기 쉬우므로 아예 실행파일로 만드는 것도 좋은 방법이다)을 제공해서 사용자를 등록할 수 있도록 해주고 있다. 사용법은 다음과 같다.

```
# adduser

Adding a new user. The username should not exceed 8 characters
in length, or you may run into problems later.
```

사용자 ID를 만들 때는 8자 이상 넣지 말라. 그것을 염두에 두기 바란다.

```
Enter login name for new account (^C to quit): foo

Editing information for new user [foo]
```

사용자에 관한 정보를 넣는다. 사용자의 본명, GID, UID, 홈 디렉토리, 사용하는 셸의 종류, 패스워드 등 아주 중요한 정보를 입력해야 한다. 특히 UID는 모든 사용자에게 유일한 수치여야 하므로 각별히 주의하기 바란다.

```
Full Name: Lee Man Yong
GID [100]: ?
```

여기서 그냥 엔터, 사용자 그룹을 만들어주길 원한다면 잠시 뒤의 설명을 보라.

```
Group 'users', GID 100
First unused uid is 506
```

adduser 스크립트가 자동으로 사용되지 않은 UID 중 가장 낮은 번호를 여러분에게 제공해준다. 그 값을 그냥 사용하면 된다. 여러분이 특별하게 해줄 필요가 있다면 원하는 수치를 적어준다. 하지만 절대 중복된 수치가 들어가서는 안된다는 것을 명심하라.

```
UID [506]: ?

Home Directory [/home/foo]: ?
```

홈 디렉토리는 보통 /home 아래에 생긴다. 하 지만 정말로 대형 서버인 경우에는 사용자들을 여러 그룹으로 나누어 /home1, /home2, /home3 이런 식으로 디렉토리를 많이 나누고 그 아래에 적절하게 배분해주기도 한다. 예를 들어 학교에서 학번 별로 관리를 하려고 한다면 /90, /91, /92 처럼 해주고 사용자의 학번을 알아서 관리하는 것이 좋지 않을까? 작은 시스템에서는 그냥 /home 하나 아래에다가 관리하는 것이 좋다.

```
Shell [/bin/bash]: ?
```

또는 C 셸을 선호하는 사람은 /bin/tcsh라고 입력해준다.

```
Password [foo]:Shell [/bin/bash]: ********?
```

패스워드를 제외하고는 대부분의 설정은 <엔터>키만으로도 해결된다. 패스워드는 영어사전에 있는 단어를 사용하지 말고 대소문자, 숫자를 섞어서 사용하는 것이 좋다. 학교 랜 환경에서 믿을 수 있는 사람들끼리 사용하는 느슨한 시스템이라면 별 상관없다. 나중에 세팅을 통해서 여러분의 랜에 물린 믿을 수 있는 컴퓨터들 빼고는 다른 곳에서 텔넷 접속을 할 수 없도록 만들 수 있으니까 그때 가서 알아보도록 하자.

```
Information for new user [foo]:
Home directory: [/home/foo] Shell: [/bin/bash]
uid: [506] gid: [100]
```

여러분이 입력한 최종 정보를 보여주며 사용자를 등록할 것인지 물어온다.

```
Is this correct? [y/N]: Y

Adding login [foo] and making directory [/home/foo]

Adding the files from the /etc/skel directory:
./.less -> /home/foo/./.less
./.lessrc -> /home/foo/./.lessrc
./.Xdefaults -> /home/foo/./.Xdefaults
./.Xmodmap -> /home/foo/./.Xmodmap
./.bash_logout -> /home/foo/./.bash_logout
./.bash_profile -> /home/foo/./.bash_profile
./.bashrc -> /home/foo/./.bashrc
./.emacs -> /home/foo/./.emacs
./.inputrc -> /home/foo/./.inputrc
./.kermrc -> /home/foo/./.kermrc
./.mailcap -> /home/foo/./.mailcap
./.mime.types -> /home/foo/./.mime.types
./.cshrc -> /home/foo/./.cshrc
```

스크립트는 마지막으로 사용자 홈디렉토리를 만들고 기본적인 세팅을 담고 있는 파일들을 자동으로 /etc/skel 디렉토리에서 사용자 홈디렉토리로 복사한다. .bashrc, .inputrc, .cshrc 등 중요한 파일을 꼭 /etc/skel에 넣어두고 사용자들이 처음부터 중요하지 않은 세팅에 시간을 쏟지 않아도 되도록 신경을 쓰면 좋다. 관리자 한 명의 위력은 대단하다. 그 사람이 한 시간만 봉사하면 나머지 사람들은 한 시간씩이 모두 절약된다.

3.3 사용자 계정을 만들고 난 후의 점검 사항

첫 번째, **퍼미션! 퍼미션! 퍼미션!** 사용자의 홈 디렉토리는 보통 **drwxr-xr-x**라는 퍼미션을 가지고 있다. 앞의 d 는 디렉토리라는 의미일 뿐이고 나머지 rwx, r-x, r-x가 각각 사용자, 그룹, 타인에 대한 퍼미션이라는 것을 알 것이다. 보안이 중요하게 느껴지는 환경에서는 타인에 대하여 x 퍼미션을 없애면 아예 디렉토리로 들어

올 수 없으니 사용자들에게 맡기기보다는 스크립트 같은 것을 하나 만들어서 모든 사용자들의 홈 디렉토리에서 타인의 실행 퍼미션을 제거해주는 것도 좋다. 사실 관리자가 해야 할 대부분의 일은 퍼미션이다. 이거 하나만 잘 관리하면 제일 안전한 서버가 된다. 여러분의 서버에 들어와서 장난을 하고 싶어하는 크랙커가 있다 하더라도 철통같은 퍼미션에 금방 싫증을 느낄 것이다. 그 크랙커는 다시는 여러분의 서버로 들어오지 않는다. 왜냐하면 좀처럼 쉽게 크랙이 되질 않아 자기 스스로 지치기 때문이다. 아마도 다른 곳을 찾아 떠날 것이다. 두 번째 root만이 사용하는 실행파일의 퍼미션을 철저히 하라. 특히 /sbin, /usr/sbin 디렉토리는 타인에 대한 실행 퍼미션(디렉토리에 대한 실행 퍼미션은 특수한 의미를 갖는다)을 없애서 들어갈 수조차 없도록 하라. /sbin 과 /usr/sbin 디렉토리에 있는 대부분의 프로그램들은 root가 아닌 사람이 사용할 필요가 전혀 없다. 특히 fdisk같은 것은 절대 타인의 실행 퍼미션이 있어선 안된다. 세 번째 쓸모없는 프로그램은 지운다. 예를 들어 사용자를 변경하여 사용할 수 있게 해주는 su는 아주 위험한 명령임에 틀림없다. su가 있다면 일반사용자로 들어와서도 루트의 패스워드만 알아내면 언제든지 루트로 돌변할 수 있기 때문이다. su와 sudo에 대해서는 잘 생각해보기 바란다.

3.4 쿼터 제한 가하기

쿼터(quota) 제한이란 사용자가 이용할 수 있는 하드디스크의 양을 제한하는 것을 말한다. 커널 컴파일할 때 쿼터 제한 기능을 꼭 집어넣어서 컴파일하고 쿼터 제한을 가할 수 있게 해주는 프로그램들(quota, edquota, quotacheck, quotaon, repquota, quotactl)을 이용하면 된다. 쿼터 제한을 하는 이유는 절대로 사용자들에게 제한을 가하려는 것이 아니다. 다중 사용자 환경에서 한 사람의 사용자가 자원을 독차지함으로써 다른 사용자에게 피해를 주는 일을 없애기 위해서이다. 악의적인 사용자가 인터네트로부터 윈도95용 오락을 잔뜩 ftp로 받아놓고 /home 파티션을 다 차지해 버리면 어떻게 할 것인가? 쿼터 제한만이 있다면 사용자들은 불평을 할 것이다. 그래서 쿼터 제한과 함께 있어야 할 것은 모든 사람이 자유롭게 사용할 수 있는 하드디스크 공간이다. 홈 디렉토리에서는 쿼터를 걸지만 그 자유공간만은 사용자들의 자율에 맡기는 것이다. 보통 여분의 하드디스크가 있으면 /var/tmp 등의 디렉토리에 마운트시 켜놓고 퍼미션을 자유롭게 해준다.

3.5 파티션 원칙

/home 파티션, /var/tmp 등의 파티션은 루트 파티션이나 기타 /usr 파티션과 구분하여 사용하는 것이 좋다. 파티션을 따로 분할해서 마운트하여 사용하든가 아니면 여분의 하드디스크가 많을 때는 아예 다른 하드디스크에 두면 좋다. 특히 /usr 파티션은 읽기 전용으로만 마운트되는 것이 제일 바람직하다. 자세한 사항은 리눅스 파일 시스템 규격인 FSSTND 문서를 구해서 읽어보기 바란다.

3.6 사용자 지우기

사용자를 지워야 할 때도 있을 것이다. 하지만 여러분이 기대하는 deluser와 같은 것은 없다. 하지만 지우는 방법은 아주 간단하다. /etc/passwd에서 사용자에 해당하는 줄을 지우고 그 사용자의 홈 디렉토리를 모두 지워주면 된다. 하지만 그 사용자의 파일이 홈 디렉토리 뿐만 아니라 다른 곳에 산재해 있을 수도 있다. 그럴 때는 다음과 같이 하면 된다.

```
find / \( -nogroup -o -nouser \) -print
```

이렇게 하면 사용자를 지우기 때문에 생긴 고아 파일들이 나열된다. 그런데 이런 방법은 상당히 조심해야 한다. 뜻하지 않는 파일을 지울 수도 있기 때문이다. 또는 미리 find / -user foo로 파일들을 검색해서 수동으로 지우자. 보통

```
find / -user foo -exec rm -f {} \;
```

하면 될 것으로 생각하는 사람이 많은데 이것은 여러분 시스템의 퍼미션이 완벽한 경우에나 제대로 되지 행여나 조금이라도 잘 못하는 경우에는 원하지 않는 파일을 지우는 수가 있다. 관리자, 그리고 관리자의 개인 아이디인 경우만 조심하면 된다. 나머지 일반 사용자에 대해서는

```
find / -user <사용자명> -exec rm -f {} \;
```

만 해도 될 것이다. 어찌되었든 여러분이 관리자라면 정말로 흥미로운 find 명령을 잘 익혀두길 바란다. 관리자가 find라는 명령을 모른다면 이미 실격이다.

3.7 패스워드 파일에 익숙해지기

/etc/passwd 파일을 한 번 살펴보기 바란다.

```
ftp:*:404:1::/home/ftp:/bin/bash
guest:*:405:100:guest:/dev/null:/dev/null
yong:JprJGlXBC1bGQ:501:100:Lee Man Yong,Home,,(02) 603-2461,:/home/yong:/bin/bash
smileguy:EbhqvUES5/4/s:502:100:J J H:/home/smileguy:/bin/bash
postgres:PbKIAttHJZqJw:503:100:Postgres95 Master:/home/postgres95:/bin/csh
```

형식은 다음과 같다.

```
사용자명:암호화된 패스워드:UID:GID:사용자정보:홈 디렉토리:사용셸
```

7개의 필드가 콜론(:)으로 분리되어 있다. adduser라는 스크립트를 쓰지 않고 그냥 수동으로 입력해도 된다. ftp, guest라는 사용자를 한 번 살펴보자. ftp라는 사용자는 익명 FTP 사용자를 말하는데 패스워드 필드에 애스터리스크 문자(*)가 있는 것을 주목하자. 이렇게 하면 그 사람은 등록되어 있기는 하지만 로그인 할 수가 없게 된다. 잠시 어떤 사람을 로그인하지 못하게 만들려면 패스워드 맨 앞에 애스터리스크 문자(*)를 넣어주면 된다. 그것을 해제하면 다시 로그인할 수 있다.

3.8 그룹 만들기

그룹으로 관리할 필요성에 대하여 위에서 얘기한 바 있다. 그것은 그렇게 어렵지 않다. /etc/group 이라는 파일을 한 번 살펴보도록 하자. 형식은 /etc/passwd 파일과 비슷하다. "그룹명:패스워드:GID:그룹회원명 리스트"라는 간단한 형식을 갖는다. 보다시피 패스워드 필드는 모두 아무 것도 넣지 않았다. 그룹에 패스워드를 부여하려는 경우를 빼고는 대부분 사용자 패스워드 만으로도 충분한 것 같다. 새로운 그룹을 만들려면 위에 있는 그룹 중 다른 이름 그리고 유일한 GID를 부여한다. 그리고 원하는 사용자를 회원 리스트에 콤마(,)로 분리하여 넣으면 된다.

```
root::0:root
bin::1:root,bin,daemon
daemon::2:root,bin,daemon
sys::3:root,bin,adm
adm::4:root,adm,daemon
tty::5:
disk::6:root,adm
lp::7:lp
mem::8:
kmem::9:
wheel::10:root
floppy::11:root
mail::12:mail
news::13:news
uucp::14:uucp
man::15:man
users::100:games
nogroup::-2:
```

3.9 셰도 패스워드 시스템

상당한 보안이 필요한 시스템의 경우에는 셰도 패스워드라는 것을 사용한다. /etc/passwd 파일은 루트 사용자만이 아닌 모든 사람들이 읽을 수 있는 파일이므로 원하는 사람은 누구나 카피본을 가질 수 있고 일명 '사전 공격 (Dictionary Attack)' 이라는 방식을 써서 간단한 패스워드를 쓴 사용자들의 암호를 깰 수 있기 때문이다. 그래서 진짜 패스워드는 다른 곳에 숨겨두고 쓸모없는 /etc/passwd를 보여주는 셰도(Shadow) 패스워드 시스템이라는 것이 나온 것이다. 셰도 패스워드를 다루지는 못하겠다. 왜냐하면 상당히 번거롭기 때문이다. 사실 셰도 패스워드 시스템으로 전환하려면 상당한 노력이 든다. 단지 패스워드 스템만 바꾸는 것이 아니라 그에 연관된 모든 프로그램들을 셰도 패스워드용으로 구해야 한다는 수고가 든다. 관련 How-To 문서가 있으니 관심 있는 사람들은 그것을 참고하기 바란다.

3.10 X 윈도 프로그램 사용하게 만들기

텔넷으로 할 수 있는 재미있는 일 중 하나가 바로 X 클라이언트/X 서버의 연결이다. 만약 여러분의 서버에 X 윈도를 작동시켰다면 그리고 적절한 세팅을 해주면 X 윈도 사용자들은 여러분의 X 윈도 프로그램을 가동시켜서 자신의 로컬 X 윈도에서 그 결과를 사용할 수 있다. 이렇게 함으로써 대형 서버만이 특정 대형 프로그램들을 구비하고 있어도 접속만 하면 마치 자기 컴퓨터의 프로그램을 쓰듯 사용할 수 있다. 물론 충분한 네트워크 속도가 나와야 한다. 일반적으로 X 서버가 작동하면 다른 호스트에서 들어와서 서버의 X 프로그램을 사용하여 자신의 X 터미널에서 결과를 볼 수 있게 되어있지 않다. 그래서 서버에서는 다음과 같은 일을 해주어야 한다.

```
xhost +호스트명
또는
xhost +
```

설정한 호스트명으로부터 X 연결을 허용한다. 플러스(+) 문자 대신 마이너스 (-) 문자를 사용하면 주어진 호스트로부터 X 연결을 불허한다. 그냥 플러스 자만 쓰면 모든 호스트로부터의 접속을 허용한다. 마찬가지로 마이너스 문자만 쓰면 모든 호스트로부터의 접속을 불허한다. 그냥 아무런 인수없이 xhost라고 실행하면 현재의 상태를 보여준다. 보통은 로컬 호스트에서만 접속을 허용한다. 보통 X 윈도 프로그램들은 엄청난 시스템 자원을 사용하기 때문에 아주 높은 사양의 컴퓨터, 충분한 램(64메가 이상, 그 이상일 때는 꼭 캐시램 512KB로 늘리기 바란다. 128 메가 이상일 때는 커널에게 옵션을 주어야만 된다는 사실을 알아야 한다), 충분한 스왑 파티션(100메가 이상)을 요구한다.

3.11 로그인하는 사용자에게 공고하기

/etc/issue.net이라는 파일에다 여러분이 원하는 공고문을 적어주면 된다. 그 파일에는 운영체계, 버전, 작동되는 머신 명칭(인텔 머신인지 알파인지 아니면 스팍인지), 날짜 등을 나타내는 특수문자가 들어갈 수 있다. 이 파일을 통해서 시스템 점검이라든지 커널 버전업으로 인한 서비스의 일시 중단 등을 공고할 수 있다.

`/etc/issue.net 예제`

```
%s %r (%m) %v (%t)
   지금 현재는 %d 입니다.
   환영합니다!
freeyong:~$ telnet localhost
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.

Linux 2.0.23 (i586) #7 Wed Oct 23 05:03:07 KST 1996 (ttyp4)
   지금 현재는 11:14 on Saturday, 02 November 1996 입니다.
   환영합니다!
```

3.12 안전 터미널

사용자 중에서 가장 특수하고 막강한 권력을 가진 사용자는 바로 루트이다. 많은 크랙커들이 유닉스의 가장 아름다운 서비스 중 하나인 텔넷을 주공격 대상으로 하는 것은 놀라운 일이 아니다. 원격지에서 로그인하여 상대 컴퓨터를 마음껏 사용할 수 있다는 것은 큰 매력이기도 하지만 커다란 보안의 문제일 수도 있다. 크랙커들이 어떤 식으로든 첫 번째 목표로 세우는 것은 루트의 비밀번호를 알아내는 일이다. 그렇게 되면 그 시스템은 완전히 그의 손안에 들게 되는 것이다. 최대한 크랙커들을 예방하기 위해 안전 터미널(Secure Terminal)이라는 개념이 있다. 루트 사용자는 특별하므로 미리 정해진 몇 개의 콘솔, 그리고 안전하다고 생각하는 터미널만을 사용할 수 있도록 제한하는 것이다. 보통은 루트가 특별히 필요한 경우가 아니라면 원격지에서의 로그인, 관리는 행하지 않는 것이 좋다. 관련 파일은 /etc/securetty라는 파일이다. 알짜슬랙웨어에는 심각하다면 심각하다고 말할 수 있는 오류가 들어있다. 다음 파일을 살펴보도록 하자.

```
console
tty1
tty2
tty3
tty4
tty5
tty6
ttyS0
ttyS1
ttyS2
ttyS3
ttyp0
ttyp1
ttyp2
ttyp3
```

우선 루트는 콘솔 즉 자기 서버 앞에서 로그인할 수 있다. 그 다음은 가상 콘솔인 tty1부터 tty6 그리고 직접 접속된 시리얼 라인 또는 모뎀 접속인 ttyS0부터 ttyS3이 있다. 알짜웨어에서 문제가 되는 것은 바로 가상 터미널인 ttyp0부터 ttyp3를 추가한 것이다. 이것은 제작자의 오류이다. 만약 원격지에서도 루트의 관리를 할 수 있을 만큼 보안의 문제가 없는 곳에서는 놔두어도 좋다. 루트로 로그인하려고 하다가 안전 터미널이 아니라고 거부하는 모습을 한 번 보자.

```
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
 Linux 2.0.25 (i586) #14 Sun Nov 10 02:32:27 KST 1996 (ttyp5)

   지금 현재는 18:58 on Monday, 11 November 1996 입니다.
   환영합니다!

freeyong login: root
root login refused on this terminal.
```

제 4 절 FTP 서비스

FTP는 TCP/IP의 아주 중요한 부분으로 남아있는 서비스이다. 웹의 등장으로 조금 무기력해진 것같이 보여도 대량 파일 전송에는 역시 FTP 서비스가 최고이다. 그리고 FTP는 사라지기보다는 웹 브라우저 안으로 통합되는 양상을 띠고 있다. FTP 서비스는 크게 두 가지로 나눌 수 있는데 서버 시스템에 등록한 사용자들을 위한 일반적인 FTP와 익명의 모든 사용자들에게 개방하는 익명 (Anonymous) FTP가 있다. 등록 사용자에 대한 FTP 서비스는 텔넷과 비슷하게 사용자명과 패스워드를 입력받고 자기 권한만큼 파일에 접근해서 받아갈 수

있으며 자신의 홈 디렉토리 같은 곳에는 업로드도 가능하다. 약간의 주의를 요하는 것이 바로 익명 FTP이다. 일단은 여러분이 갖고 있는 대부분의 소규모 FTP에 그렇게 많은 접속이 이루어지지는 않을지 모르나 커다란 FTP 사이트를 건설하려고 한다면 정말로 엄청난 시스템이 아니면 안될 것이다.

4.1 익명 FTP 서비스 준비사항

한 마디로 텔넷과 마찬가지로 리눅스 설치와 함께 FTP 서비스는 기본으로 이루어진다. 여러분의 리눅스 머신이 이미 네트워크에 존재한다면 여러분도 모르는 사이에 누군가 이미 여러분의 컴퓨터를 익명 FTP로 사용하고 있는지도 모른다. FTP 서비스 또한 수퍼 서버 inetd에 의해 관리된다. 빠른 반응 시간을 갖기 위해서는 ftpd를 그냥 띄워도 괜찮다.

4.2 새로운 FTP 데몬의 설치

/usr/sbin 디렉토리로 가보도록 하자.

```
freeyong:/usr/sbin# ls -l *ftpd
-rwxr--r--   1 root     root         8528 Sep  9 14:14 in.tftpd*
-rwxr-xr-x   1 root     bin         77444 Dec  6  1995 wu.ftpd*
```

위에서 보는 바와 같이 wu.ftpd가 없다면 네트워크 키트를 받아다가 설치해 주는 것이 좋다. 기본 설정치이다. 워싱턴 대학에서 만든 뛰어난 ftp 데몬이며 거의 모든 유닉스 계열 사이트에서 찾아 볼 수 있을 것이다. 우리가 설치할 ftp 데몬은 바로 워싱턴 대학의 wu.ftpd이다.

4.3 익명 FPT의 보안 점검

익명 FTP는 특히 안전성이 중요하다. 항상 악의를 가진 사람들이 존재한다는 식으로 생각하는 것은 좋지 않다. 보안을 하는 이유는 크랙커를 막겠다는 것보다는 시스템의 핵심부를 타인에게 드러내지 않음으로써 예기치 않은 일들을 막고자 하는 것이다. 그럼 점검을 해보자.

```
ftp:*:404:1::/home/ftp:/bin/bash
```

/etc/passwd를 보면 위와 같이 되어 있는 것을 볼 수 있을 것이다. 패스워드 필드에 애스터리스크 문자만 있으므로 ftp라는 로그인명으로는 텔넷 접속같은 것은 할 수 없도록 되어 있다. 접속 자체가 불가능하기는 하지만 안전하게 하기 위해서 셸도 /bin/bash 같은 걸로 지정하는 것보다는 그냥 /bin/false같은 것으로 지정해두는 것도 좋다. 슬랙웨어, 알짜웨어 등 모든 경우에 있어서 ftp라는 사용자는 위에서 보면 1번 그룹에 속하는 것으로 나와있는데 1번 그룹은 bin 그룹이다. 보통은 anonymous라는 그룹을 새로 만든 후에 그 그룹의 멤버로 설정하면 좋을 듯 하다.

4.4 익명 FTP 홈 디렉토리

익명 FTP의 홈 디렉토리는 위에서처럼 /home/ftp이다. 자, 잠시 여러분이 어떤 익명 FTP에 들어갔을 때를 생각해보자. 그러면 전형 적으로 다음과 같은 디렉토리가 보일 것이다. 만약 여러분이 cd / 라는 명령으로 시스템의 루트 디렉토리로 가려고 해보았자 여러분이 원하는 디렉토리로 가는 것이 아니라 실제로는 전체 시스템에서 /home/ftp에 해당하는 곳에 머무를 뿐이다. 이것이 익명 FTP와 일반 사용자들의 FTP가 다른 점이다. 익명 FTP의 경우에는 ftp 사용자에게 있어 /home/ftp가 마치 / 처럼 작동하도록 되어 있다. 내부적으로 chroot라는 것이 작동하여 /home/ftp라는 디렉토리를 루트 디렉토리처럼 인식하도록 하니까 /home/ftp 이하의 모든 디렉토리들은 접근을 할 수 없도록 한 것이다. 각 디렉토리를 점검해보도록 하자.

```
drwxr-xr-x    8 root   wheel   1024 Aug 23 20:30 .
drwxr-xr-x     8 root   wheel   1024 Aug 23 20:30 ..
drwxr-xr-x     2 root   wheel   1024 Aug 23 20:30 bin
drwxr-xr-x     2 root   wheel   1024 Aug 23 20:30 etc
drwxrwxrwx   3 root    wheel   1024 Oct 11 16:21 incoming
drwxr-xr-x     2 root   wheel   1024 Nov 17  1993 lib
drwxr-xr-x     2 root   wheel   1024 Aug 23 20:30 pub
drwxr-xr-x     3 root   wheel   1024 Aug 23 20:30 usr
-rw-r--r--     1 root   root     312 Aug  1  1994 welcome.msg
```

우선 /home/ftp 즉 ftp 디렉토리는 ftp라는 사용자가 소유하고 있어야 한다. 그리고 다른 사람들은 쓰기 퍼미션을 가져서는 안된다. ftp/bin 즉 /home/ftp/bin에는 ls라는 실행파일이 적어도 하나 들어있어야 하며 소유권은 수퍼 유저, 루트가 가지고 있으며 ls의 퍼미션은 111이다. ftp/etc는 수퍼 유저의 소유이며 쓰기 금지가 되어 있어야 한다. 내용을 보면 passwd와 group 파일이 있는데 실제로는 시스템 전체의 패스워드와 그룹 파일과는 다른 파일이다. 이 파일이 있는 이유는 dir 했을 때 숫자가 아니라 사용자명 그룹명이 나오도록 하기 위함이다. 잘 보면 패스워드 같은 건 아예 없다는 것을 알 수 있다. 이 파일을 지우면 온통 사용자명과 그룹명이 숫자로만 나올 것이다. ftp/pub는 일단 읽기만 되어야 한다. 업로드 파일은 ftp/incoming을 이용하라. 배포판들에서 incoming 디렉토리의 소유자가 잘못되어 있는데 소유자는 루트가 아니라 ftp여야 한다. 루트 권한으로 들어가서 chown ftp ftp/incoming이라고 하면 된다. 단지 ftp 사용자에게만 쓰기 권한이 있어도 업로드를 할 수 있도록 되어 있다. ftp 사용자에게만 쓰기 권한이 있도록 조정한다. 일단 한 번 올린 파일에 대해서는 지울 수 없다. 사람들에게 파일만 올릴 것이 아니라 설명서도 꼭 올려주도록 부탁한다. 관리자는 정기적으로 incoming 디렉토리를 보고 pub 디렉토리 밑에다 알맞게 범주로 나누어서 다시 사용자들이 다운로드만 할 수 있게 해준다.

4.5 사용자 환영 메시지

/home/ftp 디렉토리를 보면 welcome.msg 파일을 볼 수 있다. 그곳에다 사용자들에게 알려야 할 사항을 적어주면 된다. 시스템 정기 점검, 업그레이드 소식 등 또는 새로운 자료 소식을 항상 올려주면 좋을 것이다. 각 디렉토리로 들어갈 때마다 간단한 안내를 해주는 경우가 있다. 이 때는 .message라는 파일을 각 디렉토리에 만들어서 화면에 표시하고 싶은 내용을 적으면 된다. 피리어드(.) 문자로 시작하는 파일임에 유의하라.

4.6 시디롬 내용을 제공하려고 할 때

시디롬을 마운트시켜서 그 내용을 제공하는 것도 안전한 방법 중의 하나라고 생각한다. 그리고 하드디스크 용량을 절약할 수 있기 때문에 아주 좋다. 그런데 많은 사람들이 시디롬 내용을 익명 FTP로 제공하려고 할 때 약간의 어려움을 겪고 있는 것 같다. 여러분이 원하는 곳에 마운트를 했다고 생각하거나 또는 기존에 /cdrom으로 마운트된 것을 /home/ftp 하위 디렉토리에서 링크하면 되지 않을까 생각해도 전혀 시디롬이 있는 디렉토리를 찾지 못하기 때문이다. 왜 그럴까? 바로 익명 FTP는 /home/ftp 를 chroot 명령으로 강제로 루트(/) 디렉토리처럼 보도록 만들었기 때문이다. 따라서 여러분이 무심코 /home/ftp 밖에 있는 디렉토리를 접근하려고 하거나 링크하려고 하면 찾을 수 없다고 나온다. 따라서 시디롬을 보통은 /home/ftp 디렉토리에 .1 처럼 도트로 시작하는 디렉토리를 만든 후에 다음과 같이 해준다.

```
mount -t iso9660 /dev/cdrom /home/ftp/.1
```

그리고 /home/ftp 안에서는 모두 다 /.1을 기준으로 시디롬의 각 디렉토리를 링크해두면 된다. 일단 /home/ftp/.1이라는 디렉토리에다가 마운트를 하고 익명 FTP로 들어가게 되면 그 디렉토리는 chroot 명령에 의하여 이제부터는 /.1로만 인식된다. 그것은 FTP 클라이언트나 서버에게도 마찬가지이다. 따라서 만약에 시디롬의 slakware라는 디렉토리를 /pub/slakware라는 디렉토리로 서비스하고 싶다면 다음과 같이 한다.

```
cd /home/ftp
cd pub
ln -s /.1/slakware slakware ( 절대로 /home/ftp/.1이 아니다. )
```

또 다른 고용량 하드디스크를 마운트해서 사용하려고 한다면 마운트를 꼭 읽기 전용으로 해두기 바란다. -o ro 옵션을 꼭 붙이기 바란다.

4.7 링크 주의!

보안에서 정말로 중요시해야 할 것이 있다. 바로 링크의 문제이다. 물론 /home/ftp 라는 디렉토리 구조 밖의 디렉토리를 /home/ftp 안쪽에다 링크해보았자 아무런 소용이 없다. 앞서 말한 바와 같이 익명 FTP 세션에서 chroot가 호출되면 /home/ftp 밖의 디렉토리에 대한 보호가 이루어지기 때문이다. 그래서 위와 같이 특정 파티션을 /home/ftp 안 쪽에 마운트하는 방법을 사용하는데 그러한 파티션은 꼭 익명 FTP 용으로만 쓰기 바란다. 만약 익명 FTP가 아니라 다른 일반적인 용도로 사용하게 된다면 링크를 따라 올라가다 내려오는 순간 우연치 않게 여러분이 원하지 않는 디렉토리를 보여줄 수도 있게 된다. 예를 들어 필자는 알짜웨어 시디롬을 익명 FTP로 제공하기 위해 우선 시디롬을 /home/ftp/.1 이라는 디렉토리에 마운트하였다. 그리고 pub 디렉토리에 가서 다음과 같이 했다고 치자.

```
ln -s /.1/rootdsks rootdsks
```

자, 확인을 하기 위해서 익명 FTP로 들어가보자.

```
cd pub
cd rootdsks
cd ..
```

여기서 우리는 다시 pub로 돌아오는 것이 아니라 .1이라는 디렉토리로 들어가게 된다.

4.8 FTP 서비스의 선전

FTP 서비스의 약점을 하나 들라면 필자는 바로 선전 가능성이라고 말하고 싶다. 각광받고 있는 웹 서비스보다는 자기 자신을 화려하게 선전할 방법을 갖지 못하는 처지이기 때문이다. 따라서 훌륭한 FTP 서비스에 대해서는 웹 페이지를 이용하여 선전을 잘 해주기 바란다. 그래야 사용자들이 많이 이용할 수 있다. 웹 페이지에서 FTP 사이트로 링크를 해주는 것도 좋은 방법이다. 이렇게 하면 웹 서비스와 FTP 서비스가 조화를 이룰 수 있다.

4.9 익명 FTP 접근 권한 세팅

접근 권한에 대한 세팅은 /etc 디렉토리의 ftpusers, ftpgroups라는 파일을 통해 서 한다. ftpusers라는 파일의 내용을 한 번 살펴보자.

```
# The entire line gets matched, so no comments or extra characters on
# lines containing a username.
#
root
uucp
news

# End of ftpusers.
news
```

위에서 나열한 root, uucp, news라는 사용자에 대해서는 FTP 접근 자체를 불 허한다. 즉 ftpusrs에 등록된 사용자는 접근 권한을 받는 것이 아니라 접근 권한을 제한 받는다는 것이다. 이들 열거한 사용자들이 너무도 강력하기 때문에 커다란 문제를 일으킬 소지가 크다. 동시 사용자 제한에 대해서 알아보자. 익명 FTP를 무한

정 모든 사람들이 사용할 수 있게 할 수는 없다. 여러분의 서버 성능과 대역폭에 따라 세팅을 해야 할 것이다. 지역(local)사용자는 대역폭을 크게 요구하므로 더욱 적은 인원으로 제한해야 하며 원격(remote) 사용자는 전송률이 비교적 상당히 떨어지므로 훨씬 많은 인원을 수용할 수 있을 것이다. 그것을 설정하는 파일이 바로 /etc/ftpaccess 이다.

```
limit   local   20  Any                 /etc/msgs/msg.toomany
limit   remote  100 SaSu|Any1800-0600   /etc/msgs/msg.toomany
limit   remote  60  Any         /etc/msgs/msg.toomany
```

우선 첫줄을 보면 지역 사용자는 어느 때든 (Any) 20명으로 제한한다. 그리고 사람이 너무 많아서 접근을 거부할 때는 /etc/msgs/msg.toomany라는 파일 내용을 보여준다. 두 번째 줄의 경우 원격 사용자의 경우 SaSu 즉 토요일 (Saturday), 일요일(Sunday) 또는 어느 날이든 18시부터 06 시까지는 100명으로 제한한다. 이 조건을 만족하지 않으면 60명으로 제한된다. 조건식에 있어서 첫 번째 조건식이 유효하면 그것만 적용한다는 사실을 기억하기 바란다. 두 번째 줄이 원격 사용자에게 적용되면 세 번째 줄은 처리하지 않는다. 사용 인원을 -1로 세팅하면 인원 제한을 없애는 것이다.

제 5 절 메일 서비스

인터네트의 기초 서비스를 들라고 한다면 바로 이메일 서비스를 들 수 있을 것이다. 이메일이야말로 직장인이든 학생이든 인터네트에 대해 얼마나 알고 있냐에 상관없이 한번쯤은 들어보았을 것이고 따라서 많은 사람들이 그 사용법을 알고 있다. 편지를 평생 한 번도 쓰지 않은 사람은 없을 것이다. 그리고 이 시대에 태어나서 이메일 한 번 안보내고 죽는 사람은 없을 것이다. 메일 서비스는 FTP, 웹 서비스와 같은 것이 없어도 의사소통을 가능케 해주는 최소한의 수단이다. 그리고 개인들 사이의 메시지 전달에 쓰이는 절대적으로 필요한 서비스이다.

5.1 이메일 서버?

다른 운영체계에서는 네트스케이프 메일서버라든지 마이크로소프트 메일 등의 다양한 선택권이 있지만 유닉스 계열은 단연 sendmail뿐이라고 말할 수 있다. 다양성의 부족이라기 보다는 sendmail의 전통적인 강력함 때문이라고 말할 수 있다. sendmail이라고 하는 것은 상당히 다루기 힘든 이메일 서버이다. 심지어 센드메일을 세팅할 정도면 그 사람은 진정한 네트워크 관리자라고 말하고 있을 정도이다. 재미있는 것은 센드메일 세팅이 얼마나 사람을 겁먹게 하는지 센드메일 세팅을 2번 이상 해보는 사람은 미친 사람이라는 말도 있을 정도이다. 센드메일을 가지고 여러분은 상상할 수 없는 많은 일을 할 수 있다. 충분하게 공부하려면 마음을 굳게 먹고 오렐리 출판사에서 나온 정말로 두꺼운 센드메일 책을 사서 보면 된다. 하지만 그렇게 겁먹을 필요는 없다. 우리는 아주 간단한 몇 가지 기능만을 사용할 것이기 때문에 해당 사항만을 잠깐만 익히면 된다. 그리고 그 이상의 기능을 쓰는 일은 거의 없을 것이다. 센드메일 최신 버전은 8.9.1이다. 슬랙웨어 3.1 초기판, 알짜슬랙웨어에는 8.8.2 이전 버전이 들어있고 보안에서 약간의 버그가 있다고 보고되고 있다. 알짜웨어의 경우 한글 센드메일은 한글 패치판으로 버전이 훨씬 낮다. 따라서 본격적인 메일 서버로 리눅스 머신을 사용하려는 분들은 다음과 같이 해야 한다.

- 영문 센드메일 8.8.8 이상
- 한글 pine 3.96k

기존의 센드메일은 pkgtool을 실행한 후 remove를 선택하고 sendmail 부분을 지워주면 된다. 지우기에 앞서 우선은 ps aux 명령으로 sendmail의 프로세스를 확인한 후 프로세스를 죽인다. 그리고 나서 실행하기 바란다.

5.2 센드메일 세팅-기본

센드메일을 설치하고 setup을 실행시켜서 시스템 설정을 다시 해준다. configure 메뉴를 선택한 후에 다른 세팅은 변화시키지 말고 계속 취소시킨 후에 SENDMAIL CONFIGURATION에서 SMTP+BIND 또는 SMTP를 선택한다. 전자는 네임서버를 가지고 있는 경우이며 후자는 네임서버를 가지고 있지 않은 경우이다.

5.3 센드메일 세팅-고급

많은 사람들에게는 해당되지 않겠지만 리눅스 서버 한 대를 몇 대의 랜에 물린 컴퓨터들에 대한 메일 게이트웨이(gateway)로 사용하고자 할 때가 있을 것이다. 그리고 다른 많은 용도로 사용할 필요가 있다. 이럴 때는 바로 여러분의 네트워크 관리자로서의 심각한 도전이 이루어진 것일 것이다. 간략한 소개글에서는 충분치 않으므로 꼭 자기보다 훨씬 실력이 많은 리눅서들이나 오렐리 사의 Sendmail책을 구입해보기 바란다.

제 6 절 NFS 서비스

여러분은 Network File System 이라는 것을 들어본 적이 있는지 모르겠다. 유닉스에 관심이 많은 사람들은 이미 들어본 적이 있는 유명한 네트워크 서비스 중 하나이다. 바로 상대방의 파일 시스템 일부를 마치 자기 자신의 디렉토리인 것처럼 사용할 수 있게 해주는 것이 NFS이다. NFS 가 중요한 이유는 엄청난 고용량 하드 디스크를 탑재하고 있는 몇 대의 워크스테이션이 네트워크의 다른 컴퓨터들에게 파일 시스템 서비스를 해줌으로써 모든 컴퓨터들이 필요 이상의 자원을 가질 필요도 없고 소프트웨어를 이리 저리 깔아둘 필요도 없어지게 된다. 생각해보라. 사무실 곳곳 그리고 학교 사무실 곳곳에 아래한글을 설치했다고 치자. 종종 경험해보았을 것이다. 각 사무실에서 아래한글이 지워졌다고 다시 깔아달라고 요청하는 경우가 많다. 일일이 모든 PC에 설치를 한다는 것은 정말로 시간낭비이다. 차라리 한글과 컴퓨터사에 사용인원에 따른 라이센스료를 지불한 다음 중앙 호스트 2-3대 정도의 파티션에다가 아래한글을 설치해둔다. 그리고 그것을 NFS 로 내부 사용자들에게 개방해주는 것이다. 물론 충분한 속도가 제공되어야 할 것이다. 그래서 현재로는 손수 일일이 PC에 아래한글을 하나씩 설치하는 수고를 하고 있을 따름이다. 우선은 클라이언트 관점에서 NFS 마운트를 어떻게 할 것인지를 알아보기로 한다.

6.1 클라이언트로서 NFS 마운트하기

만약 여러분이 속한 랜 환경에서 또는 인터네트에서 누군가 여러분에게 그들의 특정 디렉토리를 NFS로 마운트할 수 있게 허가해주었다고 하자. nfs-host.xxx.co.kr라는 호스트가 자신의 /usr/local 디렉토리를 마운트하도록 허가해주었다고 가정한다. 여러분이 랜이든 PPP 든 간에 TCP/IP 접속을 이루어 냈다면 다음과 같이 하면 된다.

```
# mount -t nfs nfs-host.xxx.co.kr:/usr/local /usr/local
```

위 명령은 상대편 nfs-host.xxx.co.kr의 /usr/local 디렉토리를 자신의 /usr/local 디렉토리로 NFS 방식으로 마운트하는 것이다. 리눅스에서는 마운트 옵션 -o rsize=8192, wsize=8192을 주면 더 효율적인 속도를 얻어낼 수 있다고 한다. 이렇게 하고 나서 cd /usr/local 하면 얼마든지 여러분이 디렉토리인 것처럼 사용할 수 있다. 하지만 대부분의 경우 파일을 지우거나 변경할 수는 없도록 읽기 전용으로만 허가해줄 것이다. 여러분이 윈도95와 같은 것을 사용하고 있다면 셰어웨어나 정품 NFS 클라이언트를 구입해야 할 것이다.

6.2 NFS 서버로서의 준비

여러분이 알짜웨어 또는 대부분의 배포판으로 설치를 하면 이미 NFS 데몬들 이라고 부르는 것들이 그 즉시 사용 가능하도록 되어 있다. 단지 /etc/rc.d/rc.inet2를 약간 수정하면 그만이다.

```
# # Start the various SUN RPC servers.
if [ -f ${NET}/rpc.portmap ]; then
#  # Start the NFS server daemons.
if [ -f ${NET}/rpc.mountd ]; then
  echo -n " mountd"
  ${NET}/rpc.mountd
  fi
if [ -f ${NET}/rpc.nfsd ]; then
  echo -n " nfsd"
  ${NET}/rpc.nfsd
  fi
    .
    .
    .
```

이런 부분을 찾았는가? rpc.mountd와 rpc.nfsd가 작동할 수 있도록 주석문 처 리를 없애도록 한다. 전자는 NFS 마운트 데몬이고 후자는 NFS 파일 서버 데 몬이다. 그 다음은 여러분의 어느 디렉토리를 누구에게 어떤 옵션으로 제공할 것인지를 결정하는 일이 남았다. 여기서 중요한 파일은 바로 /etc/exports라는 파일이다.

```
# See exports(5) for a description.
# This file contains a list of all directories exported to other computers.
# It is used by rpc.nfsd and rpc.mountd.
/home/alzza2    168.126.124.228(ro)
/home/alzza2    206.48.168.180(ro)
/home/yong      localhost(ro)

# sample /etc/exports file
       /               master(rw) trusty(rw,no_root_squash)
/projects       proj*.local.domain(rw)
       /usr            *.local.domain(ro) @trusted(rw)
/home/joe       pc001(rw,all_squash,anonuid=150,anongid=100)
       /pub            (ro,insecure,all_quash)
```

man exports 해보면 나오는 예제 exports 파일의 내용이다. 호스트별로 그리고 사용자별로 괄호 안에 친 옵션을 주고 허가할 수 있도록 해주고 있다. 여러분이 일단 알아두어야 할 옵션은 바로 ro, rw 정도일 것이다. 맨 먼저 master, trusty라는 호스트에 대해서는 루트 디렉토리 / 자체를 읽기/쓰기로 허용하고 있다. 도메인명이 local.domain이고 호스명이 proj로 시작하는 호스트에 대해서는 /projects라는 디렉토리를 읽기/쓰기로 허용하고 있다. 맨 마지막을 보면 모든 호스트에 대하여 /pub 디렉토리를 읽기 전용으로 허용해주고 있다. 보는 바와 같이 / 디렉토리는 다른 호스트에 그리고 그 하부 디렉토리에 해당하는 /pub는 일반인들에게 허용하는 식으로 각 파티션이 따로 분리되지 않아도 중복해서 어떤 디렉토리와 그 하부디렉토리를 별도로 마운트를 허용할 수 있다. 너무 간단하지 않은가?

제 7 절 리눅스 서버 시스템에 대한 이해

리눅스의 인터네트 서비스에는 그에 해당되는 데몬들이 있기 마련이다. 데몬들은 정확히 서버에서 활동 중인 프로세스를 말하는데 평소에는 쿨쿨 잠만 자고 있다가 클라이언트의 접속 요구가 생기면 깨어난다. 텔넷 서비스에는 telnet 데몬이, FTP 서비스에 대해서는 ftp 데몬이, 메일 서비스에 대해서는 mail 데몬이 그리고 NFS 서비스에 대해서는 nfs 데몬이, 마지막으로 웹 서비스에 대해서는 웹 데몬이 필요하다. 메일에 대해서는 sendmail 또는 smail이라는 데몬이 그리고 NFS에 대해서는 전술한 rpc.mountd, rpc.nfsd가 협동해야 하고 웹 서비스는 보통 httpd라는 데몬이 있는데 telnet과 ftp 의 경우에는 그렇게 단독으로 뜬다기 보다는 inetd 수퍼 데몬이라는 것에 의해서 제어를 받아서 필요할 때만 실행되도록 되어 있다. 부팅 과정을 잘 살펴보면 inetd라는 것이 뜨

는 것을 볼 수 있다. inetd 데몬에 중요한 설정파일은 /etc/inetd.conf라는 파일이다. 그 내용을 일부만 살펴보도록 하자.

```
# See "man 8 inetd" for more information.
#
# If you make changes to this file, either reboot your machine or send the
# inetd a HUP signal:
# Do a "ps x" as root and look up the pid of inetd. Then do a
# "kill -HUP <pid of inetd>".
# The inetd will re-read this file whenever it gets that signal.
#
# <service_name>  <sock_type> <proto>   <flags> <user> <server_path>
<args>
#

#
# These are standard services.
#
ftp     stream  tcp     nowait  root    /usr/sbin/tcpd  wu.ftpd
telnet  stream  tcp     nowait  root    /usr/sbin/tcpd  in.telnetd
nntp    stream  tcp     nowait  root    /usr/sbin/tcpd  in.nntpd
# The comsat daemon notifies the user of new mail when biff is set to y:
comsat        dgram   udp     wait    root    /usr/sbin/tcpd  in.comsat
#
# Shell, login, exec and talk are BSD protocols.
#
shell   stream  tcp     nowait  root    /usr/sbin/tcpd  in.rshd -L
login   stream  tcp     nowait  root    /usr/sbin/tcpd  in.rlogind
# exec  stream  tcp     nowait  root    /usr/sbin/tcpd  in.rexecd
# talk  dgram   udp     wait    root    /usr/sbin/tcpd  in.talkd
ntalk   dgram   udp     wait    root    /usr/sbin/tcpd  in.talkd
#
# Pop et al
#
# pop2  stream  tcp     nowait  root    /usr/sbin/tcpd  in.pop2d
pop3    stream  tcp     nowait  root    /usr/sbin/tcpd  in.pop3d

#
# Finger, systat and netstat give out user information which may be
# valuable to potential "system crackers."  Many sites choose to disable
# some or all of these services to improve security.
# Try "telnet localhost systat" and "telnet localhost netstat" to see that
# information yourself!
#
finger  stream  tcp     nowait  nobody  /usr/sbin/tcpd  in.fingerd -w
systat  stream  tcp     nowait  nobody  /usr/sbin/tcpd  /bin/ps -auwwx
netstat stream  tcp     nowait  root    /usr/sbin/tcpd  /bin/netstat -a
```

위 설정 파일을 조금이라도 바꾸는 경우에는 앞 부분에서 설명한 것처럼 inetd를 kill -HUP 해서 프로세스를 죽인 후 다시 띄워야 한다. 위에서 열거한 것은 시스템 운영에서 아주 중요한 데몬들이라고 할 수 있다. 이 모든 것을 관리하므로 수퍼 데몬이라고 할 수 있다. 참고로 /etc/services라는 파일을 보면 여러분의 서비스가 어떤 프로토콜(TCP 인가? UDP 인가?)인가 그리고 어떤 포트를 사용하는가에 대해서 감잡을 수 있으리라 본다. 꼭 한 번 파일의 내용을 모르더라도 훑어보기 바란다. 마지막으로 위의 inetd.conf 파일을 잘 살펴보면 그냥 in.telnetd를 수행시키는 것이 아니라 /usr/sbin/tcpd라는 것에 의해서 보호되어(Wrap) 수행된다. 이렇게

tcpd라고 부르는 것에 의해 수행되는 서비스들은 접근 제어와 모니터링이 가능해진다. 맨 페이지를 보면 정말 장황하게 설명되어 있으므로 참고하기 바란다. 일단은 tcpd에 의해서 어떻게 호스트별 접근 제한을 가할 수 있는지 알아보자. 악독한 사용자들이 많은 호스트는 접근을 거부해보자. 서로 크랙킹을 많이 하는 학교 과들 사이에서는 접근 제한을 가하는 것도 좋을 것이다. 관계된 파일은 2개이다. /etc/hosts.allow 와 /etc/hosts.deny 즉 전자는 허용하는 사이트, 후자는 거부하는 사이트가 된다. 규칙은 /etc/hosts.deny에 넣는 호스트에서는 접근이 불허된다. 하지만 /etc/hosts.allow에 들어간 사이트는 전자에 거부 호스트 목록에 상관없이 접근이 허용된다. 따라서 /etc/hosts.allow는 잘 작성해야 한다. 예제를 한 번 살펴보기 바란다. 둘 다 형식은 같으니 하나만 알아보겠다.

```
데몬 리스트: 호스트 리스트
ALL: ALL EXCEPT terminalserver.foobar.edu
in.talkd: ALL
in.ntalkd: ALL
in.fingerd: ALL
in.ftpd: LOCAL, .my.domain
ALL EXCEPT in.fingerd: other.host.name
```

콜론(:)앞에다 서비스 데몬 리스트를 적어주고 뒤에는 호스트명을 적어준다. 위의 예에서 보면 ALL, EXCEPT, LOCAL 등의 특수한 명칭이 나오는데 각각은 여러분이 사전에서 찾을 수 있는 의미와 같다고 보면 된다. ALL과 EXCEPT는 데몬 리스트와 호스트 리스트 양자에 쓰일 수 있다. 보안이 정말로 문제되는 사이트에서는 우선은 hosts.deny 파일에 ALL: ALL이라고 써준다. 이렇게 하면 모든 호스트에 대하여 어떤 서비스도 일단 불허해놓는다. 그리고 hosts.allow에서 하나씩 허용해주면 된다. 자세한 사항은 맨 페이지를 보면 된다.

```
$ man 5 hosts_access
```

제 8 절 아파치(Apache) HTTP 서버 설치와 운영

단순한 클라이언트로의 전락! 인터네트 사용인구가 늘어나긴 해도 그리고 네트워크 속도가 빨라지기는 했지만 개인 사용자들은 여전히 진정한(?) 사용자 수준에 머물고 있다. 하지만 PC 한 대와 리눅스 그리고 적당한 속도의 네트워크만 있다면 이제 여러분의 리눅스 머신은 더 이상 클라이언트 머신이 아니다. 웹서비스를 상용으로 할 것이 아니라면 네트워크 속도는 모뎀속도 정도만 돼도 상관없지 않을까? SLIP이나 PPP로 동적 IP를 할당받아 가끔 운영해보는 개인 서버도 재미있을 것이다. 잠시 서버로 작동하는 여러분의 리눅스 머신으로 친구들을 초대해보는 것도 좋다. 현재 조사된 바에 의하면 전세계적으로 가장 많이 사용되는 웹 서버는 단연 아파치 서버이며 우리나라에서는 NCSA서버를 가장 많이 사용한다고 한다. 몇 가지 문서를 찾아보면 우리나라에서 NCSA서버가 가장 많이 사용되는 이유로 한글화된 문서가 있다는 사실을 들고 있다. NCSA 서버의 버그를 수정하면서 (아파치 서버의 이름은 NCSA 1.3에 대한 A PAtCHy 버전이라는 말로부터 유래된 것이라 한다), 가장 빠른 서버로 인정되고 있으며 전세계적으로 가장 많이 사용되는 아파치 서버에 대한 한글 문서가 나온다면 좋은 성능의 아파치 서버가 국내에서도 많이 사용되리라 믿으며 이 글을 적는다. 필자가 접할 수 있고 마음대로 사용할 수 있는 유일한 시스템은 리눅스이기 때문에 리눅스에서의 아파치 서버 설치/운영 방법과 사용자 확인(User Authentification)에 대해 간단히 기술한다. 아파치 서버 대부분의 기능은 NCSA 서버로부터 물려받은 것이기 때문에 NCSA 서버 문서만으로도 충분하지만 아파치 서버 특유의 기능 또는 확장된 개념들이 존재하므로 NCSA에서 찾아 볼 수 없는 부분을 주목하고 싶다. 다른 운영체계에서 설치, 운영하는 법이나 더 많은 정보를 원한다면 이 글 마지막에 있는 여러 참고자료들을 직접 찾아보기 바란다.

8.1 준비사항

- 1.1.1
- 자신에게 필요한 모듈들, 예를 들어, 관계형 데이터베이스 포스트그래스 95를 위한 mod_auth_pg95.c 등
- 리눅스 머신 한 대

- IP 주소; DNS 등록이 되어 있다면 좋고 그렇지 않아도 서버 운영은 가능하다.

8.2 설치

1) 아파치 서버 소스 풀기 먼저 아파치 소스를 /usr/local/etc에 복사하고 압축을 푼다.

```
/usr/local/etc# tar xvzf apache_1.1.1.tar.gz
```

apache_1.1.1이라는 디렉토리가 생기며 아파치 서버와 관련된 모든 파일이 풀려 나온다. 일반적으로 아파치 서버 배포본은 httpd라는 디렉토리 아래에 설치되므로 링크를 만들어주거나 httpd 라는 이름을 바꾸도록 하자.

```
/usr/local/etc# mv apache_1.1.1 httpd
```

또는

```
/usr/local/etc# ln -s apache_1.1.1 httpd
```

2) 디렉토리 설명 압축을 풀었다면 다음과 같은 내용을 볼 수 있다.

```
-rw-r--r--     1 129   kmem  3738 Mar 13  1996 CHANGES
-rw-r--r--     1 129   kmem   2604 Feb 22  1996 LICENSE
-rw-r--r--     1 129   kmem   3059 Jul  3 15:24 README
drwxr-xr-x     2 129  kmem   1024 Jul  9 04:04 cgi-bin/
drwxr-xr-x     2 129   kmem  1024 Jul  9 04:04 conf/
drwxr-xr-x     2 129   kmem  1024 Jul  9 04:04 htdocs/
drwxr-xr-x     2 129  kmem   2048 Jul  9 04:04 icons/
drwxr-xr-x     2 129  kmem   1024 Jul  9 04:04 logs/
drwxr-xr-x     2 129  kmem   2048 Jul 10 00:02 src/
drwxr-xr-x     2 129  kmem   1024 Jul  9 04:05 support/

디렉토리명   디렉토리 내용
cgi-bin      CGI 스크립트 모음
conf         아파치 실행할 때 환경 초기값 설정파일
htdocs       일반적인 서버 홈페이지 디렉토리
icons        아파치 서버 자체적으로 사용하는 아이콘
logs         여러 가지 사용 기록 저장 파일
src          아파치 소스
support      서버의 보안에 관련된 프로그램들이 들어있다.
```

8.3 컴파일 전 src/Configuration 파일 수정

먼저 Configuration.tmpl 파일을 Configuration으로 복사한다.

```
/usr/local/etc/httpd/src# cp Configuration.tmpl Configuration
```

에디터로 Configuration파일을 불러들여 수정을 시작해 보자.

- 컴파일러 설정은 gcc로 되어 있을 것이다. 리눅스는 이 세상에서 제일 멋진 C 컴파일러를 기본으로 사용하고 있다.
- CFLAGS 등 컴파일러 옵션 설정 모듈과 관련된 옵션들도 이곳에 적어 넣는다.

```
CFLAGS= -O2
LFLAGS=
EXTRA_LIBS=
```

- 우리는 리눅스에 설치를 하므로 리눅스에 해당하는 부분의 #표시를 없애도 록 한다.

```
AUX_CFLAGS= -DLINUX
```

- 모듈 설정

Configuration 파일에 있는 모듈의 목록은 활성화되든 되지 않든 상관없이 모두 src 디렉토리에 소스로 포함되어 있다. 하지만 새로운 기능의 모듈들 예를 들어 포스트그래스95용 모듈인 mod_auth_pg95을 사용하기 위해서는 해당 소스를 src 디렉토리에 포함시키고 Configuration파일에도 포함시켜야 한다. 모듈은 파일의 이름을 보면 그 기능을 대강 알 수 있으며 더 자세히 알고자 한다면 에서 많은 정보를 얻을 수 있을 것이다. 기본적으로 제공되고 활성화된 모듈은 다음과 같다.

```
mime_module
access_module
auth_module
negotiation_module
includes_module
dir_module
cgi_module
userdir_module
alias_module
env_module
common_log_module
```

기본적으로 제공되나 사용자가 명시적으로 활성화시켜야 하는 것들은 다음과 같다.

```
asis_module
imap_module
action_module
agent_log_module
referer_log_module
config_log_module
cern_meta_module
status_module
info_module
anon_auth_module
db_auth_module
dbm_auth_module
msql_auth_module
digest_module
dld_module
cookies_module
proxy_module
```

8.4 컴파일

모듈설정까지 마쳤으면 이제 저장을 하고, ./Configure를 실행시킨 뒤 컴파일을 한다.

```
/usr/local/etc/httpd/src# ./Configure
Using 'Configuration' as config file
/usr/local/etc/httpd/src# make
```

8.5 서버 환경 설정

컴파일이 성공적으로 완료되었으면 src 디렉토리에 httpd가 생성되었을 것이다. 이제는 서버의 환경을 설정해 주자. conf 디렉토리에는 4개의 환경 설정파일이 있다. 그 중 *.conf-dist 파일을 *.conf로 복사하고 그 복사본을 가지고 서버의 환경설정을 해주자.

```
/usr/local/etc/httpd/conf# cp access.conf-dist access.conf
/usr/local/etc/httpd/conf# cp httpd.conf-dist httpd.conf
/usr/local/etc/httpd/conf# cp srm.conf-dist srm.conf
```

access.conf파일 수정

Directory에 지정된 디렉토리를 서버의 홈디렉토리로 지정한다.

```
<Directory /usr/local/etc/httpd/htdocs>
```

Options Indexes FollowSymLinks

가능한 옵션은 다음과 같다.

```
None : 어떤 옵션도 이용할 수 없다.
All : MultiViews 옵션을 이용할 수 있다.
Indexes : URL에 지정된 디렉토리에 (index.html 같은)지정된 파일이 없을
          경우 디렉토리의 파일 목록을 보여주는 옵션.
Includes : 서버측의 추가적인 정보를 제공할 수 있게 한다.
FollowSymLinks : 디렉토리의 심볼릭 링크를 사용 가능하게 한다.
ExecCGI : CGI 스크립트를 실행할 수 있게 한다.
MultiViews : All 옵션이 설정되었을 때만 지정된 목록의 multiviews를 허용한다.

# This controls which options the .htaccess files in directories can
# override. Can also be "All", or any combination of "Options", "FileInfo",
# "AuthConfig", and "Limit"
AllowOverride None
```

.htaccess파일은 서버의 각 디렉토리에 대한 접근을 제어하기 위한 것으로 디렉토리에 .htaccess파일이 있으면 서버 전체에 작용하는 access.conf보다 우선권을 가진다. 다음은 .htaccess파일에 대한 Override에 대한 옵션이다.

```
None : .htaccess파일을 읽을 수 없게 한다.
All : 모든 지정에 대해 가능하게 한다.
Options : 규정된 디렉토리 형식을 컨트롤하는 지정의 사용을 허락한다.
FileInfo : 문서형식을 컨트롤하는 지정의 사용을 허용한다.
AuthConfig : 사용자 인증 지정의 사용을 허용한다. 사용자 인증 변수를 사용한다.
Limit : 호스트 접근을 컨트롤하는 지정을 허용한다.

# Controls who can get stuff from this server.
order allow,deny
allow from all

</Directory>
```

order : 서버가 액세스 컨트롤을 수행하는 순서를 나타낸다. allow기능을 먼저 수행하고, deny기능을 수행하라는 것이다.
allow from : 나열되는 주소들에 대한 액세스 컨트롤을 가능하게 한다. 사용 가능한 주소는 도메인 네임, 호스트 이름 주소, 호스트 ip 주소, ip주소의 앞부분 3바이트, 모든 주소에 해당하는 all이 있다.
deny from : allow from과 반대되는 개념이며 사용 가능한 주소는 allow from 과 같다.
require : 사용자, 그룹에 대한 접근을 통제할 수 있다.
사용방법 : require entity en1 en2 ... enn
entity에 들어갈 수 있는 것은 user, group, valid-user의 세 가지이다.
user : 지정된 사용자들에게만 접근을 허용하는 것으로 지정된 사용자에 대한 정보는 AuthUserFile에서 지정한 파일에 있다.
group : 지정된 그룹에게만 접근을 허용하는 것으로 지정된 그룹에 대한 정보는 AuthGroupFile에서 지정한 파일에 있다.
valid-user : AuthUserFile에 있는 모든 사용자에 대해 접근을 허용한다.

```
# /usr/local/etc/httpd/cgi-bin should be changed to whatever your
# ScriptAliased CGI directory exists, if you have that configured.
```

cgi-bin디렉토리에 대한 Options를 지정한다. 옵션은 htdocs 설정의 옵션과 같다.

```
<Directory /usr/local/etc/httpd/cgi-bin>
AllowOverride None
Options None
</Directory>
```

서버의 상태결과를 http://servename/status의 URL에 접근하면 볼 수 있게 해주는 옵션이다. 'allow from 서버 도메인 네임' 의 형식으로 접근이 가능하다.

```
#</Location /status>
#SetHandler server-status
#order deny,allow
#deny from all
#allow from .nowhere.com
#</Location>
```

그 외 접근 제어에 대한 관련 사항을 추가로 작성할 수 있다.

httpd.conf

```
# ServerType is either inetd, or standalone.
ServerType standalone
```

서버의 타입을 지정하는 것으로 standalone은 httpd 데몬 프로세스가 사용자의 요청을 처리하는 것이고 inetd는 inetd 데몬 프로세스가 처리하게 하는 것이다. standalone이 더 효율적인 방법이다.

```
Port 80
```

시스템에 의해 미리 httpd를 위해 예약된 포트 번호는 80번이다. 0에서 1023까지의 포트번호는 시스템에 의해 미리 예약되어 있다. 그 이상의 포트번호를 지정하여 일반사용자도 httpd을 설치, 운영 가능하다. http://aaa.bbb.ccc:8080/ 등으로 사용할 수 있다.

```
HostnameLookups on
웹서버에 대한 접근을 도메인 네임이나 ip주소 (on) 또는 ip주소만으로(off)
접근하게 할 것인지를 결정하는 것이다.

# If you wish httpd to run as a different user or group, you must run
# httpd as root initially and it will switch.
# User/Group: The name (or #number) of the user/group to run httpd as.
#  On SCO (ODT 3) use User nouser and Group nogroup User nobody
Group #-1
ServerType이 standalone일 때만 적용되는 것으로 서버가 사용자의 요청에 대해서
생성하는 child httpd 프로세스에 대한 user id, group id 이다. 일반적으로
시스템에서 사용하지 않는 것들로 지정하는 것이 바람직하다. 슬랙 3.1기반
배포본에는 /etc/group에 nogroup -2가 지정되어 있다.

# ServerAdmin: Your address, where problems with the server should be
# e-mailed.
ServerAdmin you@your.address
서버 관리자의 e-mail 주소를 지정한다.

# ServerRoot: The directory the server's config, error, and log files
# are kept in
ServerRoot /usr/local/etc/httpd
서버의 Root 디렉토리를 지정한다.

# ErrorLog: The location of the error log file. If this does not start
# with /, ServerRoot is prepended to it.
ErrorLog logs/error_log
서버에서 발생하는 에러를 기록하는 로그 파일을 지정한다.

# TransferLog: The location of the transfer log file. If this does not
# start with /, ServerRoot is prepended to it.
TransferLog logs/access_log
서버에 대한 사용자들의 엑세스 정보를 기록하는 로그파일을 지정한다.

# PidFile: The file the server should log its pid to
PidFile logs/httpd.pid
httpd 데몬 프로세스의 프로세스 id를 기록하고 있는 파일을 지정한다.

# ScoreBoardFile: File used to store internal server process information
ScoreBoardFile logs/apache_status
서버 프로세스의 정보를 담고 있는 파일을 지정한다.

# ServerName allows you to set a host name which is sent back to clients
# for your server if it's different than the one the program would get (i.e.
# use "www" instead of the host's real name).
#
# Note: You cannot just invent host names and hope they work. The
# name you define here must be a valid DNS name for your host. If you
# don't understand this, ask your network administrator.
#ServerName new.host.name
```

```
서버의 도메인 네임을 지정한다. 자신의 서버가 도메인 네임을 가지지 않았다면
놔둔다. 가졌다면 활성화 시켜주고 도메인 네임을 기입한다.

# Timeout: The number of seconds before receives and sends time out
#  n.b. the compiled default is 1200 (20 minutes !)
Timeout 400
클라이언트가 정보를 받을 때까지 소요되는 대기시간의 최대 값을 지정한다.
네트워크가 응답이 늦을 수록 수치를 늘리는 것이 좋다.

# KeepAlive: The number of Keep-Alive persistent requests to accept
# per connection. Set to 0 to deactivate Keep-Alive support
KeepAlive 5
접속된 채로 특별한 요청이 없음에도 계속 연결을 유지시킬 수치를 지정한다.

# KeepAliveTimeout: Number of seconds to wait for the next request
KeepAliveTimeout 15
KeepAlive가 지정되었을 때 다음 요청을 기다리는 수치를 초 단위로 지정한다.

# Server-pool size regulation. Rather than making you guess how many
# server processes you need, 아파치 dynamically adapts to the load it
# sees --- that is, it tries to maintain enough server processes to
# handle the current load, plus a few spare servers to handle transient
# load spikes (e.g., multiple simultaneous requests from a single
# Netscape browser).
# It does this by periodically checking how many servers are waiting
# for a request. If there are fewer than MinSpareServers, it creates
# a new spare. If there are more than MaxSpareServers, some of the
# spares die off. These values are probably OK for most sites ---
MinSpareServers 5
MaxSpareServers 10
httpd 데몬프로세스의 child 프로세스에 대해 MinSpare Servers보다 작으면 새
로운 프로세스를 생성하고 MaxSpare Servers보다 많으면 여분의 프로세스를
죽이는(kill) 것을 지정한다.

# Number of servers to start --- should be a reasonable ballpark figure.
StartServers 5
httpd 서버를 처음 실행시킬 때 여분의 프로세스를 생성시킬 수치를 지정한다.

# Limit on total number of servers running, i.e., limit on the number
# of clients who can simultaneously connect --- if this limit is ever
# reached, clients will be LOCKED OUT, so it should NOT BE SET TOO
LOW.
# It is intended mainly as a brake to keep a runaway server from taking
# Unix with it as it spirals down...
MaxClients 150
한번에 접근 가능한 클라이언트의 수치를 지정한다.

# MaxRequestsPerChild: the number of requests each child process is
# allowed to process before the child dies.
# The child will exit so as to avoid problems after prolonged use when
```

```
# 아파치 (and maybe the libraries it uses) leak. On most systems, this
# isn't really needed, but a few (such as Solaris) do have notable leaks
# in the libraries.
MaxRequestsPerChild 30
child 프로세스가 응답할 수치를 지정한다. 아마 리눅스에서는 상관 없는 듯하다.

# Proxy Server directives. Uncomment the following line to
# enable the proxy server:
#ProxyRequests On
아파치 서버는 프락시 서버의 기능도 하는데 on으로 활성화 시켜주면 된다.

# To enable the cache as well, edit and uncomment the following lines:
#CacheRoot /usr/local/etc/httpd/proxy
#CacheSize 5
#CacheGcInterval 4
#CacheMaxExpire 24
#CacheLastModifiedFactor 0.1
#CacheDefaultExpire 1
#NoCache adomain.com anotherdomain.edu joes.garage.com
프락시 서버를 지정했을 때 필요한 설정들이다.
Virtual 호스트 관련된 부분은 mini FAQ의 Virtual-Web문서를 읽어보기 바란다.
```

srm.conf

```
# DocumentRoot: The directory out of which you will serve your
# documents. By default, all requests are taken from this directory, but
# symbolic links and aliases may be used to point to other locations.
DocumentRoot /usr/local/etc/httpd/htdocs
'access.conf'의 DocumentRoot 지정부분과 동일하다.
# UserDir: The name of the directory which is appended onto a user's
# home directory if a ~user request is recieved.
UserDir public_html
시스템을 사용하는 일반 사용자들의 개인 웹 홈 디렉토리를 지정한다.
$HOME/public_html 디렉토리를 만들어서 홈페이지를 등록할 수 있다.
디렉토리명은 서버 관리자가 변경할 수 있다.

# DirectoryIndex: Name of the file or files to use as a pre-written HTML
# directory index. Separate multiple entries with spaces.
DirectoryIndex index.html
특정 파일을 지정하지 않고 디렉토리만 지정했을 때 불러들일 문서를 지정한다.

# FancyIndexing is whether you want fancy directory indexing or
# standard
FancyIndexing on
디렉토리의 내용을 보여줄 때 지정된 아이콘으로 보여줄 것인지 지정한다.

# AddIcon tells the server which icon to show for different files or
# filename extensions
AddIconByEncoding (CMP,/icons/compressed.gif) x-compress x-gzip
AddIconByType (TXT,/icons/text.gif) text/*
....(중략)
```

```
AddIcon /icons/folder.gif ^^DIRECTORY^^
AddIcon /icons/blank.gif ^^BLANKICON^^
지정된 아이콘 목록이다.

# DefaultIcon is which icon to show for files which do not have an icon
# explicitly set.
DefaultIcon /icons/unknown.gif
아이콘 목록에 없는 파일을 보여줄 때 사용될 아이콘을 지정한다.

# AddDescription allows you to place a short description after a file in
# server-generated indexes.
# Format: AddDescription "description" filename
# ReadmeName is the name of the README file the server will look for
# by default. Format: ReadmeName name
#
# The server will first look for name.html, include it if found, and it will
# then look for name and include it as plaintext if found.
#
# HeaderName is the name of a file which should be prepended to
# directory indexes.
ReadmeName README
HeaderName HEADER
디렉토리 목록을 보여줄 때 목록의 마지막 부분 뒤와 목록의 시작 전에 보여줄
내용을 담고 있는 파일을 지정한다. 여기서는 목록의 뒤에 README, 목록의
처음에 HEADER를 보여주게 지정되어 있다.

# IndexIgnore is a set of filenames which directory indexing should ignore
# Format: IndexIgnore name1 name2...
IndexIgnore */.??* *~ *# */HEADER* */README* */RCS
디렉토리를 출력할 때 무시할 파일들을 지정한다.

# AccessFileName: The name of the file to look for in each directory
# for access control information.
AccessFileName .htaccess
서버가 디렉토리를 출력할 때 참고할 파일을 지정한다.

# DefaultType is the default MIME type for documents which the server
# cannot find the type of from filename extensions.
DefaultType text/plain
사용자의 요청으로 각 파일에 대한 행동을 취할 때 서버가 알지 못하는 형식의
파일일 경우 표시 형식을 지정한다.

# AddEncoding allows you to have certain browsers (Mosaic/X 2.1+)
# uncompress information on the fly. Note: Not all browsers support this.
AddEncoding x-compress Z
AddEncoding x-gzip gz
압축 코드에 대한 인코딩정보를 지정한다.

# Redirect allows you to tell clients about documents which used to exist
# in your server's namespace, but do not anymore. This allows you to tell
```

```
# theclients where to look for the relocated document.
# Format: Redirect fakename url
```
가상 자료파일을 url에 지정된 문서로 Redirect 한다.

```
# Aliases: Add here as many aliases as you need (with no limit). The for
# mat is Alias fakename realname
#Alias /icons/ /usr/local/etc/httpd/icons/
```
파일명이나 디렉토리명을 alias 할 수 있다. 활성화 시켜준다.

```
# ScriptAlias: This controls which directories contain server scripts.
# Format: ScriptAlias fakename realname
#ScriptAlias /cgi-bin/ /usr/local/etc/httpd/cgi-bin/
```
서버에서 사용하는 cgi를 담은 디렉토리를 지정한다. 이 디렉토리의 파일들은 서버에 의해 cgi스크립트로 인식된다. 활성화 시켜준다.
AddType은 ext에 해당하는 확장자를 가진 파일을 mime.types에 기술된 mime type형태로 정의 하는 것이다.
형식 : AddType type/subtype ext1
AddHandler역시 ext에 해당하는 확장자에 대한 행동 양식을 정의하는 것이지만, 일반적인 확장자를 가진 파일들에 대한 정의를 한다.
형식 : AddHandler action-name ext1

```
# To use CGI scripts:
#AddHandler cgi-script .cgi
```
서버의 어떤 위치에 있던지 .cgi 확장자를 가진 파일은 cgi-script로 인식하게 한다.
.pl, .sh 등의 다른 확장자도 추가할 수 있다.

```
# To use server-parsed HTML files
#AddType text/html .shtml
#AddHandler server-parsed .shtml
```
Server Side Includes (SSI)를 사용할 때 필요하다. SSI는 HTML파일 속에 어떤 실행 프로그램의 결과나 특정 파일을 포함할 수 있게 한다.

```
# Uncomment the following line to enable 아파치's send-asis HTTP file
# feature
#AddHandler send-as-is asis
```
mime type sttpd/send-as-is에 대한 설정이다.

```
# If you wish to use server-parsed imagemap files, use
#AddHandler imap-file map
```
mime type application/x-httpd-imap에 대한 설정이다.

지정된 media type에 대해 해당 스크립트를 실행 가능하게 할 수 있다.
형식 : Action media/type /cgi-script/location
형식 : Action handler-name /cgi-script/location
예) Action text/html /cgi-bin/footer
CERN HTTP 서버의 meta information을 emulate해 준다. 자세한 내용은 CERN HTTP 서버문서를 읽어보길 바란다.

```
# MetaDir: specifies the name of the directory in which 아파치 can find
```

```
# meta information files. These files contain additional HTTP headers
# to include when sending the document

#MetaDir .web

# MetaSuffix: specifies the file name suffix for the file containing the
# meta information.

#MetaSuffix .meta


# Customizable error response (아파치 style)
# these come in three flavors
# 서버 에러에 대한 응답을 지정해 줄 수 있습니다.
# 각 에러 코드에 대한 응답을 cgi나 일반 텍스트로 만들어서 사용자에게
보여줄 수 있다. 어떤 서버에 접속하면 해당 URL이 없다는 등의 한글메시지가
가능한 것도 이것을 이용하는 것이다.
#    1) plain text
#ErrorDocument 500 "The server made a boo boo.
#  n.b.  the (") marks it as text, it does not get output
#
#    2) local redirects
#ErrorDocument 404 /missing.html
#  to redirect to local url /missing.html
#ErrorDocument 404 /cgi-bin/missing_handler.pl
#  n.b. can redirect to a script or a document using server-side-includes.
#
#    3) external redirects
#ErrorDocument 402 http://other.server.com/subscription_info.html
#
```

mime.types

```
mime.types파일은 서버에 의해 리턴되는 파일의 파일명과 MIME형식을 기술하는
것에 대해 규정한다.
type/subtype ext1 ext2 ext3 ...으로 기술되며 ext1 등은 파일명이 해당 확장자를
가졌을 때 클라이언트에게 전달되는 확장자이다.
예) video/mpeg                     mpeg mpg mpe
```

8.6 서버의 실행

위에서와 같이 환경설정을 마쳤으면, 아파치 서버를 실행시켜보자. 루트 권한으로

`/usr/local/etc/httpd/src/httpd -f /usr/local/etc/httpd/conf/httpd.conf`

를 실행해도 되며 /etc/ rc.d/rc.local 등에 위의 내용을 삽입하면 부팅할 때 자동으로 아파치 서버가 작동한다. 사용 가능한 옵션은 다음과 같다.

```
-d serverroot 보통 httpd.conf에 지정된 ServerRoot 변수에 대한 지정을 할
              수 있게 한다. 디폴트는 /usr/local/etc/httpd 이다.
```

-f config	환경설정 파일을 지정해서 시작하게 한다. 지정된 config파일이 / 로 시작하지 않으면 ServerRoot에 대한 상대패스로 인식한다. 디폴트는 conf/httpd.conf 이다.
-X	내부적인 테스트를 위해 싱글 프로세스 모드로 실행시킨다. 서버 데몬은 아들 프로세스를 fork하지 않는다. 웹서비스를 하려면 이 모드를 사용하지 말라.
-v	httpd의 버전을 프린트하고 실행을 마친다.
-?	httpd의 옵션을 프린트하고 실행을 마친다.

8.7 사용자 인증(User Authentication)

사용자가 문서에 접근하려 할 때 사용자를 확인하는 것을 말한다. 이것은 사용자의 아이디와 패스워드를 확인함으로써 가능하다. 사용자 인증을 위해서는 다음 환경변수를 지정해 주어야 한다. conf/access.conf파일에 지정하면 서버의 모든 디렉토리를 대상으로 하고 각 디렉토리에 conf/srm.conf의 Access FileName변수에서 지정한 파일이름(보통 .htaccess)이 존재하면 그 파일을 참조한다. conf/access.conf 파일의 디렉토리 지정부분에 AllowOverride AuthConfig지정을 해주면 아래의 환경변수를 사용할 수 있다. access.conf파일의 설명을 참조하기 바란다.

AuthType Basic

사용하는 인증의 방식을 지정한다. 현재까지 Basic만 구현되어 있다.

AuthName somedomain

사용자 인증을 확인하는 확인처의 이름을 밝힌다. 여러 이름을 나열할 수 있다.

AuthUserFile /web/users

인증된 사용자의 데이터를 보관할 파일을 지정한다. support디렉토리에 보면 몇 가지 보안에 관련된 프로그램들이 소스 형태로 존재한다. 이것을 make명령을 주어 컴파일하고 나면 실행 가능한 파일들 중 htpasswd라는 명령이 생성되며 이 명령이 사용자를 등록시키는 일을 수행한다. /web/에 지정된 디렉토리에서 ../support/htpasswd -c users username 라고 해주면 사용자가 생성되며 새로운 사용자 등록은 -c 옵션을 빼고 명령을 주면 된다. 사용자 삭제는 해당 파일에서 사용자 데이터 부분을 삭제해주면 된다.

AuthGroupFile /web/groups

인증된 그룹의 데이터를 보관한다. 이 파일은 직접 만들어 주면 된다.

```
예) group1: user1 user2 user3
    group2: user2 user5 user6
```

다음은 /usr/local/etc/httpd/htdocs 디렉토리의 접근 제어를 AuthUserFile에 지정된 사용자에 대해서만 접근을 허용하도록 설정을 마친 access.conf의 내용이다.

```
<Directory /usr/local/etc/httpd/htdocs>
Options Indexes FollowSymLinks
AllowOverride AuthConfig
order allow,deny
allow from all
AuthType Basic
AuthName liberty
AuthUserFile /usr/local/etc/httpd/conf/users
AuthGroupFile /usr/local/etc/httpd/conf/group
```

```
require valid-user
</Directory>
```

8.8 리눅스 파일 시스템 표준에 따르기

알짜웨어에 있는 아파치 1.0 을 설치하면 아파치 실행 파일을 제외하고는 cgi-bin, conf, htdocs, icons, logs 등의 디렉토리가 모두 /var/lib/httpd라는 디렉토리 아래 있음을 알 수 있다. 리눅스 파일 시스템 표준에서는 시스템 운영 도중에 갱신되는 자료들에 대해서는 /usr 파티션 아래 두는 것을 권장하지 않는다. 특히 htdocs, logs 등은 빈번하게 변화하는 디렉토리이므로 /var에 위치하고 있어야 한다. 이전 아파치 서버를 설치했다면 우선은 pkgtool을 가지고 삭제를 한다. 그리고 /usr/local/etc/httpd에 있는 것들을 모두 /var/lib/httpd로 옮긴 다음 /usr/local/etc/httpd는 단지 /var/lib/httpd에 대한 링크로 만들어두는 것이 좋다. 파일 시스템 표준에 대해서 잘 모르는 사람은 그냥 넘어가도 좋다.

8.9 마치며

가장 좋은 웹서버로 인정되는 아파치 서버가 단지 소개 글 하나 없다는 이유로 국내에서는 거의 쓰이지 않고 있다는 것에 안타까움을 느끼며 적은 지식이나마 아파치 서버를 운영해본 경험을 바탕으로 이 글을 썼다. 이후 SSL(Secure Socket Layer) 등의 아파치 서버의 많은 기능을 정리한 글을 작성할 예정이다.