Virtual Services Howto

Brian Ackerman, (brian@nycrc.net) v2.1, 15 August 1998

권태준, (linusben@bawi.org) 06 April 1999

이 문서는 시스템에서 가상 서비스(virtualize)를 제공하는 것에 대한 많은 요청들을 위해 만들어졌다.

목 차

제 1 절 소개

1.1 필요한 지식들

시스템의 가상 서비스를 제공하는 일은 어려운 일이 아니지만, 기초적인 지식 이상을 요구한다. 이 문서가 리눅스 머신을 전체적으로 설정하는 방법에 대한 입문서(primer)는 아니다.

이 하우투 문서는 당신이 아래의 내용들에 대해서 어느정도 이해를 하고 있다는 가정하에 이루어진 것이다:(한글 자료는 원본에 없는 것입니다. 당연하죠?-역자주)

- 리눅스 커널 컴파일과 IP 앨리어싱(aliasing) 지원 `IPAliasmini-HOWTO` `IPAliasingmini-HOWTO`
- 네트워크 장비의 설정 `NET-3HOWTO`
- inetd 설정 `NET-3HOWTO`
- 아래와 같은 여러가지 네트웨크 패키지 `(Sendmail)` `(Apache)` `(Apache)-` `Qmail` `(SAMBA)`
- DNS 설정 `DNSHOWTO` `DNS`
- 기초적인 시스템 관리에 대한 이해 `LinuxSystemsAdministrators'sGuide` `KLDP`
- 웹서버 설정에 대한 이해 `WWW`

만약 당신이 위의 어떤 것에 대해 확신이 없다면, 링크를 쫓아가서 내용을 먼저 확실하게 이해하기 바란다. 위의 사항에 대해서는 난 메일에 대해 결코 답장을 보내지 않을 것이다. 관련된 HOWTO의 저자에게 직접 질문을 하는 것이 좋을 것이다.

1.2 목적

가상(virtual) 서비스의 목적은 하나의 머신에 여러개의 네트워크 카드 없이도 여러개의 IP 주소를 인식하도록 하는 데 있다. IP 앨리어싱(aliasing)은 여러개의 네트워크 카드가 있을 때 각각의 장비에서 하나 이상의 IP 주소를 허용하는 커널 옵션이다. 이 때 커널은 뒤에서 다중의 정보교환이 이루어지고(이것은 매우 빠르게 이루어진다) 사용자는 마치 하나 이상의 서버를 가지고 있는 것처럼 인식하게 된다.

이런 다중 정보교환(multiplexing)은 하나의 도메인을 호스팅하는 것과 같은 비용으로 여러개의 도메인(www.domain1.com www.domain2.com 등)을 호스팅할 수 있다. 불행하게도 대부분의 서비스(FTP, web, mail)는 다중 도메인을 다룰 수 있도록 설계되어있지 않다. 이들을 적당하게 작동시키기 위해서는 설정 뿐만 아니라 소스 코드도 변형시켜야 한다. 이 문서는 어떻게 이런 변형을 시켜서 가상 머신을 설정할 것인지에 대해 기술한다.

가상 서비스 기능을 위해서는 데몬 역시 필요하다. 이 데몬(virtuald) 역시 이 문서의 후반에서 다룰 것이다.

1.3 피드백(Feedback)

이 문서는 패키지들이 업데이트되고 소스코드나 설정 내용들이 변경될 때마다 확장될 것이다. 만약 이 문서의 특정부분이 불분명하다고 생각되면 거리낌없이 나에게 당신의 제안이나 질문을 email로 보내라. 내가 이 하우투 문서의 모든 부분을 뒤지지 않도록 가급적이면 특정한(specific) 부분을 집어서 확실하게 해주고, 불분명한 부분이 위치한 섹션을 같이 표시해 주기 바란다. 또하나 중요한 점은 제목에 VIRTSERVICES HOWTO라는 문장을 넣어 메일을 보내라는 점이다. 그렇지 않은 다른 메일들은 개인적인 메일로 취급될 것이기 때문에 무시될 수도 있다. (내 친구들 역시 내가 개인적인 메일을 다 읽지 않는다는 사실을 알고 있다.)

나의 예제들은 단지 예제일 뿐이므로 그대로 복사해서 사용하지는 말기 바란다. 당신이 이 예제를 사용하고자 한다면, 당신 시스템에 맞는 값을 넣어야 할 것이다. 만약 어떤 문제가 생긴다면 나에게 메일을 보내라. 설치에 사용한 설정 파일과 설치에 발생한 에러 메세지들을 모두 나에게 보내준다면 내가 그것을 본 뒤 나의 의견을 답장에 써서 보내줄 것이다.

1.4 갱신 기록 (Revision History)

V1.0

최초의 버전

V1.1

버추얼 웹 설정 부분의 에러 수정

V1.2

날짜 수정

V2.0

HTML 링크 갱신

웹 갱신

새로운 센드메일 옵션

새로운 Qmail 섹션

Syslogd 갱신

FTP 갱신

Virtuald 기본 옵션

새로운 삼바(SAMBA) 섹션

FAQ 갱신

V2.1

모든 패스(path)를 /usr/local로 수정 Changed all paths to /usr/local.

Virtuald의 VERBOSELOG 컴파일 옵션 추가

Virtualmailfilter의 setuid/setgid 버그 수정

Virtualmailfilter의 excel 버그 수정

Virtualmailfilter의 대소문자 버그 수정.

Virtualmailfilter의 환경 변수 확인 수정

Virtualmailfilter/virtmaildelivery에서 mbox 코드 제거

Qmail에서 tcpserver.init pop 섹션 추가

FAQ에 앨리어스 도메인네임 질문 추가

Virtmailfilter가 홈 디렉토리를 virtmaildelivery로 보내는 부분 수정

1.5 저작권/배포

이 문서의 저작권은 Copyright (c) 1997 by The Computer Resource Center Inc. 에 있다.

이 문서의 복사본은 저자의 동의 없이 어떤 매체(물리적, 전자적)를 통해서도 배포 및 재생산이 가능하다. 번역 역시 비슷하게 허용되는데, 다만 번역자를 문서에 명시한다는 것을 가정한다. 상업적인 재배포 역시 허용되며 장려되지만 이런 종류의 배포에 대해서는 `ComputerResourceCenter` 에게 알려주기 바란다.

이 문서의 발췌 역시 사전 동의 없이 이루어질 수 있다.

이 문서의 복사본 제작 및 배포는 저작권에 대한 명시와 허가에 대한 이 권고가 모든 배포본에 포함될 때에만 인정된다.

요약하면, 우리는 가능한 많은 경로를 통해서 이 문서가 널리 퍼지기를 바란다. 하지만, 우리는 또한 이 문서의 저작권 역시 계속 갖고있기를 원하므로, 이 문서의 재배포에 대한 것을 알기를 바란다.

제 2 절 IP 앨리어싱(IP Aliasing)

IP 앨리어싱은 가상 호스팅 머신을 실행시키기 위한 설정으로 필요한 커널 옵션이다. 이미 이것에 대한 미니-하우투가 나와있다. `IPaliasing IPAliasingmini-HOWTO` 이것을 어떻게 설정하는지 알고 싶다면 이 문서를 참고하라.

제 3 절 Virtuald

3.1 소개

모든 네트워크 연결은 2개의 IP 어드레스/포트 쌍으로 이루어진다. 네트워크 프로그래밍을 위한 API(Applications Program Interface)는 소켓 API로 불린다. 소켓은 마치 열려있는 파일과 같이 작동하기 때문에 네트워크 연결을 통해 데이터를 주고받는 것은 소켓에 쓰고읽는 과정으로 이해된다. 로컬 소켓의 주소를 돌려주는 함수는 `getsockname`이다. Virtuald는 `getsockname`을 사용하여 로컬 시스템의 어떤 IP가 접근 가능한지를 판단한다. Virtuald는 설정 파일을 읽어서 해당 IP에 할당된 디렉토리를 돌려받는다. 그리고는 `chroot`를 통해서 모든 서비스에 대한 제어권을 넘겨준다. `Chroot`은 루트 디렉토리를 재설정하여 새롭게 지정받은 포인트를 루트로 설정하기 때문에 상위 디렉토리는 실행 프로그램들에 의해 무시된다. 네트워크 프로그

램에 대해서는 이 과정은 마치 아무일도 일어나지 않은 것처럼 투명하게 보일 것이다. Virtuald는 inetd와 같은 프로그램과 연결되어 어떤 서비스라도 가상적으로 만들 수 있다.

3.2 Inetd

Inetd는 외부에서 네트워크를 통해 연결을 시도할 때(예를 들면 POP서버의 요청) 여러 개의 포트를 감시하고 있다가 요청에 대한 적당한 연결을 구현해주는 네트워크 super server이다. Inetd는 네트워크를 전반적으로 관리하면서 특정 프로그램에 대해 네트워크를 연결해 주는 일을 한다. 이런 과정을 통해 어떤 서비스가 네트워크 연결이 필요 없을 때 불필요하게 실행되는 것을 막을 수 있다.

표준적인 /etc/inetd.conf 파일은 다음과 같다:

```
ftp stream tcp nowait root /usr/sbin/tcpd \
        wu.ftpd -l -a
pop-3 stream tcp nowait root /usr/sbin/tcpd \
        in.qpop -s
```

가상 서비스를 할 때 /etc/inetd.conf 파일은 다음과 같이 된다:

```
ftp stream tcp nowait root /usr/local/bin/virtuald \
        virtuald /virtual/conf.ftp wu.ftpd -l -a
pop-3 stream tcp nowait root /usr/local/bin/virtuald \
        virtuald /virtual/conf.pop in.qpop -s
```

3.3 설정 파일

각각의 서비스들은 어떤 IP들과 디렉토리들을 사용하고 관리할 것인지에 대한 정보를 설정 파일에서 얻게 된다. 당신은 하나의 전체 설정 파일 (master config file)을 갖던지 아니면 몇몇의 설정 파일들을 통해서 각각의 서비스에서 도메인의 다른 리스트를 얻게 할 수 있다. 설정 파일은 다음과 같다:

```
# This is a comment and so are blank lines

# Format IP SPACE dir NOSPACES
10.10.10.129 /virtual/domain1.com
10.10.10.130 /virtual/domain2.com
10.10.10.157 /virtual/domain3.com

# Default option for all other IPs
default /
```

3.4 소스코드

아래는 virtuald 프로그램에 대한 C 소스 코드이다. 이것을 컴파일하고 /usr/local/bin 아래에 0755의 퍼미션으로 설치하라.(소유자와 그룹은 모두 root로 한다.) 컴파일시에는 VERBOSELOG가 옵션으로 사용될 수 있는데 (오직 이 옵션만이 존재한다), 이것은 연결시 기록을 남길것인지를 정한다.

```
#include <netinet/in.h>
#include <sys/socket.h>
#include <arpa/inet.h>
#include <stdarg.h>
```

```
#include <unistd.h>
#include <string.h>
#include <syslog.h>
#include <stdio.h>

#undef VERBOSELOG

#define BUFSIZE 8192

int getipaddr(char **ipaddr)
{
        struct sockaddr_in virtual_addr;
        static char ipaddrbuf[BUFSIZE];
        int virtual_len;
        char *ipptr;

        virtual_len=sizeof(virtual_addr);
        if (getsockname(0,(struct sockaddr *)&virtual_addr,&virtual_len)<0)
        {
                syslog(LOG_ERR,"getipaddr: getsockname failed: %m");
                return -1;
        }
        if (!(ipptr=inet_ntoa(virtual_addr.sin_addr)))
        {
                syslog(LOG_ERR,"getipaddr: inet_ntoa failed: %m");
                return -1;
        }
        strncpy(ipaddrbuf,ipptr,sizeof(ipaddrbuf)-1);
        *ipaddr=ipaddrbuf;
        return 0;
}

int iptodir(char **dir,char *ipaddr,char *filename)
{
        char buffer[BUFSIZE],*bufptr;
        static char dirbuf[BUFSIZE];
        FILE *fp;

        if (!(fp=fopen(filename,"r")))
        {
                syslog(LOG_ERR,"iptodir: fopen failed: %m");
                return -1;
        }
        *dir=NULL;
        while(fgets(buffer,BUFSIZE,fp))
        {
                buffer[strlen(buffer)-1]=0;
                if (*buffer=='#' || *buffer==0)
                        continue;
                if (!(bufptr=strchr(buffer,' ')))
```

```
                {
                        syslog(LOG_ERR,"iptodir: strchr failed");
                        return -1;
                }
                *bufptr++=0;
                if (!strcmp(buffer,ipaddr))
                {
                        strncpy(dirbuf,bufptr,sizeof(dirbuf)-1);
                        *dir=dirbuf;
                        break;
                }
                if (!strcmp(buffer,"default"))
                {
                        strncpy(dirbuf,bufptr,sizeof(dirbuf)-1);
                        *dir=dirbuf;
                        break;
                }
        }
        if (fclose(fp)==EOF)
        {
                syslog(LOG_ERR,"iptodir: fclose failed: %m");
                return -1;
        }
        if (!*dir)
        {
                syslog(LOG_ERR,"iptodir: ip not found in conf file");
                return -1;
        }
        return 0;
}

int main(int argc,char **argv)
{
        char *ipaddr,*dir;

        openlog("virtuald",LOG_PID,LOG_DAEMON);

#ifdef VERBOSELOG
        syslog(LOG_ERR,"Virtuald Starting: $Revision: 1.49 $");
#endif
        if (!argv[1])
        {
                syslog(LOG_ERR,"invalid arguments: no conf file");
                exit(0);
        }
        if (!argv[2])
        {
                syslog(LOG_ERR,"invalid arguments: no program to run");
                exit(0);
        }
```

```
        if (getipaddr(&ipaddr))
        {
                syslog(LOG_ERR,"getipaddr failed");
                exit(0);
        }
#ifdef VERBOSELOG
        syslog(LOG_ERR,"Incoming ip: %s",ipaddr);
#endif
        if (iptodir(&dir,ipaddr,argv[1]))
        {
                syslog(LOG_ERR,"iptodir failed");
                exit(0);
        }
        if (chroot(dir)<0)
        {
                syslog(LOG_ERR,"chroot failed: %m");
                exit(0);
        }
#ifdef VERBOSELOG
        syslog(LOG_ERR,"Chroot dir: %s",dir);
#endif
        if (chdir("/")<0)
        {
                syslog(LOG_ERR,"chdir failed: %m");
                exit(0);
        }
        if (execvp(argv[2],argv+2)<0)
        {
                syslog(LOG_ERR,"execvp failed: %m");
                exit(0);
        }

        closelog();

        exit(0);
}
```

제 4 절 셸 스크립트

4.1 Virtfs

각각의 도메인은 자신만의 디렉토리 구조를 가진다. 따라서 만약 chroot를 사용하려고 한다면, 당신은 공유 라이브러리나 바이너리 실행파일들, 설정 파일 등을 복제하는 과정이 필요하다. 나는 내가 만든 각각의 도메인에 대해서 /virtual/domain1.com을 사용한다.

물론 더 많은 디스크의 공간이 필요하게 되지만, 그것이 새로운 머신과 네트워크 카드를 설치하는 것보다는 값이 저렴하다. 만약 당신이 공간을 절약하고 싶다면 하드 링크를 이용할 수 있는데, 내 경우 이 방법을 사용하면 2M가 약간 넘는 공간만을 사용하게 된다. 하지만, 이 스크립트는 가급적 일반적인 시스템을 만들기 위해 메인 파일시스템에서 모든 파일을 복사하려 할 것이다.

여기 간단한 virtfs 스크립트가 있다:

```
#!/bin/sh

echo '$Revision: 1.49 $'

echo -n "Enter the domain name: "
read domain

if [ "$domain" = "" ]
then
        echo Nothing entered: aborting
        exit 0
fi

leadingdir=/virtual

echo -n "Enter leading dir: (Enter for default: $leadingdir): "
read ans

if [ "$ans" != "" ]
then
        leadingdir=$ans
fi

newdir=$leadingdir/$domain

if [ -d "$newdir" ]
then
        echo New directory: $newdir: ALREADY exists
        exit 0
else
        echo New directory: $newdir
fi

echo Create $newdir
mkdir -p $newdir

echo Create bin
cp -pdR /bin $newdir

echo Create dev
cp -pdR /dev $newdir

echo Create dev/log
ln -f /virtual/log $newdir/dev/log

echo Create etc
mkdir -p $newdir/etc
for i in /etc/*
do
        if [ -d "$i" ]
```

```
        then
                continue
        fi
        cp -pd $i $newdir/etc
done

echo Create etc/skel
mkdir -p $newdir/etc/skel

echo Create home
for i in a b c d e f g h i j k l m n o p q r s t u v w x y z
do
        mkdir -p $newdir/home/$i
done

echo Create home/c/crc
mkdir -p $newdir/home/c/crc
chown crc.users $newdir/home/c/crc

echo Create lib
mkdir -p $newdir/lib
for i in /lib/*
do
        if [ -d "$i" ]
        then
                continue
        fi
        cp -pd $i $newdir/lib
done

echo Create proc
mkdir -p $newdir/proc

echo Create sbin
cp -pdR /sbin $newdir

echo Create tmp
mkdir -p -m 0777 $newdir/tmp
chmod +t $newdir/tmp

echo Create usr
mkdir -p $newdir/usr

echo Create usr/bin
cp -pdR /usr/bin $newdir/usr

echo Create usr/lib
mkdir -p $newdir/usr/lib

echo Create usr/lib/locale
```

```
cp -pdR /usr/lib/locale $newdir/usr/lib

echo Create usr/lib/terminfo
cp -pdR /usr/lib/terminfo $newdir/usr/lib

echo Create usr/lib/zoneinfo
cp -pdR /usr/lib/zoneinfo $newdir/usr/lib

echo Create usr/lib/\*.so\*
cp -pdR /usr/lib/*.so* $newdir/usr/lib

echo Create usr/sbin
cp -pdR /usr/sbin $newdir/usr

echo Linking usr/tmp
ln -s /tmp $newdir/usr/tmp

echo Create var
mkdir -p $newdir/var

echo Create var/lock
cp -pdR /var/lock $newdir/var

echo Create var/log
mkdir -p $newdir/var/log

echo Create var/log/wtmp
cp /dev/null $newdir/var/log/wtmp

echo Create var/run
cp -pdR /var/run $newdir/var

echo Create var/run/utmp
cp /dev/null $newdir/var/run/utmp

echo Create var/spool
cp -pdR /var/spool $newdir/var

echo Linking var/tmp
ln -s /tmp $newdir/var/tmp

echo Create var/www/html
mkdir -p $newdir/var/www/html
chown webmast.www $newdir/var/www/html
chmod g+s $newdir/var/www/html

echo Create var/www/master
mkdir -p $newdir/var/www/master
chown webmast.www $newdir/var/www/master
```

```
echo Create var/www/server
mkdir -p $newdir/var/www/server
chown webmast.www $newdir/var/www/server

exit 0
```

4.2 Virtexec

가상 환경에서 명령어를 실행시키기 위해서 당신은 chroot를 이용하여 디렉토리를 변경한 뒤 명령어를 실행해야만 한다. 여기 어떤 명령에 대해서 이런 기능을 수행할 수 있는 특별한 셸스크립트인 virtexec를 소개한다:

```
#!/bin/sh

echo '$Revision: 1.49 $'

BNAME=`basename $0`
FIRST4CHAR=`echo $BNAME | cut -c1-4`
REALBNAME=`echo $BNAME | cut -c5-`

if [ "$BNAME" = "virtexec" ]
then
        echo Cannot run virtexec directly: NEED a symlink
        exit 0
fi

if [ "$FIRST4CHAR" != "virt" ]
then
        echo Symlink not a virt function
        exit 0
fi

list=""
num=1
for i in /virtual/*
do
        if [ ! -d "$i" ]
        then
                continue
        fi
        if [ "$i" = "/virtual/lost+found" ]
        then
                continue
        fi
        list="$list $i $num"
        num=`expr $num + 1`
done

if [ "$list" = "" ]
then
        echo No virtual environments exist
```

```
        exit 0
fi

dialog --clear --title 'Virtexec' --menu Pick 20 70 12 $list 2> /tmp/menu.$$
if [ "$?" = "0" ]
then
        newdir=`cat /tmp/menu.$$`
else
        newdir=""
fi
tput clear
rm -f /tmp/menu.$$

echo '$Revision: 1.49 $'

if [ ! -d "$newdir" ]
then
        echo New directory: $newdir: NOT EXIST
        exit 0
else
        echo New directory: $newdir
fi

echo bname: $BNAME

echo realbname: $REALBNAME

if [ "$*" = "" ]
then
        echo args: none
else
        echo args: $*
fi

echo Changing to $newdir
cd $newdir

echo Running program $REALBNAME

chroot $newdir $REALBNAME $*

exit 0
```

이 스크립트가 실행되기 위해서 당신 시스템에 dialog 프로그램이 설치되어 있어야 한다는 것을 주의하라. virtexec를 사용하기 위해서는 단지 심볼릭 링크를 해주면 된다. 예를 들면 다음과 같다.

```
ln -s /usr/local/bin/virtexec /usr/local/bin/virtpasswd
ln -s /usr/local/bin/virtexec /usr/local/bin/virtvi
ln -s /usr/local/bin/virtexec /usr/local/bin/virtpico
ln -s /usr/local/bin/virtexec /usr/local/bin/virtemacs
ln -s /usr/local/bin/virtexec /usr/local/bin/virtmailq
```

링크를 시킨 후 virtvi를 실행시키면 가상 시스템의 vi를 실행시킬 것이다. 마찬가지로 virtpasswd는 가상 시스템의 사용자 비밀번호를 바꿀 것이고, virtmailq는 가상 공간의 메일 큐(queue)를 확인할 것이다. 당신은 원하는 만큼 virtexec를 만들 수 있다. 만약 당신의 프로그램이 공유 라이브러리를 필요로 한다면 바이너리와 마찬가지로 가상 파일시스템에 위치시킬 수 있다.

4.3 공고(Note)

나는 모든 스크립트들을 /usr/local/bin에 설치했다. 그 이외에 가상 파일시스템에 넣고싶지 않은 것들은 /usr/local에 두었다. 스크립트는 /usr/local에 있는 파일을 가상 파일시스템으로 복사하지 않는다. 가상 파일시스템으로 넘어오면 안되는 중요한 파일들은 반드시 가상 시스템에서 제거되어야 한다. 한 예로, 내 시스템에는 ssh가 설치되어 있는데, 모든 가상의 파일 시스템의 서버에서는 개인키(private key)를 원하지 않기 때문에, virtfs를 실행시킨 이후에 각각의 가상 파일 시스템에서 이것을 제거했다. 또한 resolv.conf를 바꾸고, 그 안에 있던 다른 도메인들의 이름을 가진 모든 것들을 삭제했다. 이런 것들로는 /etc/hosts나 /etc/HOSTNAME 등이 있다.

내가 virtexec로 심볼릭 링크시킨 프로그램들이다:

- virtpasswd – 사용자 비밀번호 변경
- virtadduser – 사용자 추가
- virtdeluser – 사용자 삭제
- virtsmbstatus – 삼바(SAMBA) 상태 보기
- virtvi – 파일 편집
- virtmailq – 메일큐 확인
- virtnewaliases – 앨리어스 테이블 새로 구축

제 5 절 DNS

DNS 역시 정상적으로 설정될 수 있다. 이것은 `DNSHOWTO.` 를 참조하라.

제 6 절 Syslogd

6.1 문제들

Syslogd는 일반적인 유닉스 시스템에서 사용되는 시스템 기록 도구이다. Syslogd는 FIFO라 불리는 특별한 파일을 여는 데몬인데, FIFO는 하나의 파이프와 같이 작동하는 특별한 파일이다. 쓰는 쪽에서 기록한 내용은 모두 읽는 쪽에 나타나게 된다. Syslogd는 읽는 쪽에서 보내는 데이터를 기다리고 있다. C의 함수 가운데에는 쓰는 쪽에 무언가를 기록할 수 있는 함수가 있는데, 프로그램에서 이 함수를 사용한다면, 당신의 출력 결과는 syslogd로 가게 된다.

`Chroot`를 사용하는 환경에서는 syslogd가 (/dev/log)에서 읽어들일 FIFO가 존재하지 않게 된다. 이것은 모든 가상 환경에서 syslogd로 들어갈 수 없다는 것을 의미한다.

6.2 Solution

6.2.1 Setup Links

Syslogd는 명령어 라인에서 설정을 해준다면 다른 FIFO를 찾을 수 있다:

```
syslogd -p /virtual/log
```

그리고는 /virtual/log를 /dev/log로 심볼링 링크시킨다:

```
ln -sf /virtual/log /dev/log
```

그리고는 모든 /dev/log의 복사본을 이 파일로 하드링크 시킨다:

```
ln -f /virtual/log /virtual/domain1.com/dev/log
```

virtfs 스크립트는 이미 이 과정을 수행한다. /virtual이 디스크 공간에 인접해 있고 /dev/log가 하드링크되어 있는 상태이기 때문에, 이들은 같은 inode 숫자를 가지고 같은 데이터를 가리키고 있다. 모든 가상의 /dev/log 들이 동시에 작동중이기 때문에 chroot도 이것을 멈출 수 없다. 모든 환경에서부터의 모든 메세지가 하나의 장소에 기록된다는 사실을 기억하라. 하지만, 이 데이터에서 원하는 자료만 걸러내는 독립적인 프로그램을 쉽게 만들 수 있을 것이다.

6.2.2 Syslogd.init

syslogd.init의 이 버전은 syslogd가 시작할 때마다 /dev/log FIFO를 지우고 새로 만들기 때문에 매번 /dev/log에 하드링크시켜야 한다. 여기 변형된 syslogd.init 파일이 있다:

```
#!/bin/sh

. /etc/rc.d/init.d/functions

case "$1" in
  start)
        echo -n "Starting dev log: "
        ln -sf /virtual/log /dev/log
        echo done
        echo -n "Starting system loggers: "
        daemon syslogd -p /virtual/log
        daemon klogd
        echo
        echo -n "Starting virtual dev log: "
        for i in /virtual/*
        do
                if [ ! -d "$i" ]
                then
                        continue
                fi
                if [ "$i" = "/virtual/lost+found" ]
                then
                        continue
```

```
                fi
                ln -f /virtual/log $i/dev/log
                echo -n "."
        done
        echo " done"
        touch /var/lock/subsys/syslogd
        ;;
  stop)
        echo -n "Shutting down system loggers: "
        killproc syslogd
        killproc klogd
        echo
        rm -f /var/lock/subsys/syslogd
        ;;
  *)
        echo "Usage: syslogd {start|stop}"
        exit 1
esac

exit 0
```

6.3 여러개의 Syslogd

6.3.1 하나의 디스크에 한개의 syslogd

만약 당신이 설정한 하나의 파일시스템이 넘쳐서 다른 디스크에 있는 공간을 가상 시스템에서 사용하고자 한다면, 하드 링크를 통해서 서로 다른 디스크를 연결할 수 없다는 사실을 명심하기 바란다. 이것은 한 도메인에 대해서도 서로 다른 디스크에서는 각각의 syslogd를 실행시켜야 한다는 것을 의미한다. 한 예로, 만약 당신이 /virtual1에 13개의 도메인을 가지고 있고, /virtual2에 15개의 도메인을 가지고 있다면, 13개의 도메인에 대해서는 /virtual1/log에 하드 링크를 시킨 후, `syslogd -p /virtual1/log` 와 같은 방법으로 각각 syslogd를 실행시킨다. 마찬가지 방법으로 /virtual2에 15개의 다른 도메인을 /virtual2/log에 하드링크 시킨 후 `syslogd -p /virtual2/log` 의 방법으로 syslogd를 실행시킨다.

6.3.2 하나의 도메인에 하나의 syslogd

만약 당신이 모든 기록들을 하나의 위치로 집중시키고 싶지 않다면 각각의 도메인에 대해서 syslogd를 따로 실행시킬 수 있다. 이것은 프로세스 ID를 낭비하기 때문에 권하고 싶은 방법은 아니지만, 이것을 구현하는 것은 쉬운 일이다. 당신은 syslogd.init 파일에서 `chroot /virtual/domain1.com syslogd`로 각각의 도메인에 대해 syslogd를 실행시킬 수 있다. 이것은 `chroot` 안에서 각각의 syslogd를 실행시키며 그 기록은 /var/log가 아니라 /virtual/domain1.com/var/log에 기록될 것이다. 메인 시스템에서 일반 `syslogd`와 커널 기록을 위한 `klogd`를 실행시키는 것을 잊지 말라.

제 7 절 가상 FTP

7.1 Inetd

Wu-ftpd는 자체적으로 가상 시스템에 맞게 구성될 수 있다. 하지만, 각각의 도메인에 대해서 분리된 비밀번호 파일을 관리할 수는 없다. 예를 들면, `bob@domain1.com` 와 `bob@domain2.com` 를 원할 때, 두 도메인의 계정 가운데 하나의 이름을 bob2로 바꾸던지 그 이외의 다른 계정 이름을 사용해야 한다. 하지만 만약 당신이 각각

의 도메인에 대해서 가상의 파일시스템을 가지고 있다면, 각각의 비밀번호 파일을 가질 수 있기 때문에 이런 문제를 해결할 수 있다. 단지 virtnewuser 스크립트를 이용하여 사용자를 만들고 virtpasswd 스크립트를 이용해서 비밀번호를 설정해 주면 위에서 말한 상황에 대해 모든 설정을 끝낼 수 있다.

wu-ftpd를 위한 inetd.conf의 엔트리는 다음과 같다:

```
ftp stream tcp nowait root /usr/local/bin/virtuald \
        virtuald /virtual/conf.ftp wu.ftpd -l -a
```

7.2 익명의 FTP(Anonymous FTP)

이것은 virtuald의 설정에 영향을 받지 않는다. 우선 /virtuald/domain1.com/etc/passwd 안에 일반적인 경우와 마찬가지로 FTP 사용자를 생성한다.

```
ftp:x:14:50:Anonymous FTP:/var/ftp:/bin/false
```

그리고는 익명의 FTP를 위한 디렉토리를 설정한다. 각각의 도메인에 대해서 서로 다른 비밀번호 파일을 가지고 있기 때문에 익명의 FTP를 제공할 도메인을 선택할 수 있다. FTP 서버는 이미 chroot에 의해 /virtual/domain1.com 디렉토리에 설정되어 있기 때문에 어떤 추가경로도 앞에 설정할 필요는 없다.

7.3 가상의 FTP 사용자들

Wu-ftpd는 방문자 그룹(guest group)이라는 것을 지원한다. 이것을 이용하면 각각의 사용자들에 대해서 다른 FTP 공간을 만들 수 있다. 또한 FTP 서버는 특정 공간에 대해 chroot를 이용해서 사용자가 정해진 디렉토리 밖으로 나갈 수 없게 만들 수 있다. 만약 당신이 가상 도메인 안에 이 방법으로 사용자들을 추가한다면 그 사용자들은 시스템 파일들을 볼 수 없게 된다.

방문자 그룹은 /virtual/domain1.com/etc/ftpaccess 파일에 추가된다.

/virtual/domain1.com/etc/passwd 안에 엔트리를 만들고 시작하는 홈 디렉토리는 chroot 디렉토리와 /./로 분리한다.

```
guest1:x:8500:51:Guest FTP:/home/g/guest1/./incoming:/bin/false
```

그 다음에는 방문자의 홈을 익명의 FTP를 설정하던 때와 마찬가지로 설정해 준다. 각각의 도메인에 대해서 비밀번호 파일을 따로 관리하기 때문에 어떤 도메인에 방문자 계정이 존재하는지, 그리고 각 도메인의 어떤 사용자들이 방문자 그룹으로 설정되어 있는지를 설정할 수 있다. 다시 한번 강조하는데, 이미 chroot에 의해서 FTP 서버가 /virtual/domain1.com 디렉토리로 설정되어 있기 때문에 경로앞에 어떤 추가적인 정보를 줄 필요는 없다.

제 8 절 가상의 웹

8.1 Virtuald와 실행하기

8.1.1 권하지 않음!(Not recommended)

아파치는 기본적으로 가상의 도메인을 위한 지원을 해준다. 아파치는 내가 내부의 가상 도메인 기작을 사용하는 것을 권하는 유일한 프로그램이다. inetd를 통해서 어떤 프로그램이 실행되는 것은 많은 대가를 필요로 하며, 프로그램이 시작하기 위해서는 항상 inetd가 시작되어야 한다는 단점이 있다. 이런 결과로 반응 시간이 느려질 것인데, 다른 네트워크 서비스에서는 이런 느려짐이 견딜만 하지만 웹 서비스에서는 그렇지 않을 것이다.

아파치는 또한 너무 많은 요청이 들어오는 경우 제어를 할 수 있는 기능을 가지고 있는데, 이 기능은 크지 않은 규모의 사이트에서도 중요한 문제가 될 수 있다.

간단히 말하면, 아파치 서버를 virtuald와 함께 사용하여 가상 시스템을 구축하는 것은 정말 안좋은 생각이다. virtuald가 지향하는 것은 제공하고자 하는 서비스가 내부에서 가상 공간에 대한 만족스러운 지원을 하지 않을 경우에 부족한 부분을 채우는 일이다. virtuald가 이미 완벽하게 가상 시스템을 지원하는 좋은 코드를 대체하기 위한 것은 아니다.

물론 위의 충고도 꼭 그렇개 하고 말리라 생각하는 무모한 사람들에게는 별 효용이 없을 것이지만.

8.1.2 Inetd

/etc/inetd.conf 를 편집하라.

```
vi /etc/inetd.conf # Add this line
www stream tcp nowait www /usr/local/bin/virtuald \
        virtuald /virtual/conf.www httpd -f /var/www/conf/httpd.conf
```

8.1.3 Httpd.conf

/var/www/conf/httpd.conf를 편집하라.

```
vi /var/www/conf/httpd.conf # Or wherever you put the Apache config files
It should say:
ServerType standalone

Replace it with:
ServerType inetd
```

8.1.4 설정

그리고는 아파치 서버을 일반적으로 설정하는 과정을 수행한다.

8.1.5 Httpd.init

inetd를 통해서 아파치가 실행될 때에는 httpd.init은 필요없다.

8.2 Apache VirtualHost와 같이 실행하기

아파치는 세 개의 설정 파일 - `access.conf` , `httpd.conf` , `srm.conf`을 가진다. 아파치의 새로운 버전은 설정 파일이 세 개일 필요는 없다. 하지만, 세 개의 설정파일로 나누어져 있는 것이 훨씬 관리하고 설정하기가 쉽다고 생각되기 때문에 이 하우투 문서에서는 이 스타일을 기준으로 설명하겠다.

8.2.1 Access.conf

이 설정파일은 웹 디렉토리 구조에서 접근을 통제하는데 사용된다. 여기 각각의 도메인에 어떻게 서로 다른 옵션을 설정할 수 있는지를 보여주는 예제 설정파일이 있다.

```
# /var/www/conf/access.conf: Global access configuration
```

```
# Options are inherited from the parent directory
# Set the main directory with default options
<Directory />
AllowOverride None
Options Indexes
</Directory>

# Give one domain a passwd protected directory
<Directory /virtual/domain1.com/var/www/html/priv>
AuthUserFile /var/www/passwd/domain1.com-priv
AuthGroupFile /var/www/passwd/domain1.com-priv-g
AuthName PRIVSECTION
AuthType Basic
<Limit GET PUT POST>
require valid-user
</Limit>
</Directory>

# Give another domain Server Side Includes
<Directory /virtual/domain2.com/var/www/html>
Options IncludesNOEXEC
</Directory>
```

8.2.2 Httpd.conf

이 설정파일은 아파치 서버의 주된 옵션을 관리한다. 여기 각각의 도메인에 대해서 서로 다른 옵션을 설정하는 것에 대한 예제 파일이 있다.

```
# /var/www/conf/httpd.conf: Main server configuration file

# Begin: main conf section

# Needed since not using inetd
ServerType standalone

# Port to run on
Port 80

# Log clients with names vs IP addresses
HostnameLookups on

# User to run server as
User www
Group www

# Where server config, error and log files are
ServerRoot /var/www

# Process Id of server in this file
PidFile /var/run/httpd.pid
```

```
# Internal server process info
ScoreBoardFile /var/www/logs/apache_status

# Timeout and KeepAlive options
Timeout 400
KeepAlive 5
KeepAliveTimeout 15

# Number of servers to run
MinSpareServers 5
MaxSpareServers 10
StartServers 5
MaxClients 150
MaxRequestsPerChild 30

# End: main conf section

# Begin: virtual host section

# Tell server to accept requests for ip:port
# I have one for each IP needed so you can explicitly ignore certain domains
Listen 10.10.10.129:80
Listen 10.10.10.130:80

# VirtualHost directive allows you to specify another virtual
# domain on your server.  Most Apache options can be specified
# within this section.
<VirtualHost www.domain1.com>

# Mail to this address on errors
ServerAdmin webmaster@domain1.com

# Where documents are kept in the virtual domain
DocumentRoot /virtual/domain1.com/var/www/html

# Name of the server
ServerName www.domain1.com

# Log files Relative to ServerRoot option
ErrorLog logs/domain1.com-error_log
TransferLog logs/domain1.com-access_log
RefererLog logs/domain1.com-referer_log
AgentLog logs/domain1.com-agent_log

# Use CGI scripts in this domain
ScriptAlias /cgi-bin/ /var/www/cgi-bin/domain1.com/
AddHandler cgi-script .cgi
AddHandler cgi-script .pl
</VirtualHost>
```

```
<VirtualHost www.domain2.com>

# Mail to this address on errors
ServerAdmin webmaster@domain2.com

# Where documents are kept in the virtual domain
DocumentRoot /virtual/domain2.com/var/www/html

# Name of the server
ServerName www.domain2.com

# Log files Relative to ServerRoot option
ErrorLog logs/domain2.com-error_log
TransferLog logs/domain2.com-access_log
RefererLog logs/domain2.com-referer_log
AgentLog logs/domain2.com-agent_log

# No CGI's for this host
</VirtualHost>
# End: virtual host section
```

8.2.3 Srm.conf

이 설정파일은 어떤 요청에 대해 서비스를 제공할 것인지와 그 형식이 어떻게 될 것인지에 대한 것을 조절한다. 가상 도메인에 대해서 이것을 바꿀 필요는 없다. 아파치에서 기본적으로 제공하는 예제 파일 형식 그대로 가상 시스템에 적용해도 작동할 것이다.

8.2.4 Httpd.init

httpd.init 파일에 대해서 특별히 해줄 것은 없다. 아파치의 표준적인 설정을 이용하라.

8.3 파일 서술자 오버플로우(File Descriptor Overflow)

8.3.1 경고

이것은 오직 독립적(standalone style)인 아파치 서버에만 해당된다. inetd를 통해서 서버가 실행될 때에는 다른 도메인과 연관이 없기 때문에 모든 파일 서술자 테이블을 갖는다.

아파치 서버가 여는 모든 로그 파일들은 프로세스에 대한 또다른 파일 서술자 (file descriptor)가 된다. 리눅스에서는 각각의 프로세스에 대해서 256개 까지 파일 기술자를 가질 수 있다는 제약이 있다. 즉 여러개의 도메인을 사용한다면 구먼큼 많은 파일 기술자들이 필요하게 된다. 만약 하나의 아파치 서버 프로세스에서 너무 많은 도메인을 설정하고 사용한다면 이 테이블이 넘쳐날(overflow) 수도 있다. 이것은 특정한 로그가 작동하지 않을 수도 있으며, CGI가 실패할 수도 있다는 것을 의미한다.

8.3.2 여러개의 아파치 서버

만약 한 도메인에 대해서 다섯개의 파일 기술자를 가지고 있다고 가정하면 당신은 아파치 서버에서 50개의 도메인을 아무 문제없이 실행시킬 수 있다. 하지만, 만약 이런 상황에서 문제가 발생한다면 /var/www1에 도메인1-도메인25 의 아파치 서버를 할당하고, /var/www2에 도메인26-도메인50의 아파치 서버를 할당하는 방식을

사용할 수 있다. 이것은 각각의 서버에 대해 자신만의 설정과 에러, 로그 디렉토리를 갖게 된다. 각각의 서버는 또한 자신만의 Listen과 VirtualHost 디렉토리를 가지고 있어야 한다. httpd.init 파일에서도 여러개의 서버를 설정해 주어야 한다는 사실을 잊지 말라.

8.4 하나의 IP에서 서버 공유

8.4.1 IP 절약하기

HTTP(HyperText Transfer Protocol) 버전 1.1에는 서버의 이름을 클라이언트에 알리는 기능이 추가되었다. 이것은 클라이언트가 서버를 찾을 때 IP 어드레스로 부터 찾을 필요가 없다는 것을 의미한다. 따라서 두개의 가상 서버가 같은 IP 어드레스를 가지고 서로 다른 웹 사이트로 꾸밀 수 있다. 아파치 설정은 위와 똑같은데, 단지 서로 다른 Listen을 지시할 필요가 없다는 점만 다르다. (두개의 도메인이 같은 IP를 쓰기 때문에 Listen 역시 같게 된다.)

8.4.2 결점

오직 문제가 되는 점은 virtuald가 도메인을 구분하기 위해서 IP 어드레스를 사용한다는 점이다. 지금 현재의 virtuald 형태로는 각각의 도메인에 대한 스풀 디렉토리를 `chroot`로 다르게 설정할 수 없다. 따라서, 메일은 하나의 IP에 대해서만 반응할 수 있기 때문에 각각의 도메인에 대해서 독자적인 스풀 디렉토리를 갖는 것은 불가능하다. IP를 공유하는 모든 웹 클라이언트는 IP의 스풀 디렉토리 역시 공유해야 한다. 또 이것은 사용자이름을 복사해서 사용하는 것에 대한 문제를 언급하게 하지만, 이정도는 IP를 공유함에 있어서 필요한 대가일 것이다.

8.5 더 많은 정보

이 하우투는 단지 어떻게 가상 시스템을 아파치 웹 서버에서 제공할 수 있는지만을 보여주고 있다. 대부분의 웹 서버들은 비슷한 인터페이스를 갖는다. 가상 웹 호스팅에 대해 더 많은 내용을 알고싶다면 `WWWHOWTO`, 를 참고하거나, 아파치에 대한 문서들 (`Apache`에서 구할 수 있다) 혹은 `ApacheWeek`를 찾아보라.

제 9 절 가상 메일/POP 서버

9.1 문제

가상 메일의 지원에 대한 요청은 날로 증가하고 있다. 센드메일은 가상 메일 시스템을 지원한다고 말한다. 하지만 그것이 지원하는 것은 여러 도메인에서 메일들을 검사하는 기능이다. 그 후 당신은 특정 메일을 다른 곳으로 포워딩 할 수 있다. 하지만, 로컬 머신으로 포워딩된 메일이나 bob@domain1.com과 bob@domain2.com에 온 메일은 같은 메일 폴더에 들어가 있게 된다. 이들이 서로 다른 메일이고 두명의 bob이 서로 다른 사람일 때에는 문제가 된다.

9.2 해결책

당신은 각각의 사용자 이름에 숫자를 붙여서, 혹은 정해진 문자를 앞에 붙여서 중복되는 사용자가 아님을 구분할 수 있는 방법이 있다.(예: bob1, bob2 혹은 dom1bob, dom2bob) 당신은 또한 mail이나 pop를 고쳐서 이런 전환이 보이지 않게 이루어질 수 있게 만들수도 있다. 외부로 나가는 메일 역시 이런 식으로 각각의 서브도메인에 대해서 그 이름을 사용하게 만들 수 있다.

내가 가진 해결책은 두가지이다. 하나는 sendmail을 이용하는 것이고, 다른 하나는 Qmail을 이용하는 것이다. Sendmail을 이용한 해결책은 이 기능을 추가하여 sendmail 을 설치하는 것이다. 하지만, 이 방법은 모든 제약이 sendmail에 똑같이 적용된다. 이 방법은 또한 각각의 도메인에 대해서 하나씩의 sendmail이 queue mode로

실행되어야 한다는 단점을 가지고 있다. 50개 혹은 그 이상의 senmail queue 프로세스는 매시간 시스템을 바쁘게 만들 것이다.

Qmail을 이용하는 해결책은 여러개의 Qmail을 필요로하지도 않고, 하나의 queue 디렉토리 이외에서도 실행가능하다. 이 방법은 Qmail이 virtuald와 맞지 않기 때문에 추가적인 프로그램을 필요로 한다. 난 sendmail을 이용한 방법 역시 비슷한 과정을 필요로 한다고 믿는다. 하지만, Qmail은 이 경우 보다 해결책을 위한 준비가 잘 되어있는 것 같다.

내가 한 프로그램이 다른 프로그램보다 낫다는 것을 보증하는 것은 아니다. Sendmail 설치는 보다 직접적인 해결책이지만, Qmail을 이용하는 방법이 아마 더 강력한 해결책이 될 수 있다.

9.3 Sendmail을 이용한 해결책

9.3.1 소개

각각의 가상 파일 시스템은 자신의 /etc/passwd 안에 도메인을 설정한다. 이것은 bob@domain1.com과 bob@domain2.com이 서로 다른 사용자로 /etc/passwd 안에 등록되어 있다는 것을 의미하며, 메일 프로그램에서 두 사용자를 구분하는 데에는 아무 문제가 없다. 또한 자신만의 스풀 디렉토리 역시 가지고 있으므로 다른 가상 파일 시스템에 대해서 서로 다른 파일로서 메일 폴더가 존재하게 된다.

9.3.2 Sendmail 설정 파일 만들기

일반적인 /etc/sendmail.cf 를 m4를 통해 만든다. 내가 사용하는 것은 다음과 같다:

```
divert(0)
VERSIONID(`tcpproto.mc')
OSTYPE(linux)
FEATURE(redirect)
FEATURE(always_add_domain)
FEATURE(use_cw_file)
FEATURE(local_procmail)
MAILER(local)
MAILER(smtp)
```

9.3.3 Sendmail 설정파일 편집하기

/virtual/domain1.com/etc/sendmail.cf 를 가상 도메인에 반응할 수 있도록 편집한다:

```
vi /virtual/domain1.com/etc/sendmail.cf # Approximately Line 86
It should say:

#Dj$w.Foo.COM

Replace it with:

Djdomain1.com
```

9.3.4 Sendmail 지역 배달

/virtual/domain1.com/etc/sendmail.cw를 지역 호스트이름으로 편집한다.

```
vi /virtual/domain1.com/etc/sendmail.cw
mail.domain1.com
domain1.com
domain1
localhost
```

9.3.5 가상 도메인 사이의 Sendmail : The Hack (PRE8.8.6)

하지만, sendmail은 작은 소스 코드 변환을 필요로 한다. Sendmail은 /etc/sendmail.cw 라는 파일을 가지고 있는데, 여기에는 sendmail이 로컬 내에서(외부의 다른 머신이 아닌) 배달할 모든 머신들의 이름이 기록되어 있다. Sendmail은 내부에서 머신의 모든 장치들에 대해 검사하여 이 리스트를 로컬 IP를 가지고 초기화한다. 이점 때문에 만약 같은 머신 내의 가상 도메인 사이에서 메일을 주고받고자 할 때 문제가 될 수 있다. Sendmail은 다른 가상 도메인을 로컬 어드레스로 생각하고 로컬 지역으로 메일을 스풀링하게 된다. 예를 들면, bob@domain1.com이 fred@domain2.com에게 메일을 보냈다고 하자. 그러면 domain1.com의 sendmail은 domain2.com을 로컬로 인식하고 메일을 domain1.com에 스풀링할 것이다. (당연히 domain2.com으로는 메일이 가지 않을 것이다.) 따라서 당신은 sendmail을 변형시켜야 한다. (이 변형은 v8.8.5에서 테스트해본 결과 아무 문제가 없었다.)

```
vi v8.8.5/src/main.c # Approximately Line 494
It should say:

load_if_names();

Replace it with:

/* load_if_names(); Commented out since hurts virtual */
```

만약 가상 도메인 사이에서 메일을 주고받을 필요가 있을 경우에만 이 설정을 이용하라. (아마 대부분의 경우 그러하겠지만)

이것은 문제점을 해결할 것이다. 하지만, 주된 이더넷 장치인 eth0는 없어지지 않는다. 따라서, 만약 당신이 가상 IP에서 eth0로 메일을 보내게 되면 이것은 로컬로 배달이 될 것이다. 따라서 나는 이것을 더미(dummy) IP인 virtual1.maindomain.com(10.10.10.157)로 이용한다. 난 절대 이 호스트로 메일을 보내지 않으며, 물론 그 가상 도메인으로도 메일은 가지 않는다. 이 방법은 또한 내가 ssh를 사용하는 IP를 가진 시스템이 정상적인지를 확인하는 방법이기도 하다.

9.3.6 가상 도메인 사이의 Sendmail : Sendmail의 새로운 기능 (POST8.8.6)

Sendmail V8.8.6부터는 추가적인 네트워크 인터페이스의 비사용 탑재(disable loading)에 대한 새로운 옵션이 생겼다. 따라서 코드를 바꿀 필요는 없게 되었는데, 이것을 `DontProbeInterfaces`라 한다.

/virtual/domain1.com/etc/sendmail.cf를 편집하라.

```
vi /virtual/domain1.com/etc/sendmail.cf # Add the line
O DontProbeInterfaces=True
```

9.3.7 Sendmail.init

Sendmail은 독립적으로 실행이 불가능하고 항상 inetd를 통해서 실행되게 된다. 이 방법은 비효율적이고 시작하는 데 시간이 걸리겠지만, 만약 당신이 운영하는 사이트가 이런 점이 문제가 될 정도로 네트워크가 빈번하다면 하나의 시스템에서 가상의 여러 도메인을 같이 사용하는 것은 좋은 방법이 아니다. `-bd` 플랙(flag)과 같이

사용하지 않도록 주의하라. 또한 각각의 도메인에 대해서 `sendmail -q` 을 실행하여 배달되지 않은 메일들에 대한 큐 작업을 가능하게 하는 것도 잊지 말라. 새로운 sendmail.init 파일은 다음과 같다:

```
#!/bin/sh

. /etc/rc.d/init.d/functions

case "$1" in
  start)
        echo -n "Starting sendmail: "
        daemon sendmail -q1h
        echo
        echo -n "Starting virtual sendmail: "
        for i in /virtual/*
        do
                if [ ! -d "$i" ]
                then
                        continue
                fi
                if [ "$i" = "/virtual/lost+found" ]
                then
                        continue
                fi
                chroot $i sendmail -q1h
                echo -n "."
        done
        echo " done"
        touch /var/lock/subsys/sendmail
        ;;
  stop)
        echo -n "Stopping sendmail: "
        killproc sendmail
        echo
        rm -f /var/lock/subsys/sendmail
        ;;
  *)
        echo "Usage: sendmail {start|stop}"
        exit 1
esac

exit 0
```

9.3.8 Inetd 설정

Pop는 다른 영향없이 정상적으로 설치될 것이다. 단지 inetd의 엔트리에서 이 항을 가상의 포트와 함께 고려할 필요가 있다. inetd.conf 엔트리에서 sendmail과 pop에 대한 것은 다음과 같다:

```
pop-3 stream tcp nowait root /usr/local/bin/virtuald \
        virtuald /virtual/conf.pop in.qpop -s
smtp stream tcp nowait root /usr/local/bin/virtuald \
        virtuald /virtual/conf.mail sendmail -bs
```

9.4 Qmail을 이용한 방법

9.4.1 소개

이 방법은 qmail-local의 배달 시스템을 차용하기 때문에, 가상의 홈 디렉토리 안의 .qmail 파일은 작동하지 않게 된다. 하지만, 각각의 도메인은 도메인 전체의 앨리어싱 (aliasing)을 통제하는 도메인 주인 사용자(domain master user)를 갖는다. 두 개의 외부 프로그램들이 도메인 주인의 .qmail-default 파일을 사용할 수 있게 해줄 것이다. 각각의 도메인에 메일이 배달되기 위해서는 이들 두 프로그램을 통해야 할 것이다.

두 개의 프로그램이 필요한데, 그 가운데 하나는 setuid root 상태로 실행된다. 이 작은 프로그램은 일단 프로세스의 소유권을 root가 아닌 사용자로 바꾸고, 다시 두번째 프로그램을 실행시킨다. 가까운 보안 관련 사이트에서 왜 이런 방식이 필요한지를 참고할 수 있을 것이다.

이 방법은 virtuald를 사용할 필요성이 별로 없다. Qmail은 매우 유동적인 프로그램이라 일반적인 virtuald 설정을 필요로하지 않는다. Qmail은 메일의 배달을 위해 프로그램들의 연결을 이용하도록 설계되었다. 이 디자인은 가상 서비스 부분을 Qmail 배달 프로세스 중간에 쉽게 삽입할 수 있게 한다.

당신이 Qmail을 사용한다면 메인 서버의 도메인에서 무제한의 도메인 이름을 만들어 낼 수 있다. 이것은 각각의 도메인에 대해 분리된 Qmail을 갖는 것이 아니기 때문에 가능하다. 메일 클라이언트 프로그램(유도라나 elm, mutt 등)에서 당신이 임의로 만들어낸 도메인 이름을 인식하는 것을 확인해 보라.

9.4.2 가상 도메인 설정

Qmail은 당신이 제공하는 각각의 가상 도메인을 받아들일 수 있도록 설정되어야 한다. 아래의 명령어들을 수행하라.

```
echo "domain1.com:domain1" >> /var/qmail/control/virtualdomains
```

9.4.3 도메인의 주인(Domain Master User) 설정

메인 /etc/passwd 파일에 domain1의 사용자들을 추가한다. 나는 /bin/false 셸을 만들어 도메인 주인(the domain master)이 로그인하지 못하게 만들었다. 도메인 주인은 domain1의 .qmail 파일들을 추가할 수 있고, 도메인의 모든 메일들은 이 계정을 통하여 발송된다. 사용자 이름은 여덟 자리까지 가능하며 도메인 이름은 더 길어지 수 있다는 것을 주의하기 바란다. 나머지 문자들은 무시된다. 이것은 domain12라는 사용자와 domain123이라는 사용자가 같은 사용자로 인식되기 때문에 Qmail이 혼동할 수 있다는 것을 의미한다. 따라서 도메인 주인 이름 결정에 주의를 기울이기 바란다.

다음과 같은 절차를 통하여 도메인 주인의 .qmail 파일을 만들자. 다른 시스템 앨리어스 - 예를 들면 웹마스터나 호스트마스터- 가 이 지점에 추가된다.

```
echo "user@domain1.com" > /home/d/domain1/.qmail-mailer-daemon
echo "user@domain1.com" > /home/d/domain1/.qmail-postmaster
echo "user@domain1.com" > /home/d/domain1/.qmail-root
```

도메인 주인의 .qmail-default 파일을 만들자. 이것은 모든 메일을 가상의 도메인으로 걸러주게 될 것이다.

```
echo "| /usr/local/bin/virtmailfilter" > /home/d/domain1/.qmail-default
```

9.4.4 Tcpserver

Qmail은 Maildir 형식을 지원하는 특별한 pop을 필요로 한다. 이 pop 프로그램 또한 가상 시스템에 맞게 되어야 한다. Qmail의 저자는 tcpserver(inetd 대용)를 Qmail과 함께 사용할 것을 권하는데, 나의 예제에서도 inetd 대신에 tcpserver를 사용하였다.

Tcpserver는 설정 파일을 필요로 하지 않는다. 모든 정보는 명령행에서 주어지게 된다. 여기 메일 데몬과 popper를 사용하기 위한 tcpserver.init이 있다.

```
#!/bin/sh

. /etc/rc.d/init.d/functions

QMAILDUSER=`grep qmaild /etc/passwd | cut -d: -f3`
QMAILDGROUP=`grep qmaild /etc/passwd | cut -d: -f4`

# See how we were called.
case "$1" in
  start)
        echo -n "Starting tcpserver: "
        tcpserver -u 0 -g 0 0 pop-3 /usr/local/bin/virtuald \
                /virtual/conf.pop qmail-popup virt.domain1.com \
                /bin/checkpassword /bin/qmail-pop3d Maildir &
        echo -n "pop "
        tcpserver -u $QMAILDUSER -g $QMAILDGROUP 0 smtp \
                /var/qmail/bin/qmail-smtpd &
        echo -n "qmail "
        echo
        touch /var/lock/subsys/tcpserver
        ;;
  stop)
        echo -n "Stopping tcpserver: "
        killall -TERM tcpserver
        echo -n "killing "
        echo
        rm -f /var/lock/subsys/tcpserver
        ;;
  *)
        echo "Usage: tcpserver {start|stop}"
        exit 1
esac

exit 0
```

9.4.5 Qmail.init

당신은 제공되는 표준 Qmail 초기 스크립트를 바로 사용할 수 있다. Qmail은 이것을 어떻게 설정해야 하는지에 대해 상당히 좋은 문서와 함께 배포된다.

9.4.6 소스(Source)

Qmail로 가상 메일 서비스를 구축하기 위해서는 두개의 서로 다른 프로그램이 필요하다. 하나는 virtmailfilter이고, 다른 하나는 virtmaildelivery이다. 여기 virtmailfilter에 대한 C 소스 코드가 있다. 이 프로그램은 /usr/local/bin에 4750의 소유권을 가지고, root 소유, nofiles 그룹으로 설치되어야 한다.

```
#include <sys/wait.h>
```

```
#include <unistd.h>
#include <string.h>
#include <stdlib.h>
#include <stdio.h>
#include <ctype.h>
#include <pwd.h>

#define VIRTPRE                 "/virtual"

#define VIRTPWFILE              "etc/passwd"
#define VIRTDELIVERY            "/usr/local/bin/virtmaildelivery"
#define VIRTDELIVERY0           "virtmaildelivery"

#define PERM                    100
#define TEMP                    111
#define BUFSIZE                 8192

int main(int argc,char **argv)
{
        char *username,*usernameptr,*domain,*domainptr,*homedir;
        char virtpath[BUFSIZE];
        struct passwd *p;
        FILE *fppw;
        int status;
        gid_t gid;
        pid_t pid;

        if (!(username=getenv("EXT")))
        {
                fprintf(stdout,"environment variable EXT not set\n");
                exit(TEMP);
        }

        for(usernameptr=username;*usernameptr;usernameptr++)
        {
                *usernameptr=tolower(*usernameptr);
        }

        if (!(domain=getenv("HOST")))
        {
                fprintf(stdout,"environment variable HOST not set\n");
                exit(TEMP);
        }

        for(domainptr=domain;*domainptr;domainptr++)
        {
                if (*domainptr=='.' && *(domainptr+1)=='.')
                {
                        fprintf(stdout,"environment variable HOST has ..\n");
                        exit(TEMP);
```

```
        }
        if (*domainptr=='/')
        {
                fprintf(stdout,"environment variable HOST has /\n");
                exit(TEMP);
        }

        *domainptr=tolower(*domainptr);
}

for(domainptr=domain;;)
{
        snprintf(virtpath,BUFSIZE,"%s/%s",VIRTPRE,domainptr);
        if (chdir(virtpath)>=0)
                break;

        if (!(domainptr=strchr(domainptr,'.')))
        {
                fprintf(stdout,"domain failed: %s\n",domain);
                exit(TEMP);
        }

        domainptr++;
}

if (!(fppw=fopen(VIRTPWFILE,"r+")))
{
        fprintf(stdout,"fopen failed: %s\n",VIRTPWFILE);
        exit(TEMP);
}

while((p=fgetpwent(fppw))!=NULL)
{
        if (!strcmp(p->pw_name,username))
                break;
}

if (!p)
{
        fprintf(stdout,"user %s: not exist\n",username);
        exit(PERM);
}

if (fclose(fppw)==EOF)
{
        fprintf(stdout,"fclose failed\n");
        exit(TEMP);
}

gid=p->pw_gid;
```

```
        homedir=p->pw_dir;

        if (setgid(gid)<0 || setuid(p->pw_uid)<0)
        {
                fprintf(stdout,"setuid/setgid failed\n");
                exit(TEMP);
        }

        switch(pid=fork())
        {
                case -1:
                        fprintf(stdout,"fork failed\n");
                        exit(TEMP);
                case 0:
                        if (execl(VIRTDELIVERY,VIRTDELIVERY0,username,homedir,NULL)<0)
                        {
                                fprintf(stdout,"execl failed\n");
                                exit(TEMP);
                        }
                default:
                        if (wait(&status)<0)
                        {
                                fprintf(stdout,"wait failed\n");
                                exit(TEMP);
                        }
                        if (!WIFEXITED(status))
                        {
                                fprintf(stdout,"child did not exit normally\n");
                                exit(TEMP);
                        }
                        break;
        }

        exit(WEXITSTATUS(status));
}
```

9.4.7 소스(Source)

여기에는 virtmaildelivery에 대한 C 소스 코드가 있다. 이것은 /usr/local/bin에 0755의 소유권으로, 소유자와 그룹 모두 root로 설치되어야 한다.

```
#include <sys/stat.h>
#include <sys/file.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <stdio.h>
#include <errno.h>
#include <time.h>
```

```
#define TEMP                    111
#define BUFSIZE                 8192
#define ATTEMPTS                10

int main(int argc,char **argv)
{
        char *user,*homedir,*dtline,*rpline,buffer[BUFSIZE],*p,mail[BUFSIZE];
        char maildir[BUFSIZE],newmaildir[BUFSIZE],host[BUFSIZE];
        int fd,n,nl,i,retval;
        struct stat statp;
        time_t thetime;
        pid_t pid;
        FILE *fp;

        retval=0;

        if (!argv[1])
        {
                fprintf(stdout,"invalid arguments: need username\n");
                exit(TEMP);
        }

        user=argv[1];

        if (!argv[2])
        {
                fprintf(stdout,"invalid arguments: need home directory\n");
                exit(TEMP);
        }

        homedir=argv[2];

        if (!(dtline=getenv("DTLINE")))
        {
                fprintf(stdout,"environment variable DTLINE not set\n");
                exit(TEMP);
        }

        if (!(rpline=getenv("RPLINE")))
        {
                fprintf(stdout,"environment variable RPLINE not set\n");
                exit(TEMP);
        }

        while (*homedir=='/')
                homedir++;
        snprintf(maildir,BUFSIZE,"%s/Maildir",homedir);
        if (chdir(maildir)<0)
        {
                fprintf(stdout,"chdir failed: %s\n",maildir);
```

```
                exit(TEMP);
        }

        time(&thetime);
        pid=getpid();
        if (gethostname(host,BUFSIZE)<0)
        {
                fprintf(stdout,"gethostname failed\n");
                exit(TEMP);
        }

        for(i=0;i<ATTEMPTS;i++)
        {
                snprintf(mail,BUFSIZE,"tmp/%u.%d.%s",thetime,pid,host);
                errno=0;
                stat(mail,&statp);
                if (errno==ENOENT)
                        break;

                sleep(2);
                time(&thetime);
        }
        if (i>=ATTEMPTS)
        {
                fprintf(stdout,"could not create %s\n",mail);
                exit(TEMP);
        }

        if (!(fp=fopen(mail,"w+")))
        {
                fprintf(stdout,"fopen failed: %s\n",mail);
                retval=TEMP; goto unlinkit;
        }

        fd=fileno(fp);

        if (fprintf(fp,"%s",rpline)<0)
        {
                fprintf(stdout,"fprintf failed\n");
                retval=TEMP; goto unlinkit;
        }

        if (fprintf(fp,"%s",dtline)<0)
        {
                fprintf(stdout,"fprintf failed\n");
                retval=TEMP; goto unlinkit;
        }

        while(fgets(buffer,BUFSIZE,stdin))
        {
```

```
        for(p=buffer;*p=='>';p++)
                ;

        if (!strncmp(p,"From ",5))
        {
                if (fputc('>',fp)<0)
                {
                        fprintf(stdout,"fputc failed\n");
                        retval=TEMP; goto unlinkit;
                }
        }

        if (fprintf(fp,"%s",buffer)<0)
        {
                fprintf(stdout,"fprintf failed\n");
                retval=TEMP; goto unlinkit;
        }
}

p=buffer+strlen(buffer);
nl=2;
if (*p=='\n')
        nl=1;

for(n=0;n<nl;n++)
{
        if (fputc('\n',fp)<0)
        {
                fprintf(stdout,"fputc failed\n");
                retval=TEMP; goto unlinkit;
        }
}

if (fsync(fd)<0)
{
        fprintf(stdout,"fsync failed\n");
        retval=TEMP; goto unlinkit;
}

if (fclose(fp)==EOF)
{
        fprintf(stdout,"fclose failed\n");
        retval=TEMP; goto unlinkit;
}

snprintf(newmaildir,BUFSIZE,"new/%u.%d.%s",thetime,pid,host);
if (link(mail,newmaildir)<0)
{
        fprintf(stdout,"link failed: %s %s\n",mail,newmaildir);
        retval=TEMP; goto unlinkit;
```

```
        }

unlinkit:
        if (unlink(mail)<0)
        {
                fprintf(stdout,"unlink failed: %s\n",mail);
                retval=TEMP;
        }

        exit(retval);
}
```

9.5 감사 (Acknowledgement)

Qmail에 의한 해결책을 가능하게 도움을 준 VicenteGonzalez(vince@nycrc.net) 에게 감사한다. 아마 Vince에게 감사의 메일 정도는 보낼 수 있겠지만, Qmail에 대한 것을 포함하여 이 HOWTO에 포함된 내용의 질문과 의견은 모두 나에게 보내도록 하라.

제 10 절 가상 삼바 (Virtual Samba)

10.1 설정

가상 SAMBA는 매우 설정하기 쉽다. 아래의 파일들이 적절하게 설정을 해줄 것이다.

- /virtual/domain1.com/etc/smb.conf FILE
- /virtual/domain1.com/var/lock/samba DIRECTORY
- /virtual/domain1.com/var/log DIRECTORY
- /usr/local/bin/virtsmbstatus SYMLINK /usr/local/bin/virtexec

10.2 Inetd

/etc/inetd.conf를 편집한다.

```
vi /etc/inetd.conf # Add this line
netbios-ssn stream tcp nowait root /usr/local/bin/virtuald \
        virtuald /virtual/conf.smbd smbd
```

10.3 Smb.init

서버가 inetd를 통해서 실행되므로 이 smb.init 파일은 필요하지 않다.

제 11 절 다른 가상 서비스

다른 가상 서비스도 비슷한 과정으로 제공될 수 있다.

- virtfs를 실행시키고 실행파일과 라이브러리를 가상의 파일 시스템에 넣는다.
- 이것을 /etc/inetd.conf에 넣는다.
- /virtual/conf.service 파일을 만든다.
- 필요한 다른 가상 스크립트를 만든다.

제 12 절 결론

이것이 필요한 모든 단계이다. 다시한번 말하는데, 회신은 ComputerResourceCenter로 보내라. 만약 당신이 질문을 가지고 있거나 이 문서를 갱신할 내용이 있다면, 나에게 그것을 알려주기 바란다.

좋은 반응들이 이 문서를 지속적으로 만들어왔다. 나에게 질문을 보낸 모든 사람들이 이 문서의 형태를 만드는데 도움을 주었고, 다시 필요성을 느끼는 많은 곳의 사람들에게 도움을 주게 한 것에 대해서 감사하게 생각한다. 질문하기 전에 FAQ를 먼저 읽어보고 해당 질문과 그에 대한 답이 있는지 확인해주기 바란다. 다시한번 문서에 관심을 가져준 여러분들에게 감사의 뜻을 전한다. Brian

제 13 절 자주 질문되는 사항들(FAQ)

Q1. sendmail.init과 syslogd.init을 만들었습니다. 이것들을 /usr/local/bin에 넣고 실행시키면 에러가 납니다.

A1. 이 파일들은 init 스크립트라고 한다. 이 프로그램들은 컴퓨터가 처음 부팅될 때 init이라는 프로그램에 의해 실행된다. init 프로그램은 /usr/local 밑의 바이너리 파일을 참조하지 않는다. 시스템에서 init 스크립트를 사용하기 위한 방법은 Linux System Administrators Guide나 Linux Getting Started Guide를 참고하라.

Q2. 다음과 같은 내용을 /etc/sendmail.cf에 넣었습니다.

```
divert(0)
VERSIONID(‘tcpproto.mc’)
OSTYPE(linux)
FEATURE(redirect)
FEATURE(always_add_domain)
FEATURE(use_cw_file)
FEATURE(local_procmail)
MAILER(local)
MAILER(smtp)
```

출력 결과가 이상하게 나오는데 그 이유가 무엇입니까?

A2. 이것을 /etc/sendmail.cf에 직접 넣으면 안된다. sendmail.cf는 sendmail이 이해하기 쉽게 쓰여졌기 때문에 보통 사람들이 보고 이해하기는 쉽지 않다. 따라서 보통 m4라는 프로그램이나 이것의 매크로를 이용하여 sendmail.cf 파일을 만들게 함으로써 설정을 보다 쉽게 하는 것이 일반적이다. 위의 라인들은 sendmail의 설정 파일을 위한 매크로이다. 매크로를 이용해서 sendmail을 설정하는 방법을 알고자 한다면 sendmail에 대한 문서를 참조하기 바란다. /etc/sendmail.cf 파일을 만든 다음 virtfs 스크립트를 이용하여 /virtual/domain1.com/etc/sendmail.cf로 파일을 복사하는 것을 잊지 말라. 그리고는 sendmail.cf 파일을 각각의 도메인에 맞게 설정해야 된다.

Q3. virtuald는 어디서 구할수 있습니까? 그것이 과연 무엇에 쓰는 것인가요? 또 그것을 어떻게 사용할 수 있습니까?

A3. Virtuald는 가상 서비스를 제공하기 위한 프로그램으로 내가 만든 C 프로그램이다. 이 프로그램의 소스는 당신이 보고 있는 바로 이 HOWTO에 포함되어 있다. 보통의 C 프로그램처럼 이것도 `make virtuald`의 방

법으로 컴파일할 수 있다. 만들어진 바이너리 파일은 /usr/local/bin에 놓는다. 일반적인 네트워크 서버 프로그램에 대한 포장(wrapper)으로 virtuald를 사용할 때에는 /etc/inetd.conf에 추가해야 할 것이 있다.

Q4. 내 시스템에는 dialog가 없는데요?

A4. dialog는 당신이 셸 스크립트에 다이얼로그 팝업 윈도를 넣을 수 있게 해주는 프로그램이다. 위에서 소개한 나의 virtual 셸 스크립트 예제를 실행시키기 위해서는 이 프로그램이 반드시 필요하다. dialog는 `sunsite.`에서 얻을 수 있을 것이다. 이 프로그램의 컴파일은 매우 쉽기 때문에 설치에 별 문제가 없을 것이다.

Q5. 가상의 syslogd가 작동중인지를 어떻게 알 수 있지요?

A5. virtuald가 작동중일 때에는 아래의 메세지가 syslogd(/var/log/messages)에 남게 된다:

```
Nov 19 17:21:07 virtual virtuald[10223]: Virtuald Starting: $Revision: 1.49 $
Nov 19 17:21:07 virtual virtuald[10223]: Incoming ip: 204.249.11.136
Nov 19 17:21:07 virtual virtuald[10223]: Chroot dir: /virtual/domain1.com
```

`Chroot dir` 메시지는 `chroot` 시스템 콜이 이루진 다음 virtuald에 의해 보내지게 된다. 만약 이 메시지가 나타난다면 가상의 syslogd는 실행중이라고 할 수 있다. 만약 당신이 가상 시스템으로 만든 서비스의 로그 메시지가 syslogd에 보내진 것을 확인한다면, 이것은 또한 가상의 syslogd가 맞게 설정된 것임을 확인할 수 있다.

만약 당신이 컴파일 시간 옵션 VERBOSELOG를 켜놓지 않았다면 virtuald는 전혀 기록을 남기지 않을 것이다. 이런 경우 가상의 syslogd가 실행되는지를 확인할 수 있는 유일한 방법은 가상 시스템을 구성하는 데몬들이 독립적으로 syslogd에 무엇인가를 기록하는 것을 참고하는 것이다.

Q6. 가상의 파일시스템에 걸쳐 쿼타(quota)를 설정하려면 어떻게 하죠?

A6. 보통 하는 방식으로 쿼타를 설정한다. `Quotamini-HOWTO` 를 참고하라.

하지만, 도메인이 다른 경우에도 uid에 혼란을 주어서는 안된다는 사실을 기억하라. 만약 uid에 혼란이 온다면 사용자들이 쿼타를 공유하게 될 것이다. 쿼타를 설정할 uid의 범위를 지정하고, 도메인에 쿼타를 설정할 사용자 이외에는 그 범위의 uid를 사용할 수 없게 만들어라.

Q7. inet.conf 엔트리에 있는 \ 표시는 무슨 뜻이죠?

A7. 단지 설정 파일에서 두 줄에 걸쳐 한 내용을 사용할 때 줄바꿈을 표시하는 것이다. 내가 이 표시를 사용한 것은 단어를 적당한 위치에 놓아 보기좋게 하고자 하는 의도였다. \을 없애고 하나의 줄로 설정 파일을 바꾸어도 상관없다.

Q8. passwd나 다른 login 프로그램을 사용할 때 `permission denied` 메시지가 나옵니다. FTP나 su를 사용할 때에는 `no modules loaded for service XXX` 라고 나옵니다. 왜 그럴까요?

A8. 이것들은 PAM 에러 메시지들이다. 나는 이 스크립트들을 PAM이 나오기 이전에 작성하였다. 내가 작성한 virtfs 스크립트는 /etc/pam.d, /usr/lib/cracklib_dict.*, /lib/security, 혹은 그밖의 PAM이 요구하는 파일들을 복사하는 부분을 가지고 있지 않다. PAM은 이들 파일들을 필요로 한다. 당신이 virtfs 스크립트를 편집하여 이들 파일들을 복사하게 만든다면 이들 문제는 사라질 것이다.

Q9. virtuald는 tcpd의 hosts.allow와 hosts.deny 파일들을 사용할 수 있나요?

A9. 물론. 약간의 수정으로 가능하다.

우선 소스의 두부분을 바꿔야 한다.

arguments를 확인하는 부분에 아래 내용을 첨가해야 한다.

```
        if (!argv[3])
        {
                syslog(LOG_ERR,"invalid arguments: no program to run");
                exit(0);
        }
```

exec 라인 역시 다음과 같이 바뀌어야 한다.

변경 전:

```
        if (execvp(argv[2],argv+2)<0)
```

변경 후:

```
        if (execvp(argv[2],argv+3)<0)
```

다음으로 inetd.conf 라인을 변경한다.

변경 전:

```
ftp stream tcp nowait root /usr/local/bin/virtuald \
        virtuald /virtual/conf.ftp wu.ftpd -l -a
```

변경 후:

```
ftp stream tcp nowait root /usr/local/bin/virtuald \
        virtuald /virtual/conf.ftp tcpd wu.ftpd -l -a
```

다음에 /virtual/domain1.com/etc/hosts.allow 와 /virtual/domain1.com/etc/hosts.deny 파일을 설정하여 이용할 수 있다.

Q10. 가상 호스트에서 CGI를 실행시킬 수 있나요?

A10. 물론 실행이 가능하지만, /cgi-bin 을 당신이 접근할 수 있는 `chroot` 의 밖에 설정할 것을 권한다. 한 예로, /var/www/cgi-bin/domain1.com을 생각해 보자. 클라이언트가 /cgi-bin에 접근할 수 있도록 허용하는 것은 당신의 서버에서 프로그램을 실행시킬 수 있는 기회를 부여하는 것이다. 이것은 보안상 큰 허점이 될 수 있다. 주의하라! 내 경우 버그를 자세히 검사하지 않는 시스템들에 대해서는 아예 cgi가 실행할 수 없도록 설정해 놓았다.

Q11. 내 설정파일들은 이 문서의 예제와 다릅니다. 어떻게 해야 하죠?

A11. 설정 파일에는 기본적으로 두가지 형태가 존재한다: 시스템V와 BSD가 그것이다. 이 문서에서 제공하는 예제는 시스템V 형태의 설정 파일에 기초하고 있다. 두 시스템 모두에서 가상 서비스는 성공적으로 작동한다. BSD 형태의 설정 파일들에 대한 정보는 시스템 구입처에 문의하거나 가까운 LDP 사이트를 참조하라.

Q12. 당신에게 메일을 보냈는데, 회신이 없거나 회신에 아주 오랜 시간이 걸립니다. 왜죠?

A12. 아마 메일의 제목에 VIRTSERVICES HOWTO라는 헤더를 넣지 않아서 그럴 것이다. 내가 네트워크 관리자로 가상 서버들과 그 클라이언트들을 관리하는 데 필요한 많은 일을 하고 있다는 점을 좀 생각해주고, 참아주기 바란다. 이 하우투 문서에 대한 메일을 정확히 보냈다면 2일에서 3일 안에 항상 답장을 받을 수 있을 것이다. 정확히 메일을 보내지 않았다면 나의 VIRTSERVICES 메일박스에 걸러져 들어오지 못했을 것이고, 그러면 며칠 혹은 몇주 동안 내가 모른 채 방치되어 있을지도 모른다.

Q13. virtuald는 100M 비트 환경에서도 작동하나요?

A13 네트워크 카드의 속도는 virtuald가 작동하는지의 여부와 아무 상관이 없다. 10M 비트 환경에서 100M 비트 네트워크 카드를 사용해도 정상적으로 작동한다는 사실을 확인해 보라.

Q14. sendmail의 virthost 테이블을 사용할 수 있나요?

A14. 안된다. 이것은 sendmail이 여러 도메인에 대한 정보를 받기 위한 기능이다. virtuald는 각각의 sendmail에 대해서 분리된 `chroot` 환경을 제공한다. virtuald를 설치한 후에는 sendmail을 각각의 도메인에 대해서 일반적인 방법으로 설정한다.

Q15. 가상 telnet을 설치할 수 있나요? 가상의 root 계정을 만들어 사용자들이 자신들의 도메인을 관리하게 만들 수 있나요?

A15. 이 질문은 굉장히 자주 나오는 질문이고, 솔직히 말하면 이 질문에 대해 답하는 것은 좀 피곤한 일이다. 이 문서에서 이미 여러번 말했듯이, 그 답은 inetd를 통하는 모든 서비스는 virtuald를 이용하여 가상 서비스로 만들수 있으며 위의 질문에 대해 안될 이유는 하나도 없다는 것이다. 상식적인 것이다. telnet 가상 서비스를 제공하여 얻는 이득이 무엇인지 모르겠지만, 이런 설정은 가상 머신의 보안 등의 문제에 많은 투자를 필요로할 것이다. 여기 이와 관련된 몇가지 내용들이 있다:

- 외부에서 접근하는 텔넷을 제어하여 여러개의 프로세스를 실행시키고, 외부로 나가는 연결에 대해 소스 IP 주소를 초기화하기 위해서는 커널을 해킹해야만 한다. 또 가상의 호스트이름을 사용하기 위해서는 gethostname에 대한 조작도 필요하다. 이런 작업이 숙련된 사용자에게는 단지 커널을 해킹하는 일일 뿐이지만, 초보자들에게는 절대적으로 권하지 않는다.
- 사용자들이 텔넷으로 접근하는 것을 허용하는 것은 임의의 프로그램을 당신의 리눅스 박스에서 실행시킬 수 있도록 허용하는 것과 같은 것이다. 알려져 있는 침입 방법으로 root의 계정이 노출되거나, 이 이유로 시스템에 손상이 가해질 수 있다.
- 가상 시스템에 대해서 root로 telnet 접근을 허용하는 것은 매우 안좋은 생각이다. 가상 시스템의 root도 raw device 파일을 읽을수 있고, `chroot`로 이것을 초기화하거나 시스템을 종료시키고 어떤 프로세스를 종료시키는 일 등을 수행할 수 있다.
- 이 텔넷 서비스가 실행되면 네트워크 서비스가 사용할 수 있는 귀중한 CPU 시간이 낭비될 것이다.
- 텔넷은 보안성이 떨어지는 네트워크 서비스이다. 평범한 텍스트 형태의 비밀번호가 네트워크를 통하여 보내지게 된다. 만약 악의있는 사용자가 이 비밀번호를 알게 된다면 그 사용자는 이를 이용해 시스템을 공격하고, 파괴시킬 수 있다.
- 당신의 가상 환경의 덩치가 커질 것이다. 보다 많은 공유 라이브러리들과 설정 파일, 그리고 보다 많은 실행 파일들을 필요하게 된다. 6G 정도의 하드 디스크도 금세 공간이 부족하게 될 것이다.

기본적으로 가상 시스템에 대해서 login을 허용하는 것은 진짜 안좋은 생각이다. 만약 이것이 허용된다면, 그 머신에서 호스팅하는 모든 사이트가 위험에 빠질 수 있다. 만약 특정 사이트를 사용자에 의해 관리될 수 있도록 설정하고자 한다면, ssh를 통해 로그인할 수 있는 사용자들을 추가, 삭제, 변경 가능하게 해줄 수 있는 가상 프로세스에 대한 프로그램을(스크립트가 아닌) 작성하는 것이 좋을 것이다. 이 프로그램은 메뉴 기동 방식(menu driven)이 되어야 하며, 콘솔을 허가하지 않고, root로 실행되는 것을 금지시켜야 한다. 그러기 위해서는 파일의 소유권을 root에서 몇몇 다른 사용자로 바꿀 필요가 있다. 이런 조치는 가상 시스템을 조금이라도 안전하게 만든다. 절대 root가 telnet이나 ssh를 통해 접속하는 것을 허용하지 말라. 이런 조치는 재앙을 불러올 것이다. 만약 가상 시스템에서 불가피하게 telnet을 사용할 필요가 있다면 피해가 최소화 될 수 있는 머신을 선택하라. 정상적인 관리자라면 이렇게까지 이야기했는데에도 텔넷을 제공하지는 않으리라 믿고, 더이상 시간낭비는 하지 않겠다.

Q16. virtuald나 Virtual-Services HOWTO에 대한 메일링 리스트, 웹페이지, rpm, tar 등의 정보가 있나요?

A16. 현재는 아무것도 없다. 이 HOWTO 문서가 내가 현재 하고있는 가상 서비스와 관련된 프로젝트의 전부이다. 나 역시 내가 다루지 못한 부분에 대한 HOWTO를 찾고 있다.

Q17. 일반 유저로 virtexec를 실행하려 했더니 `chroot: operation not permitted` 메시지가 나옵니다. 왜죠?

A17. `Chroot` 는 root에게 제한된 시스템 콜이다. 슈퍼 유저만이 이것을 실행시킬 수 있다. virtexec 스크립트는 `chroot` 프로그램을 실행시키기 때문에 이 스크립트를 실행시키고자 한다면 슈퍼 유저의 아이디가 필요하다.

Q18. pop과 sendmail을 설정했는데, pop이 제대로 작동하지 않는것 같군요. 어떻게 대처할까요?

A18. 몇몇 pop 프로그램들은 /usr/spool/mail을 메일 파일들을 놓기 위해 사용한다. 내가 알기로는 qpop의 경우 이 문제를 해결하기 위해 수동으로 편집을 해주어야 한다. 소스 코드를 다시 컴파일하거나 /virtual/domain1.com/usr/spool 을 /virtual/domain1.com/var/spool로 하드 링크 시키면 된다.

Q19. 나는 당신이 말한 프로그램을 쓰지 않고, 대신 XXX라는 프로그램을 썼는데, 작동이 되지 않는군요. 왜죠?

A19. 나는 가급적이면 널리 사용되는 일반적인 서버들에 대해 기술하려고 했다. 하지만, 나는 모든 사람들이 자신이 특별히 좋아하는 서버가 있다는 것도 알고 있다. 이런 경우 가급적 많은 정보를 나에게 보내준다면, 당신의 문제점을 풀기 위한 시도를 할 것이고, 더불어 이 FAQ에 문서화 하는 작업을 할 것이다. 나에게 보낼 정보 가운데 가장 중요한 것은 당신이 실행시킨 버전의 프로그램을 어디에서 얻었는가 하는 점이다. (ftp://ftp.domain1.com/subdir/subdir/file.tgz의 형식으로 보내라)

Q20. virtexec가 실행될 때 `symlink not a virt function` 라는 메시지가 발생합니다. 이게 어떤 뜻이고, 그것을 어떻게 수정해야 합니까?

A20. virtexec는 처음 4개의 문자를 0번째 argument로 취하고 나머지 이름을 가상 환경에서 실행시키게 된다. 한 예로 virtpaswd는 passwd를 실행시킨다. 따라서 만약 처음 제거하는 네 문자가 `virt` 가 아니라면 에러 메시지를 내보내게 된다. virtexec는 셸 스크립트로 쓰여져 있기 때문에 이해하기 쉬울 것이다. bash나 기타 셸들의 메뉴얼 페이지가 셸 스크립트 프로그래밍에 대한 질문에 답이 될 수 있을 것이다.

Q21. virtuald와는 상관이 없지만 Qmail이나 SAMBA, Apache 등에 대해서 질문이 있습니다.

A21. 위에서 말한 모든 프로그램들은 훌륭한 문서들을 이미 가지고 있다. 몇몇 프로그램의 경우에는 www.(패키지 이름).org 와 같이 전용 웹사이트까지 마련되어 있다. 그렇게 가상 서비스와 관계없는 질문들에 대해서는 이들 사이트에서 적당한 질문처를 찾기 바란다.

Q22. 몇개의 도메인 앨리어스를 domain1.com에 가지고 있는데, 매일이 자꾸 되돌아옵니다. 왜 그럴까요?

A22. Virtmaildelivery는 전달되는 환경 변수들에 의해서 메일을 보낼 /virtual/domain1.com 디렉토리를 판단한다. 이것은 메일의 주소를 확인하기 위해 어떤 DNS 확인 절차도 갖지 않는다. 하지만, 만약 주소가 sub-mail.mail.domain1.com이 된다면, virtmaildelivery는 주소에 대해서 확인을 하고, 차례대로 mail.domain1.com, domain1.com, com을 확인하면서 주소를 검색하게 된다. 이런 방법으로는 정해진 앨리어싱의 주소를 정확하게 알아낼 수 없다.

하지만, 만약 서브도메인의 형태가 아닌 도메인 앨리어스를 가지고 있다면, 다음과 같은 심볼릭 링크를 통해 구현이 가능하다:

```
cd /virtual
ln -s domain1.com domain1alias.com
```

virtmaildelivery는 두 디렉토리를 모두 존재하는 디렉토리로 인식할 것이고 (한쪽은 심볼릭 링크임에도 불구하고), 메일은 user@domain1.com과 user@domainalias.com 모두에게 보내질 수 있게 된다. virtexec는 두 도메인 모두를 다이얼로그 박스에 표시한다는 점을 주의하라. 두 파일 시스템이 같은 것이긴 하지만, 당신은 두 도메인 가운데 하나를 선택할 수 있다.