Virtual Services Howto

Table of Contents

Table of Contents

Virtual Services Howto

Brian Ackerman, `brian@nycrc.net`

v2.1, 15 August 1998

This document came about to satisfy the ever increasing need to know how to virtualize a service.

5.DNS

6.Syslogd

7.Virtual FTP

8.Virtual Web

9.Virtual Mail/Pop

10.Virtual Samba

11.Virtual Other

12.Conclusion

13.FAQ

1.Introduction

1.1 Knowledge Required

Creating a virtual services machine is not all that difficult, however, more than fundamental knowledge is required. This document is not a primer to how to fully configure a Linux machine.

In order to understand this HOWTO document it is assumed that you are thoroughly familiar with the following:

- Compiling a Linux kernel and adding IP aliasing support IP alias mini−HOWTO
- Setting up and configuring of network devices NET−3 HOWTO
- Setting up of inetd NET−3 HOWTO
- Various network packages like SendmailApacheQmailSAMBA
- Setting up DNS DNS HOWTO
- Understanding basic system administration Linux Systems Administrators's Guide
- Understanding how to setup a Web Server WWW HOWTO

If you are uncertain of how to proceed with any of the above it is STRONGLY recommended that you use the html links provided to familiarize yourself with all packages. I will NOT reply to mail regarding any of the above. Please direct your questions to the appropriate author of the HOWTO.

1.2 Purpose

The purpose of virtual services is to allow a single machine to recognize multiple IP addresses without multiple network cards. IP aliasing is a kernel option that allows you to assign each network device more than one IP address. The kernel then multiplexes (swaps between them very fast) in the background and to the user it appears like you have more than one server.

This multiplexing allows multiple domains (www.domain1.com, www.domain2.com, etc.) to be hosted by the same machine for the same cost as hosting one domain. Unfortunately, most services (FTP, web, mail) were not designed to handle muliple domains. In order to make them work properly you must modify both configuration files and source code. This document describes how to make these modifications in the setting up of a virtual machine.

A deamon is also required in order to make virtual services function. The source for this daemon (virtuald) is provided later in this document.

1.3 Feedback

This document will expand as packages are updated and source or configuration modifications change. If there are any portions of this document that are unclear please feel free to email me with your suggestions or questions. So that I do not have to go searching through the entire HOWTO please make certain that all comments are as specific as possible and include the section where the uncertainty lies. It is important that all mail be addressed with VIRTSERVICES HOWTO in the subject line. Any other mail will be considered personal and all my friends know that I do not ever read my personal mail so it will probably get discarded with theirs.

Please note that my examples are just that, examples and should not be copied verbatim. You may have to insert your own values. If you are having trouble, send me mail. Include all the pertinent configuration files and the error messages you get when installing and I will look them over and reply with my suggestions.

1.4 Revision History

V1.0

Initial version

V1.1

Fixed error in Virtual Web Section

V1.2

Fixed the date

V2.0

Updated html links.

Web updates.

New Sendmail option.

New Qmail section.

Syslogd updates.

FTP updates.

Virtuald default option.

New SAMBA section.

FAQ updates.

V2.1

Changed all paths to /usr/local.

Added virtuald VERBOSELOG compile option.

Fixed setuid/setgid bug in virtmailfilter.

Fixed execl bug in virtmailfilter.

Fixed capitialization bug in virtmailfilter.

Fixed environment variable sanity check in virtmailfilter.

Removed mbox code from virtmailfilter/virtmaildelivery.

Added tcpserver.init pop section for Qmail.

Added alias domain name question to the FAQ.

Fixed virtmailfilter to send home directory to virtmaildelivery.

1.5 Copyright/Distribution

2.IP Aliasing

IP aliasing is a kernel option that needs to be set up in order to run a virtual hosting machine. There is already a mini−HOWTO on IP aliasing. Consult that for any questions on how to set it up.

3.Virtuald

3.1 Introduction

Every network connection is made up of two IP address/port pairs. The API (Applications Program Interface) for network programming is called the Sockets API. The socket acts like an open file and by reading/writing to it you can send data over a network connection. There is a function call `getsockname` that will return the IP address of the local socket. Virtuald uses `getsockname` to determine which IP on the local machine is being accessed. Virtuald reads a config file to retrieve the directory associated with that IP. It will `chroot` to that directory and hand the connection off to the service. `Chroot` resets / or the root directory to a new point so everything higher in the directory tree is cut off from the running program. Therefore, each IP address gets their own virtual filesystem. To the network program this is transparent and the program will behave like nothing happened. Virtuald in conjunction with a program like inetd can then be used to virtualize any service.

3.2 Inetd

Inetd is a network super server that listens at multiple ports and when it receives a connection (for example, an incoming pop request), inetd performs the network negotiation and hands the network connection off to the specified program. This prevents services from running idly when they are not needed.

A standard /etc/inetd.conf file looks like this:

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

A virtual /etc/inetd.conf file looks like this:

```
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 Config File

Each service gets a config file that will control what IPs and directories are allowed for that service. You can have one master config file or several config files if you want each service to get a different list of domains. A config file looks like this:

```
# 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 Source

This is the C source code to the virtuald program. Compile it and install it in /usr/local/bin with permission 0755, user root, and group root. The only compile option is VERBOSELOG which will turn on/off logging of connections.

```
#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);
}
```

4.Shell Scripts

4.1 Virtfs

Each domain should get their own directory structure. Since you are using `chroot` you will require duplicate copies of the shared libraries, binaries, conf files, etc. I use /virtual/domain1.com for each domain that I create.

I realize that you are taking up more disk space but it is cheaper than a whole new machine and network cards. If you really want to preserve space you can hard link the files together so only one copy of each binary exists. The filesystem that I use takes up a little over 2M. However, this script attempts to copy all the files from the main filesystem in order to be as generic as possible.

Here is a sample virtfs script:

```
#!/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

To execute commands in a virtual environment you have to `chroot` to that directory and then run the command. I have written a special shell script called virtexec that handles this for any command:

```
#!/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
```

Please note that you must have the `dialog` program installed on your system for this to work. To use virtexec just symlink a program to it. For example,

```
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
```

Then if you type virtvi or virtpasswd or virtmailq it will allow you to vi a program, change a user's password or check the mail queue on your virtual system. You can create as many virtexec symlinks as you want. Please note that if your program requires a shared library it has to be in the virtual filesystem as well as the binary.

4.3 Notes

I install all the scripts in /usr/local/bin. Anything that I do not want to put on the virtual filesystem I put in /usr/local. The script does not copy any of the files in /usr/local to the virtual filesystem. Any files that are important to not cross virtual filesystems should be removed. For example, ssh is installed on my system and I did not want the private key for the server available on all the virtual filesystems so I remove it from each virtual filesystem after I run virtfs. I also change resolv.conf and remove anything that has the name of another domain on it for legal reasons. For example, /etc/hosts and /etc/HOSTNAME.

The programs that I symlink to virtexec are:

* virtpasswd −− change a user password
* virtadduser −− create a user
* virtdeluser −− delete a user
* virtsmbstatus −− see SAMBA status
* virtvi −− edit a file
* virtmailq −− check out the mailq
* virtnewaliases −− rebuild alias tables

5.DNS

You can configure DNS normally. There is a HOWTO on DNS.

6.Syslogd

6.1 Problem

Syslogd is the system logging utility commonly used on UNIX systems. Syslogd is a daemon that opens a special file called a FIFO. A FIFO is a special file that acts like a pipe. Anything that is written to the write side will come out the read side. Syslogd waits for data from the read side. There are C functions that write to the write side. If your program uses these C functions your output will go to syslogd.

Remember that we have used a `chroot` environment and the FIFO that syslogd is reading from (/dev/log) is not present. That means all the virtual environments will not log to syslogd.

6.2 Solution

Setup Links

Syslogd can look to a different FIFO if you tell it on the command line so run syslogd with the argument:

```
syslogd −p /virtual/log
```

Then symlink /dev/log to /virtual/log by:

```
ln −sf /virtual/log /dev/log
```

Then hard link all the /dev/log copies to this file by running:

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

The virtfs script above already does this. Since /virtual is one contiguous disk and the /dev/log's are hard linked they have the same inode number and point to the same data. The `chroot` cannot stop this so all your virtual /dev/log's will now function. Note that all the messages from all the environments will be logged in one place. However, you can write separate programs to filter out the data.

Syslogd.init

This version of the syslogd.init file hard links the /dev/log's each time you start it because syslogd deletes and creates the /dev/log FIFO each time it runs. Here is a modified syslogd.init file:

```
#!/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 Multiple Syslogd's

One Per Disk

If you run out of space on one filesystem and you have to break up your virtual domains onto different disks remember that hard links will not cross disks. That means you will have to run a separate syslogd for each group of domains on a disk. For example, if you had thirteen domains on /virtual1 and fifteen domains on /virtual2, you would hard link thirteen domains to /virtual1/log and run one syslogd with `syslogd -p /virtual1/log` and hard link fifteen other domains to /virtual2/log with a syslogd running with `syslogd -p /virtual2/log` .

One Per Domain

If you do not want to centralize the logs to one place you could also run one syslogd per domain. This wastes process ID's so I do not recommend it but it is easier to implement. You would have to alter your syslogd.init file to run syslogd as `chroot /virtual/domain1.com syslogd` for each domain. This will run each syslogd within the `chroot` and the logs will be in /virtual/domain1.com/var/log rather than all combined in /var/log. Do not forget to run a syslogd normally `syslogd` for the main system and a kernel logger `klogd` .

7.Virtual FTP

7.1 Inetd

Wu−ftpd comes with built in support to make it virtual. However, you cannot maintain separate password files for each domain. For example, if `bob@domain1.com` and `bob@domain2.com` both want an account you would have to make one of them bob2 or have one of the users choose a different user name. Since you now have a virtual filesystem for each domain you have separate password files and this problem goes away. Just create a virtnewuser script and a virtpasswd script in the way mentioned above and you are all set.

The inetd.conf entries for wu−ftpd:

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

7.2 Anonymous FTP

These are unaffected by the virtuald setup. For an anonymous user just create the FTP user in /virtual/domain1.com/etc/passwd like you would normally.

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

Then setup the anonymous FTP directory. You have separate password files for each domain so you can restrict which domain has an anonymous FTP account. Please note that since the FTP server is already `chrooted` into the /virtual/domain1.com directory you do not have to prefix any paths with it.

7.3 Virtual FTP Users

Wu−ftpd supports something called a guest group. This allows you to create different FTP areas for each user. The FTP server does a `chroot` to the specified area so the user cannot go outside that directory tree. If you create the users within a virtual domain this way they will not be able to view the system files.

Add the guest's group to the /virtual/domain1.com/etc/ftpaccess file.

Create an entry in /virtual/domain1.com/etc/passwd with the `chroot` dir and the starting home directory separated by `/./` :

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

Then setup guest's home like you would for anonymous FTP. You have separate password files for each domain so you can specifiy which domains have guest accounts and which users within a domain are guest users. Please note that since the FTP server is already `chrooted` into the /virtual/domain1.com directory you do not have to prefix any paths with it.

8.Virtual Web

8.1 Running With Virtuald

Not recommended

Apache has their own support for virtual domains. This is the only program I recommend using the internal virtual domain mechanism. When you run something through inetd there is a cost, the program has to start up each time you run it. This results in slower response time, which is perfectly fine for most services but is completely unacceptable for web service. Apache also has a mechanism for stopping connections when too many come in, which can be critical for even medium volume sites.

Simply stated, virtualizing Apache with virtuald is a really bad idea. The whole point of virtuald is to fill the gap created when services DO NOT have their own internal mechanism to do the job. Virtuald is not meant to replace good code that already completes the task at hand.

The above not withstanding here is how to do it for those who are foolhardy enough to do so.

Inetd

Edit /etc/inetd.conf

```
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
```

Httpd.conf

Edit /var/www/conf/httpd.conf

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

Replace it with:
ServerType inetd
```

Configuration

Then configure each instance of the Apache server like you would normally for single domain use.

Httpd.init

An httpd.init file is not needed since the server is run through inetd.

8.2 Running With Apache VirtualHost

Apache has three configuration files `access.conf`, `httpd.conf`, and `srm.conf`. Newer versions of Apache have made the three configuration files unnecessary. However, I find that breaking up the configuration into three sections makes it easier to manage so I will be keeping with that style in this HOWTO document.

Access.conf

This configuration file is used to control the accessibility of directories in the web directory structure. Here is a sample configuration file that shows how to have different options for each domain.

```
# /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>
```

Httpd.conf

This configuration file is used to control the main options for the Apache server. Here is a sample configuration file that shows how to have different options for each domain.

```
# /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
```

Srm.conf

This configuration file is used to control how requests are serviced and how results are formatted. You do not have to edit anything here for the virtual domains. The sample config file from Apache should work.

Httpd.init

Nothing special has to be done to the httpd.init file. Use a standard one that comes with the Apache configuration.

8.3 File Descriptor Overflow

Warning

This only applies to the standalone style Apache server. A server run through inetd does not interact with the other domains so it has the whole file descriptor table.

Every log file that the Apache server opens is another file descriptor for the process. There is a limit of 256 file descriptors per process in Linux. Since you have multiple domains you are using a lot more file descriptors. If you have too many domains running off of one Apache web server process you can overflow this table. This would mean that certain logs would not work and CGI's would fail.

Multiple Apache Servers

If you assume five file descriptors per domain you can have 50 domains running on your Apache server without any problems. However, if you find your server having problems like this you could create /var/www1 with an Apache server in charge of domain1 – domain25 and /var/www2 with an Apache server in charge of domain26 – domain50 and so on. This would give each server their own configuration, error, and log directory. Each server should be configured separately with their own Listen and VirtualHost directives. Do not forget to run multiple servers in your httpd.init file.

8.4 Sharing Servers With One IP

Saving IPs

The HTTP (HyperText Transfer Protocol) version 1.1 added a feature that communicates the name of the server to the client. This means that the client does not need to look up the server from its IP address. Therefore, two virtual servers could have the same IP address and be different web sites. The Apache configuration is the same as above except that you do not have to put in a different Listen directive since the two domains will have the same IP.

Drawback

The only problem is that virtuald uses IP addresses to distinguish between domains. In its current form virtuald would not be able to `chroot` to different spool directories for each domain. Therefore, mail would only be able to respond as one IP and there would no longer be a unique spool directory for each domain. All the web sharing IP clients would have to share that IPs spool directory. That would mean duplicate usernames would be an issue again. However, that is the price you pay for sharing IPs.

8.5 More Information

This HOWTO only shows how to implement virtual support on the Apache web server. Most web servers use a similar interface. For more information on virtual web hosting consult the WWW HOWTO, the documentation for Apache at Apache's Site, or the documentation at ApacheWeek.

9.Virtual Mail/Pop

9.1 Problem

Virtual mail support is in ever increasing demand. Sendmail says it supports virtual mail. What it does support is listening for incoming mail from different domains. You can then specify to have the mail forwarded somewhere. However, if you forward it to the local machine and have incoming mail to bob@domain1.com and bob@domain2.com they will go to the same mail folder. This is a problem since both bob's are different people with different mail.

9.2 Solution

You can make sure that each user name is unique by using a numbering scheme: bob1, bob2, etc or prepending a few characters to each username dom1bob, dom2bob, etc. You could also hack mail and pop to do these conversions behind the scenes but that can get messy. Outgoing mail also has the banner maindomain.com and you want each subdomain's outgoing mail banner to be different.

I have two solutions. One works with sendmail and one works with Qmail. The solution with sendmail should work with a stock install of sendmail. However, it shares all the limitations built into sendmail. It also requires that one sendmail has to be run in queue mode for each domain. Having 50 or more sendmail queue processes that wake up every hour can put a little strain on a machine.

The solution offered with Qmail does not require multiple instances of Qmail and can run out of one queue directory. It does require an extra program since Qmail does not rely on virtuald. I believe a similar procedure can be done with sendmail. However, Qmail lends itself to this solution more readily.

I do not endorse any one program over the other. The sendmail install is a little more straight forward but Qmail is probably the more powerful of the two mail server packages.

9.3 Sendmail Solution

Introduction

Each virtual filesystem gives a domain its own /etc/passwd. This means that bob@domain1.com and bob@domain2.com are different users in different /etc/passwds so mail will be no problem. They also have their own spool directories so the mail folders will be different files on different virtual filesystems.

Create Sendmail Configuration File

Create /etc/sendmail.cf like you would normally through m4. I used:

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

Edit Sendmail Configuration File

Edit /virtual/domain1.com/etc/sendmail.cf to respond as your virtual domain:

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

#Dj$w.Foo.COM

Replace it with:

Djdomain1.com
```

Sendmail Local Delivery

Edit /virtual/domain1.com/etc/sendmail.cw with the local hostnames.

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

Sendmail Between Virtual Domains: The Hack (PRE8.8.6)

However, sendmail requires one minor source code modification. Sendmail has a file called /etc/sendmail.cw and it contains all machine names that sendmail will deliver mail to locally rather than forwarding to another machine. Sendmail does internal checking of all the devices on the machine to initialize this list with the local IPs. This presents a problem if you are mailing between virtual domains on the same machine. Sendmail will be fooled into thinking another virtual domain is a local address and spool the mail locally. For example, bob@domain1.com sends mail to fred@domain2.com. Since domain1.com's sendmail thinks domain2.com is local, it will spool the mail on domain1.com and never send it to domain2.com. You have to modify sendmail (I did this on v8.8.5 without a problem):

```
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 */
```

Note only do this if you need to send mail between virtual domains which I think is probable.

This will fix the problem. However, the main ethernet device eth0 is not removed. Therefore, if you send mail from a virtual IP to the one on eth0 on the same box it will delivery locally. Therefore, I just use this as a dummy IP virtual1.maindomain.com (10.10.10.157). I never send mail to this host so neither will the virtual domains. This is also the IP I would use to ssh into the box to check if the system is ok.

Sendmail Between Virtual Domains: New Sendmail Feature (POST8.8.6)

As of Sendmail V8.8.6, there is a new option to disable loading of the extra network interfaces. This means you do NOT have to alter the code in any way. It is called `DontProbeInterfaces` .

Edit /virtual/domain1.com/etc/sendmail.cf

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

Sendmail.init

Sendmail cannot be started stand alone anymore so you have to run it through inetd. This is inefficient and will result in lower start up time but if you had such a high hit site you would not share it on a virtual box with other domains. Note that you are NOT running with the `-bd` flag. Also note that you need a `sendmail -q` running for each domain to queue up undelivered mail. The new sendmail.init file:

```
#!/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
```

Inetd Setup

Pop should install normally with no extra effort. It will just need the inetd entry for it with the virtuald part added. The inetd.conf entries for sendmail and pop:

```
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 Qmail Solution

Introduction

This solution takes over the delivery responsibilities of qmail−local, so use of the .qmail files in the virtual home directories will not work. However, each domain will still get a domain master user that will control aliasing for the whole domain. Two external programs will be used for that domain masters .qmail−default file. The mail will be passed through these two programs in order to deliver mail for each domain.

Two programs are required since one of them is run setuid root. It is a small program that changes to a non−root user and then runs the second program. Consult your nearest security related site for a discussion as to why this is necessary.

This solution bypasses the need for using virtuald. Qmail is flexible enough to not require a general virtuald setup. Qmail's design utilizes the chaining of programs together to deliver mail. This design makes it very easy to insert the virtual section into the Qmail delivery process without altering a stock install of Qmail.

A note that since you are using one Qmail any unqualified domain name will be expanded with the domain of the main server. This is because you do not have a separate Qmail server for each domain. Therefore, make sure that your client (Eudora, elm, mutt, etc.) knows to expand all of your unqualified domain names.

Setup Virtual Domains

Qmail has to be configured to accept mail for each of the virtual domains you will be serving. Type the following commands.

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

Setup Domain Master User

Add to your main /etc/passwd file the user domain1. I would make the shell /bin/false so that the domain master cannot log in. That user will be able to add .qmail files and all mail for domain1 will route through that account. Note that usernames can only be eight characters long and domain names can be longer. The remaining characters are truncated. That means that user domain12 and domain123 are going to be the same user and Qmail might get confused. So be careful in your master domain user naming convention.

Create the domain master's .qmail files with the following commands. Add any other system aliases at this point. For example, webmaster or hostmaster.

```
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
```

Create the domain master's .qmail–default file. This will filter all mail to the virtual domain.

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

Tcpserver

Qmail requires a special pop that can support the Maildir format. The pop program has to be virtualized. The author of Qmail recommends using tcpserver (an inetd replacement) with Qmail so my examples use tcpserver and NOT inetd.

Tcpserver does not require a config file. All the information can be passed to it via the command line. Here is the tcpserver.init file that you would use for the mail daemon and popper:

```
#!/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
```

Qmail.init

You can use the standard Qmail init script provided. Qmail comes with very good documentation describing how to set this up.

Source

You require two other programs to get virtual mail working with Qmail. They are virtmailfilter and virtmaildelivery. This is the C source to virtmailfilter. It should be installed in /usr/local/bin with permissions 4750, user root, and group nofiles.

```
#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));
}
```

Source

You require two other programs to get virtual mail working with Qmail. They are virtmailfilter and virtmaildelivery. This is the C source to virtmaildelivery. It should be installed in /usr/local/bin with permissions 0755, user root, and group root.

```
#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 Acknowledgement

Thank you Vicente Gonzalez (vince@nycrc.net) for helping make the Qmail solution possible. You can certainly mail your thanks to Vince, however all questions and comments including issues regarding Qmail, about this HOWTO should continue to be directed to me.

10.Virtual Samba

10.1 Setup

Virtual SAMBA is very simple to install. Make sure that the following files are setup properly:

- /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

Edit /etc/inetd.conf

```
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

An smb.init file is not needed since the server is run through inetd.

11.Virtual Other

Any other service should be a similar procedure.

- Run virtfs to add the binaries and libraries to the virtual filesystem.

* Add it to /etc/inetd.conf.
* Create a /virtual/conf.service file.
* Create any virtual scripts that need to be made.

12.Conclusion

Those are all the steps you need. Again mail any responses to Computer Resource Center. If you have a question or an update to the document let me know and I will add it.

The document has met with a very good response. I thank all the people who sent me questions as they are helping to shape the document to meet the needs of users everywhere. Before you ask a question I urge you to read the FAQ to see if it has been already asked and answered. Thanks again. Brian

---

13.FAQ

Q1. I created sendmail.init and syslogd.init. I put them in /usr/local/bin and tried to run them but I got errors.

A1. These files are called init scripts. They are run by the program init when your computer boots. They do not go with the /usr/local binaries. Consult the Linux System Administrators Guide or the Linux Getting Started Guide for information on how to use the init scripts system.

Q2. I put these lines into /etc/sendmail.cf

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

And I got really stange output. Why?

A2. You do not put these lines directly in /etc/sendmail.cf. The sendmail.cf file was written to be easy for sendmail to understand and hard for humans to read. Therefore, to make it easy to configure we use a program called m4 and its macro capabilities to create the sendmail.cf file. The FEATURE lines are actually macros that expand to sendmail configuration statements. See the sendmail docs on how to configure sendmail through this method. Also note that you create a main /etc/sendmail.cf file and the virtfs script then copies this to /virtual/domain1.com/etc/sendmail.cf. Then you edit that sendmail.cf file to respond as your domain.

Q3. Where do I get virtuald, what is it, and how do I use it?

A3. Virtuald is C source that I wrote to run a virtual service. It is included with this HOWTO. You compile it like a normal C program `make virtuald` . The resulting binary is placed into /usr/local/bin. Add lines to /etc/inetd.conf that use virtuald as a wrapper to a normal network server program.

Q4. I do not have dialog installed on my system?

A4. Dialog is a program that allows you to put dialog pop up windows into your shell scripts. It is required for my virtual shell script examples to work. You can get a copy of dialog at sunsite. It compiles very easily and should be no problem to install.

Q5. How can I know if virtual syslogd is working?

A5. When virtuald runs it should output the following messages to syslogd (/var/log/messages):

```
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
```

The `Chroot` dir message is sent by virtuald after the `chroot` system call is performed. If this message appears virtual syslogd is working. If the service you are virtualizing logs messages to syslogd and you see them that is also a sign that virtual syslogd is correctly setup.

Note that if you have not turned on the compile time option VERBOSELOG, virtuald will not log at all. The only way to tell if virtual syslogd is working at that point is if the daemon you are virtualizing independently logs something to syslogd.

Q6. How can I setup quotas across virtual filesystems?

A6. You setup quotas like you would normally. See the Quota mini−HOWTO. However, you have to make sure there are no uid conflicts across domains. If there are conflicts you will have users sharing a quota. Set aside a range of uid's that you know will have quota's enabled and tell your domains that they cannot have any users in that range except the ones registered to have a quota.

Q7. What is this \ notation in all the inetd.conf entries?

A7. That is just a method of breaking up config files across two lines. I did that so the line would word wrap in a nice place. You can just ignore the \ and join the two lines back together.

Q8. When I run passwd or other login programs I get `permission denied` . When I run FTP or su I get `no modules loaded for service XXX` . Why?

A8. Those are PAM error messages. I wrote these scripts before PAM was out. My virtfs script does not copy /etc/pam.d, /usr/lib/cracklib_dict.*, /lib/security or any of the other files PAM requires. PAM needs these to function. If you edit my virtfs script to copy these files the problem will go away.

Q9. Can virtuald work with tcpd hosts.allow and hosts.deny files?

A9. Yes it can with some modifications.

First the source has to be changed in two places.

This has to be inserted where the arguments are checked.

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

The exec line has to be changed from:

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

to:

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

Second the inetd.conf lines have to be changed from:

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

to:

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

Third edit the /virtual/domain1.com/etc/hosts.allow and /virtual/domain1.com/etc/hosts.deny files accordingly.

Q10. Can my virtual hosts run CGI's?

A10. Yes they can but I recommend putting the /cgi−bin in a place outside of the `chroot` that only you have access to. For example, /var/www/cgi−bin/domain1.com. Giving clients access to /cgi−bin is giving them the opportunity to run programs on your sever. This is a big security hole. Be careful. I do not let any cgi run on my systems that I have not personally inspected for bugs.

Q11. My configuration files are different from your examples. What do I do?

A11. There are two basic configuration styles: SystemV and BSD. The examples provided in the HOWTO are based on SystemV style configuration files. Virtual services works equally well on either system. For information on BSD style configuration files consult the origin of your distribution or the nearest LDP site.

Q12. I sent you mail and have not heard a response from you or your response took a long time. Why?

A12. Probably because you did not put VIRTSERVICES HOWTO in your subject header. Please bear in mind that I am a network administrator and that among the other things I do in my 20 hour days is administering my own virtual boxes and those of my clients. Mail that is properly addressed is always responded to within two or three days. Mail that is improperly addressed does not get filtered into my VIRTSERVICES mailbox and can lie around unnoticed for days or weeks.

Q13. Does virtuald work under 100Mbit?

A13 The speed of the network card is unrelated to whether virtuald will work or not. Try making sure that your server works under 10Mbit and that your 100Mbit network card works normally without a virtual server.

Q14. Should I use sendmail's virthost table?

A14. No. That is sendmail's feature to accept info for multiple domains. Virtuald gives each sendmail its own separate `chroot` environment. Install virtuald and then configure sendmail like you would normally for each domain.

Q15. Can I setup virtual telnet on my machine? What about creating a virtual root account so clients can administer their own domains?

A15. These questions come to me quite often and to be honest, I am getting a bit tired of them. The answer, as stated numerous times in the documentation, is that any service run through inetd can be virtualized using virtuald so there is nothing to stop you from doing either of the above. Nothing except common sense. Whatever benefits you might derive from allowing telnet are heavily outweighed by the cost to the virtual box (and thus the sites you are supposed to be hosting in a responsible manner) in terms of security. Here are just a few issues involved:

- In order to completely fool an incoming telnet session you have to hack the kernel to get multiple procs working, reset your source IP address for outgoing connections, fool gethostname so it uses the virtual hostname and not the system hostname, etc. If you are an advanced user then by all means hack the kernel. For the newbie I do not recommend it.
- By allowing users to come into your box via telnet you allow them to run arbitrary programs. Through known hacks you can get root and cause damage to the system.
- Giving a root telnet account on a virtual box is very bad. A root virtual user can still read raw device files which nullifies the `chroot` , shutdown the system, and can kill other processes on the system.
- The programs that these telnet sessions are running take up valuable CPU time that the network services could be using.
- Telnet is an insecure network service. Plain text passwords are sent out over the net. If a malicious user gets this password he/she can use the above mentioned attacks to harm your system.
- Your virtual environments will have to be bigger. You will need more shared libraries, more configuration files, and more binaries. A six gigabyte disk can run out of space really fast.

The bottom line is that allowing login's on a virtual box is a really bad idea. If permitted, every site hosted on that machine is at risk. If you want to allow a site holder to administer users then you are advised to write (not script) the code necessary to run the virtual processes that allow them to add, delete or modify users upon login through ssh. This should be completely menu driven, should never allow a console and should not run as root. In order to accomplish this you will have to change ownership of the pertinent files from root to some other user. If done in this manner it is marginally safe to incorporate into a virtual machine. There is never an acceptable time to allow root login's either through telnet or ssh. Doing so is simply an invitation to disaster. If there is an overwhelming reason to run telnet then the site should be hosted on a dedicated machine where the only risk is to the individual site. No responsible administrator would ever do otherwise and so I will waste no more time on this issue.

Q16. Is there an rpm, tar, web site, mailing list, etc. associated with virtuald and the Virtual−Services HOWTO?

A16. Currently there is nothing like that available. This HOWTO is the only source of information to everything I do concerning this project. I find the HOWTO to be fairly self contained making the need for other pieces of information superfluous.

Q17. When I try to run virtexec as a regular user I get `chroot: operation not permitted` . Why?

A17. `Chroot` is a root restricted system call. Only the superuser can execute it. The virtexec script runs the `chroot` program which is why you need to be root in order to run it.

Q18. I setup pop and sendmail but popping mail does not seem to work. How come?

A18. Some pop programs come with /usr/spool/mail as their place for mail files. I know that qpop has to be manually editted to fix this. Either recompile the source to your program or symlink /virtual/domain1.com/usr/spool to /virtual/domain1.com/var/spool.

Q19. I did not use the program mentioned in your HOWTO, I used program XXX. It does not work. Why?

A19. I tried to make sure to use the most generic of each server in my examples. However, I know that everyone has their favorite version of each server. Send me as much information as possible and I will try to figure out how to solve your problem and document it in the FAQ. The most important piece of information to send me is where to get the version of the software you are running (in the form ftp://ftp.domain1.com/subdir/subdir/file.tgz).

Q20. When I run virtexec is says `symlink not a virt function` . What does this mean and how do I fix it?

A20. Virtexec is a program that will take its zero argument, strip off the first four characters, and run the remaining name in the virtual environment. For example, virtpasswd runs passwd. If the first four characters that it strips off are not `virt` it complains and outputs that error message. Virtexec is written in shell script and should be fairly simple to follow. Refer to the manual pages on bash or whatever shell you run for questions about shell script programming.

Q21. I have a question about Qmail, SAMBA, Apache, etc. that is unrelated to the virtuald setup or how the package interfaces to virtuald.

A21. All the packages described here are fully documented. Some even have full web sites like www.packagename.org dedicated to them. Please consult them about questions dealing with the package that are unrelated to their virtual hosting functionality.

Q22. I have several domain aliases to domain1.com but mail keeps bouncing from the aliases. How come?

A22. Virtmaildelivery relies on the environment variables passed to it to determine which /virtual/domain1.com directory to deliver to. It does not perform any DNS lookups to determine the address of the mail. However, if the address is submail.mail.domain1.com, virtmaildelivery will first try that address and then mail.domain1.com and then domain1.com and then com in that order until either a match happens or there is no domain name left.

However, if you have domain aliases that are not subdomains of one another you have to create symlinks like so:

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

That way virtmaildelivery will be fooled into thinking that both directories exist even though one is a symlink and mail will be able to be delivered to user@domain1.com or user@domain1alias.com. Note that virtexec will list both of the domains in the dialog box when your run it. You can choose either one since they will be the same virtual filesystem.