Virtfs User Guide

Release 3.50

Afra Ahmad

July 29, 2001

E-mail: afra@prongs.org

Abstract

This document covers why, how and where to set up Virtfs, a Perl script which aids in setting up virtual servers. Virtfs is ideal for medium sized ISP's hosting many virtual servers. This may seem a lot to read, but there isn't really.

Contents

1 Introduction

Based on the *Virtual Services HOWTO* (Brian Ackerman), Virtfs will make hosting virtual servers in an organized and relatively easy fashion. With this Perl script, you will be able to add and delete virtual servers, as well as set up the virtual servers for Apache, FTP, and E-mail. In addition, typical system administration tasks can be carried out, such as adding/removing users and groups, and simple Sendmail configuration.

Please have Ackerman's Virtual Services HOWTO handy as you read this document. Hopefully this document will explain everything to get you up and running with Virtfs.

1.1 Is this for you?

If you are thinking of setting up one or two small virtual servers, then Virtfs is probably not for you: whilst using Sendmail's virtual email support (and the such), you should be fine. However, if you will be setting up many virtual servers (ie. an ISP setting up a virtual server for each of their clients), you may want to use Virtfs. That is, if your clients want a dedicated filesystem. For example, they would like a system with their own *passwd* file (so they can add/remove users themselves), or when they log in to their virtual server, no other client has access to their filesystem - as all virtual servers are hidden from one another. Each server can have different users and groups from one another.

However, if you are thinking of running a few virtual servers which do not require special customization, and will handle a few users, you may want to look at the *Pocket ISP based on RedHat Linux* HOWTO (written by Anton Chuvakin).

1.2 How Virtfs works

There are quite a few methods used for virtual servers, however Virtfs takes advantage of the system's *chroot* features. What *chroot* does is set any given directory on your filesystem to a root directory. For example, chroot can set */home/user* to the */* directory (although if you try it now, it may not work complaining that a shell could not be found).

By copying the main files (and directory structure) from the main server to a specific location, we create a template of what each virtual server consists of. Then, for each virtual server you want to create, we link them to the template that we created. This saves drive space as well as organizing everything.

But how does the process of Virtual Servers actually occur? If we create a virtual server (say, vserver.com) in the directory, */virtual/vserver.com*, we *chroot* to */virtual/vserver.com* and everything is set up as a virtual server. Upon executing *chroot* on */virtual/vserver.com*, the /etc directory will in fact be */virtual/vserver.com/etc* and not */etc* (as we know it on the main server). So, if a client edited vserver's */etc/passwd* file, it will in no way be affecting the main server's *passwd* file. The client will be editing their */etc/passwd* file but in reality they are editing */virtual/vserver.com/etc/passwd*. To them, when they are editing */etc/passwd*, they are actually editing */virtual/vserver.com/etc/passwd* on the main server.

In addition to this, once you are within a virtual server, you cannot see the files belonging to the main server. Therefore a client would not be able to see another client's filesystem, or even the main server. To them, their virtual server is their "main" server, and they cannot see beyond that.

Services (ie. FTP, E-mail, Samba, etc..) for each virtual system will run individually from one another, and even from the main system.

1.3 virtuald

When you FTP into a server, how does it differentiate which virtual directory to *chroot* to? This is done by a small wrapper (written in C) that informs your services (usually set up in */etc/inetd.conf*) where to start. Tha name of this nifty program is virtuald. Basically virtuald reads in a configuration file and matches the IP address of a server called upon to the directory where the files reside. Virtuald then hands this information off to the service, and the service carries on as usual.

For example, you set up a virtual server, vserver1.com, and vserver2.com. In your configuration file you map out the IP address for each of these servers, to point to their proper location on your main server (/virtual/vserver1.com and /virtual/vserver2.com). You configure your *inetd* to run a specific service through virtuald, say FTP. When a user FTP's to vserver2.com, virtuald will lookup the directory for the certain server, and *chroot* to it, setting /virtual/vserver2.com to make it seem like the root (/) directory to the user. The FTP service will then know where to start. If no matching IP address is found, the service runs through as normal logging into the main server. For each service there should be a different configuration file.

Don't worry about the configuration files, as Virtfs automatically adds or deletes the entries within it as needed. Virtuald is contained within the Virtfs package.

1.4 Advantages

One of the main advantages on seperating your virtual servers this method, is control. You, the system administrator, has total control of each filesystem.

Another advantage is the flexibility of running different services for different servers. As in the above example, we could have easily shut off FTP service for vserver2.com but kept it on for vserver1.com. Also, a user at vserver2.com, say *bob@vserver2.com* is not the same as *bob@vserver.com*, or even *bob@mainserver.com* (Sendmail will not get confused as virtuald will sort everything out, prior to Sendmail kicking in). This method causes the main server not to be cluttered up with your client's email accounts. They will be all contained within their own directory.

1.5 Disadvantages

A big disadvantage is disk space. Creating many filesystems can get expensive, but there are some ways to combat this. If users on a virtual server do not need X-Windows, don't copy it over (and files supporting it). If users do not need Emacs, don't copy it over, and so on. For each virtual server, you can pick and choose what should reside on it. Just as long as the important base system (in addition to what services each virtual server needs) exists it should be fine.

You can hard link binaries from the main server to your virtual servers, that need to be used. When I create virtual servers, they usually take up about 10-15 MB.

2 Documentation Revisions

Changes from v.3.0 to v.3.5

- Rewrote in Latex

Changes from v.2.0 to v.3.0

- Upgraded docs for Virtfs 0.70.0
- Grammar/Spelling fixes.

Changes from v.2.0 to v.2.3

- Edited the Sendmail section as Virtfs 0.60.0 now takes care of most of the editing of sendmail files.
- Described the ld.so.conf file creation

Changes from v.1.5 to v2.0

- Described the Template Directory (updated for Virtfs v.0.55.0)

Changes from v.1.0 to 1.5

- Added more information on how to use tcp_virtuald
- Some additions and changes per users feedback.

3 Installation

The virtfs package comes tarred up with:

- virtuald - The wrapper which runs when a service (FTP, Email, etc..) is called upon.

- tcp_virtuald - The same wrapper as virtuald, but this one is if you want to use tcp wrappers (ie. tcpd hosts.allow and hosts.deny files).
- virt_template.pl - A Perl script to aid in creating the template filesystem.
- virt_template.conf - The configuration script for virt_template
- virtfs - The tool used to help set up virtual server.
- virtrun - A simple script to execute a specific command on a virtual server.
- sendmail.virtfs - A daemon to start up and end Sendmail services for all your virtual servers.
- virt.conf - The configuration file used for virtfs. You may have to edit this.

To compile the source simply type *make* from the directory where you untarred the package.

make install will install the files to */usr/local/bin* except for virt.conf, which is copied to the */etc/* directory. Only root can execute these programs and edit the configuration files.

If installing from a RPM, the files will be installed just as above, but instead the location will be */usr/bin* instead of */usr/local/bin* (configuration files will still be in */etc*).

Now you are ready to set up the *template filesystem*. Please read more below on how to set this up (it's easy).

4 tcp_virtuald

As of Virtfs 0.42.0, there is a file called tcp_virtuald.c which is very similar to virtuald. This file does exactly what virtuald does using the same methods. The only difference in tcp_virtuald is the ability to configure services using TCP wrappers, making it possible to use tcpd hosts.allow and hosts.deny files.

There is one difference between both versions of virtuald and that is the method used to configure service, but this is described later in this guide.

5 Pre-Setup: Template Filesystem

5.1 Template Filesystem

Prior to setting up virtual servers, you must create a **template filesystem**, which will be used to decide what files each virtual server will contain. In other words, you must create a small Linux Filesystem. The files within the *template filesystem* will be hard linked to each virtual server you create. The template directory will be the directory and file structure for your virtual servers - whatever is in the template directory will be what each virtual server you create will contain. This may seem a hassle but there are many reasons this must be done. In order for the files to be linked over, they must all be within the same partition (most servers have multiple partitions). Virtfs uses links to create virtual servers to save hard drive space. For example, if you are running about six virtual servers and copied all the files over for each server, this would amount to about 700Mb in itself. By linking, however, this problem is eliminated. Each server will take up about 5-10Mb. However not *all* of the filesystem is linked over. For example, */etc/hosts*, */etc/HOSTNAME*, etc - system specific files are not linked and therefore are not included in the template filesystem.

Another advantage to the *template filesystem* is that you can configure what files should be copied over to the virtual servers. Would you want the ”virtual users” to have access to developmental tools? SSH? You can specify which files should be included with each virtual server by copying them to the template filesystem.

To recap, the process in creating virtual servers is simple. Firstly, a template directory is created and whatever files you specify will be copied over to the template filesystem. Then, when you want to create virtual servers, the files from the template filesystem are hard linked to the virtual server - Virtfs handles this for you upon creation of a virtual server.

You will have to make sure that the template directory is within the same partition as the virtual servers, ie within the */virtual* directory (if you stick to the suggested configuration). If you don't please make changes to the *Leading Virtual Directory* variable within */etc/virt.conf*.

The basic structure of this is:

```
                        --------------------
                       |  MAIN SERVER: /  |
                        ____________________
|
                                |
                                V direct copy
                                |
               -----------------------------------------
               | Template Filesystem /virtual/template |
               -----------------------------------------
                  |                          |     |
hard links        ^                          ^     ---------<-------------
                  |                          |                            |
                  |                 |              |
              -----------------         ------------------            ------------------
              |   a.com       |         |     b.com      |            |    c.com       |
              | /virtual/a.com |        | /virtual/b.com |            | /virtual/c.com |
              -----------------         ------------------            ------------------
```

5.2 Setting up a template filesystem

This section describes how to configure the template filesystem from the */etc/virt_template.conf* file.

5.3 ¡Template¿

To specify what files should be copied over to the template filesystem, edit the */etc/virt_template.conf* file. Within the *¡Template¿* section, you will notice, for example:

```
bin_files = ls echo bash ...
```

In the above example the *bin_files* variable holds the names of the files which are to be copied from the main server's */bin* directory to the template filesystem, ie */virtual/template/bin*. Seperate each filename with a space. Therefore, from the value of *bin_files* above, each virtual server will contain *ls, echo, bash* and whatever else was specified. Here is an outline of which directories are dealt with:

- *bin_files* - Files within the **/bin** directory to be copied over.
- *sbin_files* - Files within the **/sbin** directory to be copied over.
- *usr_bin_files* - Files within the **/usr/bin** directory to be copied over.
- *usr_sbin_files* - Files within the **/usr/sbin** directory to be copied over.
- *usr_local_bin_files* - Files within the **/usr/local/bin** directory to be copied over.
- *etc_files* - Files within the **/etc** directory to be copied over.

If you would like to copy a whole directory to the template directory, for example */usr/sbin/some_dir*, you may list *some_dir/** within the *usr_sbin_files* list. Another alternative is to edit the *¡Custom Template¿* section (below).

You must be asking yourself about the */etc/* directory. Since this directory is very specific to each server, and not general such as the executables mentioned above, we must choose the files carefully. When creating a template, the *sendmail* configuration files are copied over (*sendmail.cf*) and some are created on the fly (*sendmail.cw*).

By default the template directory will be */virtual/template* but you can always change this value within */etc/virt_template.conf*.

5.4 ¡Custom Template¿

Within the same configuration file, *virt_template.conf* there is the *¡Custom Template¿* section. In this section, you can specify what *shell commands* should be executed **after** virt_template.pl does all the copying of the files from the ¡Template¿ section. An example may be:

```
cp -a /lib !!template
```

The *!!template* will be replaced by virt_template.pl and it indicates the template directory. Therefore, the above is stating: *cp -a /lib /virtual/template* (if we choose to stick to the directory defaults). This section is for you to customise the template directory even further than the ¡Template¿ section.

5.5 ¡Passwd File¿, ¡Shadow File¿ and ¡Group File¿

In these sections, you must specify the password, shadow and group files that will be copied over. I suggest copying these file to a **secure** location on your main server first, and then editing them to delete the accounts you do not want on your virtual server. For example, you probably like to remove your normal users accounts. But please keep the **root** password, shadow and group entried unedited (you may change the password at a later time). So, for example, you may copy your main server's */etc/passwd, /etc/shadow and /etc/group* files to a secure location, ie */root*. Then edit the files to your preference. If your main server has the *joe* account, but your virtual servers should not contain this, remove the entries for the account from the passwd, shadow and group files. After the template directory is created, remove the temporary files!

If you copy over the main server's password, shadow and group files, all those accounts (except for home directories) will be copied over, so I have left this option up to you for which accounts should stay and which should be removed.

The Perl script **virt_template.pl** creates the template directory. The script is interactive and will create the template filesystem for you. Once it has been created you can make changes to the directory. Remember it is this directory structure that each virtual server will link to, and only server specific files will be copied over.

5.6 Executing virt_template.pl

If you are happy with the configuration of */etc/virt_template.conf*, you are now ready to create the filesystem. The Perl script, *virt_template.pl* will help you with this. The script will be informative and you will know what is going on.

6 Creating Virtual Servers

6.1 Please check!

Before you continue, please check that your *template filesystem* is all set and as root, try the following:

```
chroot /virtual/template
```

Try some commands within the chroot'd environment. It should appear like a mini Linux system. Make sure you are happy with the commands. If not, copy them over from the main server or edit the *virt_template.conf* file and try re-creating the template filesystem.

6.2 IP Aliasing

To set up virtual servers to reside on one main server, you will need to set up IP Aliasing. This will have to be done with any method of hosting virtual servers that you choose. There is a HOWTO on this subject: *Mini How-to on Setting Up IP Aliasing On A Linux Machine* by Harish Pillay. That HOWTO should set you in the proper direction. Once you set up IP Aliasing for a virtual server, test it, ping it. Obviously it will point to your main server for now.

6.3 DNS

DNS for the servers should be set up normally. There is an HOWTO for this: *DNS HOWTO* by Nicolai Langfeldt.

Although, as of Virtfs 0.70.0, the */etc/resolv.conf* file for your virtual server will be created. Specifically you will want to change the values of the *Primary DNS* and *Secondary DNS*. When the virtual server has been created, */etc/resolv.conf* will automatically be filled in.

6.4 Virtuald

Explained above, virtuald is written in C (Brian Ackerman, from his HOWTO) which is meant to direct a specific service to the directory where a virtual service exists. It should be placed in your */etc/inetd.conf* directory. More on this later..

6.5 tcp_virtuald

This file does exactly what virtuald achieves, but if you want your services to run through the tcpd service, you will want to use this version of virtuald. Details on implementing tcp_virtuald is described below.

6.6 Virtfs

6.7 Introduction

This is the Perl script which will help you setup and maintain your virtual servers. Virtfs uses dialog to present you with your choices in a simple but tidy look running from the command line. The main page for Virtfs has screenshots.

6.8 /etc/virt.conf Configuration

This is an important configuration file that you *must* look through before firing up Virtfs. There are two sections to this file: the **¡Configuration¿** and **¡Shell Commands¿** sections. The ¡Shell Commands¿ section is very similar to the ¡Custom Template¿ section (as described above). This section is compromised of shell commands that virtfs will execute upon the creation of a new virtual server. You may not even need it, especially if you customised your template directory well.

The ¡Configuration¿ section is well documented and easy to follow through. For example, if you would like to change the UI program from *dialog* to *Xdialog*, you may do so under the *Dialog* value.

An important variable to edit (or not to) is the **Leading Virtual Directory**. Your template directory should be contained in this. Typically, the value of this is */virtual* (your template directory would then be */virtual/template*). This way, all your virtual servers will be stored in */virtual* (for example, vserver.com will be in */virtual/vserver.com*).

Upon starting up Virtfs you have the choice to either create a new server, or maintain an existing virtual server that you have set up before. When you choose a virtual server to configure, you can delete the server, switch on/off FTP and Mail services, add/remove users and groups, set up Apache, etc. You can even log into the virtual server as a specified user. Virtfs should help in the daily administration tasks, basically.

After setting up the networking and DNS configuration (for a virtual server), you will want to run Virtfs to set up your virtual servers. After setting them up, you may want to run Virtfs again and again to take care of the system administration for a particular virtual server.

6.9 The FTP files

In order to make ftp service work on your virtual servers, you may have to edit */etc/virt.conf* to enable some files that are needed. You will have to edit the line:

```
ftp files = /etc/ftpaccess /etc/ftpusers /etc/ftpconversions /etc/ftphosts
```

Everytime a server is enabled, Virtfs copies the files from the main server (the values from the *ftp files* value) to the virtual server. These files are **not hard linked** as each server may need their own copy of these files. These files are server dependant.

6.10 The ldconfig files

As of v.0.60.0, Virtfs will create a */etc/ld.so.conf* file for your virtual server (this is for setting up your system's libraries). You can edit the directories which are first placed into the ld.so.conf file by editing /etc/virt.conf and changing the value of *ldconfig libs*. Each directory should be seperated by a space.

6.11 Virtrun

Virtrun is another Perl script (real simple) which also uses dialog. If you are in a rush you can execute a command for a specific virtual server. For example, the listing of a particular directory, or to *su* over (even thoug Virtfs can achieve this for you). Virtrun basically does the chroot for you.

7 The Virtual Server Administator

7.1 The Administrator account: admin

Upon creation of a new virtual server (and as of Virtfs 0.70.0), an administrator user is created. The group and user *admin* is added to the virtual system (not the main one). With admin rights, system administrators for that virtual server may tweak configuration files for their virtual server, and there is no worry that it will affect the main server.

7.2 Why?

Mainly for security. Giving root access to a client on a virtual service can lead to some harmful events. If there is an unknown break-in to the main server that can be done with root, this could be bad. To play things safe, it is best to give admin rights to the client.

Only you should know the root password.

7.3 The Administrator's access rights

Virtfs, by default, assigns full access permissions to the following directories to the new virtual server:

- /root - This will be the home directory for the admin account.
- /usr/local/* - This will be where admin may compile and install new applications.
- /home - admin will have full rights to all use directories.
- /etc - Certain files in this directory so that they may tweak their own servers to how they like.
- /var/ - Some directories in /var. This can be changed from */etc/virt.conf* - the *Admin Right* section.

7.4 Changing the rights of the Administrator

Within the Virtfs configuration file, you will see a section called *¡Admin Rights¿*. Within this section, you can specify what files or directories should automatically obtain full admin access rights. Be very careful in doing this, as the whole point of creating the admin account is for security.

7.5 Disadvantages

There are some disadvantages, such as the admin will not be able to add/remove a user or group. These things should go by the root user, anyhow.

8 Post Server Setup

8.1 FTP Files

Virtfs copies the ftp files (see above under *The FTP files*) **only** if you activate FTP service for the service.

After the files are copied over you will have to edit them if you want your virtual server's ftp server to act differently than the main servers.

8.2 ldconfig

Some applications require libraries to be pre-loaded. I suggest you chroot to the server (through Virtfs) as root and run *ldconfig*. You may have to edit */etc/ld.so.conf* within the virtual server to specify which directories are to be included. Again, the contents of */etc/ld.so.conf* can be altered everytime a server is created within */etc/virt.conf*.

8.3 Log directories

As each Linux flavour required their own log directory, I suggest after creating a virtual server, you compare the log directories. For example, Virtfs will not create */var/log/httpd* as Redhat Apache RPMs do. You should create the directory.

Virtfs does create the standard Sendmail directories for you (this is in the **¡Shell Commands¿** section of */etc/virt.conf*).

9 Common Virtual Services

As mentioned above, setting up services (ie. POP/SMTP, FTP) for certain virtual servers is very flexible. You can specify whether vserver2.com should have FTP service, or if vserver1.com should have email. I am going to cover the basics of how to set up virtuald through the common services.

Basically for each service in your *inetd* configuration file, you want to add: */path/to/virtuald virtuald /virtual/conf.file*

But, if you are using tcp_virtuald, you will want to use: */path/to/tcp_virtuald virtuald /virtual/conf.file tcpd*

For *xinetd*, please read the section below. It will still be helpful not to skip this section...

9.1 Configuration Files

The configuration files needed for each service should contain the IP address and the directory where the virtual service resides. For example, *xxx.xxx.xxx.xxx:/virtual/vserver2.com*. (where xxx.xxx.xxx.xxx is the IP number)

Each of these on a new line, but do not alter these files, as Virtfs will do this for you.

The configuration file for say, FTP, is by default (in the */etc/virt.conf* file) *conf.ftp* and this must reside in the leading virtual sub directory, ie. */virtual/conf.ftp*. For mail services, it is */virtual/conf.pop*.

There should only be ONE configuration file for each service, and not multiple. Virtfs will add entries for each server into the configuration file when you select to add it, so you need not edit it directly (and you probably shouldn't). When you delete a virtual server, Virtfs removes the entry for the chosen server.

9.2 Specifying Paths to Services

You must specify the full path to each service in /etc/inetd.conf. For example:

```
ftp  stream  tcp  nowait  root /path/to/virtuald virtuald /virtual/conf.ftp wu.ftpd -l -a
```

On my system, wu.ftpd is in the */usr/sbin* directory on the main server as well as the virtual server (*/virtual/vserver1.com/usr/sbin*). Make sure you follow the general rules to avoid problems:

- Your service file is on the main server as well as the virtual directories, in the right locations. If the service is on the main server's */usr/sbin* make sure it's on the same location in the virtual directory, ie. */virtual/vserver1.com/usr/sbin*.
- Normally, when editing the *inetd.conf*, it is fine just to leave *wu.ftpd* (the service), but when using it with virtuald you should note the full path name to the service.

9.3 FTP

In your */etc/inetd.conf* file, you will have to edit this to place the virtuald wrapper to accept connections for the FTP service. I changed mine to the following, where */virtual/conf.ftp* is the configuration file for servers allowed to FTP in:

```
ftp stream tcp nowait root /path/to/virtuald virtuald /virtual/conf.ftp /path/to/wu.ftpd -l -a
```

If you are using tcp_virtuald, you will have to specify the above line as such:

```
ftp stream tcp nowait root /path/to/tcp_virtuald virtuald /virtual/conf.ftp tcpd /path/to/wu.ft
```

9.4 Mail

There are two common mail packages: Sendmail and Qmail. The setup for virtual servers is different, depending on what package you use. Your IMAP/POP services should be easily configured though. Virtfs takes care of the mail configuration file (by default: *conf.pop*). You will just have to alter the lines in *inetd.conf* to:

```
pop-3 stream tcp nowait root /path/to/virtuald virtuald /virtual/conf.pop /path/to/ipop3d
```

You can configure mail services for tcp_virtuald as it has been done above for FTP.

9.5 Sendmail

Unlike using Sendmail's virtual email support, you can have multiple accounts with same names, on main server. As explained above, *bob@vserver1.com* and *bob@mainserver.com* are actually different accounts. This is a great advantage. As far as setting up a virtual server for email though, Virtfs copies over the main server's */etc/sendmail.cf* file (which you must set in virt.conf under *Sendmail CF*) to the virtual server's */etc* directory and edits the line:

#Dj$w.Foo.COM to:

Djvserver1.com (replacing vserver1.com with your server's real entry).

We set up Sendmail within *inetd* as follows:

```
smtp stream tcp nowait root /path/to/virtuald virtuald /virtual/conf.pop /path/to/sendmail -bs
```

You can configure mail services for tcp_virtuald as it has been done above for FTP.

Please note that Virtfs will automatically add/delete entries from the conf.pop configuration file. This file permits the servers listed to accept connections from the ports.

9.6 sendmail.cw

Upon creation of a virtual server, Virtfs will automatically create the sendmail.cf (as of v.0.60.0) in */virtual/vserver2.com/etc/sendmail.cw*.

Virtfs will add these lines into the sendmail.cw file (replacing vserver2.com with your virtual server's name):

```
mail.vserver2.com
vserver2.com
vserver2
localhost
```

9.7 sendmail.virtfs

You will need to replace your current sendmail file with the *sendmail.virtfs* shell script included with the Virtfs package. This script shell will start, stop and restart Sendmail services for all your virtual servers and main servers.

9.8 Qmail

Unfortunately, I am unable to test Virtfs with Qmail, however, there is a little package which will help you. You may read on it and download it from the *Virtfs Qmail page* . This additional package does not come with the standard Virtfs package. Reading the Virtual Services HOWTO will also help you in this (the sources from the Qmail page were taken from the HOWTO).

9.9 World Wide Web

Apache has an excellant method of configuring VirtualHosts, and I suggest you use that method. Setting up Apache is regardless of virtuald, although Virtfs will edit the *httpd.conf* adding the appropriate configuration. Refer to Apache's documentation on how to set this up for a virtual service. You may want to create a web directory, ie */virtual/vserver1.com/web* so that people with access to vserver1.com can access these files - do not place the web directory on the main server, as the users of vserver1.com will not be able to access them.

9.10 Other

You can acheive similar results with configuring your *inetd* file, as I have done above. In the chance you are using a service (like some FTP clients) that do not run through *inetd*, you will unfortunately have to go under their direction for virtual services.

If it does run through inetd, simply do the following:

1. Add the appropriate lines to activate virtuald for your services (as explained above) in *inetd.conf*
2. Create a */virtual/conf.service* file.
3. Add the servers to the configuration file.

10 xinetd

As of Redhat 7.x, many people have asked how to implement the virtual services into their system. This is quite straight forward, as explained in the email by Rodrigo B. Pereira:

```
As for xinetd, it’s all pretty simple, just a matter of adaptation to the
new organization of things. In this case (RH 7.0), every service is
configured on a separate file, under /etc/xinetd.d, but you can also use
just one file, just a matter of editing /etc/xinetd.conf. HoweverI think
it’s very nice this way.

Let’s take this telnet example:

# default: on
# description: The telnet server serves telnet sessions; it uses \
#       unencrypted username/password pairs for authentication.
service telnet
{
        flags           = REUSE
        socket_type     = stream
        wait            = no
        user            = root
        server          = /usr/local/bin/virtuald
        server_args     = /home/virtual/conf.telnet /usr/sbin/in.telnetd
        log_on_failure  += USERID
}

The key point here is server_args. The ’server’ line should always point
to the absolute path of virtuald. The ’server_args’ line consists of the
cvnf.service file, followed by the absolute path of the real service
daemon. This sure looks much nicer than on inetd.conf.

Every "real" server arguments, are passed along with the daemon abs. path.
For example, with wu-ftpd, the server_args line would look like this:

server_args             = /home/virtual/conf.ftp /usr/sbin/in.ftpd -l -a

Resuming:        server = /path/to/virtuald
                 server_args = /path/to/conf/service/conf.service
path.to.real.daemon/read.daemon any.daemon.args

After configuring, just type this "service xinetd restart" to activate
changes. Or "/etc/rc.d/init.d/xinetd restart", if for some weird reason
service isn’t available.
```

11 Logging

11.1 Virtfs Logging

Virtfs will log all actions carried out if you have configured it so within the *virt.conf* file. Set *Log = on* and then specify the file for you to log all your actions in under *Log File*.

11.2 Virtuald Logging

Virtuald will, by default, log actions by services requested through syslogd. I say by default, as you can shut off the logging by editing the source of virtuald.c and shutting off VERBOSELOG. By default it is on.

You will notice in your syslog when Virtuald kicks in. Something like this should be logged:

```
virtuald[4207]: Virtuald Starting v.xxxx
....
```

You will be able to tell where the service is being called to. Also, you may check on these logs for debugging purposes.

11.3 tcp_virtuald Logging

The logging for this modified version of virtuald is the same as the logging for virtuald (described above), except you will know which version of virtuald will be running (ie. tcpd enabled ot not):

```
virtuald[4220]: Virtuald Starting (tcpd enabled) v.xxx
```

12 References

1. *Virtual Services HOWTO*
2. *Mini HOWTO: IP Aliasing*
3. *DNS HOWTO*
4. *Pocket ISP based on RedHat Linux*