From Technologies to Solutions

Network Administration with FreeBSD 7

Building, securing, and maintaining networks with the FreeBSD operating system

Babak Farrokhi

PACKT PUBLISHING

Network Administration with FreeBSD 7

Building, securing, and maintaining networks with the FreeBSD operating system

Babak Farrokhi

BIRMINGHAM - MUMBAI

Network Administration with FreeBSD 7

Copyright © 2008 Packt Publishing

All rights reserved. No part of this book may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, without the prior written permission of the publisher, except in the case of brief quotations embedded in critical articles or reviews.

Every effort has been made in the preparation of this book to ensure the accuracy of the information presented. However, the information contained in this book is sold without warranty, either express or implied. Neither the author, Packt Publishing, nor its dealers or distributors will be held liable for any damages caused or alleged to be caused directly or indirectly by this book.

Packt Publishing has endeavored to provide trademark information about all the companies and products mentioned in this book by the appropriate use of capitals. However, Packt Publishing cannot guarantee the accuracy of this information.

First published: April 2008

Production Reference: 1070408

Published by Packt Publishing Ltd.
32 Lincoln Road
Olton
Birmingham, B27 6PA, UK.

ISBN 978-1-847192-64-6

www.packtpub.com

Cover Image by Nilesh Mohite (nilpreet2000@yahoo.co.in)

Credits

Author

Babak Farrokhi

Reviewer

Roman Bogorodskiy

Acquisition Editor

Rashmi Phadnis

Technical Editor

Della Pradeep

Editorial Team Leader

Mithil Kulkarni

Project Manager

Abhijeet Deobhakta

Project Coordinator

Abhijeet Deobhakta

Indexer

Hemangini Bari

Proofreader

Nina Hasso

Production Coordinator

Aparna Bhagat

Cover Work

Aparna Bhagat

About the Author

Babak Farrokhi is an experienced UNIX system administrator and Network Engineer who worked for 12 years in the IT industry in carrier-level network service providers. He discovered FreeBSD around 1997 and since then he has been using it on a daily basis. He is also an experienced Solaris administrator and has extensive experience in TCP/IP networks.

In his spare time, he contributes to the open source community and develops his skills to keep himself in the cutting edge.

You may contact Babak at `babak@farrokhi.net` and his personal website at `http://farrokhi.net/`

I would like to thank my wife, Hana, for being the source of inspiration in my life. Without her support and patience I could not finish this project.

Next I'd like to thank the Technical Reviewer of the book, Roman Bogorodskiy (`novel@FreeBSD.org`) for his thorough review, great suggestions, and excellent notes that helped me to come up with the chapters even better.

I also want to thank PACKT and everyone I worked with, Priyanka Baruah,Abhijeet Deobhakta, Rashmi Phadnis, Patricia Weir, Della Pradeep and others for their patience and cooperation. Without their help I could not turn my scattered notes into a professional looking book.

About the Reviewer

Roman Bogorodskiy lives in Russia, Saratov. He is a student of the Mechanics and Mathematics faculty at the Saratov State University. At the time of writing, he was working on a diploma project. He is working as a Software Engineer in the one of the biggest ISPs of his hometown. He takes part in various open source projects and got his FreeBSD commit bit back in 2005.

Table of Contents

Preface

This book is supposed to help Network Administrators to understand how FreeBSD can help them simplify the task of network administration and troubleshooting as well as running various services on top of FreeBSD 7 Operation System. FreeBSD is a proven Operating System for networked environments and FreeBSD 7 offers superior performance to run network services, as well as great flexibility to integrate into any network running IPv4, IPv6 or any other popular network protocol.

This book is divided into three segments—system configuration, network configuration, and network services.

The first segment of the book covers system configuration topics and talks about different aspects of system configuration and management, including disks management, patching and keeping the system up to date, managing software packages, system management and monitoring, jails and virtualization, and general improvements to system performance.

Second segment of the book actually enters the networking world by introducing basic network configuration in FreeBSD, network interface configuration for different layer 3 protocols, Tunnelling protocols, PPP over serial and Ethernet and IPv6. This segment also looks into bridging and routing in FreeBSD using various third party softwares. At the end, there is an introduction to various firewall packages in FreeBSD and details on how to configure them.

Third segment of the book deals with different daemons and network services that can be run on top of FreeBSD, including Local network services such as DHCP, TFTP, NFS, SMB as well as Internet services such as DNS, Web, Mail, FTP and NTP.

What This Book Covers

Chapter 1 looks into FreeBSD file system and disk I/O from a performance point of view. Several methods to optimize the I/O performance on a FreeBSD host are discussed in this chapter.

Chapter 2 discusses several methods and tools to keep a FreeBSD system up-to-date, including CVSUP to update source and ports tree and also customizing and updating system kernel and rebuilding the whole system from source.

Chapter 3 introduces FreeBSD ports collection, packages, and different methods to install, remove, or upgrade software packages on FreeBSD.

Chapter 4 covers basic information about daemons, processes, and how to manage them. You will also get familiar with various system tools to monitor and control process behavior and manage system resources efficiently.

Chapter 5 discusses virtualization in FreeBSD and introduces Jails from ground up. This chapter covers creating and maintaining Jails and scenarios in which you can benefit from these built-in virtualization facilities in FreeBSD.

Chapter 6 discusses performance tuning from different perspectives, including Disk I/O and Network, and how to get the most out of the modern hardware and multi-processor systems. It discusses various tweaks that can make your FreeBSD system perform much faster and more smoothly.

Chapter 7 deals with network configuration in FreeBSD in general, focusing mostly on network interface configuration for different network protocols such as IPv4, IPv6, IPX and AppleTalk. It also deals with basic network configuration and related configuration files and finally introduces some network management and testing tools.

Chapter 8 discusses tunneling in general and introduces various tunneling protocols, and mostly concentrates on GRE and IPSec tunneling.

Chapter 9 covers PPP configuration in FreeBSD including PPP over Ethernet protocol as both client and server.

Chapter 10 has a closer look at routing and bridging in FreeBSD using built-in bridging features and also different routing protocols including OSPF and BGP using third-party software.

Chapter 11 concentrates on IPv6 implementation in FreeBSD and gives more detail on interface configuration, routing IPv6 using RIP6, Multicast routing, and Tunneling protocols.

Chapter 12 introduces IPFW and PF tools for packet filtering and network address translation as well as traffic management on FreeBSD.

Chapter 13 has a quick look at various important protocols such as SSH, NTP, DNS, FTP, Mail, Web, and Proxying. It also introduces different pieces of software that you can use to set up these services on a FreeBSD host.

Chapter 14 looks into some network protocols that are mostly used inside an autonomous system or inside a datacenter or a local network, such as DHCP, TFTP, NFS, SMB, SNMP, NIS and Printing and introduces various pieces of software and setting them up on a FreeBSD host.

What You Need for This Book

Basically you need a host running FreeBSD 7 connected to your network. Your host can be any hardware platform that FreeBSD supports, including i386, sparc64, amd64, ia64, powerpc or pc98. You should download relevant FreeBSD installation CD images from FreeBSD project's FTP server at ftp://ftp.freebsd.org/pub/

There you will find ISO images for various platforms under different subdirectories (e.g. "ISO-IMAGES-i386" directory contains i386 platform ISO images). For a basic installation, the ISO image for first CD will suffice.

Once you have installed FreeBSD, you should also configure your network parameters to get connected to your existing network. This can be done during installation or later by modifying the `/etc/rc.conf` configuration file (covered in chapter 7).

Who is This Book for

For Network Administrators who would like to work with FreeBSD and are looking for skills beyond Installation and configuration of FreeBSD.

Conventions

In this book, you will find a number of styles of text that distinguish between different kinds of information. Here are some examples of these styles, and an explanation of their meaning.

There are three styles for code. Code words in text are shown as follows: "And finally, check the system's swap status using the following `swapinfo(8)` command."

A block of code will be set as follows:

```
flush
add check-state
add allow tcp from me to any setup keep-state
add allow tcp from 192.168.1.0/24 to me keep-state
add allow ip from 10.1.1.0/24 to me
add allow ip from any to any
```

When we wish to draw your attention to a particular part of a code block, the relevant lines or items will be made bold:

```
/dev/ad0s1a on / (ufs, local, noatime, soft-updates)
devfs on /dev (devfs, local)
procfs on /proc (procfs, local)
/dev/md1 on /tmp (ufs, local)
/dev/md2 on /mnt (ufs, local, read-only)
```

Any command-line input and output is written as follows:

```
# dd if=/dev/zero of=/swap0 bs=1024k count=256
```

New terms and **important words** are introduced in a bold-type font. Words that you see on the screen, in menus or dialog boxes for example, appear in our text like this: "Note that either the **userquota** or the **groupquota** can be specified for each partition in the **Options** column.".

Important notes appear in a box like this.

Tips and tricks appear like this.

Reader Feedback

Feedback from our readers is always welcome. Let us know what you think about this book, what you liked or may have disliked. Reader feedback is important for us to develop titles that you really get the most out of.

To send us general feedback, simply drop an email to `feedback@packtpub.com`, making sure to mention the book title in the subject of your message.

If there is a book that you need and would like to see us publish, please send us a note in the **SUGGEST A TITLE** form on `www.packtpub.com` or email `suggest@packtpub.com`.

If there is a topic that you have expertise in and you are interested in either writing or contributing to a book, see our author guide on `www.packtpub.com/authors`.

Customer Support

Now that you are the proud owner of a Packt book, we have a number of things to help you get the most from your purchase.

Errata

Although we have taken every care to ensure the accuracy of our contents, mistakes do happen. If you find a mistake in one of our books—maybe a mistake in text or code—we would be grateful if you would report this to us. By doing this you can save other readers from frustration, and help to improve subsequent versions of this book. If you find any errata, report them by visiting `http://www.packtpub.com/support`, selecting your book, clicking on the **Submit Errata** link, and entering the details of your errata. Once your errata are verified, your submission will be accepted and the errata are added to the list of existing errata. The existing errata can be viewed by selecting your title from `http://www.packtpub.com/support`.

Questions

You can contact us at `questions@packtpub.com` if you are having a problem with any aspect of the book, and we will do our best to address it.

1
System Configuration—Disks

Disk I/O is one of the most important bottlenecks in the server's performance. Default disk configuration in every operating system is optimally designed to fit the general usage. However, you may need to reconfigure disks for your specific usage, to get the best performance. This includes choosing multiple disks for different partitions, choosing the right partition size for specific usage, and fine-tuning the swap size. This chapter discusses how to use the right partition size and tuning file system to gain better performance on your FreeBSD servers.

In this chapter, we will look into the following:

- Partition layout and sizes
- Swap, softupdates, and snapshots
- Quotas
- File system back up
- RAID-GEOM framework.

Partition Layout and Sizes

When it comes to creating disk layout during installation, most system administrators choose the default (system recommended) settings, or create a single root partition that contains file system hierarchy.

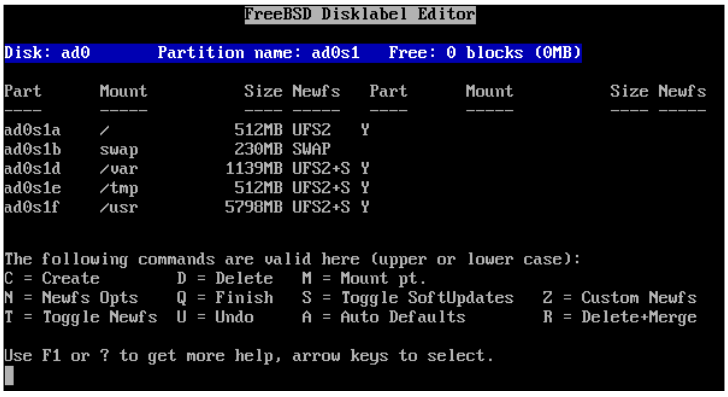
However, while the recommended settings work for most simple configurations and desktop use, it may not fit your special needs. For example, if you are deploying a mail exchanger or a print server you may need to have a `/var` partition bigger than the recommended size.

By default, FreeBSD installer recommends you to create five separate partitions as shown in the following table:

Partition	Size		Description
	Minimum	**Maximum**	
Swap	RAM size / 8	2 * RAM size	Size of swap partition is recommended to be 2 or 3 times the size of the physical RAM. If you have multiple disks, you may want to create swap on a separate disk like other partitions.
/	256 MB	512 MB	Root file system contains your FreeBSD installation. All other partitions (except swap) will be mounted under root partition.
/tmp	128 MB	512 MB	Temporary files will be placed under this partition. This partition can be made either on the disk or in the RAM for faster access. Files under this partition are not guaranteed to be retained after reboots.
/var	128 MB	1 GB + RAM size	This partition contains files that are constantly "varying", including log files and mailboxes. Print spool files and other administrative files. Creating this partition on a separate disk is recommended for busy servers.
/usr	1536 MB	Rest of disk	All other files, including home directories and user installed applications, will be installed under this partition.

These values could change in further releases. It is recommended that you refer to the release notes of the version you are using, for more accurate information.

FreeBSD disklabel editor with automatically created partitions is shown in the following screenshots:



Depending on your system I/O load, partitions can be placed on different physical disks. The benefit of this placement is better I/O performance, especially on `/var` and `/tmp` partitions. You can also create `/tmp` in your system RAM by tweaking the `tmpmfs` variable in `/etc/rc.conf` file. An example of such a configuration would look like this:

```
tmpmfs="YES"
tmpsize="128m"
```

This will mount a 128 MB partition onto RAM using `md(4)` driver so that access to `/tmp` would be dramatically faster, especially for programs which constantly read/write temporary data into `/tmp` directory.

Swap

Swap space is a very important part of the virtual memory system. Despite the fact that most servers are equipped with enough physical memory, having enough swap space is still very important for servers with high and unexpected loads. It is recommended that you distribute swap partitions across multiple physical disks or create the swap partition on a separate disk, to gain better performance. FreeBSD automatically uses multiple swap partitions (if available) in a round-robin fashion.

When installing a new FreeBSD system, you can use disklabel editor to create appropriate swap partitions. Creating a swap partition, which is double the size of the installed physical memory, is a good rule of thumb.

Using `swapinfo(8)` and `pstat(8)` commands, you can review your current swap configuration and status. The `swapinfo(8)` command displays the system's current swap statistics as follows:

```
# swapinfo -h
   Device          1K-blocks     Used    Avail Capacity
   /dev/da0s1b     4194304       40K     4.0G     0%
```

The `pstat(8)` command has more capabilities as compared with the `swapinfo(8)` command and shows the size of different system tables, under different load conditions. This is shown in the following command line:

```
# pstat -T
   176/12328 files
   0M/4096M swap space
```

Adding More Swap Space

There are times when your system runs out of swap space, and you need to add more swap space for the system to run smoothly. You will have three options as shown in the following list:

- Adding a new hard disk.
- Creating a swap file on an existing hard disk and partition.
- Swapping over network (NFS).

Adding swap on a new physical hard disk will give better I/O performance, but it requires you to take the server offline for adding new hardware. Once you have installed a new hard disk, you should launch FreeBSD's disklabel editor and create appropriate partitions on the newly installed hard disk.

To invoke the sysinstall's disklabel editor from the command line use `sysinstall diskLabelEditor` command.

If, for any reason, you cannot add new hardware to your server, you can still use the existing file system to create a swap file with the desired size and add it as swap space. First of all, you should check to see where you have enough space to create the swap file as shown as follows:

```
# df -h
   Filesystem      Size    Used   Avail Capacity  Mounted on
   /dev/ad0s1a     27G     9.0G    16G     37%       /
   devfs           1.0K    1.0K    0B     100%      /dev
   procfs          4.0K    4.0K    0B     100%      /proc
   /dev/md0        496M    1.6M   454M     0%       /tmp
```

Then create a swap file where you have enough space using the following command line:

```
# dd if=/dev/zero of=/swap0 bs=1024k count=256
   256+0 records in
   256+0 records out
   268435456 bytes transferred in 8.192257 secs (32766972 bytes/sec)
```

In the above example, I created a 256MB empty file (256 * 1024k blocks) named `swap0` in the file system's root directory. Also remember to set the correct permission on the file. Only the root user should have read/write permission on file. This is done using the following command lines:

```
# chown root:wheel /swap0
# chmod 0600 /swap0
# ls -l /swap0
-rw------- 1 root  wheel  268435456 Apr  6 03:15 /swap0
```

Then add the following `swapfile` variable in the `/etc/rc.conf` file to enable swap file on boot time:

```
swapfile="/swap0"
```

To make the new swap file active immediately, you should manually configure `md(4)` device. First of all, let's see if there is any `md(4)` device configured, using `mdconfig(8)` command as shown as follows:

```
# mdconfig -l
md0
```

Then configure `md(4)` device as shown here:

```
# mdconfig -a -t vnode -f /swap0
md1
```

You can also verify the new `md(4)` node as follows:

```
# mdconfig -l -u 1
md1     vnode     256M  /swap0
```

Please note that `-u` flag in the `mdconfig(8)` command takes the number of `md` node (in this case, 1). In order to enable the swap file, you should use `swapon(8)` command and specify the appropriate `md(4)` device as shown here:

```
# swapon /dev/md1
```

And finally, check the system's swap status using the following `swapinfo(8)` command:

```
# swapinfo -h
Device          1K-blocks     Used    Avail     Capacity
/dev/ad0s1b     1048576         0B     1.0G         0%
/dev/md1        262144          0B     256M         0%
Total           1310720         0B     1.3G         0%
```

Swap Encryption

Since swap space contains the contents of the memory, it would have sensitive information like **cleartext** passwords. In order to prevent an intruder from extracting such information from swap space, you can encrypt your swap space.

There are already two file system encryption methods that are implemented in FreeBSD 7—`gbde(8)` and `geli(8)` commands. To enable encryption on the swap partition, you need to add `.eli` or `.bde` to the device name in the `/etc/fstab` file to enable the `geli(8)` command and the `gbde(8)` command, respectively. In the following example, the `/etc/fstab` file shows a swap partition encrypted using `geli(8)` command:

```
# cat /etc/fstab
   # Device           Mountpoint    FStype   Options     Dump    Pass#
   /dev/ad0s1b.eli     none          swap      sw          0       0
   /dev/ad0s1a          /            ufs    rw,noatime    1       1
   /dev/acd0          /cdrom        cd9660   ro,noauto     0       0
```

Then you have to reboot the system for the changes to take effect. You can verify the proper operation using the following `swapinfo(8)` command:

```
# swapinfo -h
   Device                1K-blocks     Used    Avail     Capacity
   /dev/ad0s1b.eli       1048576         0B     1.0G          0%
   /dev/md0               262144         0B     256M          0%
   Total                  1310720        0B     1.3G          0%
```

Softupdates

Softupdates is a feature to increase disk access speed and decrease I/O by caching file system metadata updates into the memory. The softupdates feature decreases disk I/O from 40% to 70% in the file-intensive environments like email servers. While softupdates guarantees disk consistency, it is not recommended to enable it on root partition.

The softupdates feature can be enabled during file system creation (using sysinstall's disklabel editor) or using `tunefs(8)` command on an already created file system.

The best time to enable softupdates is before mounting partitions (that is in the super-user mode).

The following example shows softupdates enabled partitions:

```
# mount

   /dev/ad0s1a on / (ufs, local)
   devfs on /dev (devfs, local)
   /dev/ad0s1e on /tmp (ufs, local, soft-updates)
   /dev/ad0s1f on /usr (ufs, local, soft-updates)
   /dev/ad0s1d on /var (ufs, local, soft-updates)
```

In the above example, softupdates is enabled on `/tmp`, `/usr`, and `/var` partitions, but not on the root partition. If you want to enable softupdates on the root partition, you may use the `tunefs(8)` command as shown in the following example:

```
# tunefs -n enable /
```

Please note that you cannot enable or disable softupdates on an active partition (that is currently mounted partition). To do so, you should first unmount the partition or change it to read-only mode. In case you want to enable softupdates on root partition, it is recommended that you boot your system into single-user mode (in which your root partition is mounted as read-only) and then enable softupdates using the method mentioned in the above example.

Snapshots

A file system snapshot is a frozen image of a live file system. Snapshots are very useful when backing up volatile data such as mail storage on a busy mail server.

Snapshots are created under the file system that you are making snapshots from. Up to twenty snapshots can be created per file system.

The `mksnap_ffs(8)` command is used to create a snapshot from FFS partitions:

```
# mksnap_ffs /var /var/snap1
```

Alternatively, you can use the `mount(8)` command to do the same:

```
# mount -u -o snapshot /var/snap1 /var
```

Now that you have created the snapshot, you can:

- take a backup of your snapshot by burning it on a CD/DVD, or transfer it to another server using `ftp(1)` or `sftp(1)`.
- Use `dump(8)` utility to create a file system dump from your snapshot.

The `fsck(8)` command is used on a snapshot file to ensure the integrity of the snapshot before taking backups:

```
# fsck_ffs /var/snap1

   ** /var/snap1 (NO WRITE)
   ** Last Mounted on /var
   ** Phase 1 - Check Blocks and Sizes
   ** Phase 2 - Check Path names
   ** Phase 3 - Check Connectivity
   ** Phase 4 - Check Reference Counts
   ** Phase 5 - Check Cyl groups
   464483 files, 5274310 used, 8753112 free (245920 frags, 1063399
   blocks, 1.8% fragmentation)
```

Remember the following, when working with snapshots:

- Snapshots will degrade the system's performance at the time of its creation and removal, but not necessarily while running.
- Remove snapshots as soon as you finish your work.
- Snapshots can be removed in any order, irrespective of the order in which they were created.

You can also mount a snapshot as a read-only partition to view or extract its contents, using the `mount(8)` command. To mount a snapshot, you should first create a `md(4)` node as follows:

```
# mdconfig -a -t vnode -f /var/snap1

   WARNING: opening backing store: /var/snap1 readonly
   md2
```

In the above case, `mdconfig(8)` command has attached `/var/snap1` to the first available `md(8)` node and returned the name of the created node. Now you can mount the `md(8)` node as a read-only file system:

```
# mount -r /dev/md2 /mnt
```

And verify the operation using the `mount(8)` command:

```
# mount

   /dev/ad0s1a on / (ufs, local, noatime, soft-updates)
   devfs on /dev (devfs, local)
   procfs on /proc (procfs, local)
   /dev/md1 on /tmp (ufs, local)
   /dev/md2 on /mnt (ufs, local, read-only)
```

To unmount the mounted snapshot, you should first use the `umount(8)` command, and then remove `md(4)` node using `mdconfig(8)` as shown here:

```
# umount /mnt
# mdconfig -d -u 2
```

Note that `mdconfig(8)` takes the number of `md(4)` node (in this case, md2) using `-u` parameter.

Finally, to remove a snapshot file, use `rm(1) command`. It may take a few seconds.

```
# rm -f /var/snap1
```

Quotas

Quotas enable you to limit the number of files or disk space for each user or group of users. This would be very useful on multiuser systems (like virtual web hosts, shell access servers) on which the system administrator should limit disk space usage, on a per-user basis.

Quota is available as an optional feature and is not enabled, by default, in FreeBSD's `GENERIC` kernel. In order to enable quotas in FreeBSD, you should reconfigure the kernel (explained in Chapter 2) and add the following line to the kernel configuration file:

```
options QUOTA
```

You should also enable quotas in the `/etc/rc.conf` file by adding the following line:

```
enable_quotas="YES"
```

Quotas can be enabled, either for a user or a group of users, according to the file system. To enable quotas on each partition, you should add the appropriate line in the `/etc/fstab` file. Each partition may have its specific quota configuration. The following example shows different quota settings in the `/etc/fstab` file:

```
# cat /etc/fstab
# Device        Mountpoint    FStype   Options          Dump    Pass#
/dev/ad0s1b       none         swap      sw              0       0
/dev/ad0s1a         /           ufs      rw              1       1
/dev/ad0s1e       /tmp          ufs      rw              2       2
/dev/ad0s1f       /usr         ufs    rw, userquota     2       2
/dev/ad0s1d       /var         ufs    rw, groupquota    2       2
/dev/acd0         /cdrom      cd9660  ro,noauto         0       0
```

Note that either the **userquota** or the **groupquota** can be specified for each partition in the **Options** column. You can also combine both **userquota** and **groupquota** on one partition simultaneoulsy:

```
/dev/ad0s1f  /usr    ufs   rw,userquota,groupquota  2    2
```

Partition quota information is kept in the `quota.user` and `quota.group` files, in the root directories of their respective partitions.

Once you have performed the above steps, you need to reboot your system to load new kernel, and initialize the quota for appropriate partitions. Make sure `check_quotas` variable in the `/etc/rc.conf` file is not set to `NO`. Otherwise system will not create the initial `quota.user` and `quota.group` files. This can also be done by running the `quotacheck(8)` command, manually as follows:

```
# quotacheck -a

   quotacheck: creating quota file //quota.user
```

After rebooting, you can verify the quota activation by using the `mount(8)` command or use `quota(1)` utility to see the current quota statistics for each mount point:

```
# quota -v

   Disk quotas for user root (uid 0):
   Filesystem   usage quota limit grace files quota limit grace
            / 5785696     0     0       464037     0     0
```

Now that you have enabled quotas on your partitions, you are ready to set quota limits for each user or group.

Assigning Quotas

The `edquota(8)` utility is the quota editor. You can limit the disk space (block quota) and the number of files (`inode` quota) using this utility, on quota enabled partitions. Two types of quota limits can be set for both `inode` quota and block quota:

Hard limit is the implicit limit that cannot be exceeded. For example, if a user has a quota limit of 200 files on a partition, an attempt to create even one additional file, will fail.

Soft limit is the conditional limit that may be exceeded for a limited period of time, called **grace period**. If a user stays over the soft limit for more than the grace period (which is one week by default), the soft limit will turn into hard limit and the user will be unable to make any more allocations. However, if the user frees the disk space down to a soft quota limit, the grace period will be reset.

Running the `edquota(8)` command invokes your default text editor (taken from `EDITOR` environment variable), and loads current quota assignment status for the specified user:

```
# edquota jdoe

   Quotas for user jdoe:
   /: kbytes in use: 626, limits (soft = 0, hard = 0)
           inodes in use: 47, limits (soft = 0, hard = 0)
```

In the above case, user **jdoe** currently has forty seven files which use 626 kilobytes on the disk. You can modify the soft and hard values for either the block (first line) or the `inode` (second line). Once you finish setting quota limits, save and exit from your editor, and the `edquota(8)` utility will take care of applying new quota limits to the file system.

You can also change the default grace period using the `edquota(8)` utility. As in the previous example, `edquota(8)` invokes the default text editor to edit the current setting for the grace period:

```
# edquota -t0

   Time units may be: days, hours, minutes, or seconds
   Grace period before enforcing soft limits for users:
   /: block grace period: 0 days, file grace period: 0 days
```

The example, above, displays the current status of the grace period on a per-partition basis. You can edit the value of the grace period, save it, and exit from the editor to apply new grace period settings. For your new grace period settings to take effect, you should also turn quota off, for the relevant file system, and then turn it back on. This can be done using the `quotaon(8)` and `quotaoff(8)` commands.

And finally, `repquota(8)` is used to display the summary of quotas for a specified file system. The `repquota(8)` command can be used to have an overview of the current `inode` and block usage, as well as quota limits on a per-user or per-group basis (if `-g` flag on command line is specified).

When using quotas, always remember the following important notes:

- Setting a quota to zero means no quota limit to be enforced; this is the default setting for all users.
- Setting hard limit to one indicates that no more allocations should be allowed to be made.
- Setting hard limit to zero and soft limit to one indicates that all allocations should be permitted only for a limited time (grace period).
- Setting grace period to zero indicates that the default grace period (one week) should be used.
- Setting grace period to one second means that no grace period should be allowed.
- In order to use the `edquota(8)` utility to edit group quota setting, `-g` flag is specified.

File System Backup

There are different utilities in the FreeBSD base system to help system's administrators to take backups from their systems. But before starting to take backups, you should define your backup strategy.

Backups can be taken at the file-system-level, from the whole partition or physical disk, or on a higher-level. This enables you to select relevant files and directories t o be archived and moved to a tape device or a remote server. In this chapter, we will discuss different utilities and how to use them to create usable backups for your needs.

Dump and Restore

The `dump(8)` utility is the most reliable and portable backup solution to take backups on UNIX systems. The `dump` utility, in conjunction with `restore(8)`, creates your basic backup toolbox in FreeBSD. The `dump` command is able to create full and incremental backups from the whole disk or any partition of your choice. Even if your file system that you want to take backups from, is live (which in most cases is), the `dump` utility creates a snapshot of your file system before the back up, to ensure that your file system does not change during the process.

By default, `dump` creates backups on a tape drive unless you specify another file or a special device.

A typical full backup using dump may look like the following example:

```
# dump -0auL -f /usr/dump1 /dev/ad0s1a

     DUMP: Date of this level 0 dump: Sat Apr 14 16:40:03 2007
     DUMP: Date of last level 0 dump: the epoch
     DUMP: Dumping snapshot of /dev/ad0s1a (/) to /usr/dump1
     DUMP: mapping (Pass I) [regular files]
     DUMP: mapping (Pass II) [directories]
     DUMP: estimated 66071 tape blocks.
     DUMP: dumping (Pass III) [directories]
     DUMP: dumping (Pass IV) [regular files]
     DUMP: DUMP: 66931 tape blocks on 1 volume
     DUMP: finished in 15 seconds, throughput 4462 KBytes/sec
     DUMP: level 0 dump on Sat Apr 14 16:40:03 2007
     DUMP: Closing /usr/dump1
     DUMP: DUMP IS DONE
```

In the above example, dump is used to take a full backup (note the -0 flag) of the /dev/ad0s1a file, which is mounted onto the / mount point to a regular /usr/dump1 file. The -L flag indicates that the partition is a live file system; so dump will create a consistent snapshot from the partition, before performing the backup operation.

In case -L flag is specified, dump creates a snapshot in .snap directory in the root partition of the file system. The snapshot will be removed as soon as the dump process is complete. Always remember to use -L on your live file systems. This flag will be ignored in read-only and unmounted partitions.

And finally -u flag tells dump to record dump information in the /etc/dumpdates file. This information is used by dump for future backups.

The dump command can also create incremental backups using information recorded in the /etc/dumpdates file. In order to create an incremental backup, you should specify a higher backup-level from -1 to -9 in the command line. If backup-level is not specified, dump will assume a full backup (that is -0) should be taken.

```
# dump -1auL -f /usr/dump2 /dev/ad0s1a

     DUMP: Date of this level 1 dump: Sat Apr 14 15:00:36 2007
     DUMP: Date of last level 0 dump: Sat Apr 14 14:35:34 2007
     DUMP: Dumping snapshot of /dev/ad0s1a (/) to /usr/dump2
     DUMP: mapping (Pass I) [regular files]
     DUMP: mapping (Pass II) [directories]
     DUMP: estimated 53 tape blocks on 0.00 tape(s).
     DUMP: dumping (Pass III) [directories]
```

```
DUMP: dumping (Pass IV) [regular files]
DUMP: DUMP: 50 tape blocks on 1 volume
DUMP: finished in less than a second
DUMP: level 1 dump on Sat Apr 14 15:00:36 2007
DUMP: Closing /usr/dump2
DUMP: DUMP IS DONE
```

It also updates `/etc/dumpdates` with new backup dates:

```
# cat /etc/dumpdates

   /dev/ad0s1a                     0 Sat Apr 14 14:35:34 2007
   /dev/ad0s1a                     1 Sat Apr 14 15:00:36 2007
```

Once you have created dumps from your file system as regular files, you may want to move the dump file to another safe location (like a backup server), to protect your backups in case of a hardware failure. You can also create dumps directly on a remote server over SSH. This can be done by giving the following command:

```
# dump -0auL -f - /dev/ad0s1a | bzip2 | ssh admin@bkserver dd of=/usr/
backup/server1.dump
```

This will create a level 0 (or full) backup from the `/dev/ad0s1a` device over network using `ssh(1)` facility to host `bkserver` with username `admin` and uses `dd(1)` to create a file using input stream. And as we create a full backup, which may be a huge file, `bzip2(1)` is used to compress data stream to reduce the network load.

You can use your favourite compression program (for example, `gzip(1)`, `compress(1)`) with appropriate parameters, instead of `bzip2`.

Using a compression program will reduce the network load at the cost of CPU usage during dump routine.

Now that you made your backup on a tape or a remote device, you may also have to verify or restore your backup in future.

The `restore(8)` utility performs the inverse function of what `dump` does. Using `restore`, you can simply restore a backup taken using the `dump` utility, or extract your files, deleted accidentally. It can also be used to restore backups over the network.

A simple scenario for using `restore` is restoring a full backup. It is recommended that you restore your backup to an empty partition. You have to format the destination partition, using `newfs(8)`, before restoring your backup. After you restore the full backup, you can proceed to restore the incremental backups, in the order in which they were created.

A typical restore procedure would look like the following command lines:

```
# newfs /dev/da0s1a
# mount /dev/da0s1a /mnt
# cd /mnt
# restore -r -f /usr/dump1
```

The `restore` command fully extracts the dump file to your current directory. So you have to change your current directory to wherever you want to restore the backup using the `cd` command.

Another interesting feature of the `restore` utility is the **interactive mode**. In this mode, you can browse through files and directories inside the dump file, and also mark the files and directories that should be restored. This feature is very useful in restoring the files and directories, deleted accidentally.

There are a number of useful commands in the interactive restore shell to help users choose what they want to extract. The `ls`, `cd`, and `pwd` commands are similar to their equivalents, and are used to navigate through the dump file. Using `add` and `delete` commands, you can mark and unmark files and directories that you want to extract. Once you finish selecting the files, you can use the `extract` command to extract the selected files.

```
# restore -i -f /usr/dump1
   restore > ls
   .:
   .cshrc     bin/     dev/     home@    mnt/     sbin/    var/
   .profile   boot/    dist/    lib/     proc/    sys@
   .snap/     cdrom/   entropy  libexec/ rescue/  tmp/
   COPYRIGHT  compat@  etc/     media/   root/    usr/
   restore > add sbin
   restore > add rescue
   restore > extract
   restore > quit
```

The `restore` command is also used to extract dump information from the dump file using the `what` command in the interactive mode:

```
   restore > what
   Dump   date: Sat Apr 14 16:40:03 2007
   Dumped from: the epoch
   Level 0 dump of / on server.example.com:/dev/ad0s1a
   Label: none
```

The tar, cpio, and pax Utilities

There may be scenarios when you may not have to take a full dump of your hard disk or partition. Instead, you may want to archive a series of files and directories to your backup tapes or regular files. This is where `tar(1)`, `cpio(1L)`, and `pax(1)` utilities come into play.

The `tar` command is UNIX's original tape manipulation tool. It was created to manipulate streaming archive files for backup tapes. It is not a compression utility and is used in conjunction with an external compression utility such as `gzip` and `bzip2`, and compressd, in case compression is required.

Besides tape drives, you can use `tar` to create regular archive files. The `tar` archive files are called **tarball**.

Keep in mind that FreeBSD's `tar` utility, a.k.a `bsdtar(1)`, is slightly different from the GNU's tar. GNU tar or `gtar` is available in ports collection. Only BSD `tar` is covered in this chapter.

A tarball can be created, updated, verified, and extracted using the `tar(1)` utility.

```
# tar cvf backup.tar backup/
   a backup
   a backup/HOME.diff
   a backup/make.conf
   a backup/rc.conf
```

In the above example, `tar` is used to create a tarball called `backup.tar` from the `backup` directory. The `c` flag indicates `tar` should create a tar ball, `v` flag tells `tar` to be verbose and show a list of files on which the operation is being performed and `f` flag indicates the name of the output tarball (`backup.tar`) in the command.

To update a tarball, `u` flag is used:

```
# tar uvf backup.tar backup/
   a backup
   a backup/make.conf
   a backup/sysctl.conf
```

And `x` flag to extract the files from a tarball:

```
# tar xvf backup.tar
   x backup
   x backup/HOME.diff
   x backup/make.conf
   x backup/rc.conf
```

In all the above examples, the tarball archive was created as a regular file indicated by `f` flag. While omitting this flag, `tar` will use the default tape device on the `/dev/sa0` file. Other useful `tar` flags include `z` for `gzip` compression and `j` for `bzip2` compression.

You can create tarballs over network with SSH using piping technique discussed in *Dump and Restore* section.

The `cpio` utility is another important archiving utility in the FreeBSD's base system. It is similar to the `tar` utility in many ways. It was also a POSIX standard until POSIX.1-2001 and was dropped due to the 8GB file size limitation.

The `pax` utility was created by IEEE STD 1003.2 (POSIX.2) to sort out incompatibilities between tar and cpio. Pax does not depend on any specific file format and supports a handful of different archive formats including `tar`, `cpio`, and `ustar` (POSIX.2 standard). Despite being a POSIX standard that is widely implemented, it is still not as popular as a `tar` utility.

The `-w` flag is used to create archive:

```
# pax -w -f backup.pax backup/
```

And `-r` to extract (or read) the archive to current directory:

```
# pax -r -f backup.pax
```

The `pax` utility is also able to read/write different archive types that can be specified by `-x` flag. The supported parameters of `pax` are shown in the following list:

- **cpio**: New POSIX.2 `cpio` format
- **bcpio**: Old binary `cpio` format
- **sv4cpio**: System V release 4 `cpio` format
- **sv4crc**: System V release 4 `cpio` format with CRC checksums
- **tar**: BSD `tar` format
- **ustar**: New POSIX.2 `tar` format

Snapshots

Actually, taking snapshots from a file system isn't a backup method, but is very helpful in restoring accidentally removed files. Snapshots can be mounted as regular file systems (even over network) and the system administrator can use regular system commands to browse the mounted file system and restore selected files and directories.

RAID-GEOM Framework

GEOM is an abstraction framework in FreeBSD that provides the infrastructure required to perform transformation on disk I/O operations. Major RAID control utilities in FreeBSD use this framework for configuration.

This section does not provide in-depth information about RAID and GEOM, but only discusses RAID configuration and manipulation using GEOM.

Currently GEOM supports RAID0 (Striped Set without parity) and RAID1 (Mirrored Set without parity) through `geom(8)` facility.

RAID0—Striping

Striping disks is a method to combine multiple physical hard disks into one big logical volume. This is done mostly using relevant hardware RAID controllers, while GEOM provides software support for RAID0 stripe sets.

RAID0 offers improved disk I/O performance, by splitting data into multiple blocks and performing simultaneous disk writes on multiple physical disks, but offers no fault tolerance for hard disk errors. Any disk failure could destroy the array, which is more likely to happen when you have many disks in your set.

Appropriate kernel module should be loaded before creating a RAID0 volume using the following command:

```
# kldload geom_stripe
```

This can also be done through the `/boot/loader.conf` file, to automatically load the module during system boot up, by adding this line:

```
geom_stripe_load="YES"
```

Normally, you will not need to load any GEOM module manually. GEOM related utilities automatically detect all modules that are required to be loaded, and will load it manually.

The `gstripe(8)` utility has everything you need to control your RAID0 volume. Using this utility you can create, remove, and query the status of your RAID0 volume.

There are two different methods to create a RAID0 volume using `gstripe`—manual and automatic. In the manual method, the `create` parameter is used, and volumes created using this method do not persist during reboots. The volumes should be created at boot time, if persistence is required:

```
# gstripe create stripe1 /dev/da1 /dev/da2
# newfs /dev/stripe/stripe1
```

The newly created and formatted device can now be mounted and used as shown here:

```
# mount /dev/stripe/stripe1 /mnt
```

In the automatic method, the metadata is stored on the last sector of every device, so that they can be detected and automatically configured during boot time. In order to create automatic RAID0 volume, you should use `label` parameter:

```
# gstripe label stripe1 /dev/da1 /dev/da2
```

Just like manual volumes, you can now format `/dev/stripe/stripe1` using `newfs` and mount it.

To see a list of current GEOM stripe sets, `gstripe` has the `list` argument. Using this command, you can see a detailed list of devices that form the stripe set, as well as the current status of those devices :

```
# gstripe list

   Geom name: stripe1
   State: UP
   Status: Total=2, Online=2
   Type: AUTOMATIC
   Stripesize: 131072
   ID: 1477809630
   Providers:
   1. Name: stripe/stripe1
     Mediasize: 17160732672 (16G)
     Sectorsize: 512
     Mode: r1w1e0
   Consumers:
   1. Name: da1s1d
     Mediasize: 8580481024 (8.0G)
     Sectorsize: 512
     Mode: r1w1e1
     Number: 1
   2. Name: da0s1d
     Mediasize: 8580481024 (8.0G)
     Sectorsize: 512
     Mode: r1w1e1
     Number: 0
```

To stop a RAID0 volume, you should use the `stop` argument in the `gstripe` utility. The `stop` argument will stop an existing striped set ,but does not remove the metadata from the device, so that it can be detected and reconfigured after system reboots.

```
# gstripe stop stripe1
```

To remove metadata from the device and permanently remove a stripe set, the `clear` argument should be used;

```
# gstripe clear stripe1
```

RAID1—Mirroring

This level of RAID provides fault tolerance from disk errors and increased *READ* performance on multithreaded applications. But write performance is slightly lower in this method. In fact, RAID1 is a live backup of your physical disk. Disks used in this method should be of equal size.

The `gmirror(8)` facility is the control utility of RAID1 mirror sets. Unlike RAID0, all RAID1 volumes are automatic and all components are detected and configured automatically at boot time. The `gmirror` utility uses the last sector on each device to store metadata needed for automatic reconfiguration. This utility also makes it easy to place a root partition on a mirrored set.

It offers various commands to control mirror sets. Initializing a mirror is done using the `label` argument as shown here:

```
# gmirror label -b round-robin mirror1 da0 da1
```

In the above example, we created a mirror set named `mirror1` and attached the `/dev/da0` and `/dev/da1` disks to the mirror set.

The `-b` flag specifies the "balance algorithm" to be used in the mirror set. There are four different methods used as balance algorithms, which are listed as follows:

- **load**: Read from the device with the lowest load.
- **prefer**: Read from the device with the highest priority.
- **round-robin**: Use round-robin algorithm between devices.
- **split**: Split read requests that are bigger than or equal to slice size, on all active devices.

You may choose an appropriate algorithm depending on your hardware configuration. For example, if one of your hard disks is slower than the others , you can set higher priority on the fastest hard disk using `gmirror`'s `insert` argument and use the `prefer` method as the balance algorithm.

Once you finish initializing your mirror set, you should format the newly created device using `newfs` command and mount it to relevant mount point:

```
# newfs /dev/mirror/mirror1
# mount /dev/mirror/mirror1 /mnt
```

The `stop` argument stops a given mirror.

Using the `activate` and `deactivate` arguments you can active and deactivate a device that is attached to a mirror, which would be useful in removing or replacing a hot-swappable hard disk. When a device is `deactivated` inside a mirror set, it will not attach itself to the mirror automatically, even after a reboot, unless you re-activate the device using the `activate` argument.

To add a new device to the mirror set, or to remove a device permanently, the `insert` and `remove` arguments can be used, respectively. The `remove` argument also clears metadata from the given device. This is shown in the following command lines:

```
# gmirror insert mirror1 da2
# gmirror remove mirror1 da1
```

If you want to change the configuration of a mirrored volume (for example, changing balance algorithm on the fly), the `configure` argument can be used:

```
# gmirror configure -b load mirror1
```

In case of disk failure, when a device is faulty and cannot be reconnected to the mirror, the `forget` argument will tell `gmirror` to remove all faulty components. Once you replace the faulty disk with a brand new one, you can use the `insert` argument to attach a new disk to the array, and start synchronizing data.

Disk Concatenation

This method is used to concatenate multiple physical hard disks to create bigger volumes, beyond the capacity of one hard disk. The difference between this method and RAID0 's is that, in this method, data is written to the disk sequentially. This means that the system will fill the first device first, and the second device will be used only when there is no space left on the first device. This method does not offer any performance improvements or redundancy.

To create a concatenated volume, the `gconcat(8)` facility is available. As in RAID0, there are two methods to create a concatenated volume—manual and automatic.

Using the `create` parameter, you can create a manual concatenated volume and attach the desired physical disks. In this method, as no metadata will be written on the disk, the system will not be able to detect and reconfigure the volume after system reboots.

In order to create an automatic concatenated volume, the `label` parameter should be used:`# gconcat label concat1 da0 da1 da2`

Once a volume is created using either the manual or the automatic method, it should be formatted using `newfs` as shown as follows:

```
# newfs /dev/concat/concat1
# mount /dev/concat/concat1 /mnt
```

There is no way to remove a device from a concatenated volume. However, you can add new disks to an existing volume, and grow the size of the file system on the volume:

```
# gconcat label concat1 da3 da4
# growfs /dev/concat/concat1
```

To stop a concatenated volume, the `stop` argument is used. However this will not remove the volume permanently. The `clear` argument will remove the concatenated volume permanently, and also remove the GEOM metadata from the last sector of the attached devices.

Summary

The impact of disk I/O should not be overlooked when performance is a concern. A well configured storage will dramatically improve the system's overall performance. This chapter introduces the necessary tips and information a system administrator needs, to tweak the storage setup on a FreeBSD server. We have also seen how to take backups, weed out system redundancy and improve performance using RAID arrays, and ways and means of creating and managing virtual memory partitions, effectively.

2
System Configuration—Keeping it Updated

As a system administrator, you would definitely know the importance of keeping the system up-to-date to work around the security holes and bug fixes while keeping the highest service availability. Moreover, as FreeBSD gets upgraded round the clock, it is very important to know the right time and the need for an update.

Upgrading a system requires updating the local source tree from a server and compiling specific parts of a system such as a library, the FreeBSD kernel, or in some cases, the whole operating system. For those who are not interested in dealing with the source code and recompiling, there are other ways to perform the binary updates. The professionals can customize the binary updates generated for their systems by changing the source code and recompiling, to gain better performance.

For the developers and end users the source code is available on the project's CVS servers. Hence, the FreeBSD system administrator must have a basic knowledge of CVS. Developers use CVS to record the updates to the source tree and the end users check out the latest changes to their system, to update it as required. This chapter discusses different ways of tracking the security-related updates. Further, it discusses rebuilding the kernel, and the world (except the kernel), for those who prefer to create their customized and optimized systems from scratch.

In this chapter we will look into the following:

- CVSup as the synchronizing source— tracking –STABLE and –CURRENT
- Ports collection
- Security advisories
- Customizing and rebuilding kernel
- Rebuilding world
- Binary update
- Recovering from a dead kernel

CVSup—Synchronizing the Source Code

The FreeBSD project makes a heavy use of CVS to make the source tree available for different releases of the operating system on the development servers. The FreeBSD project also uses the Perforce version control system for various other (mostly experimental) projects. However, as a system administrator, there is no need to deal with the Perforce system. It is highly recommended that you keep track of the latest changes by subscribing to the appropriate mailing lists and checking the CVS tree. A list of FreeBSD project's mailing list is available at `http://lists.freebsd.org/`.

There are various tags available for different releases of FreeBSD, on the CVS server. Depending on the `revision` tag that you are tracking, you may see different volumes of traffic on the CVS server. There are two types of tags—the `branch` and `release` tags.

The `branch` tags refer to a particular line of development, while the `release` tags refer to the FreeBSD release in a specified time.

For example, the `RELENG_7` tag indicates the line of development for FreeBSD - 7.x (also known as FreeBSD 7-STABLE or –STABLE for short). Alternatively, `RELENG_7_1` is a release branch for FreeBSD-7.1, which will only be updated for the security advisories and other critical updates.

An example of `release` tags is `RELENG_7_0_0_RELEASE`, which is the release point of FreeBSD 7.0-RELEASE.

The `HEAD` tag is the main line of development and all new features are imported to this tree. This tree contains the codes that are necessary for the test reasons and may break your running system down, due to the library updates or changes in the memory structure.

It is not advisable to track –CURRENT tree on a production server as some of the new updates may render your system unstable.

It is important to choose the `revision` tag, which you want to track, to keep your servers up-to-date and stable. It is also recommended to keep track of the `release` tag of your currently installed FreeBSD version. For example, if your server is running FreeBSD 7.0 (which was installed from 7.0-RELEASE CD-ROM), it is advisable to keep it synchronized with `RELENG_7_0` tag, which contains only the critical updates. By tracking the –STABLE branch (in this case `RELENG_7`), you will have all the features and updates made to your FreeBSD release development line (FreeBSD 7.x in this case). This means that once the 7.1-RELEASE is released, you can check it out on the `RELENG_7` branch and update your system.