MAGAZINE BSD

FOR NOVICE AND ADVANCED USERS

OPENBSD

INSIDE

BUILDING A DESKTOP FIREWALL
PERFORMANCE COMPARISON ITTIA DB AND SQLITE
SOME INTERESTING ONE FLOPPY SYSTEMS
REMOTE INSTALLATION OF THE FREEBSD
OPENBSD AS A MAIL SERVER

VOL.3 NO.7
ISSUE 7/2010(13)
Price $11.99 US
1898-9144

Greenest, most energy-efficient blade server in the industry!

20 Server Compute Nodes in 7U of Rack Space

The iX-TB4X2 chassis holds 10 TwinBlade servers and each TwinBlade supports two nodes. This gives the iX-TB4X2 chassis the ability to house 20 nodes in 7U of rack space. The powerful Triton TwinBlade achieves 0.35U per dual-processor node, and is twice as dense as the previous generation of dual-processor blades.

A fully-loaded iX-Triton TwinBlade supports 40 Intel® Xeon® 5600/5500 series processors and up to 2.5 TB DDR 1333/1066/800MHz ECC Registered DIMM memory. In a 42U rack this translates into 120 nodes with 240 Intel® Xeon® 5600/5500 series processors and 15 TB DDR 1333/1066/800MHz ECC Registered DIMM memory.

$ ▸ By replacing 1U servers with TwinBlade servers, the power savings of the iX-TB4X2 can reach more than $1000* per year, per server with reduced cooling costs added in.

▸ Replacing 1U rackmount servers with an iX-TB4X2 Twin Blade can reduce carbon dioxide emissions by over 5.5 metric tons.**

▸ The iX-Triton TwinBlade delivers the most energy-efficient blade server in the industry with four N+1 redundant, high efficiency (94%) 2500W power supplies.

Key features:

- Up to 10 dual-node TwinBlades in a 7U Chassis, 6 Chassis per 42U rack
- Remotely manage and monitor TwinBlades, power supplies, cooling fans, and networking switches
- Hardware Health Monitor
- Virtual Media Over Lan (Virtual USB, Floppy/CD, and Drive Redirection)
- Integrated IPMI 2.0 w/ remote KVM over LAN/IP
- Remote Power Control
- Supports one hot-plug management module providing remote KVM and IPMI 2.0 functionalities
- Up to four N+1 redundant, hot-swap 2500W power supplies
- Up to 16 cooling fans

Each of the TwinBlade's two nodes features:

- Intel® Xeon® processor 5600/5500 series, with QPI up to 6.4 GT/s
- Intel® 5500 Chipset
- Up to 128GB DDR3 1333/ 1066/ 800MHz ECC Registered DIMM / 32GB Unbuffered DIMM
- Intel® 82576 Dual-Port Gigabit Ethernet
- 2 x 2.5" Hot-Plug SATA Drive Trays
- Integrated Matrox G200eW Graphics
- Mellanox ConnectX QDR InfiniBand 40Gbps or 10GbE support (Optional)

** Electricity costs vary by location.*

*** According to Energy Information Agency (a statistical agency of the U.S. Department of Energy), saving one kilowatt hour of electricity reduces carbon dioxide emissions by 1.43 pounds.*

Dear Readers!

The first month of summer is coming to an end, I guess most prorably many of you are on vacations having good time with your friends and family. I am happy you dont forget about BSDMagazine and download it every month. :)

BSD Magazine is growing, it has already around 22 000 subcribers all over the world. Comparing to 10 000 printed copies which were distributed in USA before January - this number has really grown! We are looking for the new ways to promote our magazine all the time and we are very grateful for every help you give us! Thank you for spreading a word abour BSD Mag!

This issue is devoted to OpenBSD mainly, but not only. A little bit of firewalling, floppy systems, sharing interesting experience and other. All these great authors worked hard to contribute to this issue: Dru Lavigne, Juraj Sipos, Daniel Gerzo, Daniele Mazzocchio, Sasan Montaseri, Joshua Ebarvia, Jesse Smith. In case you have any questions about the articles, just let us know we will forward them directly to authors.

Enjoy your reading and have a nice day!

Olga Kartseva
Editor in Chief
olga.kartseva@software.com.pl

MAGAZINE BSD

Editor in Chief:
Olga Kartseva
olga.kartseva@software.com.pl

Contributing:
Jan Stedehouder, Rob Somerville, Marko Milenovic, Petr Topiarz, Paul McMath, Eric Vintimilla, Matthias Pfeifer, Theodore Tereshchenko, Mikel King, Machtelt Garrels, Jesse Smith

Special thanks to:
Marko Milenovic, Worth Bishop and Mike Bybee

Art Director:
Ireneusz Pogroszewski

DTP:
Ireneusz Pogroszewski

Senior Consultant/Publisher:
Paweł Marciniak *pawel@software.com.pl*

National Sales Manager:
Ewa Łozowicka
ewa.lozowicka@software.com.pl

Marketing Director:
Ewa Łozowicka
ewa.lozowicka@software.com.pl

Executive Ad Consultant:
Karolina Lesińska
karolina.lesinska@bsdmag.org

Advertising Sales:
Olga Kartseva
olga.kartseva@software.com.pl

Publisher :
Software Press Sp. z o.o. SK
ul. Bokserska 1, 02-682 Warszawa
Poland
worldwide publishing
tel: 1 917 338 36 31
www.bsdmag.org

Software Press Sp z o.o. SK is looking for partners from all over the world. If you are interested in cooperation with us, please contact us via e-mail: *editors@bsdmag.org*

The editors use automatic DTP system AUPUS

Mathematical formulas created by Design Science MathType™.

Contents

GET STARTED

HOW TO'S

LET'S TALK

Building a Desktop Firewall

with pf and fwbuilder

This article is an excerpt from the Firewalls and VPNs chapter of the book The Best of FreeBSD Basics (ISBN 9780979034220), published by Reed Media Publishing.

What you will learn…

- how to build and configure a basic firewall using pf and fwbuilder on a FreeBSD system

What you should know…

- basic knowledge of TCP/IP and TCP/UDP ports

Everyone knows that you should be behind a firewall whenever you go online. However, not everyone knows that it's easy to create a personal firewall for a FreeBSD (or PC-BSD or DesktopBSD) system. This section shows how even a casual home user can get a firewall up and running in about ten minutes.

The Software

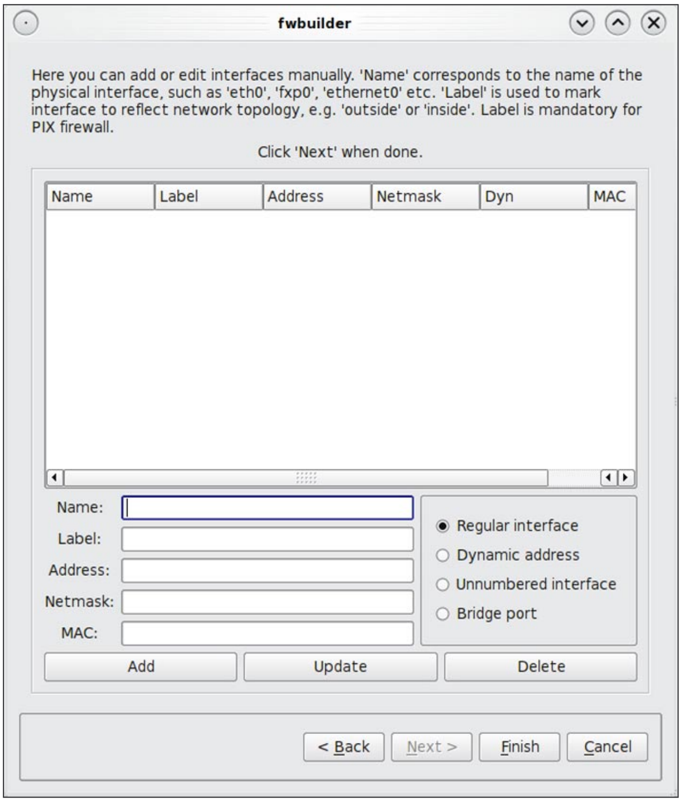
Like all of the BSDs, FreeBSD has always been security conscious. It offers several built-in firewalls to choose from: `ipfw`, `ipf`, and `pf`. I use `pf` because it is built into all of the BSDs, including OpenBSD, NetBSD, and DragonFly BSD.

I also recommend using a GUI firewall editor called `fwbuilder`. While my examples will demonstrate this utility from a FreeBSD system, it is available for Linux, Mac OS X, and Windows XP and supports `iptables`, `ipf` (IP Filter), `pf` and `ipfw`. `pf` comes with FreeBSD, but double-check that it is loaded on your system by typing the following as the superuser:

```
# kldload pf
```

If you get your prompt back, you just loaded it manually. If you're in the habit of turning off your computer, add a line to `/etc/rc.conf` to reload `pf` when your system boots:

```
pf_enable="YES"
```



fwbuilder

Here you can add or edit interfaces manually. 'Name' corresponds to the name of the physical interface, such as 'eth0', 'fxp0', 'ethernet0' etc. 'Label' is used to mark interface to reflect network topology, e.g. 'outside' or 'inside'. Label is mandatory for PIX firewall.

Click 'Next' when done.

Name	Label	Address	Netmask	Dyn	MAC

Name:
Label:
Address:
Netmask:
MAC:

● Regular interface
○ Dynamic address
○ Unnumbered interface
○ Bridge port

Figure 1. *Your new firewall*

Listing 1. *Using ifconfig to find out your interface names, IP addresses, and MAC addresses*

```
# ifconfig
xl0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> mtu 1500
options=9<RXCSUM,VLAN_MTU>
inet 192.168.2.49 netmask 0xffffff00 broadcast 192.168.2.255
ether 00:04:75:ee:e0:21
media: Ethernet autoselect (100baseTX <full-duplex>)
status: active
lo0: flags=8049<UP,LOOPBACK,RUNNING,MULTICAST> mtu 16384
inet 127.0.0.1 netmask 0xff000000
```

Listing 2. *Interface information for example firewall object*

```
Name:          xl0
Label:          external
Address:       (greyed out because I checked Dynamic address)
Netmask:      (greyed out because I checked Dynamic address)
MAC:           00:04:75:ee:e0:21

Name:         lo0
Label:          loopback
Address:     127.0.0.1
Netmask:    255.0.0.0
MAC:          (leave empty)
```

If you instead get an error like:

```
kldload: can't load pf: File exists
```

it means that your system is already configured to load pf for you.

Installation and Configuring the Firewall Object

From the GUI, become the superuser and install and start `fwbuilder`:

```
# pkg_add -r fwbuilder
# rehash
# fwbuilder
```

This will take you to the fwbuilder GUI, which is divided into two main sections. The left frame contains an Object tree and the right frame contains your firewall rules (after you have defined some objects). Using objects is a very powerful visual aid, allowing you to quickly see your networks, computers, and services, and to cut and paste these objects into firewall rules.

The first object you create should represent your firewall. Click on the New Object icon (it looks like a sheet of paper) and select New Firewall from the drop-down menu. Give your firewall a name (I called mine `my_firewall`), select PF from the drop-down menu of firewall software, and click Next. Keep the default to Configure interfaces manually, and press Next. You should see a screen like Figure 1.

Be sure to Add the interface information for each NIC in your computer as well as the loopback. If your firewall will protect only your personal computer, you need only one physical NIC installed in your computer. If you wish your computer to provide NAT to other computer(s) on your home network, you need to have two NICs installed.

If your ISP assigns you a DHCP address, check the Dynamic address option. Otherwise, enter your static IP address and subnet mask.

To determine the FreeBSD names of your interfaces as well as the associated IP addressing information, type: see Listing 1.

With my information, I entered into the New Firewall screen: see Listing 2.

When choosing a label, *external* is good for the NIC you use to access the internet, and *internal* is good for the NIC attached to your home network. If you need to add a static subnet mask, you must first convert that hex number (`0xffffff00`, for example) to decimal. Ignore the `0x`, as that simply indicates a hex number. What remains is four pairs of numbers: `ff ff ff 00`. ff is easy; it represents `255`; and `00` represents `0`. So this mask is: `255.255.255.0`. If you have a pair that isn't an ff or a `00`, use the conversions in Table 1.

Note to users of modems: your interface name will be either `ppp0` or `tun0`. Running `ifconfig` while connected to the Internet will make it easier to spot your IP address.

Once you've entered the information for a NIC, click Add and repeat for each of your NIC(s). When finished, click on the Finish button. If you take a look at your Object tree, it now contains some new objects: one for your firewall and one for each interface you defined.

You have one last change to finish the firewall object – marking one of the interfaces as a Management interface. For a personal firewall, it should be the loopback. Double-click your loopback object and check the Management interface box, then click Apply.

Table 1: *Hex Conversion Table*

Hex	Decimal
80	128
c0	192
e0	224
f0	240
f8	248
fc	252
fe	254

Creating a Simple Firewall Ruleset

You now have everything you need to create a simple firewall ruleset that allows your personal computer to access the internet and prevents anyone on the internet from accessing your computer.

Highlight the Policy object under your firewall, then click on the Rules menu and select Insert Rule (see Figure 2).

Notice that the default rule denies any source from reaching any destination using any TCP/UDP service. To allow the system running the firewall, right-click your firewall object and select Copy. Right-click inside the Source box of the rule and Paste. Your firewall should now show as the source of packets. Next, right-click the Deny word under Action and change it to Accept. In the Options box, right-click and select Logging Off – you don't want to log every one of your successful packets.

You should always add a comment to remind yourself why you made a rule. If you double-click on the box, you can type in your comment. I wrote: allow my computer to access the internet

That one rule is enough to give you a working firewall. If you want, you can add a second rule. With your existing rule highlighted, click on the Rules menu and select Add Rule Below. Add a comment: deny all other traffic.

If you don't plan on looking at your firewall logs, turn off logging in the Options box.

Note that this second rule isn't necessary for this setup, because the `pf` firewall assumes you want to deny any traffic you didn't explicitly accept. This is known as an implicit deny. You may find it useful to add the rule with a comment to remind you of this behavior.

Tip: A quick administrator's trick is to add this rule only when you are troubleshooting a problem and to leave the Logging option on.

Installing your Firewall Rules

You've just created a firewall ruleset, but it won't start working until you install it.

First, you need to configure `sshd` to allow the superuser to connect and install the firewall rules. By default, FreeBSD doesn't allow superuser ssh sessions. Change this default by typing the next line very carefully and double-checking your upper- and lowercase and your >> before pressing enter:

```
# echo "PermitRootLogin yes" >> /etc/ssh/sshd_config
```

Don't worry; no one on the internet will be able to `ssh` to your computer once you install your firewall rules.

Next, tell sshd about that change:

```
# /etc/rc.d/sshd reload
```

Reloading sshd config files.

If you see an error:

```
sshd not running? (check /var/run/sshd.pid).
```

use this command instead:

```
# /etc/rc.d/sshd start
```

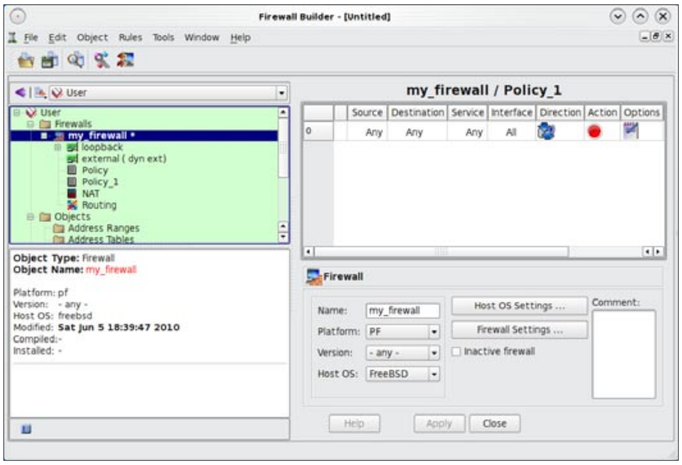
Starting sshd.

Double-check that sshd is running with:

```
# /etc/rc.d/sshd status
```

sshd is running as pid 5467.

Next, select Install from the Rules menu. Make sure both boxes are checked to Compile and Install the firewall. Click Next which will prompt you to select a filename and location to save your fwbuilder policy (it will end in a *.fwb* extension). In the next men, enter *root* for the username and type in the password for your superuser account.



my_firewall / Policy_1

	Source	Destination	Service	Interface	Direction	Action	Options
0	Any	Any	Any	All		●	

Object Type: Firewall
Object Name: my_firewall

Platform: pf
Version: - any -
Host OS: freebsd
Modified: Sat Jun 5 18:39:47 2010
Compiled:-
Installed: -

Name: my_firewall
Platform: PF
Version: - any -
Host OS: FreeBSD
□ Inactive firewall
Comment:

Figure 2. *Inserting a rule*

Review the options, then press OK. You should receive a New RSA key message:

You are connecting to the firewall `my_firewall` for the first time. It has provided you its

identification in a form of its host public key. The fingerprint of the host public key is:

`b6:76:30:aa:01:27:64:48:3b:18:28:18:5b:c9:ae:e4` You can save the host key to the local database by pressing YES, or you can cancel connection by pressing NO. You should press YES only if you are sure you are really connected to the firewall `my_firewall`.

It is safe to press Yes because you know you are connecting to your own firewall. However, it is good to know how to check a host's fingerprint in case you ever connect to a remote FreeBSD system:

```
# ssh-keygen -l -f /etc/ssh/ssh_host_dsa_key.pub
1024
b6:76:30:aa:01:27:64:48:3b:18:28:18:5b:c9:ae:e4
```

Note: You will only need to verify the fingerprint the very first time you install your firewall.

Once you click Yes, the policy will be installed and should indicate a Progress of Success. Your firewall is now running.

Note: The installation may fail if `pf` is already running. Try running the command `pfctl -d` to disable the firewall and then retry the Install from the Rules menu again.

Controlling the Firewall

Use the `pfctl` (`pf` control) command to see what's happening with your firewall and to stop and start the firewall. Use the show switch (`-s`) to view the rules currently running on the firewall:

```
# pfctl -s rules
pass out quick inet from (xl0) to any flags S/SA keep state
                  label "RULE 0 -- ACCEPT "
block drop quick inet all label "RULE 1 -- DROP "
block drop quick inet all label "RULE 10000 -- DROP "
```

If you compare that text to the rules you made in `fwbuilder`, you'll recognize rules 0 and 1. Rule 10000 is that implicit deny rule.

If you ever wish to stop your firewall, use the disable switch:

```
# pfctl -d
```

To restart the firewall, specify the name of your ruleset. It will be in `/etc` and have the same name as your firewall. In my case, it is in `/etc/my _ firewall.conf`. To start this firewall, I use `pfctl` at the command line to load the rules and enable the packet filter:

```
# pfctl -e -f /etc/my_firewall.conf
```

Alternatively, I can right-click the firewall in the Objects tree and choose Install from the drop-down menu.

Note: If you added the line to `/etc/rc.conf` mentioned at the beginning of this section, add another line to load your ruleset if you reboot your computer:

```
pf_rules="/etc/my_firewall.conf"
```

where `my _ firewall.conf` is the name of your ruleset. It is always a good idea to run `pfctl -s` rules after a reboot to double-check that your firewall is running.

Conclusion

I've demonstrated how to make a personal firewall that protects your system while allowing you to access the internet. The next section will show you how to install a NAT policy with `fwbuilder` and explore some of its other features.

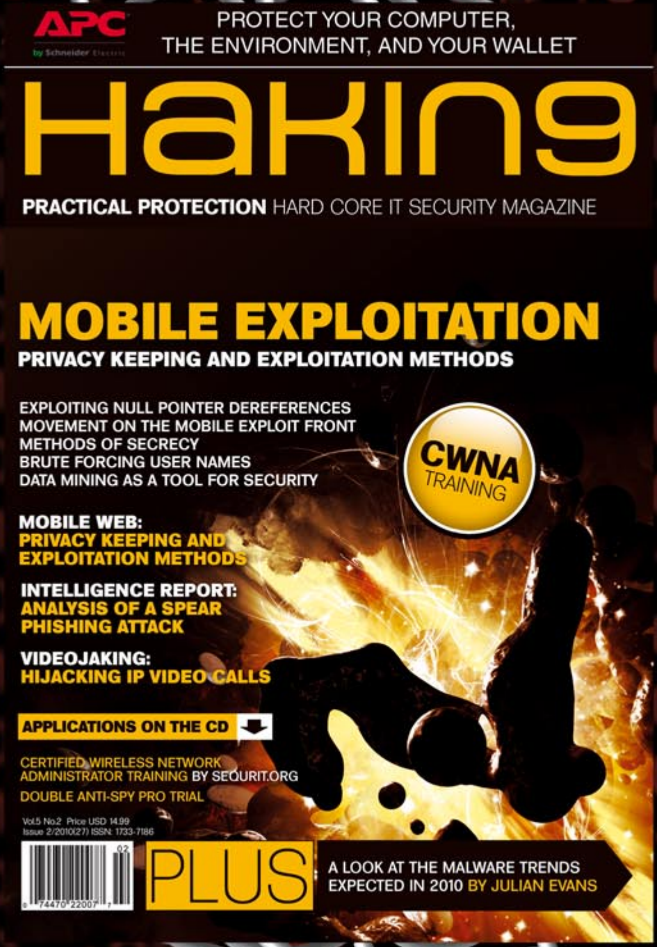
Original Article:

http://www.onlamp.com/pub/a/bsd/2006/08/03/FreeBSD Basics.html

DRU LAVIGNE

Dru Lavigne is a network and systems administrator, IT instructor, author and international speaker. She is author of O'Reilly-'s FreeBSD Basics column and the books BSD Hacks, The Best of FreeBSD Basics and the Definitive Guide to PC-BSD.

She is currently the Editor-in-Chief of the Open Source Business Resource, a free monthly publication covering open source and the commercialization of open source assets. She is founder and Chair of the BSD Certification Group Inc., a non-profit organization with a mission to create a standard for certifying BSD system administrators. Dru is a Director at the FreeBSD Foundation.



OpenBSD

Some Interesting One Floppy Systems

One floppy systems are very practical, as they usually have a specific goal, which cannot be said about all Live CD's.

What you will learn…

- Readers will learn something about how to use serial console in OpenBSD, how embedded systems can be made, or what is a transparent firewall

What you should know…

- how to setup network connections with ifconfig
- they should read manual pages of some important OpenBSD commands (pfctl, disklabel, fdisk, etc.)

To configure OpenBSD installed on your hard disk as router, you must make some configurations; with a floppy system already designed to work as router no such configuration is needed (except for the most basic one like putting the proper names of network devices into configuration files). Such a diskette is portable and easy to use. You may, too, have other wishes – an MP3 player on a diskette, or even a transparent firewall.

I am the author of SONaFR and KarmaBSD – two quite interesting one floppy systems. MaheshaBSD is a bit larger project of mine, but I already wrote an article about this LiveCD in the May issue of the BSD Magazine. SONaFR is a router based on OpenBSD 4.1 and the latter one (KarmaBSD aka 1FCD-OpBSD) is a one floppy MP3 player with a number of possibilities I describe later.

SONaFR

There are not many one floppy OpenBSD routers or firewalls and my project usually always appears on the top in Google with keywords like: one – floppy – router – OpenBSD. Another floppy router is fdgw (*http://www.fml.org/software/fdgw/*), but it is based on NetBSD. One of the best one floppy OpenBSD projects is foaf (Floppy OpenBSD Firewall) (*http://www.theapt.org/openbsd/firewall.html*).

How to use SONaFR

This floppy distro has a minimal kernel. You must have two network interface cards (NIC's) in a computer where you use this floppy. To see all the network cards available on your system, type:

```
ifconfig (from within SONaFR after it boots).
```

To see all the cards that the SONaFR kernel supports, type:

```
more etc/cards
```

The configuration scripts of SONaFR (for example, `/etc/pf.conf`) may be immediately used in any OpenBSD hard disk installation for firewall/router purposes; thus anybody can learn how to configure the OpenBSD packet filter.

Transparent Firewall

This thing may also be used as a transparent firewall (invisible firewall). If you have a computer with two NIC's (the third NIC may be used only for a SSH login with purpose to control such a transparent firewall) and you move data from one network card to another one via a bridge (without IP addresses), you work on the OSI layer 2 model (data link); thus, if you move data this way over firewall, the advantage is that you may put such a firewall anywhere – you can split any network segment without needing to configure anything (except for the transparent firewall). Such a firewall is very quick, as no decisions need to be made with respect to IP addresses a normal firewall always requires. Bandwidth, too, may be easily reduced (with use of ALTQ – ALTernate Queueing framework for BSD UNIX).

A good (and quick) overview of transparent firewalling in OpenBSD with tips and setup requirements can be found here: *http://www.dalantech.com/fusionbb/showtopic.php?tid/71026/pid/71026/post/last/m/1/*.

First, you must create the bridge (type the following command from within SONaFR):

```
ifconfig bridge0 create
```

Then activate the bridge:

```
brconfig bridge0 add rl0 add rl2 up
```

(replace "rl0" and "rl2" with real network devices present on your system.)

To activate the transparent firewall, you have to run the pfctl command (for packet filtering; the `/etc/pf.conf` file needs to be edited if you have special requirements; SONaFR has a little editor for such a purpose, just type: mg):

```
pfctl -f /etc/pf.conf
```

The behavior of such a firewall depends on rules defined in the `/etc/pf.conf` file.

To run SONaFR, your minimal requirements must be at least 9,5 MB of RAM and a Pentium (486 too) computer with two network cards and a working diskette drive. However, the SONaFR's ability to detect all possible network cards (NIC's) is limited. This is because the system has a minimalist kernel.

The `/etc/pf.conf` file in SONaFR works immediately. If you have more requirements, look into pf (Packet Filter) Internet tutorials and edit the `pf.conf` file appropriately. The purpose of this distribution was to bring something quick and easy-to-use. The only thing that the user must do is to substitute the `nfe0` and `rl0` entries in the SONaFR's pf.conf file with network devices on his or her system (`/etc/pf.conf`):

```
IntIf="nfe0"
ExtIf="rl0"
```

KarmaBSD

I often meet with broken notebooks on which the CD-ROM or floppy drive does not work. It is a pity to discard such a notebook. If some sensitive data lies on its disk and the computer's hard drive does not boot, you will hardly find a BSD tool with use of which you can copy and save data from such a notebook's USB port to a USB hard drive (or stick). You may oppose that there is quite a number of BSD Live CD's today. But if the CD-ROM drive on such a notebook is broken, such an argument does not have any weight.

KarmaBSD supports mounting of a number of disk formats:

Type of medium: USB; CD-ROM/DVD; network

Format: NTFS; ISO9660; EXT2FS; FAT MSDOS file systems; UDF; NFS file system (network)

How to prepare a disk

KarmaBSD, too, has the OpenBSD's base system tools such as newfs, disklabel, fdisk, etc., so if the hard drive (and the CD-ROM drive, too) on your notebook is broken, you may insert a new one into it and prepare your MS-DOS partition, for example:

```
fdisk -e wd0
```

then type:

```
print (or help to learn more); then:
```

`edit0` (if 0 is the partition you want to edit); then: 04 (for DOS FAT16), or 0B or 0C (both for FAT32), or A6 (for OpenBSD), A5 (for FreeBSD), A9 (for NetBSD), 83 (for Linux), 82 (for Linux swap). After you specify the operating system, type `quit` to save changes.

Finally, the partition needs one more thing – a format. To make the FAT32 format on your hard drive, type:

```
newfs_msdos -F 32 /dev/rwd0i
```

How to mount disks

To mount any drive (NTFS drive, for example), type:

```
disklabel wd0
```

or

```
disklabel sd0 (USB disk)
```

You will then see all available partitions marked with letters like j, k, l, m.

If you type:

```
mount_ntfs /dev/wd0j /mnt
```

your NTFS partition (j) will be mounted to the /mnt directory (as read only).

Serial console

This mini BSD also supports serial console (capturing output of a remote computer's screen to the screen of the local computer over a serial cable). Type:

```
set tty com0
```

at the boot prompt when KarmaBSD starts; then on your desktop computer (not the one on which KarmaBSD runs) type:

```
tip tty00
```

This way you can control any computer that has a broken keyboard or display (a very common failure with notebooks). Instead of the command `tip tty00` you must type in another OpenBSD box, you may use Putty, a free program also available for Windows.

The most important thing – music

KarmaBSD, too, can work as a one floppy MP3 player, as the purpose of this thing was to maximally utilize broken notebooks (computers). MP3 files can be played over network, too (NFS). It is all very simple. From within KarmaBSD you just type:

```
1
```

The script (`1` for `/dev/cd0a`; or `2` for `/dev/cd1a`) will mount your CD with MP3 files, it will create a playlist in /tmp, and with use of mpg123 you will automatically listen to hours of never-ending music or audiobooks by pressing one single key (1).

How to put these floppy systems on a CD

Today, when floppy drives gradually disappear from computers, both these one floppy systems may be quite easily put on a CD; to make the ISO image of any bootable diskette, just type dd (to copy the diskette's contents to a file – floppy.fs in our case); if you download the image of KarmaBSD or SONaFR, you do not have to do this, as you will have the same image you will create with the following command, but in case you first try the floppy, it may also be handy to do it this way):

```
dd if=/dev/fd0a of=floppy.fs
```

Then run:

```
mkisofs -b floppy.fs -c boot.cat -R -v -o /
                        bootableCDfromFloppy.iso .
```

(do not worry if "boot.cat" is not available)

The dot (`.`) at the end of the above mkisofs command is for the current directory, so you must run this command in the (current) directory where the floppy.fs file resides. If floppy.fs is in the `/flp` directory, for example (or use: `dd if=/dev/fd0a of=/flp/floppy.fs`), cd to it (cd `/flp`) and run the mkisofs command as you see above.

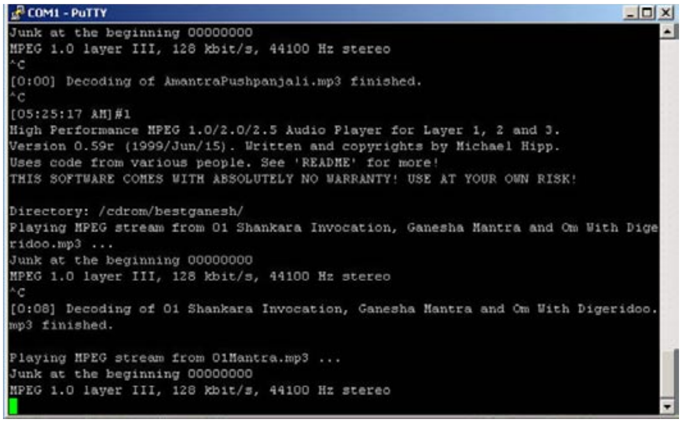
The two above-mentioned one floppy systems can be thus used on a CD. To use multiboot, apply the `-eltorito-alt-boot` option in mkisofs.

How to add packages to KarmaBSD (or SONaFR)

KarmaBSD is reviewed on the following forum: *http://www.daemonforums.org/showthread.php?t=3092*.

The writer of the text in the above forum is interested to know how to add applications into KarmaBSD but does not know how, so I decided to add this information here.

As some people think (as the author of the text in the forum) that it is a complication to install OpenBSD 4.1 today – when the version 4.7 of OpenBSD is already available and the version 4.1 is rather quite old, I decided to publish a solution how packages can be instantly (and easily) transported from any OpenBSD (or FreeBSD) system to any OpenBSD (or FreeBSD, NetBSD, etc.) system. The solution is to compile packages statically. What does it mean?



```
Junk at the beginning 00000000
MPEG 1.0 layer III, 128 kbit/s, 44100 Hz stereo
^C
[0:00] Decoding of AmantraPushpanjali.mp3 finished.
^C
[05:25:17 AM]#1
High Performance MPEG 1.0/2.0/2.5 Audio Player for Layer 1, 2 and 3.
Version 0.59r (1999/Jun/15). Written and copyrights by Michael Hipp.
Uses code from various people. See 'README' for more!
THIS SOFTWARE COMES WITH ABSOLUTELY NO WARRANTY! USE AT YOUR OWN RISK!

Directory: /cdrom/bestganesh/
Playing MPEG stream from 01 Shankara Invocation, Ganesha Mantra and Om With Dige
ridoo.mp3 ...
Junk at the beginning 00000000
MPEG 1.0 layer III, 128 kbit/s, 44100 Hz stereo
^C
[0:08] Decoding of 01 Shankara Invocation, Ganesha Mantra and Om With Digeridoo.
mp3 finished.

Playing MPEG stream from 01Mantra.mp3 ...
Junk at the beginning 00000000
MPEG 1.0 layer III, 128 kbit/s, 44100 Hz stereo
```

Figure 1. *With Windows (running on another computer) and Putty you may connect to any computer (notebook) via a serial cable (serial console with KarmaBSD) and you may thus repair or control any computer*

Ports and the base system

Before you compile packages (from ports), you may decide to add the `--enable-static` option to your makefiles. Without this, the result of compilation (binary application) of packages on OpenBSD 4.4, for example, depends always on this system's libraries. Every compilation is made against these global libraries (OpenBSD 4.7, FreeBSD 8.0, etc). Thus, if you compile packages in OpenBSD 4.7, which has different versions of libraries than OpenBSD 4.6, 4.5, etc., you cannot use these binaries in other versions of OpenBSD.

Static versus dynamic

The concept of the shared (dynamic) libraries (when many programs use certain libraries) started on the premise of saving space. But today, when a 40 GB hard drive is cheaper than ice cream in a very good European restaurant, the philosophy to save a few dozens of megabytes is, on the other hand, a very big barrier to portability of packages (if libraries are embedded in binary packages, you may use such binaries almost in any version of the system they were built for).

Programs (and sources) contained in the base system (fdisk, ifconfig, chmod, etc.) are also dependent on the kernel they are distributed with. If you deal with sources of ifconfig (or ee editor in FreeBSD) in OpenBSD 4.7, for example, you must compile them against the OpenBSD 4.7 kernel.

This is a very important hint, as we must differentiate compilation of sources from 1) the base system (binaries are not easily transportable) and from 2) ports (add-on packages). The `--enable-static` option in your makefiles must only be used in source files that are not kernel-version dependent (`/usr/ports`).

Base system tools like fdisk, mount, ifconfig, etc., are kernel(system)-version dependent (like OpenBSD 4.7) and although they, too, can be compiled statically, using them with a different kernel would bring a number of system failures.

However, packages (in `/usr/ports`), i.e. binaries of ports compiled on OpenBSD 4.5, for example, are only library dependent (they require shared libraries used in OpenBSD 4.5), but not kernel dependent. They cannot be used on other versions of OpenBSD only because they were compiled against libraries that are missing in other versions of OpenBSD. To solve the issue (backward compatibility), you may install the so-called compatibility libraries. But if you work with a small floppy (or CD) system, it is a bit hard to install such compatibility libraries into the floppy (or minimal CD) environment. To solve this, compile packages (in `/usr/ports`) statically.

When you compile your sources this way (statically), the system's libraries get embedded into the resulting binaries. Packages (not the base system) statically compiled on OpenBSD 4.5, for example, will run on any OpenBSD system including KarmaBSD. The binary of mpg123 in KarmaBSD was compiled statically too.

As most floppy systems are space limited, the best way is to make some additional room in memory and then copy such a statically compiled binary there from another floppy.

KarmaBSD and SONaFR run in memory, too, so their floppies, after you boot with them, can be taken out.

To prepare a memory file system in KarmaBSD, type (the number `4917` may vary in dependence of free RAM):

```
mount_mfs -s 4917 -o async,nosuid,nodev,noatime swap /mnt
```

The above two floppy systems can be downloaded here:

- SONaFR – *http://www.freebsd.nfo.sk/opbsd/openbsdeng.htm*
- KarmaBSD – *http://www.freebsd.nfo.sk/opbsd/karmabsdeng.htm*

They both were made with crunchgen. A crunched binary (result of crunchgen) is a program (kernel/system-version dependent) made up of many other programs (like fdisk, mount, chown, ifconfig, etc.) and libraries linked together into a single executable. The crunchgen utility is a tool that will make this binary. It is available on all BSD systems. And because it builds programs like fdisk, disklabel, mount, etc., from the heart of OpenBSD (or FreeBSD, etc.), you need sources of the base system.

My first project was 1FCDBSD – One Floppy MP3 and CD Player (KarmaBSD does not have a player for standard audio CD's). 1FCDBSD is based on FreeBSD 4.5. As I made a very handy multiboot CD with 1FCDBSD, SONaFR, and KarmaBSD, I am also happy that I can provide its download link. I thank www.rootbsd.net for allowing me to distribute my projects:

ftp://2227.x.rootbsd.net/bsd-multiboot.iso

JURAJ SIPOS

Remote Installation of the FreeBSD

Operating System without a Remote Console

This article documents the remote installation of the FreeBSD operating system when the console of the remote system is unavailable. The main idea behind this article is the result of a collaboration with Martin Matuska mm@FreeBSD.org with valuable input provided by Pawel Jakub Dawidek pjd@FreeBSD.org.

What you will learn…

- you should have a network accessible operating system with SSH access
- understand the FreeBSD installation process
- be familiar with the sysinstall(8) utility
- have the FreeBSD installation ISO image or CD handy

What you should know…

- you will know how to remotely install (using SSH) a FreeBSD system from within other operating system, such as Linux. It covers an advanced installation procedure with which one can replace Linux installation with FreeBSD OS.

There are many server hosting providers in the world, but very few of them are officially supporting FreeBSD. They usually provide support for a Linux® distribution to be installed on the servers they offer.

In some cases, these companies will install your preferred Linux distribution if you request it. Using this option, we will attempt to install FreeBSD. In other cases, they may offer a rescue system which would be used in an emergency. It's possible to use this for our purposes as well.

This article covers the basic installation and configuration steps required to bootstrap a remote installation of FreeBSD with RAID-1 and ZFS capabilities.

Introduction

This section will summarize the purpose of this article and better explain what is covered herein. The instructions included in this article will benefit those using services provided by colocation facilities not supporting FreeBSD.

- As we have mentioned in the Background section, many of the reputable server hosting companies provide some kind of rescue system, which is booted from their LAN and accessible over SSH. They usually provide this support in order to help their customers fix broken operating systems. As this article will explain, it is possible to install FreeBSD with the help of these rescue systems.
- The next section of this article will describe how to configure, and build minimalistic FreeBSD on the local machine. That version will eventually be running on the remote machine from a ramdisk, which will allow us to install a complete FreeBSD operating system from an FTP mirror using the sysinstall utility.
- The rest of this article will describe the installation procedure itself, as well as the configuration of the ZFS file system.

Requirements

To continue successfully, you must:

- Have a network accessible operating system with SSH access
- Understand the FreeBSD installation process
- Be familiar with the `sysinstall(8)` utility
- Have the FreeBSD installation ISO image or CD handy

Preparation – mfsBSD

Before FreeBSD may be installed on the target system, it is necessary to build the minimal FreeBSD operating system image which will boot from the hard drive. This way the new system can be accessed from the network, and the rest of the installation can be done without remote access to the system console.

Carry the card that supports BSD events around the world

BSD Fund is proud to sponsor of BSDCan 2010 and meetBSD California 2010 thanks to revenue from the BSD Fund Visa. A donation is made every time you use the card and simply charging your travel to an event can help sponsor that event.

BSD Fund also raises money through direct donations on behalf of BSD projects such as the pcc compiler.

Find our more at www.bsdfund.org

BSD Fund Visa currently available in the USA • BSD Fund is a 501(c)(3) nonprofit organization

The mfsBSD tool-set can be used to build a tiny FreeBSD image. As the name of mfsBSD suggests (*mfs* means *memory file system*), the resulting image runs entirely from a ramdisk. Thanks to this feature, the manipulation of hard drives will not be limited, therefore it will be possible to install a complete FreeBSD operating system. The home page of mfsBSD, at *http://people.freebsd.org/~mm/mfsbsd/* , includes pointers to the latest release of the toolset.

Please note that the internals of mfsBSD and how it all fits together is beyond the scope of this article. The interested reader should consult the original documentation of mfsBSD for more details.

Download and extract the latest mfsBSD release and change your working directory to the directory where the mfsBSD scripts will reside:

```
# fetch http://people.freebsd.org/~mm/mfsbsd/mfsbsd-
                  latest.tar.gz
# tar xvzf mfsbsd-1.0-beta3.tar.gz
# cd mfsbsd-1.0-beta3/
```

Configuration of mfsBSD

Before booting mfsBSD, a few important configuration options have to be set. The most important that we have to get right is, naturally, the network setup. The most suitable method to configure networking options depends on whether we know beforehand the type of the network interface we will use, and the network interface driver to be loaded for our hardware. We will see how mfsBSD can be configured in either case.

Another important thing to set is the root password. This can be done by editing the `conf/rootpw.conf` file. Please keep in mind that the file will contain your password in the plain text, thus we do not recommend to use real password here. Nevertheless, this is just a temporary one-time password which can be later changed in a live system.

The conf/interfaces.conf method

When the installed network interface card is unknown, we can use the auto-detection features of mfsBSD. The startup scripts of mfsBSD can detect the correct driver to use, based on the MAC address of the interface, if we set the following options in `conf/interfaces.conf`:

```
initconf_interfaces="ext1"
initconf_mac_ext1="00:00:00:00:00:00"
initconf_ip_ext1="192.168.0.2"
initconf_netmask_ext1="255.255.255.0"
```

Do not forget to add the defaultrouter information to the `conf/rc.conf` file:

```
defaultrouter="192.168.0.1"
```

The conf/rc.conf method

When the network interface driver is known, it is more convenient to use the `conf/rc.conf` file for networking options. The syntax of this file is the same as the one used in the standard `rc.conf(5)` file of FreeBSD.

For example, if you know that a `re(4)` network interface is going to be available, you can set the following options in `conf/rc.conf`:

```
defaultrouter="192.168.0.1"
ifconfig_re0="inet 192.168.0.2 netmask 255.255.255.0"
```

Building an mfsBSD image

The process of building an mfsBSD image is pretty straightforward.

The first step is to mount the FreeBSD installation CD, or the installation ISO image to `/cdrom`. For the sake of example, in this article we will assume that you have downloaded the FreeBSD 8.0-RELEASE ISO. Mounting this ISO image to the `/cdrom` directory is easy with the `mdconfig(8)` utility:

```
# mdconfig -a -t vnode -u 10 -f 8.0-RELEASE-amd64-disc1.iso
# mount_cd9660 /dev/md10 /cdrom
```

Next, build the bootable mfsBSD image:

```
# make BASE=/cdrom/8.0-RELEASE
```

Note: The above make command has to be run from the top level of the mfsBSD directory tree, i.e. `~/mfsbsd-1.0-beta3/`.

Booting mfsBSD

Now that the mfsBSD image is ready, it must be uploaded to the remote system running a live rescue system or pre-installed Linux® distribution. The most suitable tool for this task is scp:

```
# scp disk.img root@192.168.0.2:.
```

To boot mfsBSD image properly, it must be placed on the first (bootable) device of the given machine. This may be accomplished using this example providing that sda is the first bootable disk device:

```
# dd if=/root/disk.img of=/dev/sda bs=1m
```

If all went well, the image should now be in the MBR of the first device and the machine can be rebooted. Watch for the machine to boot up properly with the `ping(8)` tool. Once it has came back on-line, it should be possible to access it over `ssh(1)` as user root with the configured password.

Installation of The FreeBSD Operating System

The mfsBSD has been successfully booted and it should be possible to log in through ssh(1). This section will describe how to create and label slices, set up gmirror for RAID-1, and how to use sysinstall to install a minimal distribution of the FreeBSD operating system.

Preparation of Hard Drives

The first task is to allocate disk space for FreeBSD, i.e.: to create slices and partitions. Obviously, the currently running system is fully loaded in system memory and therefore there will be no problems with manipulating hard drives. To complete this task, it is possible to use either sysinstall or `fdisk(8)` in conjunction to `bsdlabel(8)`.

At the start, mark all system disks as empty. Repeat the following command for each hard drive:

```
# dd if=/dev/zero of=/dev/ad0 count=2
```

Next, create slices and label them with your preferred tool. While it is considered easier to use sysinstall, a powerful and also probably less buggy method will be to use standard text-based UNIX® tools, such as `fdisk(8)` and `bsdlabel(8)`, which will also be covered in this section. The former option is well documented in the Installing FreeBSD chapter of the FreeBSD Handbook. As it was mentioned in the introduction, this article will present how to set up a system with RAID-1 and ZFS capabilities. Our set up will consist of a small `gmirror(8)` mirrored / (root), `/usr` and `/var` file systems, and the rest of the disk space will be allocated for a `zpool(8)` mirrored ZFS file system. Please note, that the ZFS file system will be configured after the FreeBSD operating system is successfully installed and booted.

The following example will describe how to create slices and labels, initialize `gmirror(8)` on each partition and how to create a UFS2 file system in each mirrored partition:

```
# fdisk -BI /dev/ad0 1
# fdisk -BI /dev/ad1
# bsdlabel -wB /dev/ad0s1 2
# bsdlabel -wB /dev/ad1s1
# bsdlabel -e /dev/ad0s1 3
# bsdlabel /dev/ad0s1 > /tmp/bsdlabel.txt && bsdlabel -R
                    /dev/ad1s1 /tmp/bsdlabel.txt 4
# gmirror label root /dev/ad[01]s1a 5
# gmirror label var /dev/ad[01]s1d
# gmirror label usr /dev/ad[01]s1e
# gmirror label -F swap /dev/ad[01]s1b 6
# newfs /dev/mirror/root 7
# newfs /dev/mirror/var
# newfs /dev/mirror/usr
```

1 Create (*http://www.freebsd.org/doc/en_US.ISO8859-1/articles/remote-install/installation.html#FDISK*) a slice covering the entire disk and initialize the boot code contained in sector 0 of the given disk. Repeat this command for all hard drives in the system.

2 Write (*http://www.freebsd.org/doc/en_US.ISO8859-1/articles/remote-install/installation.html#BSDLABEL-WRITING*) a standard label for each disk including the bootstrap code.

3 Now (*http://www.freebsd.org/doc/en_US.ISO8859-1/articles/remote-install/installation.html#BSDLABEL-EDITING*), manually edit the label of the given disk. Refer to the `bsdlabel(8)` (*http://www.freebsd.org/cgi/man.cgi?query=bsdlabel&sektion=8*) manual page in order to find out how to create partitions. Create partitions a for `/ (root)` file system, b for swap, d for `/var`, e for `/usr` and finally f which will later be used for ZFS.

4 Import (*http://www.freebsd.org/doc/en_US.ISO8859-1/articles/remote-install/installation.html#BSDLABEL-RESTORE*) the recently created label for the second hard drive, so both hard drives will be labeled in the same way.

5 Initialize (*http://www.freebsd.org/doc/en_US.ISO8859-1/articles/remote-install/installation.html#GMIRROR1*) `gmirror(8)` (*http://www.freebsd.org/cgi/man.cgi?query=gmirror&sektion=8*) on each partition.

6 Note (*http://www.freebsd.org/doc/en_US.ISO8859-1/articles/remote-install/installation.html#GMIRROR2*) the -F option used for swap partition. This instructs `gmirror(8)` (*http://www.freebsd.org/cgi/man.cgi?query=gmirror&sektion=8*) to assume that the device is in the consistent state after the power/system failure.

7 Create (*http://www.freebsd.org/doc/en_US.ISO8859-1/articles/remote-install/installation.html#NEWFS*) a UFS2 file system on each mirrored partition.

System Installation

This is the most important part. This section will describe how to actually install the minimal distribution of FreeBSD on the hard drives that we have prepared in the previous section. To accomplish this goal, all file systems need to be mounted so sysinstall may write the contents of FreeBSD to the hard drives:

```
# mount /dev/mirror/root /mnt
# mkdir /mnt/var /mnt/usr
# mount /dev/mirror/var /mnt/var
# mount /dev/mirror/usr /mnt/usr
```

When you are done, start `sysinstall(8)`. Select the Custom installation from the main menu. Select Options and press Enter. With the help of arrow keys, move the cursor on the Install Root item, press Space and change it to /mnt. Press Enter to submit your changes and exit the Options menu by pressing q.

Warning: Note that this step is very important and if skipped, sysinstall will be unable to install FreeBSD.

Go to the Distributions menu, move the cursor with the arrow keys on the Minimal option, and check it by pressing Space. This article uses the Minimal distribution in order to save network traffic, because the system itself will be installed over ftp. Exit this menu by choosing Exit option.

Note: The Partition and Label menus will be skipped, as these are useless now.

In the Media menu, select FTP. Select the nearest mirror and let sysinstall assume that the network is already configured. You will be returned back to the Custom menu. Finally, perform the system installation by selecting the last option, Commit. Exit sysinstall when it finishes the installation.

Post Installation Steps

The FreeBSD operating system should be installed now; however, the process is not finished yet. It is necessary to perform some post installation steps in order to allow FreeBSD to boot in the future and to be able to log in to the system. You must now `chroot(8)` into the freshly installed system in order to finish the installation. Use the following command:

```
# chroot /mnt
```

To complete our goal, perform these steps:

- Copy the GENERIC kernel to the `/boot/kernel` directory:

  ```
  # cp -Rp /boot/GENERIC/* /boot/kernel
  ```

- Create the `/etc/rc.conf`, `/etc/resolv.conf` and `/etc/fstab` files. Do not forget to properly set the network information and to enable sshd in the `/etc/rc.conf` file. The contents of the `/etc/fstab` file will be similar to the following:

```
# Device          Mountpoint FStype Options     Dump    Pass#
/dev/mirror/swap none         swap    sw           0        0
/dev/mirror/root /            ufs     rw           1        1
/dev/mirror/usr  /usr         ufs     rw           2        2
/dev/mirror/var  /var         ufs     rw           2        2
/dev/cd0         /cdrom       cd9660   ro,noauto   0        0
```

- Create the `/boot/loader.conf` file, with the following contents:

  ```
  geom_mirror_load="YES"
  zfs_load="YES"
  ```

- Perform the following command, which will make ZFS available on the next boot:

```
# echo 'zfs_enable="YES"' >> /etc/rc.conf
```

- Add additional users to the system using the `adduser(8)` tool. Do not forget to add a user to the wheel group so you may obtain root access after the reboot.
- Double-check all your settings.

The system should now be ready for the next boot. Use the `reboot(8)` command to reboot your system.

ZFS

If your system survived the reboot, it should now be possible to log in. Welcome to the fresh FreeBSD installation, performed remotely without the use of a remote console!

The only remaining step is to configure `zpool(8)` and create some `zfs(8)` file systems. Creating and administering ZFS is very straightforward. First, create a mirrored pool:

```
# zpool create tank mirror /dev/ad[01]s1f
```

Next, create some file systems:

```
# zfs create tank/ports
# zfs create tank/src
# zfs set compression=gzip tank/ports
# zfs set compression=on tank/src
# zfs set mountpoint=/usr/ports tank/ports
# zfs set mountpoint=/usr/src tank/src
```

That's all. If you are interested in more details about ZFS on FreeBSD, please refer to the ZFS (*http://wiki.freebsd.org/ZFS*) section of the FreeBSD Wiki.

DANIEL GERZO

I am a FreeBSD user and enthusiast since around 2003. I received a documentation commit bit in 2006 which makes me a FreeBSD developer as well. I am living in central Europe, the capital of Slovakia – Bratislava. I currently study Computer science at the Slovak University of Technology. I also own a small company providing consultancy, administration, and other IT services for FreeBSD servers.

11th Libre Software Meeting

Free Admission

RMLL.info

http://rmll.info

Bordeaux - Pessac - Talence

July 6th - 11th , 2010

Tuesday 6 to Friday 9: Enseirb-Matmeca and Université Bordeaux 1
Saturday 10 and Sunday 11: Central Bordeaux
And also lots of events all around Bordeaux metropolitan area and Aquitaine

Accessibility - Sustainable development - Administration and public policies - Corporate - Technology - Science and education - Free art and culture

MANIFESTATION RESPONSABLE
Gironde CONSEIL GENERAL

Live on...
Radio - RMLL
http://radio.rmll.info

ABUL.ORG AFUL Scideralle
April promouvoir et défendre le logiciel libre
GIROLL
gandi.net CEMEA ADEISO
CNLL Conseil National du Logiciel Libre
AGENCE
LANDINUX
OpenEirb
aliénor.net FDN médias-cité

Gironde CONSEIL GENERAL gironde.fr
REGION AQUITAINE
LACUB
Pessac
MACIF
BORDEAUX
Crédit Agricole Aquitaine
Mairie Talence
UNIVERSITÉ BORDEAUX 1
IPB ENSCBP BORDEAUX
IPB ENSEIRB MATMECA BORDEAUX
ville de gradignan
INRIA
LaBRI Laboratoire Bordelais de Recherche en Informatique

OpenBSD

as a Mail Server

In a previous document, we built redundant firewalls using the CARP and PFSYNC protocols; these were the first building blocks of a hypothetical, OpenBSD-based, small private networkthat we are going to build step by step across several documents.

What you will learn…

- Installing a full-featured mail server
- Basic mail server security

What you should know…

- A good knowledge of OpenBSD administration
- Basic MySQL database administration

Now that we have raised the *defensive walls* of our network, it's time to think about the services we want to provide. Offering a reliable and secure email service is probably one of the top priorities of most system administrators; therefore, in the next chapters, we will build a full-featured mail server, based on open-source software and focusing on security. The following is the list of the pieces of software we will use:

OpenBSD

http://www.openbsd.org/ – the *secure by default* operating system, with *only two remote holes in the default install, in a heck of a long time!*;

Postfix

http://www.postfix.org/ – an MTA *that started life at IBM research as an alternative to the widely-used Sendmail* (*http://www.sendmail.org/*) *program* and which *attempts to be fast, easy to administer, and secure*;

MySQL

http://www.mysql.com/ – the *world's most popular open source database*;

Courier-IMAP

http://www.courier-mta.org/imap/ – a *fast, scalable, enterprise IMAP server* that supports MySQL and maildirs;

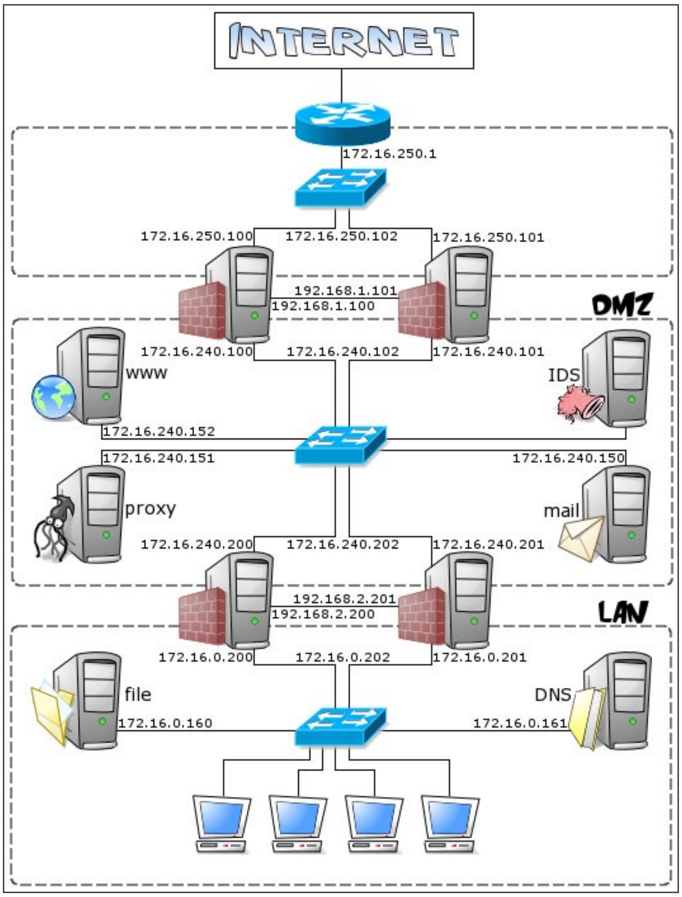


Figure 1. *Network layout*

Cyrus SASL

http://asg.web.cmu.edu/sasl/ – the Cyrus (*http://cyrusimap.web.cmu.edu/*) implementation of the SASL (*http://asg.web.cmu.edu/sasl/*) protocol;

Amavisd-new

http://www.ijs.si/software/amavisd/ – a *high-performance interface between mailer (MTA) and content checkers* (antivirus and antispam), written in Perl and optimized for Postfix;

SpamAssassin

http://spamassassin.apache.org/ – a Perl-based *mail filter to identify Spam*, using *a variety of mechanisms including header and text analysis, Bayesian filtering, DNS blocklists, and collaborative filtering databases*;

ClamAV

http://www.clamav.net/ – a fast and easy-to-use open-source virus scanner.

A good knowledge of OpenBSD is assumed, since we won't delve into system management topics such as base configuration (*http://www.openbsd.org/cgi-bin/man.cgi?query=afterboot&sektion=8*) or packages/ports (*http://www.openbsd.org/faq/faq15.html*) installation.

Preliminary installation steps

Before delving into the installation and configuration of all the mail-handling software, we will take a brief look at the operating system that will host it.

As usual, my choice goes to OpenBSD for its proven security, reliability and ease of use. Needless to say, all these features are essential for a system that will have to handle a large volume of email traffic while still making life hard for spammers and malicious users.

We won't dwell upon the installation procedure here, which is documented in full detail on the OpenBSD web site (*http://www.openbsd.org/faq/faq4.html*). Just a couple of notes:

- while partitioning the hard drive, bear in mind that we will configure Postfix to use virtual domains (*http://www.postfix.org/VIRTUAL_README.html*) and, consequently, it will store all users' mail folders in a single directory (`/var/vmail`). Therefore, it is recommended to assign a (large) dedicated slice to this filesystem, in order to prevent mails from filling up any critical filesystem, should quotas fail. Furthermore, if you choose to install MySQL on the mail server itself, it is usually recommended to assign one of the first slices to `/var/mysql`, in order to allow for faster disk access by the database engine;
- the only file sets we will need to install are those marked as *required* on the documentation (*http://www.openbsd.org/faq/faq4.html#FilesNeeded*), i.e. bsd (the kernel), `baseXX.tgz` (the base system), and `etcXX.tgz` (the configuration files in `/etc`) plus `compXX.tgz` (the C compiler), since we will also have to install some ports (*http://www.openbsd.org/ports.html*) not available as precompiled packages for licensing reasons.

Note: since leaving a compiler on a publicly accessible server is a definite security risk, it is recommended that you remove the compiler when the installation is over or that you compile on another machine.

After the first reboot, we can disable some default network services managed by inetd(8) (*http://www.openbsd.org/cgi-bin/man.cgi?query=inetd&sektion=8*): see Listing 1.

by commenting them out in `/etc/inetd.conf` (*http://www.openbsdorg.com/cgi-bin/man.cgi?query=inetd&sektion=8*) and reloading `inetd(8)` (*http://www.openbsdorg.com/cgi-bin/man.cgi?query=inetd&sektion=8*):

```
# pkill -HUP inetd
```

Listing 1. *Default services on OpenBSD*

```
$ grep -v ^# /etc/inetd.conf
ident           stream  tcp     nowait  _identd /usr/
                   libexec/identd     identd -el
ident           stream  tcp6    nowait  _identd /usr/
                   libexec/identd     identd -el
127.0.0.1:comsat dgram  udp     wait    root    /usr/
                   libexec/comsat     comsat
[::1]:comsat    dgram   udp6    wait    root    /usr/
                   libexec/comsat     comsat
daytime         stream  tcp     nowait  root
                   internal
daytime         stream  tcp6    nowait  root
                   internal
time            stream  tcp     nowait  root
                   internal
time            stream  tcp6    nowait  root
                   internal
$
```

Listing 2. *Postfix configuration*

```
/etc/postfix/main.cf
# Directory containing all the post* commands
command_directory = /usr/local/sbin
# Directory containing all the Postfix daemon programs
daemon_directory = /usr/local/libexec/postfix
# Full pathnames of various Postfix commands
sendmail_path = /usr/local/sbin/sendmail
newaliases_path = /usr/local/sbin/newaliases
mailq_path = /usr/local/sbin/mailq
# Directories containing documentation
html_directory = /usr/local/share/doc/postfix/html
manpage_directory = /usr/local/man
readme_directory = /usr/local/share/doc/postfix/readme
# The owner of the Postfix queue and of most Postfix
                    daemon processes
mail_owner = _postfix
# The group for mail submission and queue management
                    commands
setgid_group = _postdrop
# The myhostname parameter specifies the internet
                    hostname of this mail system. It
# is used as default for many other configuration
                    parameters (default = system's
# FQDN)
myhostname = mail.kernel-panic.it
# The internet domain name of this mail system. Used
                    as default for many other
# configuration parameters (default = $myhostname minus
                    the first component)
mydomain = kernel-panic.it
# The domain name that locally-posted mail appears to
                    come from, and that locally
# posted mail is delivered to. As you can see,
                    a parameter value may refer to other
# parameters
myorigin = $myhostname
# Network interface addresses that this mail system
                    receives mail on
inet_interfaces = all
# Network interface addresses that this mail system
                    receives mail on by way of a
# proxy or NAT unit
proxy_interfaces = router.kernel-panic.it
# List of domains that this machine considers itself
                    the final destination for.
# Virtual domains must not be specified here
mydestination = $myhostname, localhost.$mydomain, localhost
```

```
# List of "trusted" SMTP clients allowed to relay mail
                    through Postfix.
mynetworks = 127.0.0.0/8, 172.16.0.0/24, 172.16.240.0/24
# What destination (sub)domains this system will relay
                    mail to
relay_domains = $mydestination
# The default host to send mail to when no entry is
                    matched in the optional
# transport(5) table. Square brackets turn off MX lookups
relayhost = [smtp.isp.com]
# List of alias databases used by the local delivery
                    agent
alias_maps = hash:/etc/postfix/aliases
# Alias database(s) built with "newaliases" or
                    "sendmail -bi". This is a separate
# configuration parameter, because alias_maps may
                    specify tables that are not
# necessarily all under control by Postfix
alias_database = hash:/etc/postfix/aliases
# SMTP greeting banner
smtpd_banner = $myhostname ESMTP $mail_name
# Postfix is final destination for the specified list of
                    "virtual" domains
virtual_mailbox_domains = kernel-panic.it
# Virtual mailboxes base directory
virtual_mailbox_base = /var/vmail
# Optional lookup tables with all valid addresses in
                    the domains that match
# $virtual_mailbox_domains.
virtual_mailbox_maps = hash:/etc/postfix/vmailbox
# The minimum user ID value accepted by the virtual(8)
                    delivery agent
virtual_minimum_uid = 2000
# User ID that the virtual(8) delivery agent uses
                    while writing to the recipient's
# mailbox
virtual_uid_maps = static:2000
# Group ID that the virtual(8) delivery agent uses
                    while writing to the recipient's
# mailbox
virtual_gid_maps = static:2000

# Optional lookup tables that alias specific mail
                    addresses or domains to other
# local or remote address
virtual_alias_maps = hash:/etc/postfix/virtual
```

Anyway, OpenBSD is considered secure also with those services turned on and the mail server should be placed behind a firewall; nevertheless, I prefer staying on the safe side and disable them all (including `comsat(8)` (*http://www.openbsdorg.com/cgi-bin/man.cgi?query=comsat&sektion=8*), since we won't have any interactive user receiving mail on the system).

To modify the server network configuration, please refer to the related chapter (*http://www.kernel-panic.it/openbsd/carp/carp3.html*) in the previous document about redundant firewalls (*http://www.kernel-panic.it/openbsd/carp/*) or to the networking (*http://www.openbsd.org/faq/faq6.html*) FAQ.

Postfix

Postfix (*http://www.postfix.org/*) is a MTA (*Mail Transport Agent*) developed by Wietse Venema (*http://www.porcupine.org/wietse/*) *as an alternative to the widely-used Sendmail (http://www.sendmail.org/) program*. It *attempts to be fast, easy to administer, and secure, while at the same time being sendmail compatible enough to not upset existing users. Thus, the outside has a sendmail-ish flavor, but the inside is completely different*. Postfix also comes with excellent documentation (*http://www.postfix.org/documentation.html*) and a lot of howtos (*http://www.postfix.org/docs.html*).

Our mail server requirements will be quite simple: it will be final destination solely for its canonical domains (*http://www.postfix.org/docs.html*) and it will only relay mail from systems on the internal network (though we will also consider relaying from untrusted networks by means of SMTP authentication). Canonical domains include the hostname (in our case, `mail.kernel-panic.it`) and the IP address (172.16.240.150) of the machine that Postfix runs on, and the parent domain of the hostname (`kernel-panic.it`).

Canonical domains are usually implemented with the Postfix local domain address class (*http://www.postfix.org/ADDRESS_CLASS_README.html#local_domain_class*), which, unfortunately, has one major drawback for me: it requires that each e-mail account have a corresponding Unix account. On the contrary, I prefer:

1. keeping Unix and e-mail accounts apart and
2. having all mailboxes well-ordered inside a single directory.

Therefore, we will use *Postfix Virtual Domain Hosting* (*http://www.postfix.org/VIRTUAL_README.html*), which is normally used for hosting multiple internet domains on the same server, but will also allow us to achieve our two goals.

Configuration

In this paragraph, we will configure Postfix to work standalone, with no back-end database. Then, in the next chapter, when everything will be working fine, we will hook up Postfix to a MySQL database; this will allow us to centrally store configuration information that both Postfix and Courier-IMAP will need to access.

There are a few packages we need to install:

- `mysql-client-x.x.x.tgz`
- `pcre-x.x.tgz`
- `postfix-x.x.x-mysql.tgz`

Note: if you're planning to use SMTP authentication, you will need to compile Postfix from the ports, because there's no pre-compiled package available with both MySQL and SASL support:

Listing 3. *Replacing Sendmail with Postfix*

```
# /usr/local/sbin/postfix-enable
old /etc/mailer.conf saved as /etc/mailer.conf.pre-
                    postfix
postfix /etc/mailer.conf enabled

NOTE: do not forget to add sendmail_flags="-bd" to
      /etc/rc.conf.local to startup postfix correctly.

NOTE: do not forget to add "-a /var/spool/postfix/dev/
                    log" to
      syslogd_flags in /etc/rc.conf.local and restart
                    syslogd.

NOTE: do not forget to remove the "sendmail
                    clientmqueue runner"
      from root's crontab.
#
```

Listing 4. *Postfix startup commands*

```
/etc/rc.conf.local
# Specify a location where syslogd(8) should place an
                    additional log socket
# for Postfix
syslogd_flags="-a /var/spool/postfix/dev/log"

# Make Postfix start in background and process queued
                    messages every 30 min
sendmail_flags="-bd"
```

```
# cd /usr/ports/mail/postfix/snapshot
# env FLAVOR="mysql sasl2" make install
```

The installation will create the `/etc/postfix` directory, containing all the configuration files. Postfix has several hundred configuration parameters that are controlled via the `/etc/postfix/main.cf` file, but don't worry: for the vast majority of these parameters, the default value is the best option (see `postconf(5)` (*http://www.postfix.org/postconf.5.html*) for a detailed list of all the available configuration parameters, their description and their default value) and we will only have to override a very small subset of them: see Listing 2.

Listing 5. *Testing the basic functionality*

```
# chgrp _postdrop /usr/local/sbin/postqueue /usr/
                    local/sbin/postdrop
# chmod 2755 /usr/local/sbin/postqueue /usr/local/
                    sbin/postdrop
# pkill syslogd
# syslogd -a /var/empty/dev/log -a /var/spool/postfix/
                    dev/log
# pkill sendmail
# /usr/local/sbin/sendmail -bd
postfix/postfix-script: starting the Postfix mail system
and test our hard work!
# telnet mail.kernel-panic.it 25
Trying 172.16.240.150...
Connected to mail.kernel-panic.it.
Escape character is '^]'.
220 mail.kernel-panic.it ESMTP Postfix
HELO somedomain.org
250 mail.kernel-panic.it
mail from: someone@somedomain.org
250 Ok
rcpt to: d.mazzocchio@kernel-panic.it
250 Ok
data
354 End data with <CR><LF>.<CR><LF>
From: someone@somedomain.org
To: d.mazzocchio@kernel-panic.it
Subject: Test mail

It works!
.
250 Ok: queued as 548D7286
quit
221 Bye
Connection closed by foreign host.
# tail /var/log/maillog
Dec 16 10 15:26:35 mail postfix/smtpd[29212]:
                    connect from ws1.lan.kernel-
                    panic.it[172.16.0.15]
Dec 16 15:26:53 mail postfix/smtpd[29212]: 57076222:
                    client=ws1.lan.kernel-
                    panic.it[172.16.0.15]
Dec 16 15:27:02 mail postfix/cleanup[13428]: 57076222:
                    message-id=<20070210142653.5707622
                    2@mail.kernel-panic.it>
Dec 16 15:27:02 mail postfix/qmgr[26776]: 57076222:
                    from=<someone@somedomain.org>,
                    size=392, nrcpt=1 (queue active)
Dec 16 15:27:02 mail postfix/virtual[14381]: 57076222:
                    to=<d.mazzocchio@kernel-panic.it>,
                    relay=virtual, delay=15,
                    delays=15/0.28/0/0.03, dsn=2.0.0,
                    status=sent (delivered to maildir)
Dec 16 15:27:02 mail postfix/qmgr[26776]: 57076222:
                    removed
Dec 16 15:27:06 mail postfix/smtpd[29212]:
                    disconnect from ws1.lan.kernel-
                    panic.it[172.16.0.15]
# cat /var/vmail/kernel-panic.it/d.mazzocchio/new/1118
                    146014.V3I9448M811660.mail
.kernel-panic.it
Return-Path: <someone@somedomain.org>
X-Original-To: d.mazzocchio@kernel-panic.it
Delivered-To: d.mazzocchio@kernel-panic.it
Received: from somedomain.org (ws1.lan.kernel-panic.it
                    [172.16.0.15])
        by mail.kernel-panic.it (Postfix) with SMTP id
                    57076222
        for <d.mazzocchio@kernel-panic.it> Sat, 16 Dec
                    2007 15:26:47 +0100 (CET)
From: someone@somedomain.org
To: d.mazzocchio@kernel-panic.it
Subject: Test mail
Message-Id: <20070210142653.57076222@mail.kernel-
                    panic.it>
Date: Sat, 16 Dec 2007 15:26:47 +0100 (CET)

It works!
#
```

Let's take a closer look at some of the above configuration parameters.

One of the goals we had was to avoid having a separate Unix account for each e-mail account. We have achieved this by configuring Postfix to write to the mailboxes using uid 2000 and gid 2000 (see the `virtual_uid_maps` and `virtual_gid_maps` parameters above). Now we only have to create a user with this pair of uid and gid:

```
# useradd -d /var/vmail -g =uid -u 2000 -s /sbin/nologin \
> -c "Virtual Mailboxes Owner" -m vmail
```

Our second goal was having all mailboxes grouped together in a single directory; this is achieved by setting the value of the `virtual_mailbox_base` parameter to the path of that directory (in our configuration, `/var/vmail`). In matter of fact, this parameter is a prefix that the `virtual(8)` (*http://www.postfix.org/virtual.8.html*) agent prepends to all pathname results from `virtual_mailbox_maps` table lookups.

In our configuration, the `virtual_mailbox_maps` parameter refers to the `/etc/postfix/vmailbox` file, containing the list of all valid addresses in the virtual domains (`virtual_mailbox_domains` parameter) and the path to the corresponding mailboxes or maildirs (a mailbox is a single file containing all the emails; a maildir (*http://www.qmail.org/man/man5/maildir.html*), instead, is a directory, with a defined structure, containing all the emails in separate files):

```
/etc/postfix/vmailbox
info@kernel-panic.it          kernel-panic.it/info/
d.mazzocchio@kernel-panic.it  kernel-panic.it/
                    d.mazzocchio/
[...]
```

Please pay attention to the trailing slashes: they tell Postfix that the pathname refers to a maildir instead of a mailbox file, and maildirs are our only option, since Courier-IMAP doesn't support mailbox files.

The `virtual_alias_maps` parameter allows to alias specific mail addresses or domains to other local or remote address. Its value is the pathname to a file (in our case `/etc/postfix/virtual`) containing the alias mappings:

```
/etc/postfix/virtual
root@kernel-panic.it            root@localhost.kernel-
                    panic.it
postmaster@kernel-panic.it      postmaster@localhost.kerne
                    l-panic.it
abuse@kernel-panic.it           postmaster@localhost.kerne
                    l-panic.it
[...]
```

Finally, the `/etc/postfix/aliases` file contains the addresses to which Postfix will redirect mail for local recipients (see `aliases(5)` *http://www.postfix.org/aliases.5.html*). Since many accounts point to root's email address, you should check root email frequently or forward it all to another account. E.g.:

```
/etc/postfix/aliases
root: d.mazzocchio@kernel-panic.it
MAILER-DAEMON: postmaster
postmaster: root
```

Listing 6. *MySQL installation and configuration*

```
# /usr/local/bin/mysql_install_db
[ ... ]
# mysqld_safe &
[ ... ]
# /usr/local/bin/mysql_secure_installation
[ ... ]
Enter current password for root (enter for none):
                    <Enter>
OK, successfully used password, moving on...
[ ... ]
Set root password? [Y/n] Y
New password: root
Re-enter new password: root
Password updated successfully!
[ ... ]
Remove anonymous users? [Y/n] Y
 ... Success!
[ ... ]
Disallow root login remotely? [Y/n] Y
 ... Success!
[ ... ]
Remove test database and access to it? [Y/n] Y
 - Dropping test database...
 ... Success!
 - Removing privileges on test database...
 ... Success!
[ ... ]
Reload privilege tables now? [Y/n] Y
 ... Success!
[ ... ]
#
```

```
bin: root
[...]
```

Now we only have to reload Postfix lookup tables:

```
# /usr/local/sbin/postmap /etc/postfix/vmailbox
# /usr/local/sbin/postmap /etc/postfix/virtual
# /usr/local/sbin/newaliases
```

replace Sendmail: see Listing 3.

and follow the above advice, by commenting out the *sendmail clientmqueue runner* in root's crontab:

```
# sendmail clientmqueue runner
#*/30 * * * * /usr/sbin/sendmail -L sm-msp-queue -Ac -q
```

and adding a couple of variables in the `/etc/rc.conf.local(8)` (*http://www.openbsd.org/cgi-bin/man.cgi?query=rc.conf.local&sektion=8*) file see Listing 4.

Now we can change a few permissions and restart the processes (or simply reboot): see Listing 5.

MySQL

If Postfix is working fine, we can proceed to the next step and install MySQL. MySQL is *the world's most popular open source database*, combining performance, reliability and ease of use. It will ensure faster data access times and allow us to centralize configuration information that both Postfix and Courier-IMAP will need to access.

There are a few packages we need to install:

Listing 7. *Creating the database*

```
# mysql -u root -p
password: root
mysql> CREATE DATABASE mail;
Query OK, 1 row affected (0.01 sec)

mysql> use mail
Database changed
mysql> CREATE TABLE domains (
    ->        id        INT NOT NULL PRIMARY KEY AUTO_
                   INCREMENT,
    ->        domain    VARCHAR(255) NOT NULL UNIQUE);
Query OK, 0 rows affected (0.02 sec)

mysql> CREATE TABLE users (
    ->       id        INT NOT NULL PRIMARY KEY AUTO_
                   INCREMENT,
    ->        login     VARCHAR(255) NOT NULL UNIQUE,
    ->        name      VARCHAR(255) NOT NULL,
    ->        password CHAR(13) NOT NULL,
    ->        uid       SMALLINT NOT NULL DEFAULT 2000,
    ->        gid       SMALLINT NOT NULL DEFAULT 2000,
    ->        home      VARCHAR(255) NOT NULL DEFAULT
                   '/var/vmail',
    ->        maildir   VARCHAR(255) NOT NULL,
    ->        quota     VARCHAR(10)  NOT NULL DEFAULT
                   '10000000S');
Query OK, 0 rows affected (0.01 sec)

mysql> CREATE TABLE alias_maps (
    ->        id        INT NOT NULL PRIMARY KEY AUTO_
                   INCREMENT,
    ->        account  VARCHAR(255) NOT NULL UNIQUE,
    ->        alias    VARCHAR(255) NOT NULL);
Query OK, 0 rows affected (0.00 sec)

mysql> GRANT SELECT ON mail.* to 'vmail'@'localhost'
                   IDENTIFIED BY 'vmail';
Query OK, 0 rows affected (0.01 sec)

mysql> INSERT INTO domains (domain) VALUES ('kernel-
                   panic.it');
Query OK, 1 row affected (0.01 sec)

mysql> INSERT INTO users (login, name, password,
                   maildir)
    -> VALUES ('d.mazzocchio@kernel-panic.it',
                   'Daniele Mazzocchio',
    -> ENCRYPT('danix'), 'kernel-panic.it/
                   d.mazzocchio/');
Query OK, 1 row affected (0.01 sec)

mysql> INSERT INTO alias_maps (account, alias)
    -> VALUES ('postmaster@kernel-panic.it',
    ->         'postmaster@localhost.kernel-
                   panic.it');
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO alias_maps (account, alias)
    -> VALUES ('root@kernel-panic.it',
                   'root@localhost.kernel-panic.it');
Query OK, 1 row affected (0.00 sec)
```

- `p5-Net-Daemon-x.xx.tgz`
- `p5-PlRPC-x.xxxx.tgz`
- `p5-DBI-x.xx.tgz`
- `p5-DBD-mysql-x.xxxx.tgz`
- `mysql-server-x.x.xx.tgz`

After the installation, you will find various sample configuration files in the `/usr/local/share/mysql` directory; choose the most suitable to your needs and copy it to /etc/my.cnf. E.g.:

```
# cp /usr/local/share/mysql/my-small.cnf /etc/my.cnf
```

The socket dilemma

Choosing a good location for the MySQL socket file is sometimes hard because of chrooted processes, which need to access it from inside their *reduced* filesystem. But Postfix goes even further: of its many processes, most are chrooted to the `/var/spool/postfix` directory, but a few are not! As a consequence, by default, part of the Postfix processes will look for the socket file in the `/var/run/mysql/` directory, while the others will look for it in the `/var/spool/postfix/var/run/mysql/` directory!

Anyway, there are many possible workarounds:

- if the database runs on a remote server, there is no need to bother with the socket file! We will later see how to configure Postfix and Courier-IMAP for connecting to a remote database;
- if you want to preserve the defaults as much as possible, you can create a symbolic link to the socket inside the chroot before the database startup:

```
# mkdir -p /var/spool/postfix/var/run/mysql/
# ln -f /var/run/mysql/mysql.sock /var/spool/postfix/var/
                    run/mysql/mysql.sock
```

- Remember to add the above commands to the `/etc/rc.local(8)` (*http://www.openbsd.org/cgi-bin/man.cgi?query=rc&sektion=8*) script to automatically create the link at boot time.
- you can place the socket inside the Postfix chroot (by setting the value of the socket variable in the [mysqld] section of `/etc/my.cnf` to the path of the socket, e.g. `/var/spool/postfix/mysql/mysql.sock`), and give Postfix the possibility to choose between two distinct paths: `/var/spool/postfix/mysql/mysql.sock`, for non-chrooted processes, and `/mysql/mysql.sock`, for chrooted processes;
- finally, you can forget about socket files and connect through the loopback network interface.

Listing 8. *Additional Postfix configuration for MySQL*

```
/etc/postfix/mysql_virtual_domains.cf
user = vmail
password = vmail

# solution 1:
# hosts = db_server_name
# Solution 2: skip this parameter
# Solution 3 (this file is required only by chrooted
                    processes):
# hosts = unix:/mysql/mysql.sock
# Solution 4:
hosts = 127.0.0.1

dbname = mail
query = SELECT domain FROM domains WHERE domain='%s'
/etc/postfix/mysql_virtual_alias_maps.cf
user = vmail
password = vmail

# solution 1:
# hosts = db_server_name
# Solution 2: skip this parameter
# Solution 3 (this file is required only by chrooted
                    processes):
# hosts = unix:/mysql/mysql.sock
# Solution 4:
hosts = 127.0.0.1

dbname = mail
query = SELECT alias FROM alias_maps WHERE
                    account='%s'
/etc/postfix/mysql_virtual_mailboxes.cf
user = vmail
password = vmail

# solution 1:
# hosts = db_server_name
# Solution 2: skip this parameter
# Solution 3 (this file is required by both chrooted
                    and non-chrooted processes):
# hosts = unix:/mysql/mysql.sock unix:/var/spool/
                    postfix/mysql/mysql.sock
# Solution 4:
hosts = 127.0.0.1

dbname = mail
query = SELECT maildir FROM users WHERE login='%s'
```

Listing 9. *Configuring MySQL-based authentication for Courier*

```
/etc/courier/authmysqlrc
MYSQL_SERVER             127.0.0.1
MYSQL_USERNAME           vmail
MYSQL_PASSWORD           vmail
# If you connect through the socket:
#MYSQL_SOCKET            /path/to/mysql.sock
#MYSQL_PORT              0
MYSQL_PORT               3306
MYSQL_OPT                0
MYSQL_DATABASE           mail
MYSQL_USER_TABLE         users
MYSQL_CRYPT_PWFIELD      password
MYSQL_DEFAULT_DOMAIN     kernel-panic.it
MYSQL_UID_FIELD          uid
MYSQL_GID_FIELD          gid
MYSQL_LOGIN_FIELD        login
MYSQL_HOME_FIELD         home
MYSQL_NAME_FIELD         name
MYSQL_MAILDIR_FIELD      maildir
MYSQL_QUOTA_FIELD        quota
# MYSQL_WHERE_CLAUSE     field=value AND field=value...
```

Mmmh... what to choose? After a moment's thought, I chose the latter solution, which is probably the simplest. Therefore, I left `skip networking` commented out in `/etc/my.cnf` and added the following line in the [`mysqld`] section :

```
/etc/my.cnf
bind-address = 127.0.0.1
```

thus preventing MySQL from listening on the external network interfaces.

Configuration

First and foremost, we need to install the default databases, change the password of the MySQL root user (don't take my passwords as an example!): see Listing 6, and configure the system to start MySQL on boot:

```
/etc/rc.local
if [ -x /usr/local/bin/mysqld_safe ]; then
    echo -n ' MySQL'
    /usr/local/bin/mysqld_safe >/dev/null 2>&1 &
fi
```

Next, we will hook Postfix up to the database. In particular, we will modify the value of a few parameters in the `/etc/postfix/main.cf` file:

```
/etc/postfix/main.cf
virtual_mailbox_domains = mysql:/etc/postfix/mysql_virtual_
                    domains.cf
virtual_mailbox_maps = mysql:/etc/postfix/mysql_virtual_
                    mailboxes.cf
virtual_alias_maps = mysql:/etc/postfix/mysql_virtual_
                    alias_maps.cf
```

We will see in a moment the contents of those files; but first, we are going to create the database. Tables don't need to have any particular structure, since we will tell Postfix which queries to use to extract the data. Therefore, this will actually be just one among the many

Listing 10. *Testing IMAP functionality*

```
IMAP_test.py
#!/usr/bin/env python

import imaplib

# Constants
IMAP_SRV = "mail.kernel-panic.it"
USER     = "d.mazzocchio@kernel-panic.it"
PASSWD   = "danix"

# Connect to server
imap_srv = imaplib.IMAP4(IMAP_SRV)
imap_srv.login(USER, PASSWD)

# Select the INBOX folder
imap_srv.select()

# Retrieve message list
msg_nums = imap_srv.search(None, 'ALL')[1]

# Print all messages
for num in msg_nums[0].split():
    msg = imap_srv.fetch(num, '(RFC822)')[1]
    print 'Message %s\n%s\n' % (num, msg[0][1])

# Disconnect from server
imap_srv.close()
imap_srv.logout()
```

meetBSD
2010 Conference

Starting from May 1st, 2010 users and developers of UNIX systems (in particular the BSD family) can register for the 7th edition of the meetBSD Conference. This time the event will take place in a beautiful city of Krakow, Poland, on July 2nd and 3rd, 2010 (Friday and Saturday). Similar to previous editions, conference speakers will include people actively involved in BSD development, with international audience and guests, including representatives from companies cooperating with the Open Source community. During this year's edition it will be possible for participants to take the BSD Certification Group exam and obtain the well recognized and valued certificate.

meetBSD is an annual event dedicated to UNIX-based operating systems, especially from the BSD family. It also supports projects related to the BSD systems, people behind them and Open Source communities. meetBSD is a strictly technical event with focus on high quality content, with opportunities for both seasoned professionals and those only planning to learn the BSD way. The truly open world of BSD is waiting for you.

Besides the conference proper, meetBSD has always been a great social event, full of attractions and surprises. Hurry up and register for the conference yet today, as the registration deadline is approaching shortly. See you all in Krakow, July 2-3!

meetBSD 2010 Conference
July 2nd and 3rd 2010, Kraków, Poland

www.meetBSD.org

organizers:

@outsourceme SEMIHALF EMBEDDED SYSTEMS

possible implementations: feel free to modify it according to your taste and needs.

Note: Postfix obtains the full pathname of the maildirs by joining the values of the `virtual_mailbox_base` and `virtual_mailbox_maps` parameters, while Courier-IMAP obtains it by joining the values of the `MYSQL_HOME_FIELD` and `MYSQL_MAILDIR_FIELD` parameters. As a consequence, we will create two separate fields in the `users` table (`home` and `maildir`) and make those variables point to them in order for Postfix and Courier-IMAP to get along see Listing 7.

Now let's take a brief look at the new Postfix configuration files, which include the configuration settings for MySQL see Listing 8.

That's all: now we can reload Postfix configuration:

```
# postfix reload
postfix/postfix-script: refreshing the Postfix mail system
```

and test our work; everything should run exactly as before!

Courier-IMAP

Now that our server can send and receive email, it may be useful to let users read it! For this purpose, we're going to install Courier-IMAP (*http://www.courier-mta.org/imap/*), *a fast, scalable, enterprise IMAP server that uses Maildirs*. This is the same IMAP server that comes with the Courier mail server (*http://www.courier-mta.org/*), but configured as a standalone IMAP server that can be used with other mail servers, such as Postfix.

Installation and configuration

The following is the list of the required packages:

- `gdbm-x.x.x.tgz`
- `libltdl-x.x.x.tgz`
- `tcl-x.x.x.tgz`
- `expect-x.x.x-no _ tk.tgz`
- `courier-authlib-x.x.tgz`
- `courier-imap-x.x.x.tgz`
- `courier-authlib-mysql-x.x.x.tgz`

Once you have added all the packages, you will find a fresh new `/etc/courier/` directory containing Courier IMAP's configuration files. Let's take a brief look at each of them.

The `/etc/courier/authdaemonrc` configuration file sets several operational parameters for the `authdaemond` process (the resident authentication daemon); fortunately, we only need to edit the `authmodulelist` parameter, which specifies the list of the authentication modules available; set it to `authmysql` to allow for MySQL based authentication:

```
/etc/courier/authdaemonrc
[ ... ]
authmodulelist="authmysql"
[ ... ]
```

The `/etc/courier/authmysqlrc` configuration file contains the `authmysql` database connection parameters; below is a sample configuration file: see Listing 9.

The next step is creating the SSL certificate for the IMAPS protocol. To make your life easier, Courier-IMAP comes with a script, `mkimapdcert(8)` (*http://www.courier-mta.org/mkimapdcert.html*), which will create the certificate after reading all the necessary information from the /etc/courier/imapd.cnf configuration file. Therefore,

Listing 11. *Testing the POP service*

```
# telnet mail.kernel-panic.it 110
Trying 172.16.240.150...
Connected to mail.kernel-panic.it.
Escape character is '^]'.
+OK Hello there.
user d.mazzocchio@kernel-panic.it
+OK Password required.
pass danix
+OK logged in.
list
1 2531
[ ... ]
quit
+OK Bye-bye.
Connection closed by foreign host.
#
```

Listing 12. *Freshclam configuration*

```
/etc/freshclam.conf
DatabaseDirectory       /var/db/clamav
DatabaseOwner           _clamav
DNSDatabaseInfo         current.cvd.clamav.net
DatabaseMirror          db.it.clamav.net
DatabaseMirror          database.clamav.net
MaxAttempts             3
checks                  24
```

you should first customize the latter file (in particular, pay close attention to the *common name* (CN) parameter, which must match the name of the server the users will connect to) and then run `mkimapdcert(8)` (*http://www.courier-mta.org/mkimapdcert.html*):

```
# /usr/local/sbin/mkimapdcert
[ ... ]
```

Now we only have to start the daemons:

```
# mkdir -p /var/run/courier{,-auth}/
# /usr/local/sbin/authdaemond start
# /usr/local/libexec/imapd.rc start
# /usr/local/libexec/imapd-ssl.rc start
```

configure the system to start Courier-IMAP on boot:

```
/etc/rc.local
echo -n ' Courier-IMAP'
/bin/mkdir -p /var/run/courier{,-auth}/
[ -x /usr/local/sbin/authdaemond ] && /usr/local/sbin/
                    authdaemond start
[ -x /usr/local/libexec/imapd.rc ] && /usr/local/libexec/
                    imapd.rc start
[ -x /usr/local/libexec/imapd-ssl.rc ] && /usr/local/
                    libexec/imapd-ssl.rc start
```

Listing 13. *Updating virus signatures with freshclam*

```
# freshclam
ClamAV update process started at Tue Dec 18 00:35:25 2007
WARNING: Your ClamAV installation is OUTDATED!
WARNING: Local version: 0.90.3 Recommended version: 0.92
DON'T PANIC! Read http://www.clamav.net/support/faq
Downloading main.cvd [100%]
main.cvd updated (version: 45, sigs: 169676, f-level:
                    21, builder: sven)
WARNING: Your ClamAV installation is OUTDATED!
WARNING: Current functionality level = 16, recommended
                    = 21
DON'T PANIC! Read http://www.clamav.net/support/faq
Downloading daily.cvd [100%]
daily.cvd updated (version: 5160, sigs: 8698, f-level:
                    21, builder: sven)
WARNING: Your ClamAV installation is OUTDATED!
WARNING: Current functionality level = 16, recommended = 21
DON'T PANIC! Read http://www.clamav.net/support/faq
Database updated (178374 signatures) from
                    db.it.clamav.net (IP:
                    193.206.139.37)
#
```

...and test our hard work! I suggest using a simple Python script, just to give our weary fingers a break: see Listing 10.

Adding POP3 access

It is usually desirable that email users be offered the choice between IMAP and POP3 remote access; after all, POP3 users tend to use less disk space, bandwidth and resources on the server.

Adding POP3 support to our mail server is fairly simple; first, we need to add the appropriate package:

```
courier-pop3-x.x.x.tgz
```

Then, we have to run `mkpop3dcert(8)` (*http://www.courier-mta.org/mkpop3dcert.html*) to generate the SSL certificate for POP3 over SSL (similarly to `mkimapdcert(8)` (*http://www.courier-mta.org/mkimapdcert.html*), SSL parameters are read from a configuration file, `/etc/courier/pop3d.cnf`) and start the daemons:

```
# /usr/local/sbin/mkpop3dcert
[ ... ]
# /usr/local/libexec/pop3d.rc start
# /usr/local/libexec/pop3d-ssl.rc start
```

Add the following lines to `/etc/rc.local(8)` (*http://www.openbsd.org/cgi-bin/man.cgi?query=rc.local&sektion=8*) to start the POP3 server on boot:

```
/etc/rc.local
[ -x /usr/local/libexec/pop3d.rc ] && /usr/local/libexec/
                    pop3d.rc start
[ -x /usr/local/libexec/pop3d-ssl.rc ] && /usr/local/
                    libexec/pop3d-ssl.rc start
```

Finally, we can perform a quick test to make sure everything works as expected: see Listing 11.

Managing disk space

Quotas allow you to specify the maximum size of maildirs, in order to prevent the `/var/vmail` filesystem from filling up. The best option would certainly be to use the operating system's built-in quota support (*http://www.openbsd.org/faq/faq10.html#Quotas*), but we can't, because we have a single user writing to all the maildirs. Therefore, we must rely on the mail software to get quota support on maildirs.

Courier-IMAP comes with built-in quota support, but this solves only one half of the problem: in fact, also Postfix must be able to reject mail sent to over-quota users. To achieve this, we will rely on the `deliverquota(8)` (*http://www.courier-mta.org/deliverquota.html*) utility, which delivers mail taking into account any software-imposed quota on maildirs.

The first step is assigning a quota to each maildir with `maildirmake(1)` (*http://www.courier-mta.org/maildirmake.html*). E.g.:

```
# /usr/local/bin/maildirmake -q 10000000S \
> /var/vmail/kernel-panic.it/d.mazzocchio
```

The above command installs an (approximately) 10MB quota on the `/var/vmail/kernel-panic.it/d.mazzocchio` maildir. Note: `maildirmake(1)` (*http://www.courier-mta.org/maildirmake.html*) also allows you to create and initialize maildirs, thus allowing users to access them; otherwise, a user's maildir will be created upon receiving his first email.

Next, we need to define, in `/etc/postfix/master.cf(5)` (*http://www.postfix.org/master.5.html*), a special Postfix daemon for delivery through `deliverquota(8)` (*http://www.courier-mta.org/deliverquota.html*):

```
/etc/postfix/master.cf
[ ... ]
qdeliver  unix  -     n     n     -     -     pipe
  flags=uh user=vmail argv=/usr/local/bin/deliverquota -c
                 -w 90
  /var/vmail/${domain}/${user}
```

and tell Postfix to use this daemon for final delivery to virtual domains, by setting the value of the virtual_transport parameter in `/etc/postfix/main.cf`:

```
/etc/postfix/main.cf
virtual_transport = qdeliver
```

`deliverquota(8)` will place a warning message into the maildir if, after the message is successfuly delivered, the maildir is at least 90 percent full (`-w 90`). The body of the warning message is copied verbatim from the `/etc/courier/quotawarnmsg` file.

Please note that, as reported by Giovanni Bechis, `deliverquota(8)` (*http://www.courier-mta.org/deliverquota.html*) fails to correctly deliver emails sent to an alias that maps to multiple accounts, one of which has the same name as the alias itself, unless you set the following parameters in `/etc/postfix/main.cf`:

```
/etc/postfix/main.cf
qdeliver_destination_concurrency_limit = 1
qdeliver_destination_recipient_limit = 1
```

Listing 14. *Amavisd configuration*

```
/etc/amavisd.conf
# COMMONLY ADJUSTED SETTINGS:

$max_servers = 2;
$daemon_user  = '_clamav';      # Run under the same
                    user as ClamAV
$daemon_group = '_clamav';      # Run under the same
                    group as ClamAV

$mydomain = 'kernel-panic.it';

$MYHOME   = '/var/amavisd';
$TEMPBASE = "$MYHOME/tmp";    # working directory,
                    needs to be created manually
$ENV{TMPDIR} = $TEMPBASE;
$QUARANTINEDIR = '/var/clamav/quarantine';

[...]

# Leave only ClamAV uncommented
@av_scanners = (
  ['ClamAV-clamd',
    \&ask_daemon, ["CONTSCAN {}\n", "/var/clamav/
                    clamd.sock"],
    qr/\bOK$/, qr/\bFOUND$/,
    qr/^.*?: (?!Infected Archive)(.*) FOUND$/ ],
);

[...]

# Leave only ClamAV uncommented
@av_scanners_backup = (
  ['ClamAV-clamscan', 'clamscan',
    "--stdout --disable-summary -r --tempdir=$TEMPBASE
                    {}", [0], [1],
    qr/^.*?: (?!Infected Archive)(.*) FOUND$/ ],
);

1;
```

MAGAZINE
BSD

Listing 15. *Running amavisd*

```
# mkdir /var/amavisd/tmp
# chown -R _clamav:_clamav /var/amavisd/
# /usr/local/sbin/amavisd debug
Dec 18 22:07:11 mail.kernel-panic.it /usr/local/sbin/
                    amavisd[24429]: starting.
/usr/local/sbin/amavisd at mail.kernel-panic.it
                    amavisd-new-2.3.2 (20050629),
                    Unicode aware
Dec 18 22:07:11 mail.kernel-panic.it /usr/local/sbin/
                    amavisd[24429]: user=,
EUID: 0 (0);  group=, EGID: 0 31 20 5 4 3 2 0 (0 31 20
                    5 4 3 2 0)
Dec 18 22:07:11 mail.kernel-panic.it /usr/local/
                    sbin/amavisd[24429]: Perl version
                    5.008008
[...]
```

Listing 16. *Adding the content-filtering services to Postfix*

```
/etc/postfix/master.cf
smtp-amavis unix -     -      -      -      2  smtp
    -o smtp_data_done_timeout=1200
    -o smtp_send_xforward_command=yes
    -o disable_dns_lookups=yes
    -o max_use=20

127.0.0.1:10025 inet n  -      -      -      -  smtpd
    -o content_filter=
    -o local_recipient_maps=
    -o relay_recipient_maps=
    -o smtpd_restriction_classes=
    -o smtpd_delay_reject=no
    -o smtpd_client_restrictions=permit_mynetworks,reject
    -o smtpd_helo_restrictions=
    -o smtpd_sender_restrictions=
    -o smtpd_recipient_restrictions=permit_mynetworks,reject
    -o mynetworks_style=host
    -o mynetworks=127.0.0.0/8
    -o strict_rfc821_envelopes=yes
    -o smtpd_error_sleep_time=0
    -o smtpd_soft_error_limit=1001
    -o smtpd_hard_error_limit=1000
    -o smtpd_client_connection_count_limit=0
    -o smtpd_client_connection_rate_limit=0
    -o receive_override_options=no_header_body_
                    checks,no_unknown_recipient_checks
```

Setting these parameters to `1` disables parallel deliveries to the same recipient.

Content filtering

Now we have a fully-functional mail server, able to send and receive email and providing remote access to users' mailboxes. However, if we don't want our server to become an immune carrier of computer viruses or to be drowned under a sea of spam, we need to install all the necessary content-filtering tools.

Though Postfix natively supports multiple content inspection mechanisms (*http://www.postfix.org/BUILTIN_FILTER_README.html*), the documentation (*http://www.postfix.org/CONTENT_INSPECTION_README.html*) itself *encourages the use of external filters and standard protocols because this allows you to choose the best MTA and the best content inspection software for your purpose*. Therefore, we will rely on third-party software for content filtering; in particular, we will use SpamAssassin to filter spam, ClamAV to check emails for viruses and Amavisd-new to coordinate it all. Below is the outline of the whole architecture: see Figure 2.

SpamAssassin

SpamAssassin (*http://wiki.apache.org/spamassassin/*) is a *mature, widely-deployed open source project that serves as a mail filter to identify Spam. SpamAssassin uses a variety of mechanisms including header and text analysis, Bayesian filtering, DNS blocklists, and collaborative filtering databases*.

There are quite a few packages we need to install:

- `p5-Compress-Raw-Zlib-x.x.tgz`
- `p5-IO-Compress-Base-x.x.tgz`
- `p5-IO-Compress-Zlib-x.x.tgz`
- `p5-Compress-Zlib-x.x.tgz`
- `p5-IO-Zlib-x.x.tgz`
- `p5-IO-String-x.x.tgz`
- `p5-Algorithm-Diff-x.x.tgz`
- `p5-Text-Diff-x.x.tgz`
- `p5-Archive-Tar-x.x.tgz`
- `re2c-x.x.x.tgz`
- `p5-Net-CIDR-Lite-x.x.tgz`
- `p5-Net-IP-x.x.tgz`
- `p5-Digest-SHA1-x.x.tgz`
- `p5-Digest-HMAC-x.x.tgz`
- `p5-Net-DNS-x.x.tgz`
- `p5-Sys-Hostname-Long-x.x.tgz`
- `p5-URI-x.x.tgz`
- `p5-Mail-SPF-Query-x.x.tgz`

- `p5-Socket6-x.x.tgz`
- `p5-IO-INET6-x.x.tgz`
- `bzip2-x.x.x.tgz`
- `libiconv-x.x.x.tgz`
- `gettext-x.x.x.tgz`
- `libidn-x.x.x.tgz`
- `curl-x.x.x.tgz`
- `gnupg-x.x.x.tgz`
- `p5-Net-SSLeay-x.x.tgz`
- `p5-IO-Socket-SSL-x.x.tgz`
- `p5-HTML-Tagset-x.x.tgz`
- `p5-HTML-Parser-x.x.tgz`
- `p5-Crypt-SSLeay-x.x.tgz`
- `libghttp-x.x.x.tgz`
- `p5-HTTP-GHTTP-x.x.tgz`
- `p5-libwww-x.x.tgz`
- `p5-Mail-SpamAssassin-x.x.x.tgz`

After the packages installation, you will find the main SpamAssassin configuration file (`local.cf`) in the fresh new `/etc/mail/spamassassin` directory. The configuration phase can be very complex and goes beyond the scope of this document; anyway, you can find all the details in the man page (Mail::SpamAssassin::Conf *http://spamassassin.apache.org/full/3.1.x/dist/doc/Mail_SpamAssassin_Conf.html*).

Like Postfix, SpamAssassin has a lot of configuration parameters, although, in most cases, default values can be preserved and only a few parameters need to be overridden:

```
/etc/mail/spamassassin/local.cf
rewrite_header  Subject ***** SPAM *****
report_safe     1
lock_method     flock
required_score  8.0
```

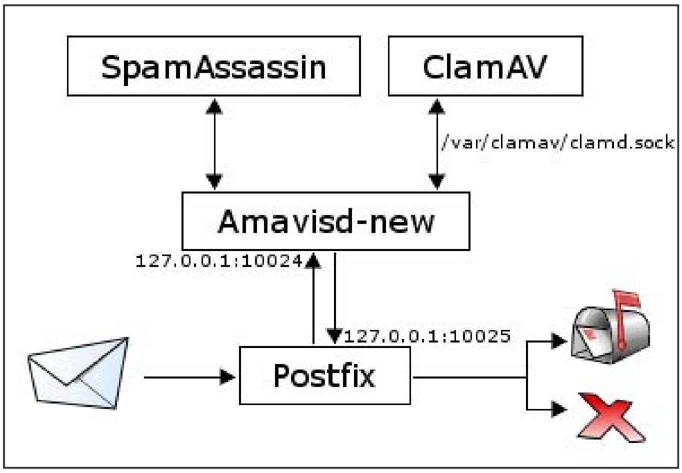


Figure 2. *Mail filtering*

ClamAV

ClamAV (*http://www.clamav.net/*) is an *open source (GPL) anti-virus toolkit for UNIX*; the main purpose of this software is the integration with mail servers (i.e. attachment scanning). All the antivirus tasks are handled by three processes:

- freshclam – which automatically updates the virus definitions, by connecting to one of the ClamAV mirrors (*http://www.clamav.net/mirrors.html*); its configuration file is `/etc/freshclam.conf`;
- clamd – a flexible and scalable multi-threaded antivirus daemon; its configuration file is `/etc/clamd.conf`;
- clamscan – a command line antivirus scanner.

The required packages are:

- `arc-x.xx.tgz`
- `lha-x.xx.xxxxxx.tgz`
- `unzip-x.x.tgz`
- `zoo-x.x.x.tgz`
- `gmp-x.x.x.tgz`
- `unarj-x.x (from the ports)`
- `unrar-x.x (from the ports)`
- `clamav-x.x.tgz`

The `freshclam.conf` configuration file requires only a few parameters: see Listing 12.

Now we can update the virus definition database by running the `freshclam` command. Please make sure you have installed the latest release of ClamAV, or you'll get warning messages about reduced functionality, like the following: see Listing 13.

The reduced *functionality level* means that you may not be able to use all the available virus signatures and, consequently, fail to detect the latest viruses. To automatically update the database, we simply have to schedule freshclam in crontab every hour (preferably not on the hour, just to avoid traffic peaks):

```
16 * * * * /usr/local/bin/freshclam >/dev/null 2>&1
```

Also the `/etc/clamd.conf` configuration file needs editing only very few parameters:

```
/etc/clamd.conf
DatabaseDirectory       /var/db/clamav
LocalSocket             /var/clamav/clamd.sock
User                    _clamav
[...]
```

Bibliography

- *http://www.openbsd.org/faq/* – OpenBSD Documentation and Frequently Asked Questions
- *http://www.postfix.org/uce.html* – Postfix Configuration – UCE Controls
- *http://www.postfix.org/BASIC_CONFIGURATION_README.html* – Postfix Basic Configuration
- *http://www.postfix.org/MYSQL_README.html* – Postfix MySQL Howto
- *http://www.postfix.org/VIRTUAL_README.html* – Postfix Virtual Domain Hosting Howto
- *http://spamassassin.apache.org/full/3.0.x/dist/doc/Mail_SpamAssassin_Conf.html* – Mail::SpamAssassin::Conf
- *http://www.ijs.si/software/amavisd/README.postfix* – How to use amavisd-new with Postfix
- *http://www.clamav.net/doc/latest/html/* – Clam AntiVirus 0.92 User Manual
- *http://www.flakshack.com/anti-spam/wiki/index.php* – Fairly-Secure Anti-SPAM Gateway Using OpenBSD, Postfix, Amavisd-new, SpamAssassin, Razor and DCC

Now we can run clamd:

```
# touch /var/log/clamd.log
# chown _clamav /var/log/clamd.log
# clamd
Running as user _clamav (UID 539, GID 539)
```

and add the following lines to `/etc/rc.local(8)` (*http://www.openbsd.org/cgi-bin/man.cgi?query=rc&sektion=8*) to start it on system boot:

```
/etc/rc.local
if [ -x /usr/local/sbin/clamd ]; then
    echo -n ' clamd'
    [ -S /var/clamav/clamd.sock ] && rm -f /var/clamav/clamd.sock
    /usr/local/sbin/clamd >/dev/null 2>&1
fi
```

Amavisd-new

Amavisd-new (*http://www.ijs.si/software/amavisd/*) is a high-performance interface between mailer (MTA) and content checkers.

We will configure it to bind to port 10024 on the loopback interface, where Postfix will forward all incoming e-mails. If the e-mail successfully passes all the checks, it will be forwarded back to Postfix, listening on localhost port 10025; otherwise, mails may be deleted or quarantined and the administrator and recepients may be notified.

The following is the list of the required packages:

- `cabextract-x.x.tgz`
- `freeze-x.x (from the ports)`
- `p5-Convert-BinHex-x.x.tgz`
- `p5-IO-stringy-x.x.tgz`
- `p5-Mail-Tools-x.x.tgz`
- `p5-Time-TimeDate-x.x.tgz`
- `p5-MIME-tools-x.x.tgz`
- `p5-Convert-TNEF-x.x.tgz`
- `p5-Convert-UUlib-x.x.tgz`
- `rpm2cpio-x.x.tgz`
- `p5-Net-Server-x.x.tgz`
- `p5-Unix-Syslog-x.x.tgz`
- `amavisd-new-x.x.x.tgz`

References

- *http://www.kernel-panic.it/openbsd/carp/index.html* – Redundant firewalls with OpenBSD, CARP and pfsync
- *http://www.openbsd.org/* – OpenBSD, a FREE, multi-platform 4.4BSD-based UNIX-like operating system
- *http://www.postfix.org/* – Postfix, an Open source email server for Unix
- *http://www.postfix.org/TLS_README.html* – [TLS] – Postfix TLS Support
- *http://www.mysql.com/* – MySQL, the world's most popular open source database
- *http://www.courier-mta.org/imap/* – Courier-IMAP, a fast, scalable, enterprise IMAP server that uses Maildirs
- *http://asg.web.cmu.edu/sasl/* – Cyrus SASL, the Cyrus SASL API implentation
- *http://www.ijs.si/software/amavisd/* – Amavisd-new, a high-performance interface between mailer (MTA) and content checkers: virus scanners and/or SpamAssassin
- *http://spamassassin.apache.org/* – SpamAssassin, a mail filter to identify spam using a wide range of heuristic tests on mail headers and body text
- *http://www.clamav.net/* – ClamAV, an open source (GPL) anti-virus toolkit for Unix
- *http://www.postfix.org/VIRTUAL_README.html* – Postfix Virtual Domain Hosting Howto
- *http://tools.ietf.org/html/rfc4954* – [RFC4954] – RFC 4954, SMTP Service Extension for Authentication
  http://www.postfix.org/SASL_README.html – [SASL] – Postfix SASL Howto

The installation procedure creates a new user and group called _vscan; however, the easiest way to get Amavisd-new to cooperate with ClamAV, is to run them both under the same user (_clamav). The configuration file is /etc/amavisd.conf, which is actually a perl script (so pay attention to the semi-colons at the end of the lines!); below are the options you will most likely want to tweak: see Listing 14.

After manually creating Amavisd-new's working directory (/var/amavisd/tmp), we can start the daemon in debug mode (i.e. in foreground), just to check any errors: see Listing 15.

Now we can configure the system to start Amavisd-new on boot:

```
/etc/rc.local
if [ -x /usr/local/sbin/amavisd ]; then
    echo -n ' amavisd'
    /usr/local/sbin/amavisd >/dev/null 2>&1
fi
```

The last step is to update Postfix configuration to enable interfacing between Postfix and Amavisd-new. To achieve this, we have to add a couple of services to the /etc/postfix/master.cf(5) (*http://www.postfix.org/master.5.html*) configuration file: one to forward all incoming emails to Amavid-new, and the other to get emails back again:see Listing 16.

Finally, we need to tell Postfix to start forwarding all the emails it receives to amavisd-new for content inspection and reload the configuration.

```
# postconf -e 'content_filter=smtp-amavis:[127.0.0.1]:
                    10024'
# postfix reload
postfix/postfix-script: refreshing the Postfix mail system
```

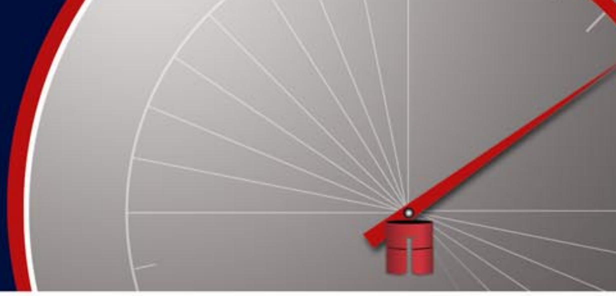
Appendix

Special thanks to TomazZ for his detailed notes on configuring TLS in Postfix.

DANIELE MAZZOCCHIO

Latest version: http://www.kernel-panic.it/openbsd/mail/

ITTIA
The Power to Manage Data



Performance Comparison

ITTIA DB and SQLite

ITTIA DB SQL and SQLite are used by software developers to manage information stored in applications and devices. Designed to be hidden from the end-user, these embedded relational database management systems are linked into the application or firmware as self-contained software libraries.

This greatly simplifies the application code responsible for organizing data and sharing it between concurrent tasks, while protecting data from corruption and race conditions.

SQLite is an open source software library that provides the basic features for storing information in an SQL database. SQLite is designed for self-contained applications that require an SQL database, but do not frequently modify or share access to the database.

ITTIA DB SQL is a scalable database engine that supports a wide range of features, providing high performance for self-contained applications without limiting the application's ability to scale up as requirements change. ITTIA DB SQL can be used as either a stand-alone software library, or with a lightweight server.

This white paper explores how differences between ITTIA DB SQL and SQLite affect the performance, features, and maintenance of database-driven applications.

Criteria for Selecting a Lightweight Relational Embedded Database

Every software development project has unique requirements, including performance objectives, reliability expectations, and time-to-market constraints. Careful planning, and the incorporation of the right database technology, can dramatically reduce project development time while increasing the performance and reliability of an application.

Databases can be implemented in many different ways, and the choice of algorithms in a particular database product has a profound impact on performance characteristics and what features are available. Areas most impacted by the implementation of the database include:

- Frequency of costly disk or flash media I/O operations.
- Time required to recover from a crash or power loss.
- Ability to manage large amounts of data without severe performance degradation.
- Performance impact of sharing data between tasks and other applications.
- Relative performance of read and write operations.
- Portability of the storage format and application code.
- Effort required to integrate database technology into the application.

Database software must carefully balance the flow of data between persistent storage and memory. If information is not saved regularly, recovery from a power loss will be slower. If information is saved too often, performance will suffer and some media, such as flash memory, will wear out more quickly.

Table 1. *Benchmark Hardware*

	Operating System	Architecture	Speed	RAM	Media
Windows XP PC	Windows XP	x86	1.7GHz	256 MiB	Hard Disk
Linux PC	Fedora Core Linux	x86	2.8GHz	1 GiB	Hard Disk
Linux Device	Angstrom Linux	ARM9	400Mhz	64 MiB	Flash

Scalability is a concern for many applications, both for the amount of data that can be stored and for the impact of sharing data. If scalability is not important, a database can greatly optimize performance. If scalability is needed, or may be needed in the future, the performance cost can be very high if the database has not provisioned for it.

By considering the current and future requirements of an application, an embedded software developer can select the most appropriate database technology.

Benchmark

This benchmark will measure elapsed time for three operations:

- Insert rows into an empty database.
- Select rows using indexed search.
- Update rows using indexed search.

Test Environment

The benchmark is performed on the following platform: see Table 1.

Test Methodology

Product Configuration

For this benchmark, the database products are configured as follows: see Table 2.

Database Schema

The following database schema is used for the benchmark: see Listing 1.

Table 2. *xxxxxxx*

	ITTIA DB SQL	SQLite
Version	3.1	3.6.17
Storage Type	Disk, Memory	Disk, Memory
Connection Method	Stand-alone single-user library	Stand-alone library
Connections	1	1
API	ITTIA DB C API	SQLite C API
Access Method	Table Cursor, SQL	SQL

Parameters

Each product is tested for both disk-based and in-memory storage. Additionally, ITTIA DB is tested using both table cursors and SQL queries. SQLite only supports SQL queries. The benchmark is run with the following parameters for disk-based storage:

- inserts = 10,000
- selects = 10,000
- updates = 10,000
- max_inserts_per_tx = 40
- max_selects_per_tx = 120
- max_updates_per_tx = 80

The number of operations is increased for in-memory storage:

- inserts = 100,000
- selects = 100,000
- updates = 100,000

Test Algorithms

SQL queries are prepared once at the beginning of the test. When table cursors are used, SQL queries are replaced with equivalent ITTIA DB C API syntax see Listing 2 and Listing 3.

Results

Results are measured as a ratio of elapsed time for SQLite to elapsed time for ITTIA DB. A result of 1.00 indicates equivalent performance. Values greater than 1.00 favor ITTIA DB. Values less than 1.00 favor SQLite.

Metrics for elapsed time are available upon request. Contacting ITTIA Research and Development using the form at: *http://www.ittia.com/contact*

- Disk Storage see Figure 1 and Figure 2.
- Memory Storage see Figure 3 and Figure 4

Feature Comparison

Atomic Transactions

In an embedded database, related database operations are grouped together into transactions, each of which will be saved either completely or not at all. To the application, it is as though all changes made in the transaction are saved immediately and simultaneously when the transaction is committed. This feature is known as atomic commit.

ITTIA DB SQL and SQLite both support atomic commit, but with different performance characteristics. SQLite creates a rollback journal for each transaction that remembers the original value before each change is made. ITTIA DB stores both the original value and the new value in a write-ahead log that can be shared by

Listing 1. *Schema*

```
create table benchmark_table (
   a integer not null,
   b integer,
   unused1 integer,
   unused2 integer,
   unused3 integer,
   unused4 integer,
   unused5 integer,
   unused6 integer,
   unused7 integer,
   unused8 integer,
   unused9 integer,
   unused10 integer,
   unused11 integer,
   unused12 integer,
   unused13 integer,
   unused14 integer,
   unused15 integer,
   unused16 integer,
   unused17 integer,
   unused18 integer
);
create unique index ix1 on benchmark_table (a);
```

Listing 2. *Benchmark Insert Algorithm*

```
const int inserts;
const int max_inserts_per_tx;

begin transaction;

for(long i = 1; i <= inserts; i++) {
   insert into benchmark_table(a, b) values(i, i+2);

   if (max_inserts_per_tx > 0 && i % max_inserts_per_
                    tx == 0) {
      commit transaction;
      begin transaction;
   }
}

commit transaction;
```

Listing 3. *Benchmark Select Algorithm*

```
const int selects;
const int max_selects_per_tx;

begin transaction;

for (long i = 1; i <= selects; i++) {
   long value = (rand() % inserts) + 1;

   select b from benchmark_table where a = value;

   if (i % max_selects_per_tx == 0) {
      commit transaction;
      begin transaction;
   }
}

commit transaction;
```

Listing 4. *Benchmark Update Algorithm*

```
const int updates;
const int max_updates_per_tx;

begin transaction;

for (long i = 1; i <= updates; i++) {
   long value = (rand() % inserts) + 1;

   update benchmark_table set b = value + 8 where a =
                   value;

   if (i % max_updates_per_tx == 0) {
      commit transaction;
      begin transaction;
   }
}

commit transaction;
```

multiple transactions. This has some surprising results on overall performance.

When a transaction is committed, the database must write enough information to disk to guarantee that no changes made in the transaction are lost. Data is organized into pages to optimize access to block devices such as disks and flash memory. In most cases, changes are scattered throughout the database, only modifying a small portion of each page.

When SQLite commits a transaction, it must write all modified pages to disk in their entirety and then wait for the operation to finish. This is extremely costly, but necessary to support recovery with only a rollback journal.

When ITTIA DB SQL commits a transaction, only the write-ahead log must be written immediately because it contains a copy of all the changes made in the transaction. Log entries are written sequentially and only contain information about changes, so fewer pages are written to disk. Other modified pages are written to disk later, after accumulating changes from many transactions.

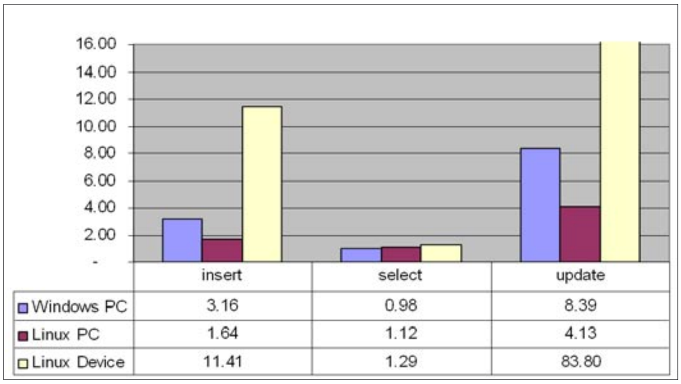
Under high load, ITTIA DB SQL can significantly reduce the amount of write activity compared to SQLite, both boosting performance and reducing wear on flash storage media.

Shared Access

Device applications usually perform many different tasks on data stored in the database. Some tasks perform best when run in parallel, allowing long-running tasks such as synchronization to happen without completely stopping normal operations. Tasks can be performed by a single application with multiple threads or instances, multiple applications on a device, or even remote applications across a network.

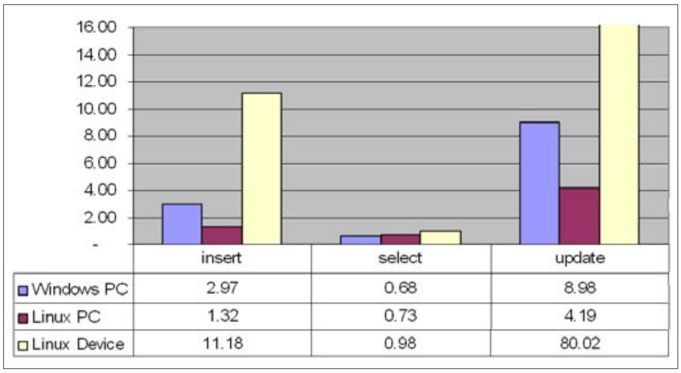
SQLite uses the locking mechanism built-in to the file system to protect database files during modification, which allows any process on the device with access to the file to use the database. This approach relies on the stability of the file system locking framework, and therefore cannot be used with files that are shared over a network. Many operating systems do not fully implement the range locks required to safely share an SQLite database.

ITTIA DB SQL uses a lightweight data server to negotiate shared access. The server is self-contained, requiring little or no configuration so that it is straightforward to use on a device. To access the database, an application is linked with a small client library in place of the stand-alone library. No code changes are required to access a database file on the same device and the application can also open database files remotely over a TCP/IP network.



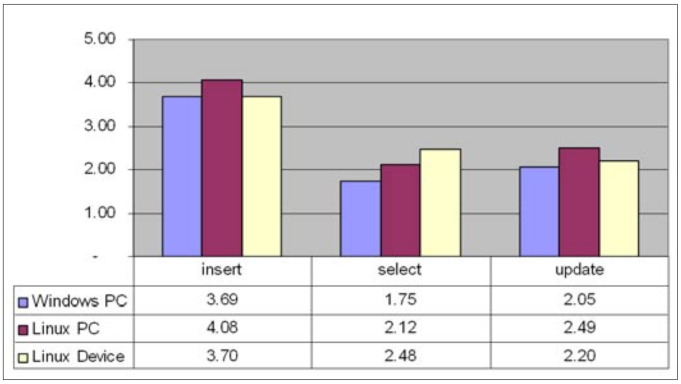
	insert	select	update
□ Windows PC	3.16	0.98	8.39
■ Linux PC	1.64	1.12	4.13
□ Linux Device	11.41	1.29	83.80

Figure 1. *SQLite vs. ITTIA DB Table Cursors (Disk)*



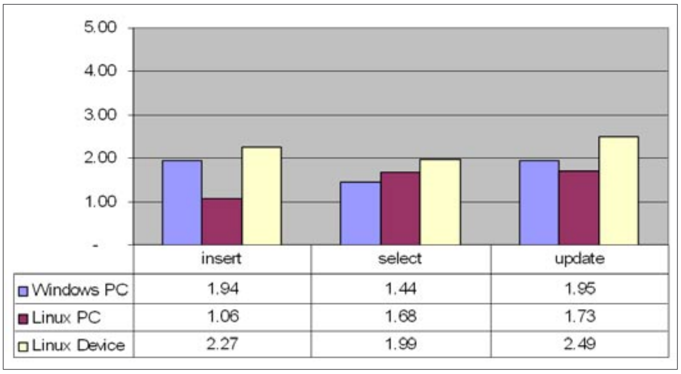
	insert	select	update
□ Windows PC	2.97	0.68	8.98
■ Linux PC	1.32	0.73	4.19
□ Linux Device	11.18	0.98	80.02

Figure 2. *SQLite vs. ITTIA DB SQL Queries (Disk)*



	insert	select	update
□ Windows PC	3.69	1.75	2.05
■ Linux PC	4.08	2.12	2.49
□ Linux Device	3.70	2.48	2.20

Figure 3. *SQLite vs. ITTIA DB Table Cursors (Memory)*



	insert	select	update
□ Windows PC	1.94	1.44	1.95
■ Linux PC	1.06	1.68	1.73
□ Linux Device	2.27	1.99	2.49

Figure 4. *SQLite vs. ITTIA DB SQL Queries (Memory)*

File system locks operate on an entire file at once, so when an SQLite transaction begins to modify the database, it must obtain exclusive access to the entire database file until the transaction is finished. This is not a problem when sharing is infrequent and every transaction only involves one or a few rows. However, one long-running transaction, such as a synchronization task, can block all other activity in the database, even when there is no real conflict.

ITTIA DB SQL uses a less restrictive locking technique: row-level locking with isolation levels. The database automatically tracks all rows that are read or modified in a transaction. At the highest level of isolation, known as *serializable*, rows are locked in such a way as to prevent all possible conflicts. And for most simple transactions, the isolation level can be reduced to minimize locking even further. This ensures that a transaction is only blocked when it would create a conflict with another transaction already in progress. In addition, an entire table can be locked manually.

Row-level locking is also available when an ITTIA DB SQL database is shared between threads in an application. SQLite allows an open database to be shared between threads, but only one thread can modify the database at a time. ITTIA DB SQL permits multiple threads to concurrently read and modify different rows in the same database without risk of conflict.

In-Memory Databases

Databases are typically designed to store data on a block device, such as a hard disk or flash media. Some applications have sufficient memory to store data entirely in main memory. To optimize for this scenario, both ITTIA DB SQL and SQLite support in-memory storage.

When a database is created in memory, SQLite uses its normal paging algorithms to organize the data, but does not write any pages to disk. ITTIA DB SQL uses algorithms that are specialized for memory tables. In this way, ITTIA DB SQL is able to take advantage of optimizations such as direct pointers to significantly improve performance.

ITTIA DB SQL also supports hybrid in-memory/on-disk databases that contain a mixture of memory and disk tables. A similar result can be achieved with SQLite by attaching an existing database file to a memory database. When tables are stored on disk, ITTIA DB uses row-level locking for higher concurrency.

Data Typing

Embedded databases can store many types of information, the most common being numbers, text, date,

and time. A value's data type must be known when it is read, whether it is used in an expression, displayed as text, or accessed natively in a programming language. Many types are incompatible: a string of text cannot always be used where a number is expected, and large numbers will not fit in an 8-bit variable.

To prevent a type mismatch, the database can check a value's data type either when it is written to the database, or when it is read. In the first case, the database must have some information about how the value will be used later so that it can check incoming values for conformance, and when a type mismatch occurs, the error must be reconciled when the value is stored in the database. In the case where the value is checked at read time, the database must be able to accommodate values of arbitrary type, and type mismatch errors are not handled until the value is read.

SQLite uses dynamic run-time typing, which means that data types are only checked when a value is read from the database and used. This is useful in prototyping, before the application's requirements are fully formed, because it allows data to be written to the database without much regard for how it will be used later. Production code, however, must be carefully audited to ensure that type mismatches are not possible or can be dealt with in a reasonable way.

ITTIA DB SQL uses static typing, where type information is stored in the database schema as part of a table's description. Each column can contain only a specific type of data. This ensures that type mismatch errors are identified early, when there is the best chance to successfully fix the mistake. This is important when the database is shared between applications that are developed separately, as the database schema forms a contract by which all parties must abide.

Application Programming Interface

ITTIA DB SQL and SQLite each have an interactive utility that provides access to database files through standard SQL commands. While this is suitable for testing and maintenance, applications need a native interface to access the database.

To access data in an SQLite database, applications use SQL queries. ITTIA DB SQL similarly supports SQL queries, but also provides direct access to tables and indexes with low-level table cursors. Table cursors have lower overhead than SQL queries and allow modifications to be made directly while browsing a table, without constructing an update query. In many cases, an application can use table cursors to both improve performance and minimize data layer source code.

To protect a database from unauthorized access, the database file can be encrypted. ITTIA DB SQL provides encryption callbacks for an application to plug in any desired page-level encryption library. SQLite users must purchase the *SQLite Encryption Extension* (SEE), which utilizes password-based AES encryption only. Both products decrypt data when it is read from disk and encrypt data before it is written to disk.

Developers of embedded systems and intelligent devices find it important to protect their database from unauthorized access. This significant requirement was recognized by the architects of ITTIA DB SQL. ITTIA DB SQL provides encryption callbacks that an application utilizes to plug in any desired page-level encryption library. This important feature is not included in the open source version of SQLite.

Technical Support

The problems solved by an embedded database are not trivial. Even though the database software hides most of the details of data management from the application, learning to use the database effectively takes some time. The problem is magnified on restricted hardware, where small configuration details can have a big impact on footprint.

The involvement of a database expert can greatly improve the quality of the application by identifying the best strategies for design and implementation. Except for large enterprise companies, the only way to receive support for SQLite is through the open source community. While this is sometimes sufficient for small, specific questions, the community rarely provides high-level guidance and cannot provide confidentiality.

The commercial-grade support offered by ITTIA provides critical assistance from database experts throughout all phases of development, from laying down the best approach in the initial design, through development and testing, and into deployment of the application.

How to Reach Us

Home Page: www.ittia.com
Contact: www.ittia.com/contact
USA:
ITTIA
1611 116th Ave
Bellevue, WA 98004
425-462-0046

Information in this document is provided solely to enable system and software implementers to use ITTIA products. No express or implied copyright license is granted hereunder to design or implement any database management system software based on the information in this document.
ITTIA reserves the right to make changes without further notice to any products described herein. ITTIA makes no warranty, representation or guarantee regarding the suitability of its products for any particular purpose, nor does ITTIA assume any liability arising out of the application of or use of any product, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. Statistics and parameters provided in ITTIA white papers and data sheets can and do vary in different applications and actual performance may vary over time. All operating parameters must be validated for each customer application by customer's technical experts.

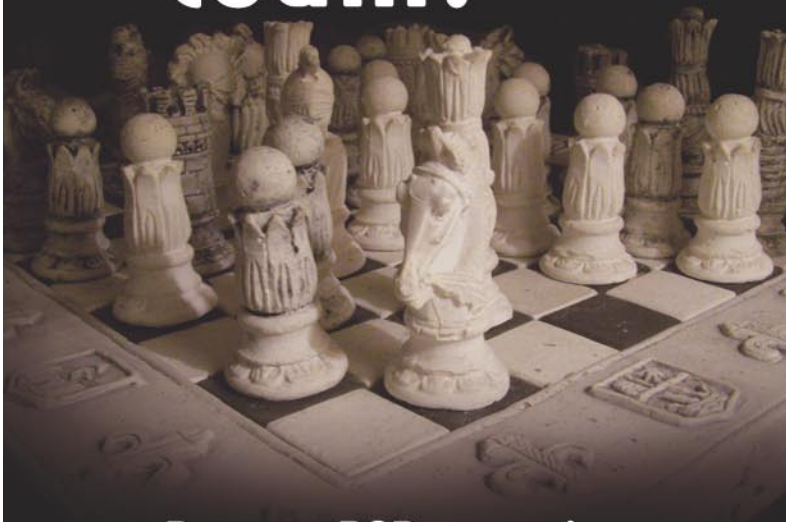
Conclusion

Often used as a stand-alone database software library, ITTIA DB SQL greatly simplifies data management for applications on embedded systems and devices, but without the complexity of a back-end database server.

SQLite is suited for simple projects that need basic transactional storage using SQL. SQLite takes advantage of self-imposed limitations to optimize for the most basic use cases. Advanced projects with robust or undefined requirements can benefit from ITTIA DB SQL's solid framework for formalizing, updating, and sharing data. Benchmarks show that even for simple operations, ITTIA DB outperforms SQLite.

Each embedded application has unique requirements for data management and storage. Selecting the right tools requires a careful problem analysis and a thorough understanding of database technology. Using a technology with the right features makes a significant impact on the performance, maintainability, and extensibility of the application.

For any questions about this white paper or benchmark, please contact ITTIA Research and Development using the form at: *http://www.ittia.com/contact*



Interview with

Jeff Roberson

Any administrator who has rushed to bring a system back on-line after a crash knows how frustrating it can be to sit through a filesystem check. It can be a painfully slow, yet necessary process. One BSD developer, Jeff Roberson, has found a way to make all our lives easier and system recovery faster. Jeff took some time out of his very busy schedule to explain some of the bottlenecks in filesystem recovery and how he has gone about speeding up the process.

To start off, could you tell our readers a little about yourself. Where you're from and how you got involved with BSD?

JR: I've been writing software professionally since 1997 and I've been an independent contractor since 2002, the same year I became a FreeBSD committer. Before BSD I did some hacking on Linux but switched and never looked back after taking a job working on a FreeBSD-based product. I wrote the ULE scheduler, UMA kernel memory allocator, parts of the thr threading support, as well as quite a lot of work on SMP scalability. I live on the island of Maui with my wife, dog, and cat but I'm originally from Virginia.

Could you please explain the previous limitations to the FreeBSD filesystem? What was the motivation behind your work?

JR: FreeBSD's FFS (*Berkeley Fast File System*) implementation has a feature called soft-updates which permits filesystem meta-data to be written asynchronously without fear of corruption after an unclean shutdown. While this feature prevents unsafe filesystem states it can potentially leak blocks or inodes in the event of a crash or power loss. To recover from this, a background fsck is run which finds unclaimed blocks and inodes and frees them. This process is relatively expensive and time consuming which detracts from the utility of the near instant boot provided by the ordering guarantees.

You've added a journal to the existing file system, could you please explain how this will help end-users and system administrators?

JR: With the addition of a small intent log the background filesystem check step is eliminated. The fsck program has been augmented to parse the journal and recover only those operations which were pending at the time of the crash. On a large and fully populated filesystem this can reduce fsck time from many hours to few seconds. The recovery time actually scales with the size of the journal and not the size of the filesystem. Bigger filesystems need not take longer to recover. The journal size itself is limited by how many dirty or uncommitted meta-data changes can be in memory at once.

Will your changes be compatible with existing BSD file systems? For instance, will it be possible for people to use your journal on existing file system without re-formatting?

JR: The meta-data changes were very minimal. The journal itself is just a special file in the filesystem. Enabling journaling may be done with tunefs on an unmounted filesystem. Journaling may then be disabled also with tunefs. There are no significant compatibility issues. An older FFS implementation can even fsck and mount a journaled filesystem, although care should be taken to ensure that it is clean when returning to an implementation that does support journaling as there could otherwise be stale recovery information.

Was it difficult to add a journal to the existing filesystem code? Were there moments when you considered starting a new FS from scratch?

JR: While I never considered starting a whole new filesystem, it was indeed very difficult. This took roughly 500 hours to get to a state where it could be committed to head and used by the general public. This is not the simple full-block journaling that many filesystems implement. The recovery operation has to parse the available logs and the current filesystem state and make an determination of what it should be. The journal writes in the kernel had to be ordered properly with other meta-data writes in soft-updates. The end result was nearly 10,000 lines of code which favourably compares to advanced journaling implementations in other operating systems.

To take advantage of your work, will people need to upgrade to FreeBSD-current, or will your work be ported to existing stable versions, ie FreeBSD 8?

JR: There are currently backports in the project's SUJ FreeBSD svn repository. We are still discussing how long these will be maintained and if there will be an official 8.x release with SUJ. A lot of that depends on how stable and performant the code turns out to be in -current. For now I would suggest those that want to try SUJ run current and report any bugs or issues that arise on the freebsd-current email list.

You've mentioned on your blog that some companies (iXsystems and Yahoo) were sponsoring your work. How did that come about?

JR: I had an initial idea of hinting fsck with a journal at BSDCan and discussed it with a few people. I was considering at the time ways that filesystems are limited by sticking to only one meta-data coherency protocol (softdep, journaling, copy-on-write). After I was convinced that the idea would work I produced a proposal and draft of a technical document and began to pursue contributions. Ultimately about half of my time was paid for by corporate sponsors but the benefit to FreeBSD and the experience gained was worth it.

If I understand your latest blog post, you've made recovery from a filesystem crash up to 1,200 times faster. That's a huge improvement. I have to wonder why there wasn't a strong push to do this sooner. What made you be the first?

JR: I can't say exactly that I was the first. The gjournal project by pjd provided full block logging with FFS many years prior. There are certain overheads with full block logging which meant it didn't see as much use as it could, and perhaps should, have. Many were deterred from implementing journaling because it is a big task no matter how you do it. I think several people attempted to make fully journaled versions of FFS but I was probably the first to reduce the scope by integrating with soft-updates rather than replacing it. Many companies have an interest in this type of infrastructure work but lack the expertise or man power to ultimately tackle it.

This was a big task, were you working with anyone else? Other FS gurus or people testing changes with you?

JR: I worked closely with the original author of FFS and soft-updates, Kirk McKusick. He reviewed my changes, and provided design feedback as well as essential details about the rationale and design of the existing mechanisms. Peter Holm was instrumental in testing. His kernel stress suite and new custom tests made this a far more stable and higher-quality system than I could've done alone. Scott Long, Matt Olander, and Debbie Chu should also be thanked for their parts in organizing funding for the project, without which I could not have done it. Many others have provided excellent bug reports and performance feedback.

Do you have any other projects on the go? Other things you'd like to tackle?

JR: I'm just starting a project to bring Infiniband to FreeBSD that will consume some time. One day I would love to have the funding and support required to write a new filesystem from scratch. Until then I spend my free time on other infrastructure projects in FreeBSD.

JESSE SMITH

FreeBSD

Experience and Success Story

In 2007, I was hired as a programmer at University of the Philippines Open University (UPOU). I came from a Microsoft Windows platform and Visual Basic background. At UPOU, there was no room for for my skills since they use various distributions of Linux for servers and open source programming languages for applications development.

Being a new employee (that time), I was never part of server discussions, since I never knew what they were talking about. Fedora, Ubuntu, Debian, NFS, FTP, those were the things I heard, but I was all at lost in those technical terms.

After a few months, I decided to try my hands on those system so that I may be able to join the discussion and be a part of the systems administration meeting and team. I tried my hands on a popular Linux distribution using KDE. I was quite impressed by KDE, but to be honest, I got a lot of system crashes(KDE related) so I decided to look for another one. I came across at this website(I can't remember) with a little devil logo. I clicked the link and it brought me to the freebsd website.

I downloaded the three (3) installation discs of version 6.2 for i386. At first I was afraid to install it since, the installer was not GUI-based, unlike my previous experience with the popular Linux distribution. So this was my second experience of a non-Windows operating system, and my first experience of installing using a non-GUI based installed.

After a couple of failure installations on my laptop, I decided to study the documentation first. I read the handbook to keep me going. And finally, after a lot of disc-switching in my CD-drive, I was able to do a complete installation of FreeBSD 6.2 with KDE as my window manager.

Since then, I used FreeBSD 6.2 for my laptop. To cut the story short, I became a novice FreeBSD user and I joined the FreeBSD Forum to seek more knowledge and skills for systems administration, since that was my dream job (not being a programmer). I was able to get a grasp of how to setup a DHCP, DNS and an FTP server. I was also looking at how the Squid Caching server works through the FreeBSD forum. I also bought the *The Book of PF* by P. N. M. Hansteen, to learn about setting up a firewall, port forwarding and NAT. I also downloaded PDF files from the BSD Magazine website to learn from the articles.

A couple of months ago, our primary gateway computer crashed, leaving our entire organization without a way to connect to the Internet. The whole IT team was required to setup a new one at the soonest possible time. Our system administrator back then, wasn't around and I was the only one with technical know-how to do the job.

I saw the old Sun Ultra 10 Workstation on the server room doing nothing. I decided to use that for the gateway. I downloaded FreeBSD 8.0 for Sparc and immediately installed it. I quickly downloaded the ports collection and installed ISC-DHCP 3.1 server first, then DNSMasq, and Squid. I recompiled the kernel with PF, created pf.conf and tested the system. To no avail, I was able to setup a gateway/firewall, DHCP server, DNS server, and Squid proxy server in less than half a working day.

Our organization was back *online* once again and the management was happy because the setup I made includes a proxy server which can block unwanted web sites. I was able to document my setup and experience. I created a generalized how-to article and submitted it to BSD Magazine for review and publication. I was surprised that my how-to article was published in the April, 2010 issue of the BSD Magazine and I'm very thankful for it. You can download a copy of the how-to in BSD Magazine website.

When our system adminstrator left for another company, I took charge of the entire systems and network and I became the new system administrator. All the thanks to FreeBSD, the members of the Documentation Team, active members of the FreeBSD forum, BSD Magazine, and the author of the Book of PF.

I now enjoy my job as a system administrator and I continue to read the FreeBSD documentation, BSD magazine and read the posts and help others in the FreeBSD forum. Once again, thanks.

JOSHUA EBARVIA

Joshua Ebarvia is a systems administrator, programmer and part-time lecturer at the University of the Philippines Open University. He enjoys working with different operating systems specially the UNIX variants and clones. You can reach him at joshua.ebarvia@gmail.com

In the next issue:

- MidnightBSD
- Ubuntu vs FreeBSD
- and Other !

Next issue is coming in August!

Orion II
⊕
Optional 45 Drive
JBOD Expansion Unit