MAGAZINE

BSD

FOR NOVICE AND ADVANCED USERS

MEMORY FILE SYSTEM IN FREEBSD

INSIDE

VOL.4 NO.8
ISSUE 08/2011(25)
1898-9144

TrueNAS™ Pro Storage Appliance: You are the Cloud

With a rock-solid FreeBSD® base, Zettabyte File System support, and a powerful Web GUI, TrueNAS™ Pro pairs easy-to-manage software with world-class hardware for an unbeatable storage solution.

Expansion Shelves Available

TrueNAS™ 2U Pro System

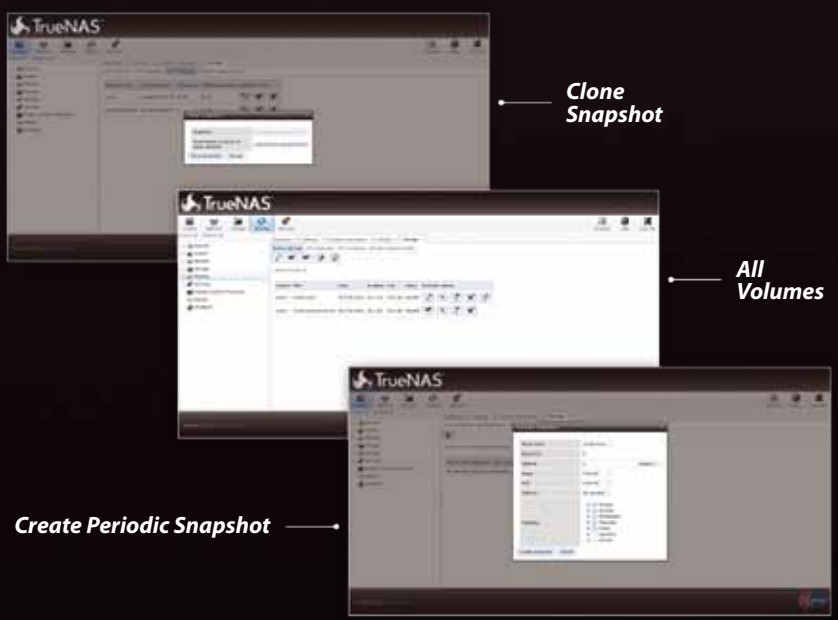
TrueNAS™ 4U Pro System

FreeNAS 8 POWERED

Storage. Speed. Stability.

In order to achieve maximum performance, the TrueNAS™ Pro 2U and 4U Systems, equipped with the Intel® Xeon® Processor 5600 Series, support Fusion-io's Flash Memory cards and 10GbE Network Cards. Titan TrueNAS™ Pro 2U and 4U Appliances are an excellent storage solution for video streaming, file hosting, virtualization, and more. Paired with optional JBOD expansion units, the TrueNAS™ Pro Systems offer excellent capacity at an affordable price.

For more information on the **TrueNAS™ 2U Pro** and **TrueNAS™ 4U Pro**, or to request a quote, visit: **http://www.iXsystems.com/TrueNAS**.



TrueNAS™ 2U PRO KEY FEATURES

- Supports One or Two Quad-Core or Six-Core, Intel® Xeon® Processor 5600 Series
- 12 Hot-Swap Drive Bays - Up to 36TB of Data Storage Capacity*
- Periodic Snapshots Feature Allows You to Restore Data from a Previously Generated Snapshot
- Remote Replication Allows You to Copy a Snapshot to an Offsite Server, for Maximum Data Security
- Up to 4.48TB of Fusion-io Flash Memory
- 2 x 1GbE Network Interface (Onboard) + Up to 4 Additional 1GbE Ports or Single/Dual Port 10GbE Network Cards

TrueNAS™ 4U PRO KEY FEATURES

- Supports One or Two Quad-Core or Six-Core, Intel® Xeon® Processor 5600 Series
- 24 or 36 Hot-Swap Drive Bays - Up to 108TB of Data Storage Capacity*
- Periodic Snapshots Feature Allows You to Restore Data from a Previously Generated Snapshot
- Remote Replication Allows You to Copy a Snapshot to an Offsite Server, for Maximum Data Security
- Up to 14.08TB of Fusion-io Flash Memory
- 2 x 1GbE Network Interface (Onboard) + Up to 4 Additional 1GbE Ports or Single/Dual Port 10GbE Network Cards

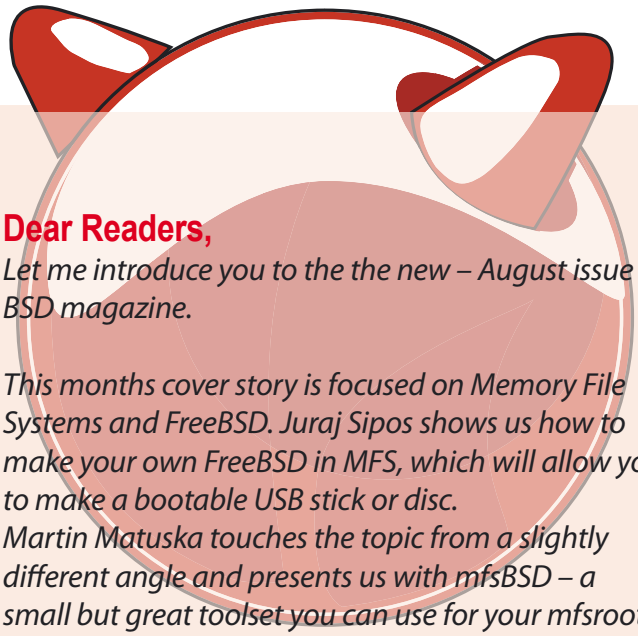
JBOD expansion is available on the 2U and 4U Pro Systems

** 2.5" drive options available; please consult with your Account Manager*

Call iXsystems toll free or visit our website today!

1-855-GREP-4-IX | www.iXsystems.com

Powerful. Intelligent.



Dear Readers,

Let me introduce you to the the new – August issue of BSD magazine.

This months cover story is focused on Memory File Systems and FreeBSD. Juraj Sipos shows us how to make your own FreeBSD in MFS, which will allow you to make a bootable USB stick or disc.
Martin Matuska touches the topic from a slightly different angle and presents us with mfsBSD – a small but great toolset you can use for your mfsroot.

We will also dig deeper into the FreeNAS 8 configuration and use in the article written by Dru Lavigne, explore new options of GIS in third part of Rob Somerville series, and find Paul Ammann's tips to avoid potential security problems with IPv6.

In 'Let's Talk' section Antoine Jacoutot will tell us about challenges of porting GNOME3 to OpenBSD. You will also find a third part of Jim Brown's Certification Series there.

We wish you to enjoy the reading and remember to share your thoughts with us!

Thank you.

Zbigniew Puchciński
Editor in Chief
zbigniew.puchcinski@software.com.pl

MAGAZINE BSD

Editor in Chief:
Zbigniew Puchciński
zbigniew.puchcinski@software.com.pl

Contributing:
Dru Lavigne, Justin C. Sherrill, Martin Matuška, Juraj Sipos, Rob Somerville, Paul T. Ammann, Antoine Bouthors, Jim Brown

Proofreaders:
Sander Reiche, Tristan Karstens

Special Thanks:
Denise Ebery

Art Director:
Ireneusz Pogroszewski

DTP:
Ireneusz Pogroszewski

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

CEO:
Ewa Dudzic
ewa.dudzic@software.com.pl

Production Director:
Andrzej Kuca
andrzej.kuca@software.com.pl

Executive Ad Consultant:
Ewa Dudzic
ewa.dudzic@software.com.pl

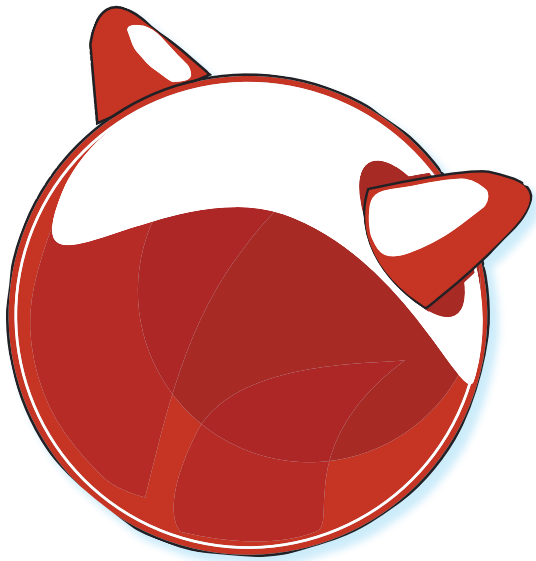
Advertising Sales:
Zbigniew Puchciński
zbigniew.puchcinski@software.com.pl

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 AUPOS

Mathematical formulas created by Design Science MathType™.



Get Started

Developers Corner

How To's

Tips and Tricks

Let's Talk

Getting Started with FreeNAS™ 8.0.1

This article provides a big picture overview of the steps that are performed when configuring a FreeNAS™ 8 storage appliance. Subsequent articles will demonstrate specific configuration scenarios.

What you will learn…

- which sharing services are available in FreeNAS™
- which filesystems and RAID levels are supported by FreeNAS™
- where to create users and setup permissions in FreeNAS™

What you should know…

- what a NAS (network attached storage) is used for

The graphical administrative interface used by FreeNAS™ was redesigned in version 8. If you are used to the .7 interface, this article will demonstrate where to perform configuration tasks using the new interface. If you are new to FreeNAS™, this article will provide you with the workflow that is used when configuring a FreeNAS™ system.

Once FreeNAS™ has been installed and you have logged into the FreeNAS™ administrative interface, the basic configuration workflow is as follows:

- Determine the type of share or service that will be used to provide storage to the clients in the network.
- Create volumes or datasets; in other words, setup the storage disks.
- Create users and groups.
- Assign permissions.
- Configure the share or service.
- Start the service.
- Test the configuration.

An overview of each of these steps is provided.

Installation and Initial Setup

As of this writing, the most recent version of FreeNAS™ is 8.0.1-BETA4. This version is recommended over 8.0-RELEASE as it addresses many of the bugs reported by the community and adds additional features such as cron jobs, rsync, S.M.A.R.T, and UPS support. You can download FreeNAS™ 8.0.1 from *https://sourceforge.net/projects/freenas/files/FreeNAS™-8.0.1/*.

In order to install FreeNAS™, you'll need a USB stick of at least 2 GB in size and a system with at least one hard disk. A minimum of 4 GB of RAM is recommended. A more complete description of the hardware requirements can be found at *http://doc.freenas.org/index.php/Hardware_Requirements*.

The easiest way to install is to download the .iso for your architecture (32- or 64-bit), burn the ISO to a CDROM, and boot from the CDROM to start the installation program. Once the installation menu appears, press enter to select the default option of *1 Install/Upgrade to hard drive/flash device, etc.* The installer will then display all possible disk media available on the system. In the example shown in Figure 1, FreeNAS™ is being installed into VirtualBox which has been prepared with a 4GB virtual disk to hold

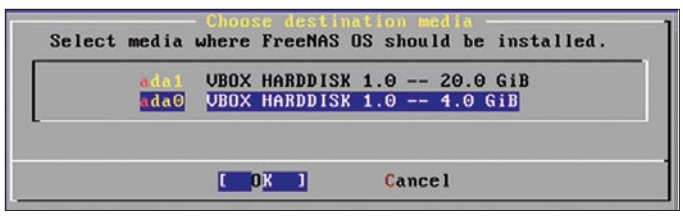


Figure 1. *Choosing Where to Install FreeNAS™*

the operating system and a 20GB virtual disk to provide data storage. Typically, one would install FreeNAS™ onto a USB stick – be sure not to install onto a disk drive as the operating system will *take over* that disk, making it unavailable for data storage, regardless of the disk's size.

Once you select where to install FreeNAS™, you'll receive a warning message reminding you that any existing data on the destination media will be destroyed. Once you confirm your selection, the installer will extract the image to the device and indicate when you can reboot. The entire installation takes only a few minutes.

The first time you boot into FreeNAS™, make sure that it is connected to the network. It will automatically try to obtain an IP address from a DHCP server and will indicate its IP address, as seen Figure 2.

In this example, typing http://10.0.2.15 into a web browser will access the FreeNAS™ graphical login. Input *admin* for the username and *freenas* for the password. You should then have access to the graphical configuration interface shown in Figure 3.

Since the initial FreeNAS™ password is a known value, you should change it by going to A*ccount->My Account-> Change Password*.

Determining Which Type of Share or Service to Configure

FreeNAS™ supports several types of shares and sharing services for providing storage data to the clients in a network. It is recommended that you select only one type to configure in order to prevent possible conflicts between different types of shares. The type of share you choose will depend upon the types of operating systems in your network, your security needs, and your expectations for network transfer speeds. You can choose from the following types of shares and services:

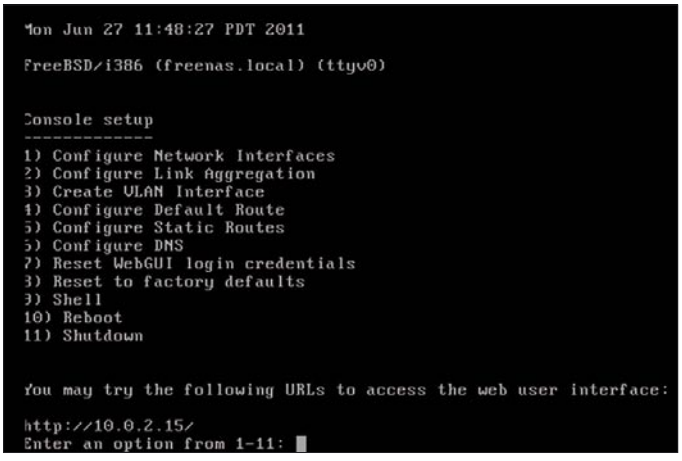


Figure 2. *Determining the IP Address*

AFP

FreeNAS™ uses Netatalk to provide sharing services to Apple clients. This type of share is a good choice if all of your computers run Mac OS X.

CIFS

FreeNAS™ uses Samba to provide the SMB/CIFS sharing service. This type of share is accessible by Windows, Mac OS X, Linux, and BSD computers, but it is slower than an NFS share. If your network contains only Windows systems, this is a good choice.

NFS

This type of share is accessible by Mac OS X, Linux, BSD, and professional/enterprise versions of Windows. It is a good choice if there are many different operating systems in your network.

FTP

This service provides fast access from any operating system, using a cross-platform FTP and file manager client application such as Filezilla. FreeNAS™ supports encryption and chroot for FTP.

SSH

This service provides encrypted connections from any operating system using SSH command line utilities or the graphical WinSCP application for Windows clients.

iSCSI

FreeNAS™ uses istgt to export disk drives that are accessible by clients running iSCSI initiator software.

Creating Volumes and Datasets

When configuring the data storage disks, FreeNAS™ supports the following:

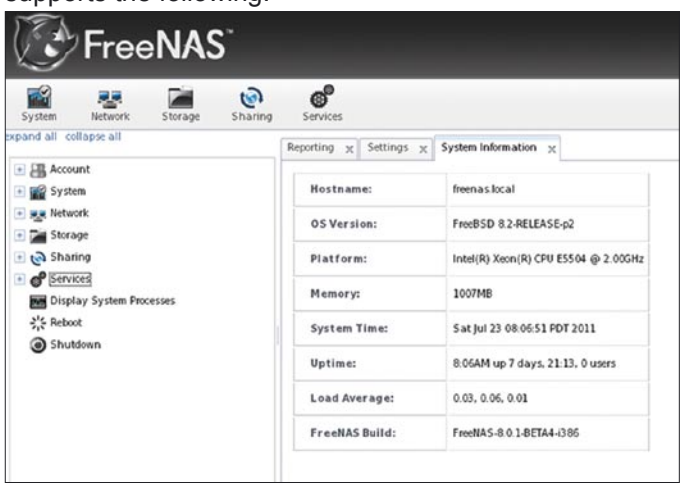


Figure 2. *FreeNAS™ Graphical Configuration Interface*

- The creation of UFS and ZFS volumes.
- The ability to import existing UFS, NTFS, MSDOS, and EXT2 volumes.
- The ability to create or import existing UFS gstripe (RAID0), gmirror (RAID1), and graid3 (RAID3) configurations.
- The ability to create or import existing ZFS RAIDs 0, 1, 5, 10, 60 and RAIDZ1 configurations.
- The ability to create ZFS datasets. Each dataset can be assigned its own user/group, compression level, and disk quota, allowing for complex sharing scenarios.
- The ability to create ZFS zvols. A zvol allows a portion of a volume to be exported as an iSCSI device extent rather than having to export the entire raw disk.
- The ability to create ZFS snapshots which can be used to restore the volume to a specific point in time.
- The ability to prepare a ZFS volume (export) so that it can be removed and installed into another system.

All disk and volume management can be performed in *Storage->Volumes*. Figure 4 shows a screenshot of this menu on a system that has one ZFS volume already created.

In the tree menu in the left frame, one can:

- change the permissions of an existing volume
- auto import an existing software RAID
- create a new UFS or ZFS volume with or without RAID
- create a ZFS dataset (requires a ZFS volume to be created first)
- import a disk containing an existing filesystem
- view all configured volumes

Several configuration icons appear in the right frame. The top 5 icons allow you to (from left to right):

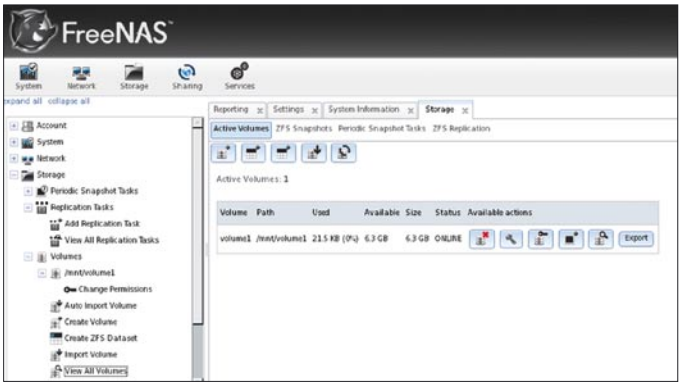


Figure 4. *Volume Management Screen*

- create a new volume
- create a new ZFS dataset
- create a new ZFS volume (zvol)
- import a volume
- auto import a volume

If a ZFS volume has been created, it will contain 6 configuration icons specific to that volume. They allow you to (reading from left to right):

- destroy the volume and all of its data
- edit ZFS options such as compression level and disk quota
- change permissions to the volume
- create a ZFS snapshot
- view the disks that comprise the volume; for each disk you can view its device name, serial number, device ID, tranfer mode, standby mode, power management, acoustic level, and S.M.A.R.T options
- export the ZFS volume

More information about volume management can be found at *http://doc.freenas.org/index.php/Volumes*.

Creating Users and Groups

FreeNAS™ supports a variety of user access scenarios:

- the use of an anonymous or guest account that everyone in the network uses to access the stored data
- the creation of individual user accounts where each user has access to their own ZFS dataset
- the addition of individual user accounts to groups where each group has access to their own volume or ZFS dataset
- support for existing accounts through the OpenLDAP or Active Directory directory service

When configuring your FreeNAS™ system, you need to decide how many users will be accessing the system and if all of the data should be accessible to all of the users. For example, if your home network uses FreeNAS™ to store photos and home videos, you may decide to create a guest account that is used by all family members. This is very easy to setup as it only requires the creation of one volume and one user account. If you require a more complex scenario you will need to create the required number of volumes/datasets and user accounts.

User accounts are created in *Account->Users->Add User*. This opens the screen shown in Figure 5.

When creating a user account, keep the following points in mind:

- the user ID will be automatically generated and can be left as-is
- the username can be descriptive (e.g. guest, anonymous, ftp) or should match an existing username on a client operating system
- the primary group should be left blank as this will create a group with the same name as the username; if you wish to add users to groups in order to manage permissions, use *Account->Groups* to create custom groups and add existing users as members of your custom groups
- *permissions will not work if you do not change the default home directory;* the home directory needs to be changed to the path of the volume or dataset that you wish the user to have access to (e.g. `/mnt/volume1` in the example shown in Figure 4)
- the shell can be left as-is unless the user will be logging in and has a shell preference
- the full name can either be a description (e.g. anonymous access) or the user's full name
- if the user is to have login access, input and confirm their password; otherwise, check the box to disable logins

Setting Permissions

Once you have created your volumes/datasets and users/groups, you need to associate each volume or dataset with the users/groups that will have permission to the data stored on that volume/dataset.

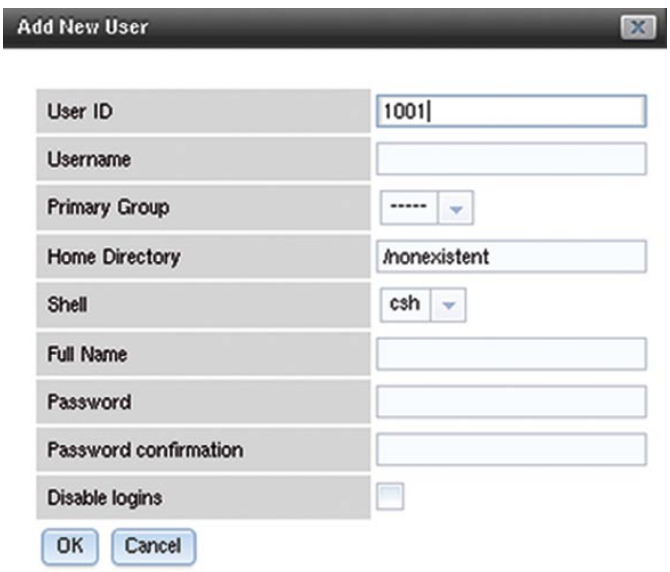


Figure 5. *Creating a User Account*

To do so, go to Storage->Volumes->View All Volumes. Click the permissions icon for the volume or dataset that you wish to configure. This will open the screen shown in Figure 6.

Check that the owner for both user and group is correct and review the mode to ensure that the desired permissions are checked. FreeNAS™ supports both Unix ACLs, which are understood by all operating systems, and Windows ACLs which add a superset of permissions to the Unix ACLs but are only understood by Windows systems.

The only time you should change to Windows ACLs is when your entire network is comprised of Windows operating systems.

You should check the set permission recursively box to ensure that your selected permissions are inherited by all subdirectories within the volume/dataset.

Creating Shares and Configuring Services

Once your volumes have been configured with permissions, you are ready to configure the type of share or service that you determine is suitable for your network. The locations within the administrative interface and the documentation for setting the various types of shares and services are as follows:

- AFP shares are configured in *Sharing->AFP Shares->Add AFP* Share and are described at *http://doc.freenas.org/index.php/AFP_Shares*
- CIFS shares are configured in *Sharing->CIFS Shares->Add CIFS Share* and are described at http://doc.freenas.org/index.php/CIFS_Shares

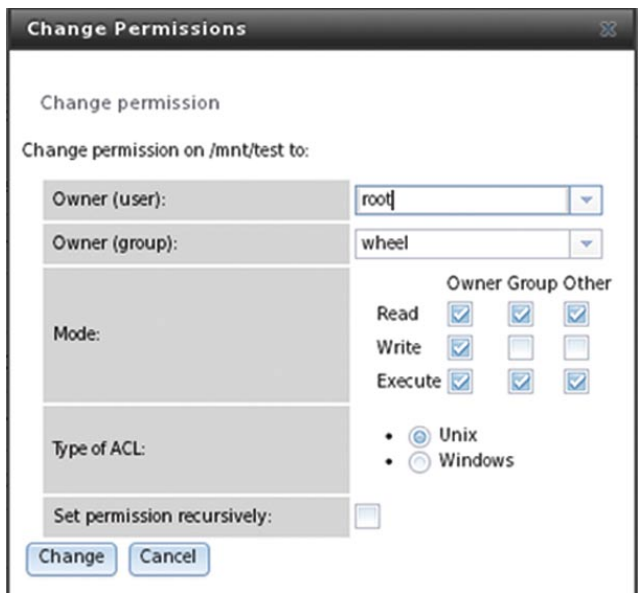


Figure 6. *Setting Permissions*

Figure 7. *Starting the Service*

- NFS shares are configured in *Sharing->NFS Shares->Add* NFS Share and are described at *http://doc.freenas.org/index.php/NFS_Shares*
- the FTP service is configured in *Services->FTP* and is described at *http://doc.freenas.org/index.php/FTP*
- the SSH service is configured in *Services->SSH* and is described at *http://doc.freenas.org/index.php/SSH*
- the iSCSI service is configured in *Services->iSCSI* and is described at *http://doc.freenas.org/index.php/ISCSI*

Starting Services and Testing Your Configuration

Once you have configured your share or service you will need to start its associated service in order to implement your configuration. Before doing so, you should enable console messages within the administrative interface so that you can view any error messages that occur when starting the service. To do so, go to *System->Settings->Advanced*, check the box *Show console messages in the footer (Requires UI reload), and click OK. Refresh your browser and a text console will appear at the bottom of your screen*.

Then, go to Services->Control Services which will open the screen shown in Figure 7.

To start a service, click its red OFF button. After a second or so, it will change to a blue ON, indicating that the service has been enabled. Watch the console messages as the service starts to determine if there are any error messages. If everything appears to be working, try to make a connection to the service from a client system. For example, use Windows Explorer to try to connect to a CIFS share, use an FTP client such as Filezilla to try to connect to an FTP share, or use Finder on a Mac OS X system to try to connect to an AFP share.

If the service starts correctly and you can make a connection but receive permissions errors, check that the user has permissions to the volume/dataset being accessed.

Summary

This article described the basic workflow when configuring a FreeNAS™ system. It demonstrated how to: install version 8.0.1, determine the type of sharing service that is appropriate for your network's needs, create volumes and datasets to manage storage, create users and groups to manage permissions, and configure and test a sharing service.

The next article in this series will demonstrate some common configuration scenarios.

DRU LAVIGNE

Dru Lavigne is author of BSD Hacks, The Best of FreeBSD Basics, and The Definitive Guide to PC-BSD. As Director of Community Development for the PC-BSD Project, she leads the documentation team, assists new users, helps to find and fix bugs, and reaches out to the community to discover their needs. She is the former Managing Editor of the Open Source Business Resource, a free monthly publication covering open source and the commercialization of open source assets. She is founder and current Chair of the BSD Certification Group Inc., a non-profit organization with a mission to create the standard for certifying BSD system administrators, and serves on the Board of the FreeBSD Foundation.

a-team systems

FreeBSD Server Specialists

Over 15 Years of Professional FreeBSD® Experience!

http://www.ateamsystems.com/

Full Server Management

- Support for Web & DB Servers
- Troubleshooting
- OS & Security Updates
- Encrypted Off-Site Backups
- Scripting & Automation

Server Monitoring

- Custom Tailored Plans
- 24x7 On Call Available
- Performance Trends
- Report Email Monitoring
- Kernel Log & HW Monitoring

We are a FreeBSD focused shop:
FreeBSD is not just another bullet point for us!

Your BSD 'App Store' with pbulk:

Building everything in pkgsrc with automation using DragonFlyBSD

The „app store" concept seemingly is the latest fad with Apple, Google, Amazon, Valve and so on. Each of them creating individual platforms for moving software to incidental customer gadgets completely through electronic download. Well, as with many other technologies, a pioneer of this software technology trend first showed up as an open source concept.

The resemblances to the app store concept are striking! FreeBSD's ports system, OpenBSD's ports, and NetBSD's pkgsrc are all similar systems, designed for *incidental* BSD users to wander through and pick out desired software.

These systems, for those who are unfamiliar with them, are a collection of files and 3rd-party software set up to automatically download, build, and install on your favorite BSD machine.

We will focus on Pkgsrc. While it originated on NetBSD, can run on a large variety of UNIX-like computers. (may be placed here: *The instructions of this article are specific to DragonFly BSD. It may vary with the many other platforms supported for pkgsrc.*)

What can one do with pkgsrc? Many things, but this article is just going to focus on one use: building it all. Generously, Joerg Sonnenberger's pbulk program can automate the entire package bulk process for you.

Over Nine Thousand!

As of this writing, pkgsrc has 10,480 packages in the most recent release. Releases happen on a quarterly basis, though it's possible to follow it in day-to-day development if you don't mind the occasional problems.

That's a lot of software to deal with. Building every last package will show just where the problems, are, however, and the resulting binary package can be used later for installing a package without having to go through a building process. This is especially popular with people who don't have a lot of processing power, or patience.

An individual package can be built in pkgsrc by navigating to the correct folder inside pkgsrc – net/wireshark for Wireshark, for instance, or editors/vim for Vim – and typing *bmake install*. bmake is NetBSD's flavor of make, and is installed as part of the pkgsrc bootstrap process, which we aren't covering here.

Typing *bmake install* 10,480 times and capturing the results is a bit more hassle than most people want to endure. Joerg Sonnenberger's pbulk program will automate the entire process for you. It's available in pkgsrc under pkgstools/pbulk.

To keep from spraying thousands of packages onto your existing computer, pbulk and all these programs can be installed into a (separate) chroot (environment). I'm using DragonFly, so these instructions are specific to DragonFly. It may vary with the many other platforms supported for pkgsrc.

Notice that the chroot installation process as shown in Listing 1, actually puts you into the chroot, where all commands happen from here on. It also starts up linux emulation, which is not necessary but can be useful for some packages.

Now that you are in the chroot, there are several things to put in place. First, create directories to hold the results of the build, such as reports and binary packages (Listing 2).

The next step is to download the pkgsrc files, using cvs (Listing 3). It's also possible to grab a tarball of the files for a slightly less complex step. 'setenv' applies only to csh users; set these environment variables in a way appropriate for your shell.

Note that this setup is downloading what is, at the time of this writing, the most recent quarterly release of pkgsrc: pkgsrc-2011Q2. If you omit the -r argument completely, you get pkgsrc as it looks that minute, which may mean you are downloading some packages as they are being updated. You may have a slightly lower success rate building those packages, but it will be the most up-to-date. Your mileage may vary.

Edit the file `/root/mk-base.conf`, for setting up your custom pkgsrc bootstrap (Listing 4). Your building will be done under a *pbulk* user identity, so make sure to create that user in the chroot, and set WRKOBJDIR to a writeable location for that user.

Next, install pkgsrc's tools within this chroot, and then install a separate set for pbulk to use. (Listing 5)

The last step is to configure `/usr/pkg_bulk/etc/pbulk.conf` so that the report and generated binaries are uploaded, if you want to upload them to a public location. By default, pbulk uses rsync to copy the data around.

Make sure the other settings look correct for your particular setup. If you aren't planning to place this on a public site or only want to deal with an email report, report_recipients is the only setting you need to configure.

Listing 1. *Building a chroot.*

```
mkdir /build/pbulk_chroot
cd /usr/src
make DESTDIR=/build/pbulk_chroot buildworld
make DESTDIR=/build/pbulk_chroot installworld
cd etc
make DESTDIR=/build/pbulk_chroot distribution
cp /etc/resolv.conf /build/pbulk_chroot/etc
kldload linux.ko
chroot /build/pbulk_chroot
/etc/rc.d/ldconfig start
mount_devfs /dev
```

Listing 2. *Directory creation in the chroot*

```
mkdir /bulklog
mkdir /scratch
mkdir /distfiles
mkdir /packages
```

Listing 3. *Installing pkgsrc files*

```
setenv CVSROOT anoncvs@anoncvs.NetBSD.org:/cvsroot
setenv CVS_RSH ssh
cd /usr
cvs -q checkout -rpkgsrc-2011Q2 -P pkgsrc
```

Listing 4. *basic configuration changes*

```
KOBJDIR = /scratch
PKGSRCDIR = /usr/pkgsrc
DISTDIR = /distfiles
PACKAGES = /packages

FAILOVER_FETCH           = yes

SKIP_LICENSE_CHECK       = yes
ALLOW_VULNERABLE_PACKAGES= yes
WRKOBJDIR               ?= /tmp/
```

Listing 5. *Installing pkgsrc tools*

```
mkdir /usr/pkg_bulk/
cd /usr/pkgsrc/bootstrap
./bootstrap --prefix /usr/pkg_bulk --pkgdbdir /usr/pkg_bulk/.pkgdb

cd /usr/pkgsrc/pkgtools/pbulk
env PATH=/usr/pkg_bulk/bin:/usr/pkg_bulk/sbin:${PATH} bmake package

cd /usr/pkgsrc/bootstrap
./cleanup
```

Listing 6. *Example lines to change in pbulk.conf*

```
base_url=http://something/you_should_change_here

pkg_rsync_args="-av --delete-excluded"
pkg_rsync_target="/archive/packages/DragonFly-2.10/pkgsrc-current"
report_rsync_args="-avz --delete-excluded"
report_rsync_target="/archive/packages/DragonFly-2.10/pbulk_report"

report_recipients="you@your.email"
```

Run `/usr/pkg_bulk/bin/bulkbuild`, and that's it. Building all packages will probably take days, so it's wise to do this on a computer that doesn't get powered down frequently, and within a persistent terminal like screen or tmux. If something goes wrong, `/usr/pkg_bulk/bin/bulkbuild-restart` will restart the building process.

For an example of the reports in web format, visit: *http://avalon.dragonflybsd.org/reports/* and drill down by version and architecture. Look for the *meta* folder, and the report.html file in there will show you the exact output format, including what failed build has the most dependencies.

The end result

Congratulations! After some days, you'll have a lot of software! These can be installed directly using `pkg_add` from pkgsrc, much faster than building individually from source. You'll also have a comprehensive list of what pkgsrc items do not build on your platform, to supply to the proper people for fixing. Or, if you're an extremely helpful person, a to-fix list you can tackle yourself.

JUSTIN C. SHERRILL

Justin Sherrill has been publishing the DragonFly BSD Digest since 2004, and is responsible for several other parts of DragonFly that aren't made out of code. He lives in the northeast United States and works over a thousand feet underground.

mfsBSD

The Swiss Army Knife for FreeBSD system administrators

mfsBSD is a toolset to create small-sized but full-featured mfsroot based distributions of FreeBSD that store all files in memory (MFS) and load from hard drive, usb storage device, optical media or network.

What you will learn…

- What is mfsBSD and how you can use it
- How to build your own mfsBSD
- How to setup a network boot with mfsBSD

What you should know…

- FreeBSD system administration basics
- Computer networking basics

It can be used for a variety of purposes, including diskless systems, recovery partitions and remotely overwriting other operating systems.

History

The idea behind mfsBSD originates from a script set called *depenguinator*, created by Collin Percival in December 2003. The goal of depenguinator was to allow installing FreeBSD on dedicated servers that pre-install only Linux distributions. The customly crafted image file was written to the boot area of the boot disk and after rebooting a FreeBSD system was fully loaded into memory. Afterwards, the user could partition and format disk drives and install FreeBSD the way he liked. The original depenguinator was created for FreeBSD 5.x. This version was incompatible with FreeBSD 6.x and 7.x. An updated Depenguinator (2.0) was released in January, 2008. Depenguinator is designed to create FreeBSD disk images on Linux computers and uses makefs from NetBSD.

The mfsBSD project was started by myself in late 2007, trying to make a depenguinator-like tool that works on FreeBSD 6.x. The approach was significantly simplified utilizing new features of the rcng startup system and in addition the `geom_uzip` class was used to compress the `/usr` filesystem. To save even more space, the latest version uses lzma compression in several places. mfsBSD requires FreeBSD to create images and unlike depenguinator, it can create ISO images and tar-gz compressed distribution files too. As the ZFS file system gained popularity, a shell script script called zfsinstall was added to mfsBSD to easily deploy ZFS-only installations of FreeBSD.

In April 2008, Daniel Gerzo published a mfsBSD-related article called *Remote Installation of the FreeBSD Operating System without a Remote Console* that is now a part of the FreeBSD documentation collection.

Availability

mfsBSD can be downloaded from the mfsBSD webpage (*http://mfsbsd.vx.sk*).

There are several downloadable ISO-files for the i386 and amd64 architectures. Files called special edition contain an installable image of FreeBSD. To be able to call the tool a Swiss Army Knife for FreeBSD system administrators, the standard ISO files are supplied with the following additional packages: cpdup, dmidecode, e2fsprogs, ipmitool, nano, rsync, smartmontools, tmux.

Advanced users can download the source code and build their very own distribution (and e.g. add custom packages to it).

Use cases

mfsBSD can be used in a variety of ways. I would like to list some of the areas where I use it in:

- Rescue System (with full ZFS support).
- Installation of FreeBSD (focus on ZFS-only installs).
- Diskless boot over network.
- Transforming a running Linux installation into FreeBSD.

Rescue System

There is no perfect hardware, there is no perfect software and there is us, system administrators, who occasionally make mistakes. Some of these errors or mistakes may prevent a FreeBSD system from booting. In this case, mfsBSD can be booted from a partition on the harddrive, an USB-stick, CD-ROM media, over the network (PXE) or from any other block device supported by FreeBSD. It provides near full functionality of FreeBSD (including some additional packages) and the damaged installation can be repaired, broken configuration scripts fixed or boot drives repartitioned. The maintenance tasks can be performed via local console, over the network using SSH or after making some adjustments via a serial connection. The default setup contains a `mfsbsd_autodhcp` setting that puts all discovered network cards into DHCP mode. The OpenSSH daemon is automatically started and enables a remote login.

FreeBSD Install

The special edition ISO images contain a single file that includes a complete FreeBSD installation. Extracting this file to a bootable partition and adjusting the `/etc/rc.conf` file creates a complete and fully working installation of FreeBSD.

One of the main mfsBSD features is support for simplified ZFS-on-root installations, provided by a script called zfsinstall. This script does everything for the user, from creating GPT partitions and the ZFS pool (supporting single-disk, mirror and raidz modes) to deploying and correctly configuring a new FreeBSD installation.

Diskless boot over network

Another ability of mfsBSD is to boot over network. Users may create their own, custom distributions that boot from the network and do not require a hard drive. I have successfully booted mfsBSD using the FreeBSD's pxeboot boot loader, standalone or chained via pxelinux (sysutils/syslinux). Later in this article I am providing a configuration example of a PXELINUX chained mfsBSD with a nice boot menu. This combination may be very useful for dedicated server providers – as many of these use the PXELINUX environment for their rescue and installation systems, they can easily add FreeBSD image deployment and rescue system support to their servers.

The german dedicated server provider hetzner.de already uses mfsBSD for their FreeBSD support. There may be other providers, too.

Requirements

To run mfsBSD from a standard ISO-Image, an i386 or amd64 system with at least 192MB RAM is required. For testing purposes VirtualBox can be used.

To build mfsBSD on your own, an existing FreeBSD installation version 8.x or higher is required (7.x is not supported anymore). To create an ISO image, mkisofs from the sysutils/cdrtools port must be installed. To speed up xz compression, sysutils/pxz is recommended.

Building your own mfsBSD

To build your own mfsBSD images, you need to download the source code from the project website. mfsBSD uses a Makefile-based build system, so basically all you have to do is run *make* with various flags and options. It is advised to read the documentation and then to adjust your configuration files and add custom packages. All scripts are easy to customize – advanced users and system administrators can add their own functionality as well.

The toolset supports the creation of three types of output images:

- bootable raw images – useful for overwriting Linux installations
- deployable tar.gz files – useful for network boot (PXE)
- bootable ISO images – useful for booting from optical media or via management consoles supporting ISO emulation

Example commands to build a custom mfsBSD image:

The mfsBSD build process

- Preparation of kernel and base distribution – kernel and base distribution files are extracted from a FreeBSD CD-ROM (may be a mounted ISO image) or are optionally custom-built and/or installed (make installworld + make installkernel + make distribution) – for the special edition, the distribution file is created.
- Removal of unnecessary files – to keep mfsBSD small, a (user-adjustable) list of files is removed from the distribution.
- Processing of configuration files – required and optional configuration files are processed and installed to the target image.
- Compression of the `/usr` filesystem – to save space, the `/usr` directory is compressed (ideally using xz).
- Build of mfsroot – a mfsroot image is built.
- Creation of a deployable output image.

```
# cd /usr/src
# make buildworld && make buildkernel
# cd ~
# fetch „http://mfsbsd.vx.sk/release/mfsbsd-1.1.tar.gz”
# tar xfz mfsbsd-1.1.tar.gz
# cd mfsbsd-1.1
# make iso COMPRESS=xz CUSTOM=1
```

Tutorial: Installing ZFS-on-root FreeBSD with mfsBSD

mfsBSD can be used to install a bootable ZFS-on-root FreeBSD – the whole system will be on ZFS.

The magic is done by the supplied tool called *zfsinstall*.

What do you need:

- A system with (at least one) bootable and writable block device (e.g. hard drive).
- A booted mfsBSD image (e.g. CD-ROM or network boot).
- Access to the distribution file from mfsBSD special edition (see mfsBSD homepage).

Instructions:

Log in to mfsbsd as root via SSH or local console (default password: mfsroot). After booting mfsBSD, check if your hard drives contain GPT partitions with the gpart command. The drive(s) need to be clean before installing. The zfsinstall script will not install if it finds any problems.

```
# gpart show
```

If your target drive(s) already contain partitions, they need to be cleaned. The destroygeom script does the job:

```
# destroygeom -d ad0
```

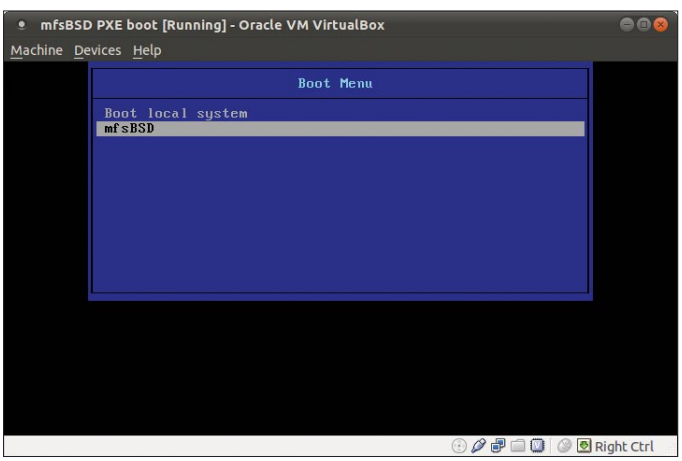


Figure 1. *mfsbsd pxeboot*

You cannot have a ZFS pool on your system with the same name as the pool you want to install. You have to destroy it first or choose a different name with the -p flag.

If you booted from an ISO file (optical drive, management console), you can mount your optical drive to gain access to the distribution tarfile. Otherwise you will have to obtain the tarfile from other sources (NFS, SCP, etc.)

```
# mount_cd9660 /dev/acd0 /cdrom
```

Now you can use the zfsinstall install script to install FreeBSD on your drive(s). The script can create a GPT swap partition, supports single-drive, mirrored and raidz installs and has many other options. Running the script without any options gives you a list of supported options. Option -h delivers a detailed help screen. Examples:
Single-drive install with a 2GB swap partition:

```
# zfsinstall -d ad0 -t /cdrom/8.2-RELEASE-amd64.tar.xz -s 2G
```

Two-drive mirror with a 1GB swap partition on each drive and lzjb compression enabled:

```
# zfsinstall -r mirror -d ad0 -d ad1 -t /cdrom/
   8.2-RELEASE-amd64.tar.xz -s 1G -c
```

If no errors occured, your installation should be ready. You can adjust `/mnt/etc/rc.conf` and/or chroot to `/mnt` and set the root password. If you want to SSH to the newly created system I recommend enabling root login in `/mnt /etc/ssh/sshd _ config`. The nano editor is bundled with all mfsBSD ISO files available from the webpage.

```
# chroot /mnt
# passwd
# vi /etc/rc.conf
# vi /etc/ssh/sshd_config
```

Tutorial: Setting up network boot with mfsBSD (pxeboot and PXELINUX)

In this tutorial I am going to show how to boot mfsBSD over the network.

This tutorial expects the DHCP server (ISC) and TFTP server to be on a i386/amd64 FreeBSD computer (server) using the same IP adress. If this is not the case, configuration may differ, you have to adjust it on your own.

What do you need:

- A boot server (DHCP, TFTP) with FreeBSD sources
- mfsBSD (ISO from webpage or self-built)

- some files from an installed sysutils/syslinux port (if you want pxelinux)
- a network bootable (virtual) machine for testing and demonstration

How to prepare the boot server (DHCP, TFTP)

The following examples expect a 192.168.0.0/24 network with gateway and dns-server at 192.168.0.1. If you are using different network parameters, you'll have to alter the configuration accordingly.

Install the FreeBSD sources into `/usr/src`.

Install the port: `net/isc-dhcp31-server` or `net/isc-dhcp41-server`.

Example of a minimal `/usr/local/etc/dhcpd.conf`:

```
authoritative;
subnet 192.168.0.0 netmask 255.255.255.0 {
   range 192.168.0.100 192.168.0.200;
   option routers 192.168.0.1;
   option domain-name-servers 192.168.0.1;
   next-server 192.168.0.1;
   filename „pxeboot”;
}
```

You may adjust the options to your needs and/or add another options.

Enable the tftp daemon in your `/etc/inetd.conf` by uncommenting the following line:

```
tftp   dgram   udp  wait  root  /usr/libexec/tftpd  tftpd
                    -l -s /tftpboot
```

Enable dhcpd and inetd by adding the following two lines to `/etc/rc.conf`:

```
inetd_enable="YES"
dhcpd_enable="YES"
```

Create the `/tftpboot` directory:

```
# mkdir -p /tftpboot
```

Extract the contents of the mfsBSD ISO image to `/tftpboot`

```
# tar -x -C /tftpboot -f mfsbsd-8.2-amd64.iso
# chmod -R go=u-w /tftpboot/*
# chmod u+w /tftpboot/boot/loader.conf
```

The standard FreeBSD `/boot/pxeboot` uses NFS instead of TFTP. In this tutorial, we are going to prepare a tftp-enabled loader. If you want to use the standard loader, you'll have to set-up a NFS server on the same IP address.

```
# mv /boot/pxeboot /boot/pxeboot.orig
# cd /usr/src/sys/boot
# make clean && make depend && make -DLOADER_TFTP_SUPPORT
# cd /usr/src/sys/boot/i386/pxeldr
# make install
# mv /boot/pxeboot /tftpboot/pxeboot
# mv /boot/pxeboot.orig /boot/pxeboot
```

Start the inetd and dhcpd daemons

```
# /etc/rc.d/inetd start
# /usr/local/etc/rc.d/isc-dhcpd start
```

Your boot setup is ready, you can boot from network now! The file `/tftpboot/boot/loader.conf` can be customized to your needs.

VirtualBox with a bridged ethernet interface is a fast and easy way to test if your setup is working. For PXE boot support, you need the VirtualBox Extension Pack installed (see VirtualBox website) Just press the [*F12*] key

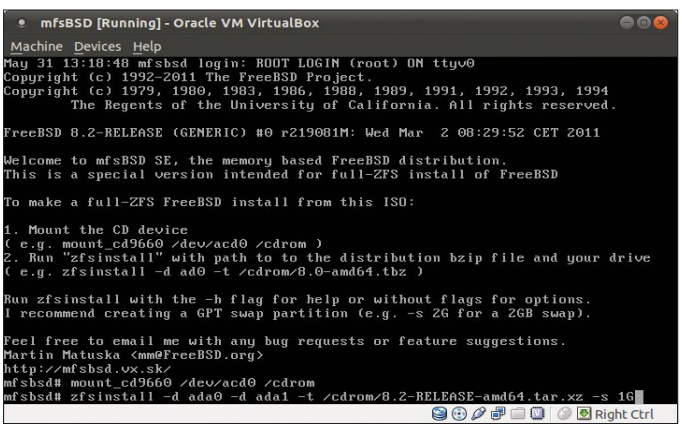


Figure 2. *zfsinstall virtualbox*

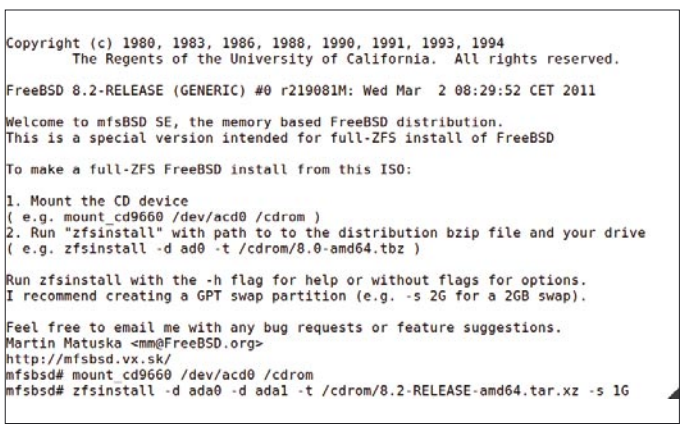


Figure 3. *zfsinstall virtualbox shell*

at virtual machine bootup and then press the [*L*] key (boot from LAN).

Intelligent boot loader: mfsBSD boot with PXELINUX support

With PXELINUX, you get a nice boot menu and the ability to configure individual boot options for different computers or whole subnets. Booting from the local harddrive into multiple operating systems or different mfsBSD versions (e.g. i386 and amd64) is supported, too. If you want to boot via PXELINUX, follow all the steps in the previous section (*Preparing the boot server*) and continue with the following instructions:

- Install the port: sysutils/syslinux (you need only two files from the port, no need to keep it).
- Copy `pxelinux.0` and `menu.c32` to the `/tftpboot` directory:

```
# cp /usr/local/share/syslinux/pxelinux.0 /tftpboot/pxelinux.0
# cp /usr/local/share/syslinux/menu.c32 /tftpboot/menu.c32
```

- Rename pxeboot to `pexboot.0`:

```
# mv /tftpboot/pxeboot /tftpboot/pxeboot.0
```

- Symlink `pxelinux.0` to pxeboot (we are now booting pxelinux):

```
# ln -s pxelinux.0 /tftpboot/pxeboot
```

- Create the `/tftpboot/pxelinux.cfg` directory:

```
# mkdir /tftpboot/pxelinux.cfg
```

- Create `/tftpboot/pxelinux.cfg/default`, you may use the following example:

```
default menu.c32
allowoptions 0
totaltimeout 100

menu title Boot Menu

label local
   menu label Boot local system
   localboot 0

label mfsbsd
   menu label mfsBSD
   menu default
   KERNEL pxeboot.0
```

On the 'Net

- *http://mfsbsd.vx.sk* – mfsBSD homepage
- *http://blog.vx.sk* – author's blog with focus on ZFS and FreeBSD
- *http://syslinux.zytor.com* – Syslinux homepage with PXELINUX documentation
- *http://www.virtualbox.org* – VirtualBox homepage

The parameter `totaltimeout 100` fires the default option if no selection is made in 10 seconds.
The parameter `allowoptions 0` disables the TAB-key in the boot-prompt.

- Now you are ready to boot from chained PXELINUX with a nice boot menu!

Alternative: NFS

To boot from NFS instead of TFTP, you'll need a working NFS server and the standard pxeboot.

Copy the standard pxeboot from your `/boot` directory to `/tftpboot`:

```
# cp /boot/pxeboot /tftpboot/pxeboot
```

Add the following to the subnet section of `/usr/local/etc/dhcpd.conf`:

```
   option root-path „192.168.0.1:/tftpboot”;
```

As of this example, the NFS server runs on 192.168.0.1 and the `/tftpboot` directory must be NFS-exported (`/etc/exports`).

Alternative: DHCP on a separate server

If your TFTP server is on a different server than your DHCP server, you need change following line in the subnet section of `/usr/local/etc/dhcpd.conf`:

```
   next-server 192.168.0.2;
```

In this example, your TFTP server is at 192.168.0.2.

MARTIN MATUŠKA

Martin Matuška (mm@FreeBSD.org) is an IT manager, systems administrator and FreeBSD committer. He is the founder of the mfsBSD project, part of the FreeBSD ZFS team and maintainer of several FreeBSD ports. He is managing a system administration company VX Solutions s. r. o. (http://www.vx.sk) with focus on deploying and maintaining ZFS systems and providing solutions based on FreeBSD and Solaris family operating systems. He writes at http://blog.vx.sk.

FreeBSD Mall

Your FreeBSD & PC-BSD Resource

www.FreeBSDMall.com

FreeBSD 8.2 Jewel Case CD/DVD

Set contains:

- **Disc 1**: Installation Boot (i386)
- **Disc 2**: LiveFS (i386)
- **Disc 3**: Essential Packages (i386)
- **Disc 4**: Essential Packages (i386)

FreeBSD 8.2 CD $39.95
FreeBSD 8.2 DVD $39.95
FreeBSD 7.4 CDROM $39.95
FreeBSD 7.4 DVD $39.95

FreeBSD Subscriptions

Save time and $$$ by subscribing to regular updates of FreeBSD!

FreeBSD Subscription , start with CD 8.2 ..... $29.95
FreeBSD Subscription, start with DVD 8.2 ..... $29.95
FreeBSD Subscription, CD 7.4 ..... $29.95
FreeBSD Subscription, DVD 7.4 ..... $29.95

PC-BSD 8.2 DVD (Hubble Edition)

PC-BSD 8.2 DVD $29.95
PC-BSD Subscription $19.95

BSD Magazine

BSD Magazine $11.99

The FreeBSD Handbook

The FreeBSD Handbook, Volume 1 (User Guide) $39.95
The FreeBSD Handbook, Volume 2 (Admin Guide) $39.95
☆ Special: The FreeBSD Handbook, Volume 2 (Both Volumes) $59.95
☆ Special: The FreeBSD Handbook, Both Volumes, & FreeBSD 8.2 ..... $79.95

The FreeBSD Bundle

Inside the Bundle, you'll find:

- FreeBSD Handbook, 3rd Edition, Users Guide
- FreeBSD Handbook, 3rd Edition, Admin Guide
- FreeBSD 8.2 4-disc set
- FreeBSD Toolkit DVD

☆ Special: The FreeBSD CD Bundle $99.95
☆ Special: The FreeBSD DVD Bundle $99.95

The FreeBSD Toolkit DVD $39.95

FreeBSD Mousepad $10.00

FreeBSD Caps $20.00

PC-BSD Caps $20.00

For *MORE* FreeBSD & PC-BSD items, visit our website at FreeBSDMall.com!

CALL 925.240.6652 Ask about our software bundles!

How To Make Memory File System In FreeBSD

Memory File System is a very good option for your own – customized FreeBSD system. Soon after FreeBSD CD (or USB) boots with MFS, it loads all the necessary files from the root directory in memory the same way as if these files were in the root directory on your hard disk (usually /dev/ad0s1a).

What you will learn…

- How to make your own customized FreeBSD in MFS
- How to make your own bootable CD
- How to make your own bootable USB stick (disk)

What you should know…

- Some knowledge on mdconfig
- Some knowledge on mkisofs

This article will help readers make their own MFS (Memory File System) that can be used for building a bootable CD or USB. Tips on how to make a bootable CD or USB are included, too.

If you have the following code in your `/boot/loader.conf` on your CD, your CD, which you made bootable with `mkisofs`, will use its root (`/`) directory upon booting (all directories in/will be in one mountpoint – that is, in memory) from the `mfsroot.gz` file:

```
#/boot//loader.conf

mfsroot_load="YES"
mfsroot_type="mfs_root"
mfsroot_name="/boot/mfsroot"
```

You must have the `mfsroot.gz` file in your `/boot` directory on your CD (or USB stick) and the system, after booting, will then exclusively run in memory. This means that all directories will be writable and you will not receive *read-only system* warnings. To make a bootable CD, run the following command (in the directory of your choice, but with files you want to have in the ISO image):

```
mkisofs -R -b boot/cdboot -no-emul-boot \
     -c boot.catalog -boot-load-size 4 -o /mnt/
                    bootableiso.iso .
```

The minimal requirement is to include your `/boot` directory (copied from a physical FreeBSD partition) into the ISO image. However, before you run the `mkisofs` command, you must prepare MFS.

Terminology

In this text, when a reference to a file put onto a CD (ISO) is made, it must be clear that such a file is physically on

Listing 1. *Files needed from /usr/bin*

```
alias, at, atq, atrm, awk, basename, bsdtar, bunzip2,
 bzip2, bzip2recover, cap_mkdb, chfn, chgrp, chkey,
chpass, chsh, cmp, csplit, cu, dialog, diff, du, edit,
env, false, find, ftp, getopt, grep, gunzip, gzip, id,
killall, ld, ldd, locale, logger, login, logins, mkfifo,
mktemp, mkuzip, more, msgs, nc, netstat, newgrp, nice,
nohup, opieinfo, opiekey, opiepasswd, passwd, patch,
pgrep, pkill, printf, procstat, read, readlink, reset,
rlogin, rsh, scp, script, sdiff, sed, sftp, slogin,
sockstat, split, ssh, ssh-add, ssh-agent, ssh-keygen,
su, tar, tip, top, touch, tput, tr, true, tset, tsort,
tty, umask, unalias, uname, uptime, users, vmstat, w,
wait, wc, who, ypcat, ypchfn, ypchpass, ypchsh, ypmatch,
yppasswd, ypwhich
```

Listing 2. *Files needed from /usr/lib*

```
cd /copy/usr/lib # we expect that /copy/usr/lib
                   already exists
mkdir aout
mkdir compat
cd /copy
cp -RP /usr/lib/libalias.so ./usr/lib
cp -RP /usr/lib/libarchive.so* ./usr/lib
cp -RP /usr/lib/libasn1.so* ./usr/lib
cp -RP /usr/lib/libauditd.so* ./usr/lib
cp -RP /usr/lib/libbsdxml.so* ./usr/lib
cp -RP /usr/lib/libbsm.so* ./usr/lib
cp -RP /usr/lib/libbz2.so* ./usr/lib
cp -RP /usr/lib/libcom_err.so* ./usr/lib
cp -RP /usr/lib/libcrypt.so ./usr/lib
cp -RP /usr/lib/libcrypto.so ./usr/lib
cp -RP /usr/lib/libcurses.so ./usr/lib
cp -RP /usr/lib/libcursesw.so ./usr/lib
cp -RP /usr/lib/libdevinfo.so* ./usr/lib
cp -RP /usr/lib/libdialog.so* ./usr/lib
cp -RP /usr/lib/libdialog.so* ./usr/lib
cp -RP /usr/lib/libdwarf.so* ./usr/lib
cp -RP /usr/lib/libelf.so* ./usr/lib
cp -RP /usr/lib/libfetch.so* ./usr/lib
cp -RP /usr/lib/libform.so* ./usr/lib
cp -RP /usr/lib/libformw.so* ./usr/lib
cp -RP /usr/lib/libftpio.so* ./usr/lib
cp -RP /usr/lib/libgcc_s.so ./usr/lib
cp -RP /usr/lib/libgnuregex.so* ./usr/lib
cp -RP /usr/lib/libgomp.so* ./usr/lib
cp -RP /usr/lib/libgpib.so* ./usr/lib
cp -RP /usr/lib/libgssapi.so* ./usr/lib
cp -RP /usr/lib/libgssapi_krb5.so* ./usr/lib
cp -RP /usr/lib/libgssapi_ntlm.so* ./usr/lib
cp -RP /usr/lib/libgssapi_spnego.so* ./usr/lib
cp -RP /usr/lib/libhdb.so* ./usr/lib
cp -RP /usr/lib/libheimntlm.so* ./usr/lib
cp -RP /usr/lib/libhistory.so* ./usr/lib
cp -RP /usr/lib/libhx509.so* ./usr/lib
cp -RP /usr/lib/libkadm5clnt.so* ./usr/lib
cp -RP /usr/lib/libkadm5srv.so* ./usr/lib

cp -RP /usr/lib/libkiconv.so ./usr/lib
cp -RP /usr/lib/libkrb5.so* ./usr/lib
cp -RP /usr/lib/libkvm.so ./usr/lib
cp -RP /usr/lib/liblwres.so* ./usr/lib
cp -RP /usr/lib/libmagic.so* ./usr/lib
cp -RP /usr/lib/libmemstat.so* ./usr/lib
cp -RP /usr/lib/libmenu.so* ./usr/lib
cp -RP /usr/lib/libmenuw.so* ./usr/lib
cp -RP /usr/lib/libmilter.so* ./usr/lib
cp -RP /usr/lib/libmp.so* ./usr/lib
cp -RP /usr/lib/libncp.so* ./usr/lib
cp -RP /usr/lib/libncurses.so ./usr/lib
cp -RP /usr/lib/libncursesw.so ./usr/lib
cp -RP /usr/lib/libnetgraph.so* ./usr/lib
cp -RP /usr/lib/libngatm.so* ./usr/lib
cp -RP /usr/lib/libobjc.so* ./usr/lib
cp -RP /usr/lib/libopie.so* ./usr/lib
cp -RP /usr/lib/libpam.so* ./usr/lib
cp -RP /usr/lib/libroken.so* ./usr/lib
cp -RP /usr/lib/libssh.so* ./usr/lib
cp -RP /usr/lib/libmilter.so* ./usr/lib
cp -RP /usr/lib/libthr.so ./usr/lib
cp -RP /usr/lib/libwrap.so* ./usr/lib
cp -RP /usr/lib/libypclnt.so* ./usr/lib
cp -RP /usr/lib/libz.so ./usr/lib
cp -RP /usr/lib/libzfs.so ./usr/lib
cp -RP /usr/lib/libzpool.so ./usr/lib
cp -RP /usr/lib/pam_chroot.so* ./usr/lib
cp -RP /usr/lib/pam_deny.so* ./usr/lib
cp -RP /usr/lib/pam_echo.so* ./usr/lib
cp -RP /usr/lib/pam_exec.so* ./usr/lib
cp -RP /usr/lib/pam_ftpusers.so* ./usr/lib
cp -RP /usr/lib/pam_group.so* ./usr/lib
cp -RP /usr/lib/pam_guest.so* ./usr/lib
cp -RP /usr/lib/pam_krb5.so* ./usr/lib
cp -RP /usr/lib/pam_ksu.so* ./usr/lib
cp -RP /usr/lib/pam_lastlog.so* ./usr/lib
cp -RP /usr/lib/pam_login_access.so* ./usr/lib
cp -RP /usr/lib/pam_nologin.so* ./usr/lib
cp -RP /usr/lib/pam_opie.so* ./usr/lib
cp -RP /usr/lib/pam_opieaccess.so* ./usr/lib
cp -RP /usr/lib/pam_passwdqc.so* ./usr/lib
cp -RP /usr/lib/pam_permit.so* ./usr/lib
cp -RP /usr/lib/pam_radius.so* ./usr/lib
cp -RP /usr/lib/pam_rhosts.so* ./usr/lib
cp -RP /usr/lib/pam_rootok.so* ./usr/lib
cp -RP /usr/lib/pam_securetty.so* ./usr/lib
cp -RP /usr/lib/pam_self.so* ./usr/lib
cp -RP /usr/lib/pam_ssh.so* ./usr/lib
cp -RP /usr/lib/pam_tacplus.so* ./usr/lib
cp -RP /usr/lib/pam_unix.so* ./usr/lib
```

a CD and not in MFS. The `mfsroot.gz` file, for example, is put onto a physical CD together with the `/boot` directory. However, almost all files mentioned in this tutorial are placed into MFS (`mfsroot.gz`)

How to prepare MFS?

Although there is a very good project called mfsBSD, this step-by-step howto provides an independent way to learn how MFS is prepared from scratch. If you later jump into a conclusion that you have been unsuccessful in something, check other projects such as mfsBSD (*http://mfsbsd.vx.sk/*) or my MaheshaBSD Live CD (*http://www.freebsd.nfo.sk/maheshaeng.htm*), which has a working mfsroot.gz file in it. I prepared my own mfsroot.gz file (for MaheshaBSD) on a trial-by-error basis and I did not find the mfsBSD project helpful very much. I emphasize again that it is a very good project, but I somehow need a different approach – for example, to learn which files are needed for MFS, have this information presented in a document like this tutorial, since all other information is rather redundant for me.

You may argue that, *I can copy files from my root directory (ad0s1a) into MFS*, but this is not quite our goal, as many people are interested in smaller (embedded) systems. I prepared mfsroot in MaheshaBSD from scratch and here I provide you with information that I learned. It is not perfect (mfsroot), but it works. The term *perfect* has rather a relative meaning here, because with embedded systems, in order to save some space, you must delete some files you consider redundant.

The following instructions will help you create Memory File System, but I need to emphasize again that the files you will use (copy) from the `/root`, `/etc`, `/var`, `/usr`, `/usr/local`, (etc.) directories will be *working files* in our situation – that is, your MFS, to say it in a very simple language, will be just working. Since not all of the files in the above-mentioned directories are necessary (in order to save space), your MFS may lack some functionalities, so it is up to you to decide whether you copy other files into your MFS or not. Please, consider this tutorial as a spring board into the universe of new choices that will be exclusively yours.

Listing 3. *Files needed from /usr/libexec*

```
/usr/libexec/atrun
/usr/libexec/ftpd
/usr/libexec/getty
/usr/libexec/ld-elf.so.1
/usr/libexec/mknetid
/usr/libexec/pppoed
/usr/libexec/rpc.rwalld
/usr/libexec/rshd
/usr/libexec/sftp-server
/usr/libexec/smrsh
/usr/libexec/ssh-keysign
/usr/libexec/tcpd
/usr/libexec/telnetd
/usr/libexec/tftpd
```

Listing 4. *Directories needed from /usr/share*

```
/usr/share/locale, /usr/share/misc, /usr/share/nls, /usr/
share/security, /usr/share/skel, /usr/share/syscons,
/usr/share/tabset, /usr/share/tmac
```

Steps

1) Make a copy of your root (`/` or `/dev/ad0s1a`) directory somewhere within your local files on your hard disk (for example, in `/copy` or `/usr/copy`). Keep all directories (in `/copy`), which point to other partitions, empty (`/var`, `/usr`, `/tmp`, `/home`).
2) Keep only the following directories in your /copy/var dir (and their subdirectories):

   `crash, cron, db, empty, ftp, games, heimdal, lib, log, mail, msgs, named, preserve, run, rwho, spool, tmp, yp`

   `/copy/vary/db/pkg` may be deleted if it is too big.
3) Keep the full content of the `/bin` directory (which you copied into `/copy/bin`).
4) Keep the `/copy/sbin` dir, too, as it is.
5) Keep all the libraries copied from `/lib` in `/copy/lib`. Apply the same rule to `/libexec` (copied into `/copy/libexec`).
6) The following files from `/usr/bin` suffice for a working MFS: see Listing 1.
7) Use this script to copy libraries from `/usr/lib` to `/copy/usr/lib` (in case your root partition [`/`] is small, use the path in `/usr/copy/usr/lib`): see Listing 2.
   Copy these files to `/copy/usr/libexec`: see Listing 3.
8) Copy these directories from `/usr/share` into `/copy/usr/share`: see Listing 4.
9) Copy the following files in `/usr/sbin` to `/copy/usr/sbin`: see Listing 5.
10) chroot into your `/copy` directory and test your system (change password, etc.). Do not forget to edit the `/etc/fstab` file (which will be in MFS) – put the following two lines into it (works also with USB):

```
/dev/md0          /           ufs      rw          0       0
/dev/acd0       /cdrom       cd9660   ro,noauto    0       0
```

The above two lines are not harmful if you have a copy of the same fstab file in the `/etc` directory on your CD (it is not necessary, but you may have other ideas).

If you want to save some space, the `/copy/etc directory` (that you will put into MFS) will work with the following directories and files:

Directories in `/copy/etc`: X11, defaults, devd, mail, pam.d, security, skel, ssh, ssl

Files in `/copy/etc`: see Listing 6.

11) You may also add some files into your `/copy/usr/local` dir. However, this is rather optional (not really necessary). It is better to do so on a fresh installation of FreeBSD (just to avoid confusion with so many files and libraries you may later deal with) and on the basis of your personal preferences. Midnight Commander appears useful, for example.

12) Prepare the boot directory in `/copy/boot` (which will be in MFS) – only the following files are needed (all `/boot`

Listing 5. *Files needed from /usr/sbin*

```
IPXrouted, ac, accton, acpiconf, adduser, apm, apmd,
arp, audit, boot0cfg, burncd, chkgrp, chown, chroot,
cpucontrol, cron, daemon, devinfo, diskinfo, extattrctl,
faithd, fdcontrol, fdformat, fdread, fdwrite, gstat,
hcseriald, inetd, iostat, ip6addrctl, jail, jexec, jls,
kbdmap, kgzip, mailwrapper, memcontrol, mixer, mountd,
moused, ndp, newsyslog, pciconf, pw, pwd_mkdb, rmt,
rmuser, rrenumd, rtsold, sa, sendmail, sshd, swapinfo,
sysinstall, syslogd, traceroute, vidcontrol, vipw, watch,
watchdog, watchdogd, zzz
```

Listing 6. *Files needed from /etc*

```
aliases, amd.map, apmd.conf, auth.conf, csh.login, ddb.conf,
devd.conf, devfs.conf, dhclient.conf, disktab, dumpdates,
fbtab, fstab, ftpusers, gettytab, group, hosts, inetd.conf,
libalias.conf, localtime, locate.rc, login.access,
login.conf, login.conf.db, mac.conf, mail.rc, master.passwd,
motd, netconfig, netstart, network.subr, networks,
newsyslog.conf, nscd.conf, nsswitch.conf, opieaccess,
passwd, pccard_ether, profile, protocols, pwd.db, rc,
rc.conf, rc.d, rc.local, rc.resume, rc.sendmail,
rc.shutdown, rc.subr, rc.suspend, remote, resolv.conf,
rpc, services, shells, spwd.db, sysctl.conf, syslog.conf,
termcap, ttys
```

Listing 7. *Directories needed from /etc:*

```
X11, defaults, devd, mail, pam.d, security, skel, ssh, ssl
```

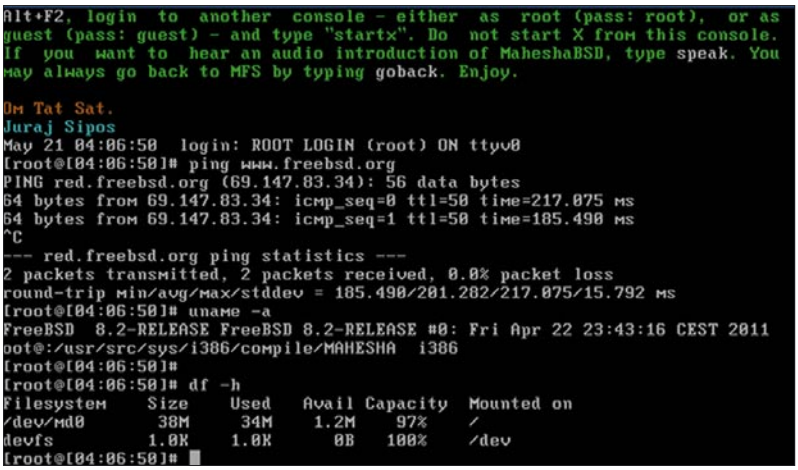


Figure 1. *This MFS was created from FreeBSD 8.2 RELEASE and tested under VMware and Qemu, where it worked without problems (also changing local password for root worked)*

`/kernel` files will be available on your CD or USB stick after it boots; you do not need them all in MFS; use `mount _ nullfs` to mount the `/boot` directory on your CD):

```
/boot/boot
/boot/boot0
/boot/boot0sio
/boot/boot1
/boot/boot2
/boot/mbr
/boot/loader.conf #keep it identical with loader.conf on a
                  CD/USB
```

In case you would like to use compression later, these files will appear useful (symbol files are not needed if you recompile the kernel with the commented

```
„#makeoptions DEBUG =-g" option" line in your kernel
                  configuration file):

/boot/kernel/geom_label.ko
/boot/kernel/geom_label.ko.symbols
/boot/kernel/geom_uzip.ko
/boot/kernel/geom_uzip.ko.symbols
/boot/kernel/zlib.ko ./boot/kernel
/boot/kernel/zlib.ko.symbols
```

13) Now we will create MFS (40 MB)

```
dd if=/dev/zero of=mfsroot bs=1024k count=40
```

You may, too, choose a bigger number than 40. We will format it and mount it:

```
mdconfig -a -f mfsroot md0
newfs md0
mount /dev/md0 /mnt
```

14) We will copy all the files from `/copy` to `/mnt` (where our mfsroot file is mounted). With the above instructions, I have still 1.2 MB free space in my mfsroot.

15) Now we will umount the `/mnt` dir and will gzip the mfsroot file (its size will be about 13,6 MB after compression):

```
umount /mnt3
mdconfig -d -u /dev/md0
gzip mfsroot
```

16) Copy the mfsroot.gz file into the directory (`/ISO/boot`, for example) from which you will make a bootable ISO. Do not forget that the file mfsroot.gz must be in the `../boot` dir. To build the ISO image, use the same command as suggested in the beginning of this article. Imperative is to make the ISO image from the directory that contains the copy of `/boot` dir with all kernel files that reside on your physical (`/dev/ad0s1a`) root partition. Putting any other directory into your ISO is a matter of choice (you may mount it later as a regular dir on a CD – just use the `mount_cd9660` command).

If you want to work with USB devices, use the same `mfsroot.gz` file as described in this article – copy it together with the `/copy/boot` directory onto your USB stick, but before you do so, prepare your USB device (make it bootable, etc.):

```
fdisk -BI /dev/da0
bsdlabel -B -w da0s1
newfs -U -O1 /dev/da0s1a
boot0cfg -v -B da0
```

Some notes on copying files onto a USB stick. *-U -O1* [*O* like in Olympus, not zero] is for the UFS1 format, which provides much shorter copying time (only with USB sticks) than UFS2; if you decide to choose UFS2, type *-U -O2*, but expect that the copying time (if you want to work with many files) will be much longer. Do not forget to use the suggested copy of `loader.conf` (shown in the beginning of this tutorial) in the `/boot` directory on your USB stick, too.

JURAJ SIPOS

Juraj lives in Slovakia, where he works in a library (in an educational institute). He has been writing and selling computer articles for over ten years. He wrote an xmodmap how to (www.faqs.org/docs/Linux-mini/Intkeyb.html) and in addition to computers he is also interested in spirituality, but not really the guru side of things, but more-so freedom and self-actualization. His website says more: www.freebsd.nfo.sk

EuroBSDcon 2011

The **Anniversary**

BSD Daemon © Marshall Kirk McKusick. Used with permission. http://www.mckusick.com/copyright.html

6 until 9 October, 2011

Meeting Plaza, **Maarssen**

Address: **Planetenbaan** 100
3606 AK Maarssen
The Netherlands

GPS: **N**52.12840, **E**5.0360

10th European BSD Conference

http://2011.eurobsdcon.org/

Manipulating map data using QGIS

In this article, we will examine how to create and manipulate shapefiles

In the previous two articles we configured Geoserver and Postgres with PostGIS extensions to serve our map data.

What you will learn…

- A rudimentary introduction to the Quantum GIS Geographic Information System QGIS and how to import shapefiles and rasters into Geoserver manipulating the raster file with GDAL

What you should know…

- What you need to know: Basic FreeBSD administration skills, Previous FreeBSD GIS tutorials in this series

We also examined using SLD to highlight values on our `nyc_buildings` map. While this is a good introduction into the server end technology, a great deal of work is involved at the client end creating and modifying custom maps, often with *difficult* data e.g. overlapping or irregular post codes. Very few systems in the enterprise are GIS or spatially aware, and as a result considerable skill is required creating quality maps. One of the most widely available tools for this purpose is the *Quantum GIS Geographic Information System* (QGIS) an Open Source suite that supports an extensive range of formats. With QGIS you can:

- View and overlay vector and raster data in different formats and projections
- Create maps and interactively explore spatial data via a GUI
- Create, edit and export spatial data
- Perform spatial analysis
- Publish your map

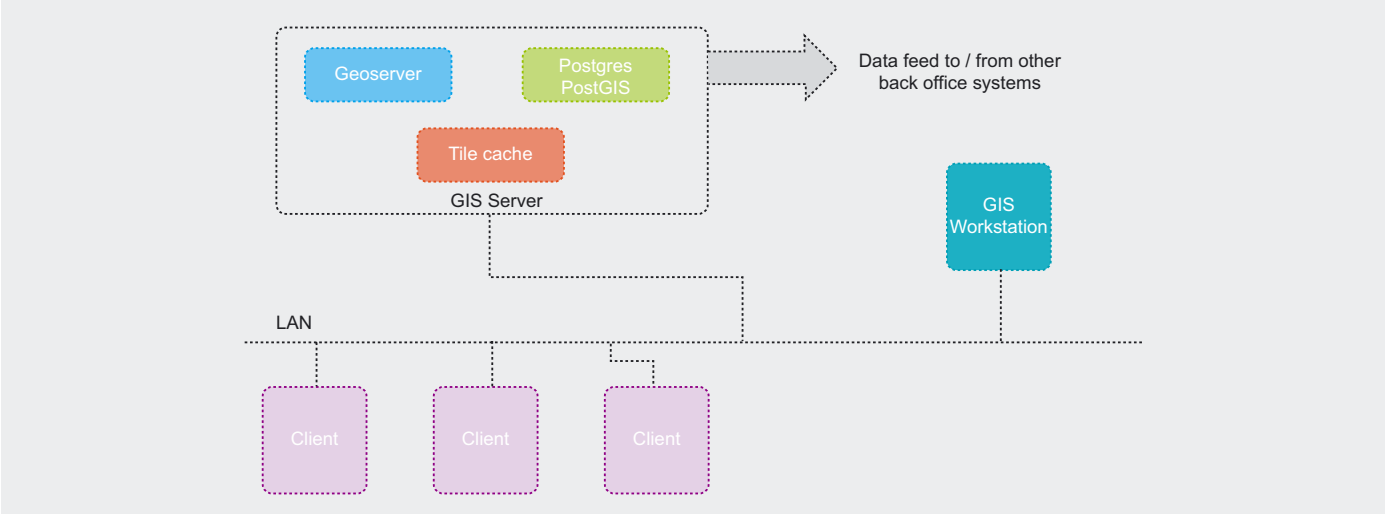


Figure 1. *Typical GIS topology*

Table 1. *Contents of QGIS sample data file*

Directory	Contents
Climate	A point shape layer with some climate data for the diagram overlay feature collected from *http://climate.gi.alaska.edu/Climate/Temperature/mean_temp.html.*
CVS table	This layer was generated from the gtopo30 dataset in GRASS and can be used with the delimited text and the interpolation plugin in QGIS.
GML	A polygon type GML layer of the alaska lakes. The vector layer was derived from VMAP0 data.
GPS	This layer was created for the QGIS demo dataset using the digitizing feature of the QGIS GPS plugin. The waypoints show 4 national monuments in Alaska. The CRS of the gpx file is latlon, WGS84.
GRASSDATA	A collection of vector and raster layers was created for the QGIS demo dataset. The vector layers were derived from VMAP0 data..
Raster	Information on AVHRR data (Global Land Cover Classification) are available at: *http://glcf.umiacs.umd.edu/data/landcover/description.shtml* The data are classified and colored according to information from glcf. Both color and reclass tables are provided with the data.
VMAP0_shapefiles	The VMAP0 dataset has been converted from VPF format to SHP format and kindly provided by Andras Fabian. Then the data were adapted for alaska dataset. For further information about the original dataset, please refer to *http://gis-lab.info/qa/vmap0-eng.html*

QGIS is available for Windows, Linux, Unix and Mac OSX platforms. See *http://www.qgis.org* for more detailed information.

GIS system topology

Typically, a GIS system will comprise of multiple servers, clients (in our case data is served over the web rather than via a custom application) and one or more GIS workstations (Figure 1). In the previous articles we have configured Geoserver and Postgresql on the same server, which in reality under heavy workloads would be a major bottleneck. While Geoserver comes with its own tilecache (GeoWebCache) running this on a separate server or using a different product (e.g. TileCache by MetaCarta) may be desirable.

QGIS is available as a client application under FreeBSD, but unfortunately the FreeBSD package does not have Postgresql support compiled in, so if this is desired, QGIS will need to be compiled from source. Due to the extensive number of heavyweight libraries involved, this is a lengthy and time consuming process. There also seems to be some issue with the Python plugins under the latest FreeBSD, but without testing I cannot be sure if this would be resolved by compiling from scratch and allowing QGIS to sort out the dependencies. As a test I performed an install on my Ubuntu 11 box and Postgresql / Python was supported as standard, so this is a possible solution for those who wish to go down the Open Source route without a very long wait.

As we will be concentrating on the FreeBSD version, we will be limited to processing ESRI shapefiles and raster

Figure 2. *QGIS Splash screen*

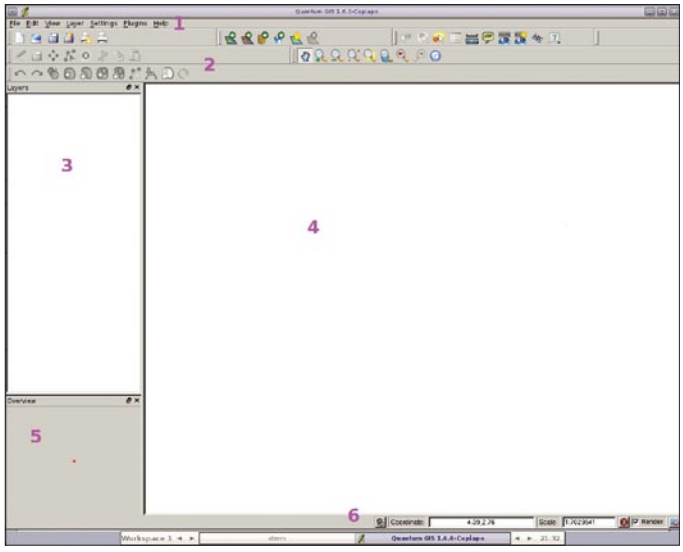


Figure 3. *QGIS User interface*

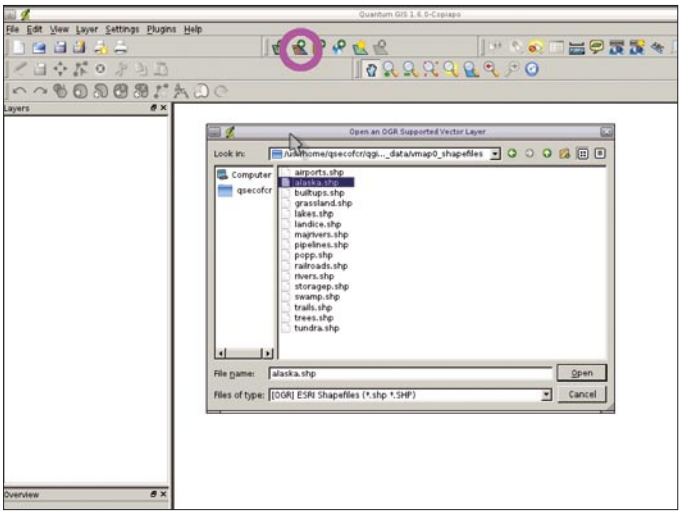


Figure 4. *Creating a vector data source by opening a shapefile*

data. These can be manually uploaded to Geoserver after modification. It is probably not advisable to edit the shapefiles *in situ* as these are accessed directly by Geoserver.

What you will need

If you have been following the series, the pre-configured Geoserver is useful, but not necessary unless you wish to access the Postgres data (`nyc_buildings`) we imported in the previous article in QGIS or import the modified maps. You will need a pre-configured FreeBSD box with X configured and running, and with internet access. For this tutorial, I used a lightweight window manager Fluxbox. Some sample ESRI shapefiles will also be required, In this example I have used the QGIS sample data which is available at: *http://download.osgeo.org/qgis/data/qgis_sample_data.tar.gz*.

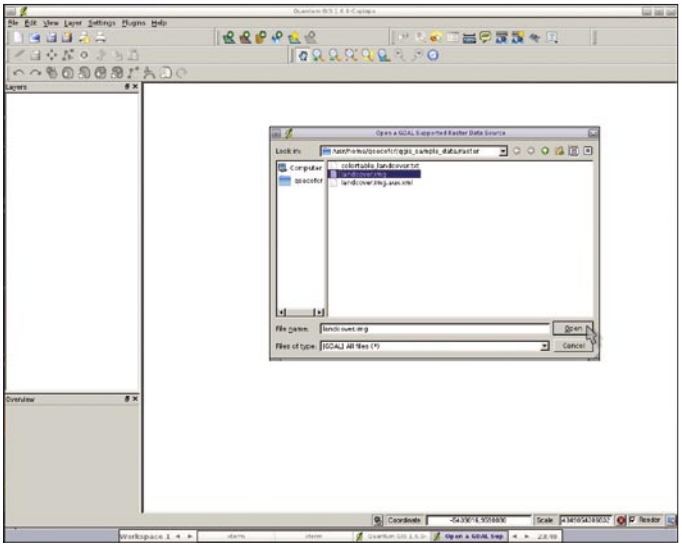


Figure 5. *Creating a raster data source by opening a GDAL image*

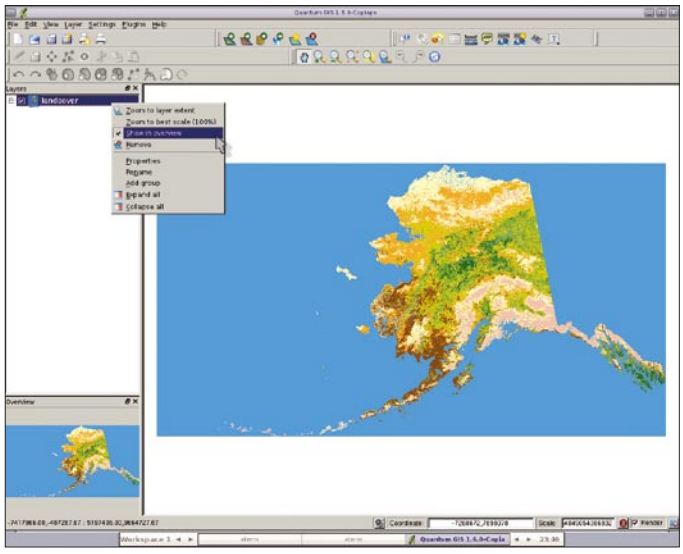


Figure 6. *Toggling the Overview*

Installation

Within a X-windows terminal, su to root and perform the following:

```
pkg_add -r qgis
pkg_add -r wget
exit
cd ~
wget http://download.osgeo.org/qgis/data/qgis_sample_data.tar.gz
tar -xvzf qgis_sample_data.tar.gz
qgis
```

You should be presented with the QGIS splash screen and the user interface (Figure 2 & 3). The GUI is divided into 6 areas as follows:

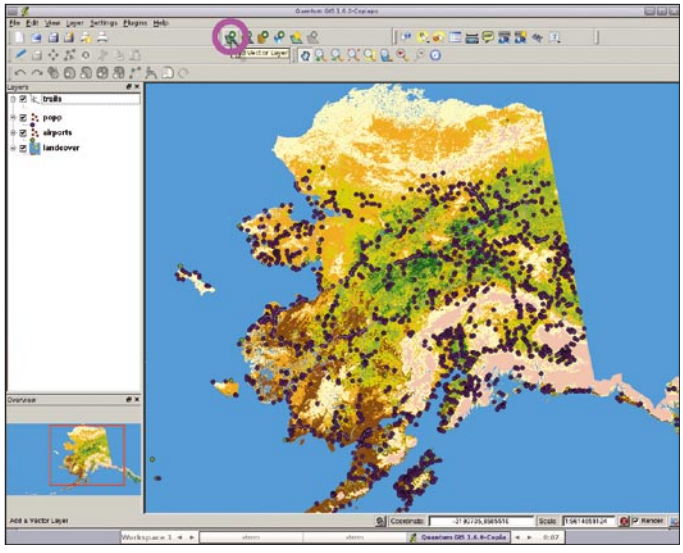


Figure 7. *Map with raster and vector data loaded with Vector layer button highlighted*

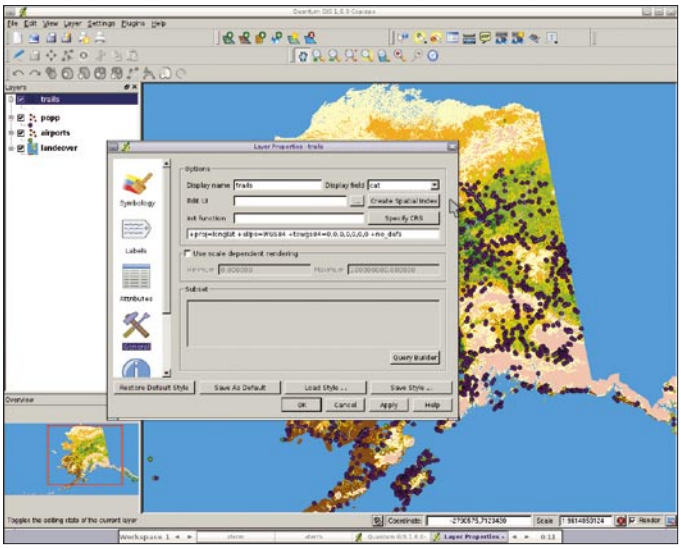


Figure 8. *Layer properties dialogue*

1 Menu bar: This provides access to the hierarchical menus
2 Toolbar : The toolbars provide quick access to most menu items, plus additional tools for interacting with the map
3 Map Legend: This displays the layers visibility, and whether or not we are in editing mode
4 Map View: Where the maps are displayed and edited
5 Map Overview: Shows the full extent of the current map. Drag the square to navigate around the map. If not enabled, this can be selected under Settings/ Panels
6 Status Bar: Displays rendering progress, current co-ordinates etc.

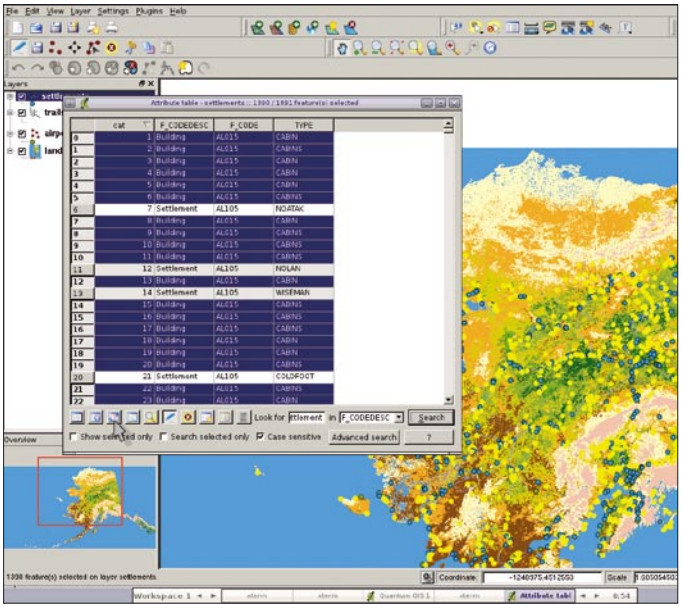


Figure 9. *Non-Settlement data selected*

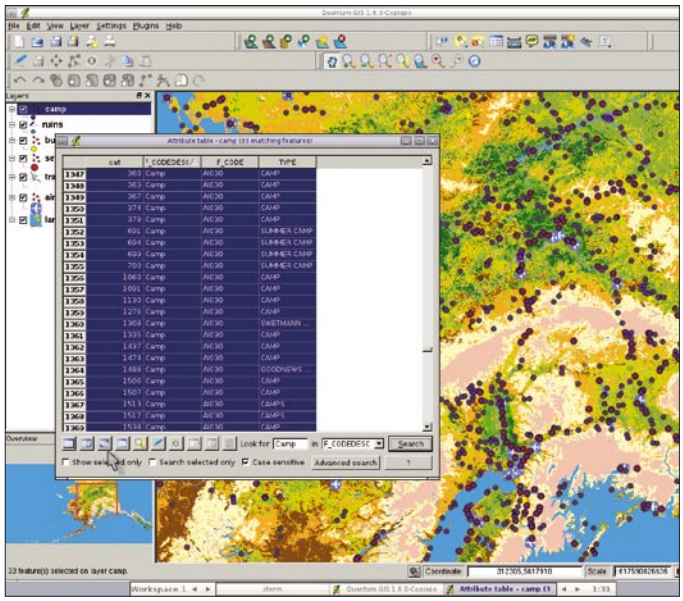


Figure 10. *Camp data selected*

QGIS sample data

The sample data is extracted into the `qgis_sample_data` directory. All data are projected in Alaska Albers Equal Area with (unit feet) unless otherwise specified (EPSG Code 2964). The dataset contains a collection of geodata from several sources with different licenses. These licenses have to be respected by the user (Table 1).

Loading the base raster layer

Raster data is pixel based, each pixel represents a geographic point. Raster is best suited for large geographic area that hold a lot of variation that would be impractical to digitise into vector format, e.g. terrain.

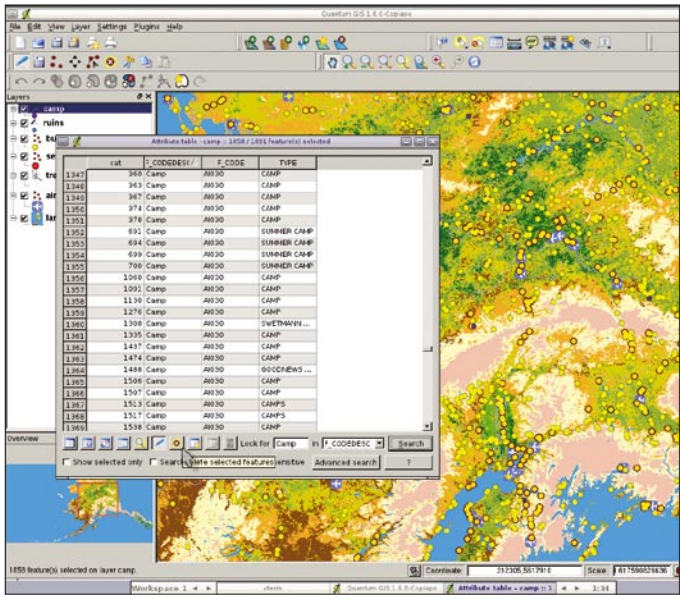


Figure 11. *Attribute table*

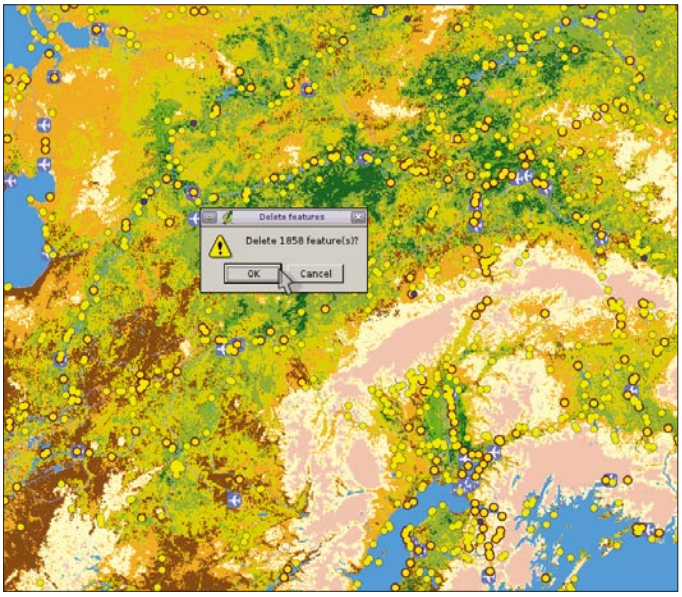


Figure 12. *Deleting features*

Load the landcover.img raster file by clicking on the Add Raster Layer button. Right click on the layer once loaded and enable Show in overview. You can now freely pan around the map, zoom in and out etc. (Figure 3-5).

Loading a vector layer

Shapefiles are used to hold vector data. For example where raster data (which is pixel based) would be used to display satellite or aerial imagery, vectors would be used on a separate layer to display geometry features associated with the raster layer e.g. towns, roads and boundaries. The town would be represented by a point, the road by a line (or polyline) and the boundaries by a polygon. As most GIS systems do not allow the *mixing and matching* of geometry features, each geometry

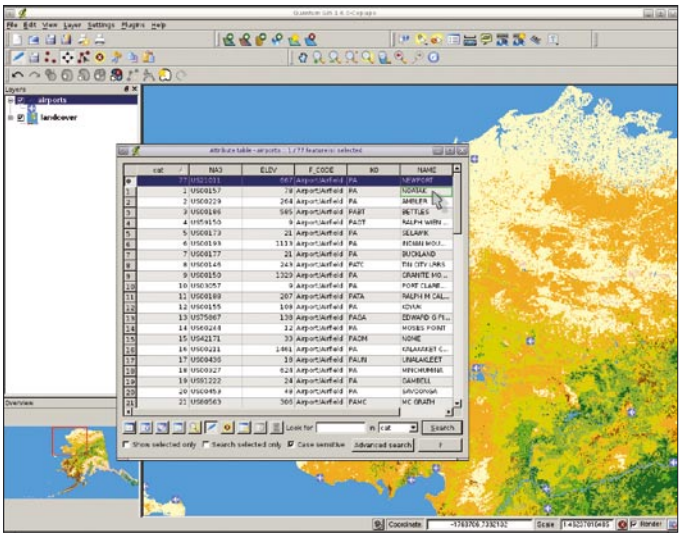


Figure 13. *Airports data*

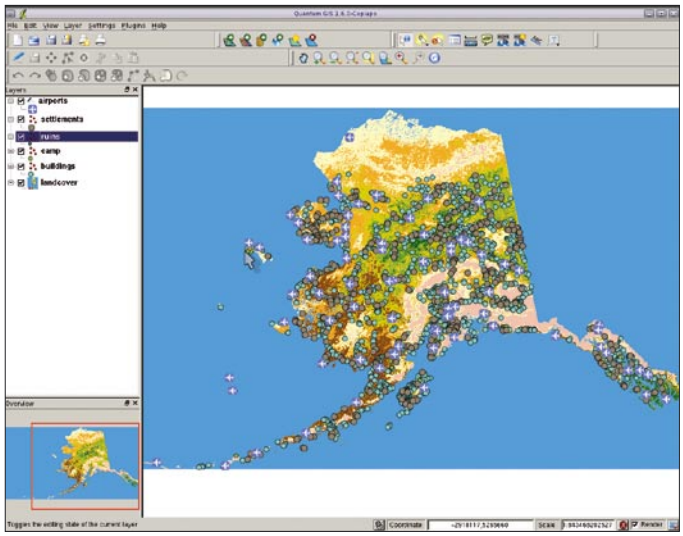


Figure 14. *Vector data with Airports highlighted*

feature and corresponding attribute data is saved as a separate layer.

Click on the Add Vector Layer and import the trails, popp and airports layers (Figure 6). To improve performance, generate a spatial index by right clicking on the vector layer, selecting properties and Create Spatial Index (Figure 7).

Manipulating the layers

Points: The popp layer holds points relating to settlements, buildings, cabins etc. At 100% scale these are not too densely packed, but as we zoom out these become blurred. We will split the data into separate layers, and use different icons to represent each attribute.

- Right click on the popp layer and save as settlement. Repeat for camp, building and ruins.

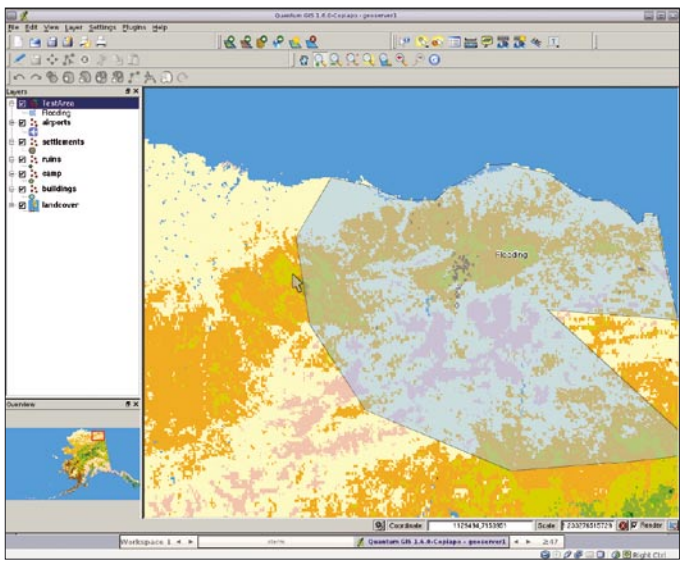


Figure 15. *Fictional flood polygon*

- Remove the popp layer
- Add the settlement layer
- Toggle editing
- Open the attribute table and search for Settlement in F_CODEDESC
- Invert the selection
- Delete (This will take a few minutes as QGIS will have to remove > 1300 entries.)
- Toggle editing and save changes

Repeat steps 3-9 for cabin, building and ruins (Figure 8-11).

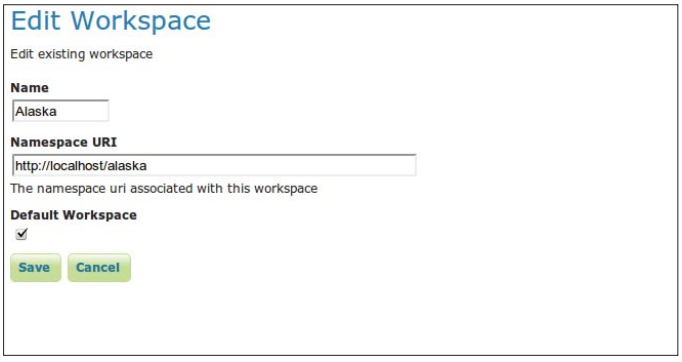


Figure 16. *Editing the workspace in Geoserver*

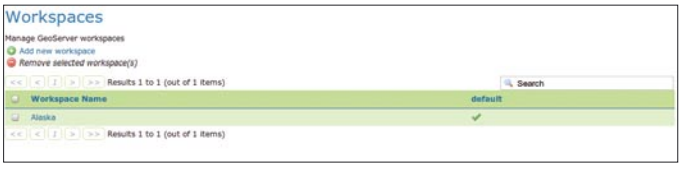


Figure 17. *Editing the workspace in Geoserver*

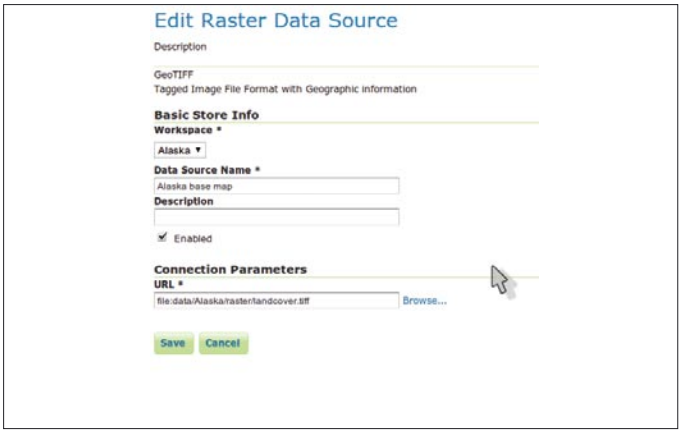


Figure 18. *Editing the Raster datasource*

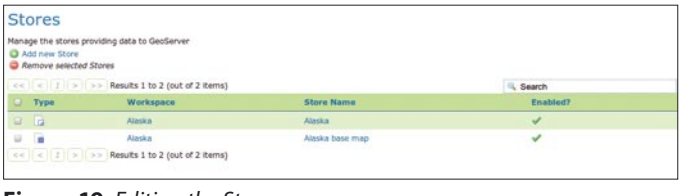


Figure 19. *Editing the Stores*

Duplicating a point

- Toggle editing
- Click on select features
- Click on the feature you want to copy
- Click on Copy Feature
- Click on Paste Feature
- Click on the Move tool to relocate
- Click on select features
- Click on Open Attribute table and edit the record as appropriate [Figure 12 & 13]
- Toggle editing and save changes

Adding a polygon

I have created a fictitious flooding area at the top RHS of the map. I wasn't terribly accurate mapping the edges, but in a real map I would habe been more precise.

- Add a new shapefile layer
- Toggle editing
- Click on the capture polygon
- Double click to place point, right click to finish
- Toggle editing and save (Figure 14).

Note that while we can manipulate the positions and attributes of the features, some changes (e.g.

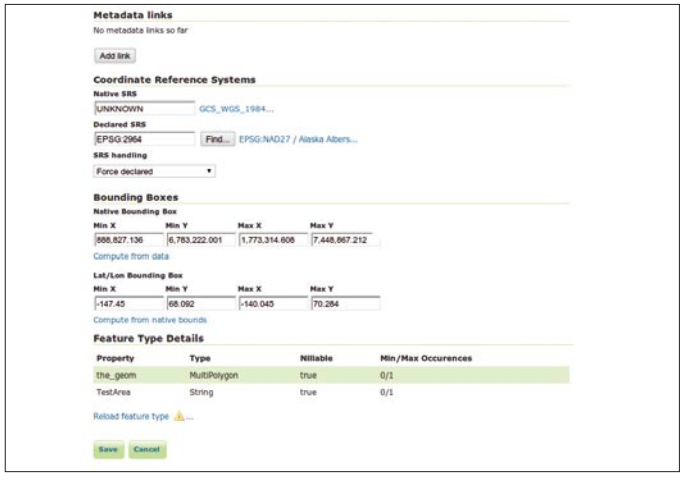


Figure 20. *Setting the bounding box and SRS*

Figure 21. *Managing the layers*

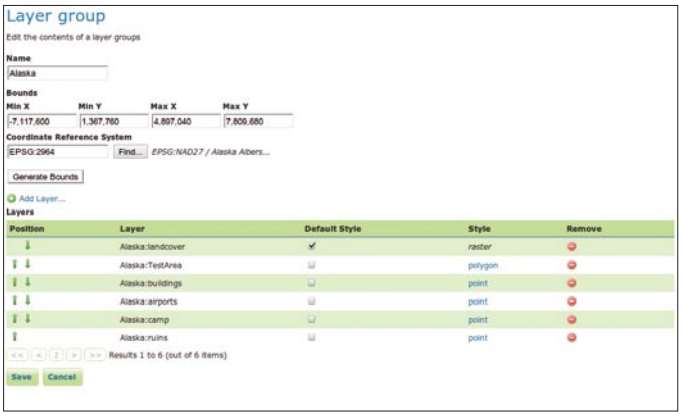


Figure 22. *Geoserver layer groups*

transparency/ icons) will not be imported into Geoserver. Such changes can be performed using SLD and / or WMS styles as detailed in the previous tutorial.

The same process can be performed to draw roads, rivers etc. using the line tool. If you have used a drawing oe CAD package QGIS is similar in concept and design.

Import to Geoserver

We will import the shapefiles into Geoserver as a directory of spatial files (shapefiles).

- Copy the shapefiles and rasters into two seperate directories called shapefiles and raster under a directory called Alaska
- Copy Alaska into `/usr/local/apache-tomcat-7.0/webapps/geoserver-2.1.0/data/data` on the Geoserver box
- chown -R www:www Alaska to flag the files with correct permissions

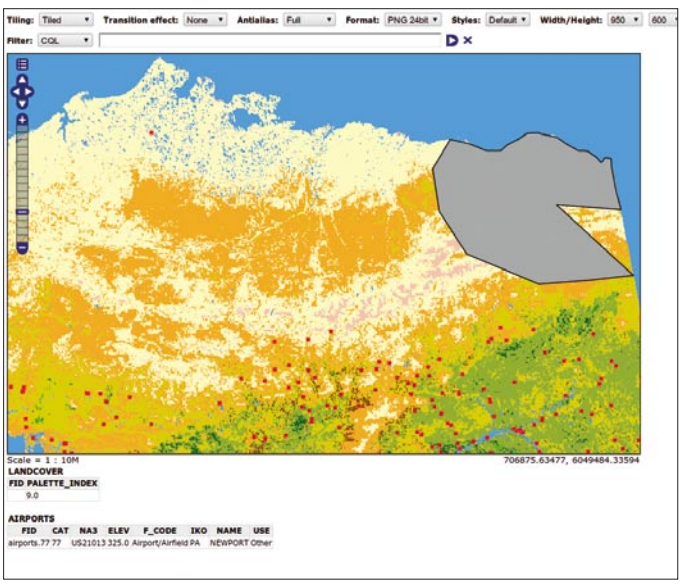


Figure 23. *QGIS edited map in Geoserver*

While the shapefiles will import straight into Geoserver, the raster image requires the GDALI extension to import the file. For the time being, we will convert this with the GDAL (*Geospatial Data Abstraction Library*). Install GDAL on the Geoserver box and convert the raster file from IMAGINE format to GEOTIFF:

```
pkg_add -r gdal
cd /usr/local/apache-tomcat-7.0/webapps/geoserver-2.1.0/
                    data/data/Alaska/raster
gdal_translate -of GTiff landcover.img landcover.tiff
```

In Geoserver

- Create a workspace called Alaska with URI *http://localhost/alaska* (Figure 15-16)
- Create a store from the vector data source in Workspace Alaska called Alaska that is a directory of shapefiles and point that at the Alaska shapefiles directory.
- Create a store from the raster data source GeoTIFF called Alaska Base Map and publish this with Declared SRS of EPSG:2964 (Figure 17-18).
- Create and publish each layer / shapefile in turn using a Declared SRS of EPSG:2964. Geoserver will automatically decide the best style to use, you may change some styles but a WMS error will be generated if you try to display a point using polygon style etc (Figure 19 – 20)
- Create a layer group with the layers you have imported (Figure 21)
- Check each layer in preview mode that you get what is expected, finally preview the layer group and you should see you new airport and the flooding area (Figure 22).

Further challenges

All the QIS sample data should work OK in the FreeBSD version, I will leave it to the reader to experiment with the other data sets.

ROB SOMERVILLE

Rob Somerville has been passionately involved with technology both as an amateur and professional since childhood. A passionate convert to *BSD, he stubbornly refuses to shave of his beard under any circumstances. Fortunately, his wife understands him (she was working as a System/36 operator when they first met). The technological passions of their daughter and numerous pets are still to be revealed.


Looking for help, tip or advice?
Want to share your knowledge with others?
BSD MAGAZINE
Give us your opinion about the magazine's content
and help us create the most useful source for you!

IP Version 6: Open with Care

Despite the promise of improved security, the move to the next-generation Internet protocols will create short-term problems for your network. Here are six tips to keep in mind in planning your transition to IPv6

What you will learn…

- Why IPv6 is already a latent threat in your IPv4-only network
- Plan ahead to avoid IPv6 security problems before widespread deployment
- Identify known areas of weakness in IPv6 security

What you should know…

- IPv6 is becoming a reality.
- IPSec is not the answer to every IPv6 security issue
- A new protocol brings new security issues with it

Ready or not, the move to the next generation of Internet protocols is upon us. Despite the promise of improved security, the move to the next-generation Internet protocols will create short-term problems for your network. Over the long term, IPv6 promises (it's design is some 15 years old, so from when does it hold its promises, from 15 years way back?!) to increase security with improved encryption and features such as IPSec (van Hauser pejorated IPSec as a *pain in the as*!), an end-to-end scheme offering mutual authentication between hosts. Until we get to a pure IPv6 environment, we will probably be slightly more at risk. IPv6 is not a silver bullet to solve all security problems. It's not going to solve your user and application problems. In some instances there will be hardware problems at first hand.

For some of us, there are two drivers for the transition to IPv6. First, the pool of available IPv4 addresses is rapidly drying up, so the growth in the Internet will increasingly be in the IPv6 address space. Second, for those working in US government, the White House has mandated that agencies enable IPv6 on public facing servers and services by September of 2012, and enable the new protocols on internal applications within two years after that. When talking about transition, don't gloss over security.

Although IPv6 offers new features, until the foreseeable future administrators will still have to maintain existing IPv4 infrastructure – which might be described as nothing but a numbering scheme to which security has been added (this seems to be the way IPv6 will follow, too). That means the familiar firewalls, access control lists and other security barriers now in place will have to be maintained. At the same time, the new protocols will have to be managed and maintained with a dearth of experience, expertise, and tools. That in turn could expand attack surfaces and open up new vectors for the black hat guys.

Six security tips for your transition to IPv6

Here is a listed collection of high-level suggestions concerning security issues to keep in mind when planning your transition to IPv6. It is not comprehensive but a consensus of high-level suggestions that I have gathered from my reading.

Planning and Policy

Introduction of IPv6 will create a separate but not necessarily equal network in the enterprise that will require its own security policies. New policies will need to be as stringent as existing ones and appropriate controls will have to be applied all over again, because existing controls may not translate to the new environment.

One must pay attention to the IPv6 environment and keep the IPv6 filters and policies up to date and parallel to the IPv4 policies. Otherwise, you might have IPv6 vulnerabilities that you assumed were closed because they were closed in IPv4.

You have to spend the time and effort to examine your existing security and access policies and how they can be adapted to IPv6.

One area of concern is likely to be access policy, which often associates an address with a user in assigning privileges. That system is beginning to show wear with the proliferation of personal mobile devices that are increasingly being used to access network resources, which might not be possible with IPv6. With the larger address space available in IPv6, the addressing is more dynamic in IPv6 and almost constantly changing.

By enabling IPv6 for internal applications, thought will have to be given to just what services should be available through IPv6 and what to maintain under current access controls for IPv4. As more resources are made available on IPv6, authentication and authorization schemes and technology at the same level of security will have to be tailored for the new protocols.

Workforce and Experience

Managing and securing two networks running different protocols will require trained workers who might not be readily available.

It's the human element, having people who know not only how to implement the new protocols but manage and maintain them as well. The onus is on you to hire that resource.

Is there an adequate pool of IT professionals with training and experience in IPv6 from which to draw? I don't believe so. Engineers today are grounded in version 4.

Fortunately, although versions 4 and 6 of the Internet protocols are not interoperable, they still are IP and if you know one, learning the other should not be that difficult. If you are a network engineer today, the leap to IPv6 is not a huge leap, but it does require hitting the books.

The situation is not the same as the move from switched circuit telephone service to Internet telephony. That move came suddenly and the differences between traditional service and voice-over IP were great. For those who choose not to adapt, it may be a career ending decision.

This does not change the fact that additional manpower is likely to be needed to oversee two networks or two versions of a network, however, and despite available training for IPv6, practical experience in running a production network with the protocols still is scarce.

Breaking Things

IPv6 is much more complex and complexity equals problems. When you are making a transition this complex, the potential for mistakes and unexpected issues is great.

The problems can come in two broad areas. There are bound to be new and unexpected flaws and vulnerabilities in the coding and configuration of the networking stack and in applications and services.

We are going to see an array of bugs that in some cases will become security vulnerabilities. We will see new exploits evolve. There is little that can be done to prevent that, but it must be taken into account when implementing the protocols and forming policy.

The other area of threat comes from breaking things already in place or allowing existing policies to break new things. Take, for example, the Internet Control Message Protocol (ICMP), which is used to send error messages and is not typically used by end-user applications.

If a security administrator is overly conservative, blocking everything he isn't certain is needed, ICMPv6 might get completely blocked, impeding discovery, routing and more. ICMPv6 cannot be blocked arbitrarily. The good news is that ICMPv6 doesn't contain the vulnerabilities found in ICMPv4.

While equivalent security must be maintained for both sets of protocols, the policies might not be transferable without creating problems.

Tools and Testing

What is lacking is maturity. While IPv6 capability is theoretically available, few networks have been using it. Will network management and security tools work as advertised? Will they perform on par with IPv4 tools or will they create bottlenecks and roadblocks?

It's too new to be an established set. It needs to be used more in production.

Development of a fully mature suite of tools will require real world experience that will not be available until the transition to IPv6 is well under way. In the meantime, thorough testing will be needed to eliminate the most obvious problems and improve performance. Poorly implemented IPv6 stacks and tunneling or translation plans will be difficult to properly secure and monitor.

Breaking some glass in a test environment will be necessary.

Spam and Blacklisting

Spam, like the poor, will always be with us, and the transition to IPv6 could make it worse.

Every time there is a change, it gives the spamers a new way to figure out how to get through a firewall. A lot of the spam tools won't be ready to address these tricks.

One of those tools is blacklisting: The blocking of IP addresses and URLs that are known to be sources of spam or other malicious traffic. Blocking addresses, as

well as monitoring traffic to identify and filter malicious traffic, could become more difficult in the dynamic IPv6 environment.

Dynamic content now being delivered via IPv4 is making blacklisting an imperfect tool. The approach is already ineffective in IPv4 and it will become less effective with IPv6.

One Web page request can be subject to twenty or more links and bad guys can take advantage of this to hide the source of malicious traffic. Another complication with blacklisting is the use of distributed botnets as well as legitimate resources that have been compromised to distribute spam and malicious code. By limiting the volume of suspicious traffic from any one source, identifying and blacklisting that source can be made more difficult.

As inadequate as blacklisting is by itself, it remains a useful tool and is not likely to be abandoned with IPv6. However, as monitoring suspect traffic and its source becomes more complex in a fully IPv6 world, it will require cloud-based services to provide the granularity of control and scale to the volumes needed for effective blocking.

Fortunately, the full impact of this change is not likely to be felt for some time. The level of IPv6 traffic on the Internet so far is minuscule and for the next couple of years we are going to be seeing a trickle rather than a flood.

Security through Obscurity

There is a constant tension in networking between functionality and convenience on one side and security on the other. The improved visibility and end-to-end connectivity offered by IPv6 could have a down side in the form of increased risks.

One of the unforeseen advantages of IPv6 has been *Network Address Translation* (NAT), a technology for placing multiple private addresses behind a single public IPv4 address as a way to extend increasingly scarce addressing resources. NAT has been criticized as a Band-Aid fix that breaks the end-to-end connectivity of the Internet and interferes with network management but it also provides a degree of security through obscurity by shielding much of the network from outsiders.

Putting NAT in an IPv6 network would be like putting a buggy whip holder on an automobile. However, if you get rid of NAT, you are going to open up the attack surface of your network.

NAT-based policies for address allocation and management will no longer apply and outsiders are given a potentially unobstructed view of the network. One solution could be to take advantage of the large address space available in IPv6 to restore some of the obscurity.

A block of addresses could be remapped to a proxy that would make it more difficult for an outsider to correlate traffic and see what is going on side the network, and to inject himself into a particular stream. That could restore some of the security provided by NAT at the cost of additional network complexity. There has always been a conflict of interest between visibility and security and that is not necessarily going to change with the adoption of IPv6.

Conclusion

IPv6 is becoming a reality. The many years of early protocol research have paid dividends with products that easily interoperate. Several early IPv6 research groups have disbanded because the protocol is starting to move into the transition phase. The 6BONE (phased out with RFC 3701) and the KAME (*http://www.kame.net*) IPv6 research and development projects have wound down and given way to more IPv6 products from a wide variety of vendors. Deployment of IPv6 is not a question of if but when. IPv6 is an eventuality.

The transition to IPv6 continues to take place around the world. The protocol is gaining popularity and is being integrated into more products. There are many IPv6-capable operating systems on the market today. Linux, BSD, Solaris, and Microsoft Windows server 2008 operating systems all have their IPv6 stacks enabled by default, and IPv6 operates as the preferred protocol stack.

IPv6 will eventually be just as popular as IPv4, if not more so. Over the next decade as IPv6 is deployed, the number of systems it is deployed on will surpass those on IPv4. While early adopters can help flesh out the bugs, there are still many issues to resolve. IPv6 implementations are relatively new to the market, and the software that has created these systems has not been field tested as thoroughly as their IPv4 counterparts. There is likely to be period of time where defects will be found, and vendors will need to respond quickly to patching their bugs. Many groups are performing extensive testing of IPv6, so they hopefully can find many of the issues before it is time to deploy IPv6. However, all the major vendors of IT equipment and software have published vulnerabilities in their IPv6 implementations. Microsoft, Juniper, Linux, Sun, BSD, and even Cisco all have published vulnerabilities in their software. As IPv6 has been adopted, it is evident that these major vendors have drawn the attention of the hackers.

PAUL T. AMMANN

Paul lives in New Fairfield, CT with his wife Eve and two cats. He recently converted from Linux to OpenBSD although he still misses his TI 99/4A and Timex Sinclair.

dotlike.net

Linux
Netzwerk
Sicherheit
Programmierung

Puffy The Hobbit

The Challenge of Porting GNOME 3 to OpenBSD

After a recent proposal from Lennart Poettering [1] for GNOME to include more Linux specific technologies like systemd (and basically become a Linux-based OS), I thought it would be interesting to show some of the challenges and constant battle that is to port this Desktop to BSD systems and more specifically OpenBSD.

What you will learn…

- some OpenBSD internals
- some OpenBSD specifics compared to other Unixes

What you should know…

- GNOME core technologies
- basic knowledge of libc functions

This article is meant to be an easy-read. I'm not going to go into deep technical details on purpose, the point being to show the most common and obvious problems as well as the current *state of the Union*.

Please note that opinions expressed in this article are mine entirely, I'm not the ambassador of the BSD crowd nor anything else ;-)

First, a word regarding this particular thread. On one hand, I can understand Lennart's position, he wants to use the technology available to him, even if that implies dropping support for software or configurations that he does not use. We can't blame him for developing whatever he wants, but the real question is what should the upstream projects do with it.

So what should GNOME really be? It is of course up to the developers to decide how they want it to grow and to be honest, currently things aren't very clear. Is GNOME supposed to grow as a complete Operating System based on the Linux kernel? Is it supposed to be a generic Unix Desktop? Or something else?

The first questions that are often asked about porting GNOME to OpenBSD are *why would anyone want to do that?*, *who cares about such a bloated piece of software* or *people who want to use GNOME should just run Linux* ...

Somehow this stupid behavior and short-sighted mind (not to mention the lack of thought) is one of the reasons that I keep working on porting GNOME to our favorite Operating System: I like the challenge and I want to prove it is useful for everyone to keep supporting platforms other than Linux.

The other reason is that it is part of my daily work to build and support Desktop systems based on OpenBSD and GNOME which are used by thousands of users around the world (that said, if GNOME was to become a Linux-only software, there would be alternatives).

OpenBSD specific challenges

People working on porting software to OpenBSD know that several functions that are taken for granted on other Unixes aren't available here. This is actually the trickiest part because writing support from scratch is one thing, but having to deal with functions that behave differently or are not present on a particular OS is much harder to deal with since it usually touches generic code also used by Linux and upstream tends to be very cautious (rightly).

Here we will disclose some issues related to OpenBSD specifics.

Includes order, missing headers

Very often we run into undefined macros or functions that are due to the order of the included headers. Linux isn't really picky about it but OpenBSD is, which would trigger a compilation failure. On a somewhat related subject, Linux

usually likes to include lots of headers into its header files, which is not our case and we often run into breakage because of missing defines.

This issue is quite common but upstream is usually open to include our patches to fix it (along with the usual mockery that we are not modern or so).

PAM

For the common crowd, Unix authentication means PAM (Pluggable Authentication Modules). While it's true that most systems use it, this is not the case in OpenBSD which instead uses `bsd_auth(3)` (BSD Authentication). While providing the same core functionality (user authentication against several sources, like flat password file, Kerberos, YP...) both work in a totally different ways.

Most of the time, porting to `bsd_auth(3)` is pretty straightforward when the actual software also supports shadow authentication because all that is needed is replace and adapt the shadow functions by the ones needed for BSD Authentication. Porting directly from PAM is more complex.

The problem nowadays is that PAM is declared as a required dependency, most of the time even shadow authentication is not supported anymore (GDM, gnome-screensaver) ... so some autoconf work is also needed in this case.

It's interesting to notice that the guys porting GNOME to Slackware Linux face the same issue as we do because Slackware does not come with PAM either.

mmap

We ran into an interesting issue porting dconf (the gconf replacement for storing application configurations in a local DB, similar to the Windows registry). For some reason, changes made to the configuration were not seen by the Desktop, e.g. changing the background would have no effect until after we logged back in. Trying to get the value with *dconf read* would always return the former one instead of the newly modified one.

Eventually we found out that the reason was due to our `mmap(2)`: OpenBSD does not implement a coherent file system buffer cache, so we were missing an `msync(2)` call with the `MS_INVALIDATE` flag to force invalidating the cache and prevent returning the old value.

This was actually fun to debug and upstream was very helpful on the matter.

Socket credentials

Several releases ago, GLib implemented *Gcredentials*, a wrapper for native credentials used for identifying, authenticating and authorizing other processes. It's used for example in the communication between dbus (via GDBus) and PolicyKit.

Unlike Linux (struct ucred) and FreeBSD (struct cmsgcred) OpenBSD does not support passing `SCM_CREDS` over Unix sockets and trying to do so would result in an ENOTSUP error (where the GLib code would assume any Unix would be able to do so).

But we needed this to implement `g_unix_connection_send_credentials()` and `g_unix_connection_receive_credentials()` that expect being able to send and receive a `GunixCredentialsMessage`.

After some discussion with upstream, it turned out there was another way of getting the credentials by looking them up from the remote peer of the communication endpoint using `g_socket_get_credentials()` (i.e. `getsockopt(2)` with `SO_PEERCRED`). This function was never called because as soon as `g_unix_connection_*_credentials()` didn't succeed, we were left without credentials.

With input from upstream we finally managed to patch the code so that it would fall back to `g_socket_get_credentials()` if sending/receiving the `GunixCredentialsMessage` failed.

pthreads

Beware, we are entering a world of hacks and despair...

GNOME uses threads heavily (via Gthread from Glib) which tends to push our userland threads to their limits. Indeed, on OpenBSD we don't have support for kernel threads yet (rthreads is still under development and not fully functional yet), which means we have to cheat with file descriptors.

Most threading related issues we face are due to the fact that `pthreads(3)` changes all the file descriptors to non blocking mode and the result is that blocking functions will fail with EINTR. A good example of that are the `g_daemon_file_{in,out}put_stream_new()` from gvfs.

Currently we are forced to patch them and use `fcntl(2)` to remove the `O_NONBLOCK` flags from the FDs. This works but it's hackish. Similar workarounds can be found in other areas of the ports tree and there is currently no *good* fix for it.

As a side effect, our Perl is not threaded either, which means that p5-GLib (the Perl bindings for GLib) are not functional either which prevents us from porting some applications.

getpt, ptsname, unlockpt...

Despite being defined by POSIX, OpenBSD always refused to implement the `posix_openpt` family of functions. The reason being that these functions could be subject to a race condition (i.e. not guaranteed to open a pty atomically) which is a security hazard.

Manipulating ttys on OpenBSD is done with `openpty(3)` but unfortunately it's not used in the same way as `posix_openpt`: some code remain unportable as such without a rewrite.

Some features are currently suffering from this:

- VTE, the terminal emulator has to use a setgit(3) helper: gnome-pty-helper, to be able to work.
- Vinagre (VNC client) and gvfs (virtual FS, replacement for gnomevfs) lose the functionality to open an ssh(1) session using a password.

time_t

Another case where OpenBSD deviates from most Unix systems is that our `time_t` is not 64 bits wide. This needs careful inspection of GCC warnings during compilation because some applications would happily overflow since no check is made to see what `time_t` is defined as.

This particular issue triggered crashes in at least evolution and gnome-keyring.

Monotonic clock

OpenBSD has support for the MONOTONIC clock, however applications cannot detect it properly because we are missing most of the `_POSIX_*` defines in our unistd.h. We also don't have `sysconf(3)` variables support for it.

Until we fix this, we have to patch the applications but where it gets tricky is that some of them assumes that if we define `POSIX_MONOTONIC_CLOCK` we also support the `pthread_condattr_setclock(3)` function which we currently don't because of our userland threads implementation.

/proc

OpenBSD comes with minimal support for the *process file system* (i.e. /proc), however it is only available on the i386 platform, not mounted by default and generally considered as deprecated and non maintained. So in a word, we do not use it.

Instead of accessing `/proc` to get information about processes, what we do is patch the code to use our kernel memory interface (`kvm(3)`).

A very good example can be seen in the commit `ID a d9da2727d3e3243fd052c9feb0c55645e87d384` in gtk+ which adds support for `GmountOperation::show-processes` on OpenBSD.

Linux-isms

In this section we will talk about the most common (usually trivial but annoying) issues introduced by developers only knowing or caring about Linux.

Harcoded paths

Everywhere in GNOME applications code you can find hardcoded paths, to binaries, libraries, include directories... Linux people assume that everything is always installed under the `/usr` PREFIX. That is not the case on any of the BSDs, the default PREFIX for external applications (i.e. not included in the base system) is different (OpenBSD and FreeBSD use `/usr/local` while NetBSD uses `/usr/pkg`).

While trivial to fix, these require some `grep(1)` work to make sure that everything the application is looking for at build and run time is correctly found.

Note that this particular issue is not really BSD related, some Linux distributions (not based on Redhat/Fedora or Debian) may suffer from this as well in some cases.

Another minor path related issue is that we do not follow the Linux Filesystem Hierarchy Standard, so we have to patch applications according to our `mtree(8)` definitions (e.g. `/var/lib › /var/db`).

GNU vs BSD userland

Another common problem for us is the use of non-standard options in basic utilities such as `cp(1)`, `grep(1)`, `sed(1)`, `find(1)` … Several Makefiles and/or configure scripts use GNU extensions which are not recognized by OpenBSD leading to build failure or creation of corrupted files. While spotting build failures is obviously easy, it can take a while to debug an issue introduced by a corrupted script or header file.

We usually end up rewriting the incriminated block using only standard options and push it upstream to prevent adding a dependency on the GNU versions of the aforementioned tools (which we have external packages for).

Same goes with the use of the `bash(1)` shell with non standard Bourne shell extensions. While BSDs provide `bash(1)` as an external package, it doesn't come with the base system and instead of introducing a new dependency, we do the same as above and rewrite the block is a more portable way when possible.

From time to time we also run into issues with `tar(1)`: some projects use GNU options that aren't available to us and create archives with a slightly different format (e.g. Pax headers). Usually fixing this is just a matter of nicely asking upstream to use standard (i.e. portable) options when they create their archives.

Linux technologies

This is where lies the core of the most complex issues, several key functionalities of GNOME are tied to Linux features.

One (in)famous example is the udev device manager (which deprecated HAL). None of the BSDs has full support for it. There are several reasons to that, first because some of the features it implements are already present in BSDs but under some other form (like `hotplugd(8)` on OpenBSD). Also, porting this kind of interface usually requires intrusive kernel changes which cannot always be made due to the intrinsic differences of the systems. Another reason is that the documentation is often incomplete or outdated which means the porter needs to have a good understanding of the Linux kernel itself to see how it is implemented there. Finally Linux has a tendency to deprecate their hardware abstraction layer very often and keeps re-inventing it. Not long after FreeBSD got HAL support working it was pretty much starting to get obsoleted by udev.

The main issue here is the lack of communication between the projects, basically BSDs systems are forced to implement whatever Linux came up with, instead of discussing beforehand how such feature could be done, which functions should do what... Linux being a leader at implementing new functionalities, it is up to the BSDs people to try and be closer to their development so we can add our voice about how things could be done to benefit most systems.

Of course, when dealing with software that were designed to be run only on Linux and whose author refuses portability patches as is the case with systemd, there isn't much room for discussion and this is why such software should never become a mandatory dependency in my opinion.

Another thing is often noticed, and more recently spelled out by some people in *the Linux crowd*: being a leader in implementing new technologies and features, they tend to think that *BSDs just need to catch up*. However one cannot expect everyone to modify their kernel to run the latest desktop technologies...

I think the Xfce 4.8 release announcement [2] makes a very short yet accurate summary of the problem non Linux systems are facing with regard to desktop environments.

Another subsystem worth mentioning that can be an issue is inotify (filesystem change monitoring). BSDs systems implement such a feature using `kqueue(2)` which has a completely different interface. That said, some applications have been successfully ported (like gamin). There are of course more to it than these two examples...

Unimplemented features

Several functions defined in GNOME applications are `ifdef __linux__` and left as a porting exercise for other systems, while using fallback stubs in the meantime.

The difference with the previous chapter is that these functions usually do not make use of Linux specific interfaces, but wouldn't work on other systems out of the box because the system calls can be different, or any other reason. That is perfectly OK and it is part of our job to port these functions to whatever OS we want to support.

Working with upstream

Bugzilla: *https://bugzilla.gnome.org/* is where all major contact with upstream takes place.

GNOME is not really a fully bundled Desktop environment but instead consists of separate modules that can be put together. This in an important fact when dealing with upstream because the feedback can be totally different between applications maintainers.

We've had some great experience working with some upstream projects where we learned a lot about the corresponding application itself or the way GNOME code is structured. Some maintainers are very opened to supporting other systems and committing portability patches.

Unfortunately this is not always the case and we have several bug reports that are rotting without any feedback from the developers. There is no easy way around that and it can become quite a frustrating and challenging task to find someone willing to review a patch, let alone commit it.

How can OpenBSD benefit upstream projects

Despite the obvious benefit to get more users (hence more testers), GNOME can also take advantage of supporting OpenBSD in several ways.

First OpenBSD is a very good platform to find bugs :-) Since we support many different hardware architectures, it is not uncommon for us to run into issues not seen under x86/x86-64 like different endianness for example. Some of our architectures run on very slow hardware (ARM, SPARC...) which are usually good at finding race conditions in software.

Our toolchain, `malloc(3)` and limits (which are way lower than on Linux, i.e. not unlimited) are also very good at finding buffer overflows, memory leaks and alike: we usually run into these very fast where it would be detected on Linux only under the right circumstances.

As we will see in the next section, gnome-shell isn't available on OpenBSD yet. As a *nice* side-effect this means that we are forced to use the gnome-fallback session as our daily desktop. Obviously, upstream doesn't test the fallback session as intensively as the regular one: as a result we encountered some bugs and pushed a

On the 'Net

- systemd as external dependency – *https://mail.gnome.org/archives/desktop-devel-list/2011-May/msg00427.html* [1]
- *Xfce 4.8 released – http://www.xfce.org/about/news/?post=1295136000* [2]
- *GNOME portability matrix – http://live.gnome.org/PortabilityMatrix* [3]

couple of patches that fixed some minor yet annoying issues.

Some examples of recent projects where OpenBSD helped discovering issues not seen on most Linux configurations are pygobject and dconf.

Current showstoppers

Mesa: As mentioned before, gnome-shell hasn't been ported to OpenBSD yet. The reason is that we cannot update the clutter package due to a bug in Mesa (the OpenGL graphics library used by *X.org*). This bug is quite bad as it would either crash the application itself or the entire X session.

Several people are actively working on this but the issue has yet to be found.

GDM

GDM 3.X is currently in a non functional state on OpenBSD. It relies on many PAM functions and the porting to `bsd_auth(3)` hasn't been done yet. Before moving forward on this we are waiting on feedback from the gnome-screensaver developers regarding our `bsd_auth(3)` patch there. So far we've been ignored...

Misc

While being worked on, several other functionalities are still either missing or incomplete on OpenBSD.

- pulseaudio – This (in)famous and controversial sound server hasn't been ported yet. Most of the time we can do without it, except for the media key plugin in gnome-control-center that we have to patch out since unfortunately pulseaudio is a required dependency (used to control some multimedia aspects of the desktop like volume... with keys from a multimedia keyboard)
- network-manager – This is a piece that will probably never be ported to OpenBSD. So far we haven't lost much by not having it. We did get scared for a little while when the epiphany developers added it as a required dependency but this was made optional after we told them this would prevent epiphany from running on systems which do not have network-manager.
- libgtop2 – A huge amount of work was made recently in porting missing functions. Despite the use of a temporary hack to get open files statistics (using `lsof(8)`), everything now works natively.
- avahi – Basic functionality works fine, but we don't have support for the autoipd feature (and probably never will). There is one important missing feature that we have to port: the chrooting of the daemon. Currently it is done using capabilities which we have no support for yet.
- accountsservice – While technically a *Freedesktop.org* project, it's used heavily by gnome-settings-daemon and the gnome-control-center. This software is a good example of how people develop having only Linux in mind. Most paths are hardcoded and spawned utilities assume Linux behavior (`usermod(8)`, chkconfig...). There is currently work to create an external helper for OpenBSD.

Conclusion

As we've seen there are numerous challenges in porting GNOME to OpenBSD. In the end this effort is good for both sides: upstream who gets more users and more bug reports and fixes for issues that were hidden or not easily seen on Linux and OpenBSD because it forces us to keep up with all the required technologies.

I think it is very nice that we can provide a *Joe User* Desktop on top of the most secure Operating System and according to the GNOME portability matrix [3], we are getting very close :-)

I would like to mention the tremendous work done by the *gnome@FreeBSD.org* team which really helped us starting on the right tracks some years ago, and more specifically Joe Marcus Clarke (*marcus@FreeBSD.org*) for his great contributions into making BSDs a better supported platform for GNOME.

Note that I'm not the only one involved in all this work, I've been sharing the entire burden for several years with another hobbit: Jasper Lievisse Adriaanse (*jasper@openbsd.org*), so big thanks to him obviously!

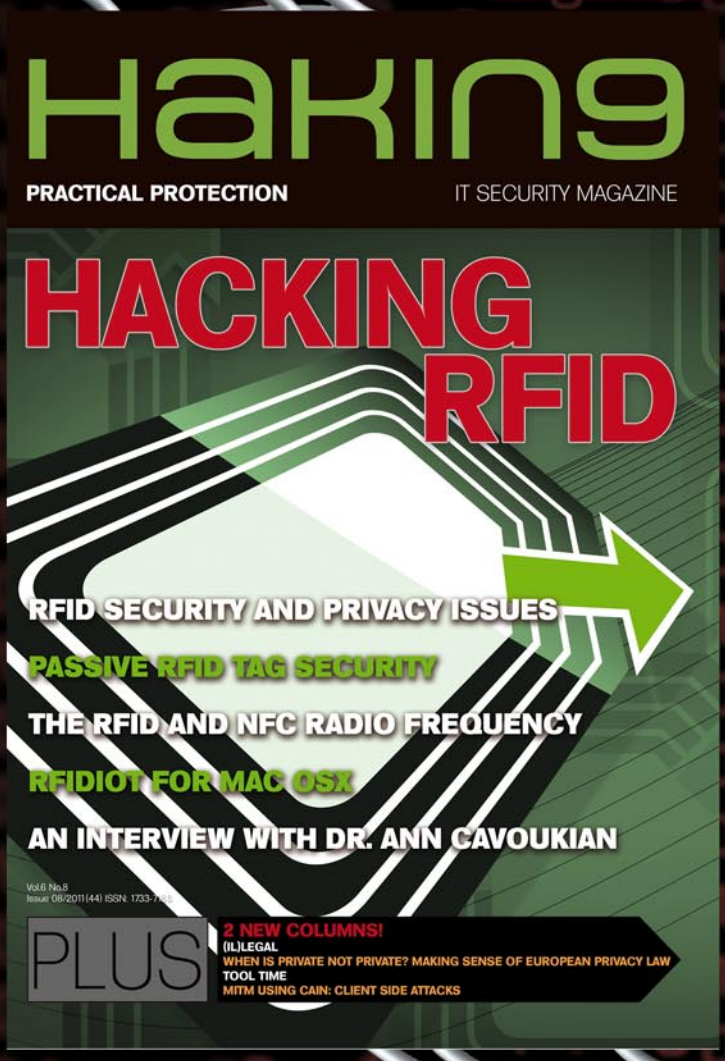
Currently we have two OpenBSD developers with commit access to the GNOME git repository, Jasper and Jonathan Matthew (*jmatthew@openbsd.org*) so I will try and convince them of removing Linux as a supported platform and help making GNOME an OpenBSD only Desktop ;-)

ANTOINE JACOUTOT

Antoine Jacoutot is an OpenBSD committer who lives in Paris, France. He is responsible for more than 300 packages, wrote the sysmerge(8) utility and is part the OpenBSD rc.d(8) framework development. He runs OpenBSD for pretty much everything.

Hakin9

PRACTICAL PROTECTION



IT SECURITY MAGAZINE

What It Takes

Starting and Running an Open Source Certification Program, Part III

This is the third part in our series on what it takes to run an Open Source Certification Program. In Part I we discussed "People", the kinds of people you will need to help you run a Certification Program for your most excellent software.

Then in Part II we talked about *Processes* – how to get the Certification Program set up as a business and how to start up the exam development effort. This month we'll look at *Technology* – what kinds of tech you can use to get things done. Most of the items in this article are about well known topics- websites, monitoring, collaboration tools, and so forth, so we won't go into elaborate detail. If you would like more detail, drop us a line at *info@bsdcertification.org* and we'll be happy to oblige.

Website

It seems every business has a website these days. It's the easiest, simplest way to *hang out your shingle* for everyone to see. Because it's a public facing system, you'll want to consider the security of the system in its design and operation. For our *BSD Certification Group* (BSDCG) website, we naturally chose a BSD operating system. Fortunately, BSD systems have several different packages for webservers and administration tools. As with any website, it's best to keep the operational requirements to a bare minimum. In Part II we noted that while we do have a website, none of our certification test items (questions or answers) are on that site. We wanted to make sure our site was always available, so we engaged with a group in Brazil to set up a mirror. We currently use *rsync* to synchronize any changes we make on the primary system to the synchronized secondary system. There is no cost and relatively little overhead with this approach .

Monitoring

You should install intrusion detection and prevention / monitoring tools such as Snort or Suricata. These tools can alert you when you have a problem such as a denial of service attack or an unauthorized login. In some cases, your hosting provider can provide additional protection by filtering upstream traffic. Keep in mind that any configurable security measures should be under some form of change control. This will help you if you decide to get your Certification Program accredited by ANSI/ISO as we discussed in Part II.

Surveys

You'll occasionally need to poll your user community for information, such as what kinds of certifications to develop, or what the name(s) should be. Taking a survey and producing usable reports can be a challenge. There are commercial tools such as SurveyMonkey, SurveyFrog, Zoomerang, and others. If, as we did, you want to stay as much open source as possible, there are tools such as FreeOnlineSurveys, and phpESP.

We found that we could not use some of the 'free' tools since they provided only limited service such as only 12 questions, 100 responses, and 10 days active. If your needs go beyond the limitations of these *free* tools, consider hosting your own survey with your own software. We found phpESP to be easy to integrate into one of our BSD hosts. Survey creation and use was tolerable, but we found ways to automate the basic tasks using some perl scripting and knowledge of SQL. This allowed us to rapidly create surveys with hundreds of questions, and since we hosted it ourselves, we could control the active time and number of responders. If this is the way you want to go, keep in mind that you'll need some technical know-how for databases, scripting, other technical tasks.

Collaboration

By now, you've got hundreds, maybe even hundreds of thousands of people using your software and there is a lot of buzz. People need to communicate, and it's your job to enable them to do that in multiple ways. Fortunately, that's what the Internet excels at- communication and collaboration. We set up several mailing lists with GNU Mailman. We have one for our public discussions about our Certification Program and our announcements, one for our private business discussions, one for our infrastructure team to keep up with configurations and changes, and one for our *Subject Matter Experts* (SMEs). Mailman is easy to administer and has proven capable of easily handling our workload. For real time discussion and collaboration, we use *Internet Relay Chat* (IRC). We have our `#bsdcert` channel set up on freenode.net. This also serves as a friendly chat forum for us as well.

With the rise of social networking we've taken advantage of groups on LinkedIn and Facebook. These groups allow us to immediately reach hundreds, even thousands of people with daily or weekly updates. It's also where we can reach out to the community for advice and support our initiatives such as new surveys, new exams, or collaboration on editing reports.

For additional collaboration, consider an open source document management system (also known as a content management sytsem – CMS). There are some excellent CMS systems in use such as Plone or Drupal. Some of them are very easy to use. If you decide to manage your content in one of these systems, consider getting a support contract with a firm that specializes in supporting that software. You will get timely notification of bugs and security holes- and hopefully fixes for your money.

Candidate Registration

All certification systems need a way to identify each candidate with some form of identification number. The BSDCG developed an on-line application to take the candidate information and give them a registration number. This number is used to identify the candidate within the BSDCG internal systems, not on any public website. Registration numbers are also used when printing exam results and certificates.

Eventually we plan to have another web based system to enable employers to verify whether a prospective employee is BSDCG certified. For now, this function is performed manually, via email to *chair@bsdcertification.org*.

Test Construction and Delivery

Test construction and delivery will be your biggest challenge over time. It is possible to start and run an Open Source Certification Program with nothing but paper and pencil, but it very quickly becomes unsustainable. There will be hundreds, possibly thousands, of items and answers. And there will be plenty of errors and corrections. Computerizing your construction and delivery are essential.

The discussion in this section centers on creating and maintaining your item collection within your own systems. If / when you migrate to an online test creation systems such as *Schroeder Measurement Technologies* (SMT) your test creation, delivery, and scoring procedures will be completely different. However, the basic concepts will remain the same.

Item Collection and Management

The BSDCG started out with item collections as just word processing documents, mostly in Open Office. These were easy to organize and exchange with our psychometrician. They were also easy to distribute to our SMEs when we had question writing sessions. And while these SME sessions generated a number of edits across several documents, it wasn't that much work to keep it all straight. The problem was pushing out questions into an actual exam.

Form Creation

Psychometricians and exam creators use special terminology when describing exam construction. A *form* is the name given to the collection of items that will be put onto an exam. Creating our exam forms required several steps as outlined in this *simplified view*:

- Collecting items by knowledge domain. Knowledge domains and objectives were developed during the Job Task Analysis. In this step, we are collecting actual items to be used on the exam.
- Ensuring that all the objectives are covered by at least one item, and ensuring that only the highest quality items get put onto an exam. This is done in item writing sessions with Subject Matter Experts.
- Drafting up a form by copying selected items from each domain and randomizing the order on the form.
- Double checking to ensure the percentages of all items for each domain are correct. The exam objectives document contains the percentage for each domain established by the psychometrician.
- All forms are vetted and approved by the psychometrician before use. This usually requires a beta period where items are tested for psychometric correctness.

This results in the actual form to be used by the exam candidate. That form is now ready for packaging and delivery.

Exam Survey

At the end of every exam form we include a short survey. The exam instructions ask the candidate to fill out the survey before returning the exam materials back to the proctor. This survey is about the exam session itself – Was the room quiet? Was it too hot or too cold? Did the proctor explain the procedures clearly? Were there any disruptions during the exam, etc. Getting personal feedback about the exam session is important, and is required for accrediting your Certification Program. Plus, it gives the candidate a chance to communicate directly with the exam program administrators. With all BSDCG exams, we read every survey and take note of every comment.

Answer Sheets

You will want to discuss scoring the results of the exam with your psychometrician. To be effective and reliable, the scoring must be independent of the exam delivery process. Your psychometrician may want to use specific equipment or have you engage with a separate company to score the exam and report the results.

In our case, we engaged Dr. Sandra Dolan to score the exam for us as part of her contract. She uses a scoring machine that reads the *Pearson NCS MM4521-3 Answer Sheet*. These forms, familiar to almost anyone who has ever taken a certification exam, are commonly known as *bubble sheets*. Each item has a row of tiny circles (*A, B, C, D, E*) for answer marks. The form also has a place for the candidate name, birthdate, and some identification codes. Note that these forms are copyrighted by Pearson NCS and must be purchased, not copied, for use.

Packaging and Test Delivery

Paper and pencil exams have very specific procedures that must be followed in order to protect your exam content. You can't just send a bunch of copies in an envelope and mail it to the proctor. If you do that, you have no way of knowing if the contents were copied or stolen by another candidate, the proctor, or someone else. In fact, when followed, the procedures below permit only the exam shipper who has already signed a Non-Disclosure Agreement and the candidate themselves to actually view the exam. No one else, including the proctor, will be able to view the actual exam with these procedures.

Here are the packaging procedures we use in the BSDCG:

- A copy of the exam form is printed and stapled together by the exam shipper. Each BSDA exam is about 12 pages long.
- Immediately after printing, the exam form is sealed with an adhesive tab.
- The first page of the exam form consists of a Non-Disclosure Agreement indicating that, by opening the seal, the candidate agrees not to discuss any aspects of the exam. The candidate is required to sign the agreement before continuing with the exam.
- The sealed exam form, a single answer sheet, and one blank colored sheet of paper are put into a numbered manilla envelope. The colored sheet is available for the candidate to write down any issues they have with any question. It gives them the chance to indicate that a question was unclear, if they thought the answers were all wrong, or any similar issue.
- The envelope is sealed. Special BSDCG packing tape is placed across the seal to prevent tampering. Only the exam candidate is permitted to open the envelope seal, and the exam seal. Once they are finished with the exam, they are required to place the exam form, answer sheet, and colored paper back into the envelope and reseal the envelope, writing their name across the seal.
- We maintain an internal spreadsheet that contains the candidate name, exam form number, and the envelope number as well as the date and location of the exam and the name of the proctor. These details are important if it later becomes obvious that there were problems with the exam session.
- The same procedures above are performed for all candidates at a particular exam session. The collection of sealed exam packs are then boxed up and mailed directly to the proctor. Included in the box are the proctor directions and a list of candidates expected for that session.

If you are mailing internationally, you'll need to allow plenty of time for the materials to reach the proctor before the exam. Six weeks lead time is highly recommended.

Proctoring the Exam Session

On the appointed day, the proctor arrives at the exam location, and sets up the exam session room. Proctors are required to check the identification of each candidate (a government issued photo ID is required). The candidate is given their exam packet and told to have a seat until the start of the exam.

Once the proctor begins the exam session, no further candidates are allowed into the exam session. The

proctor reads the exam instructions, verifies that everyone is ready, and starts the clock.

At the end of the exam, the proctor (as noted above) directs the candidates to place the exam form, answer sheet, and colored paper back into the envelope and reseal it, writing their name across the seal. Sealed exam packs are boxed back up and returned to the BSDCG for scoring.

Scoring and Reporting

The exam packs are returned to the exam shipper who checks each pack for completeness, reads the sign-in sheets, the exam survey, and all notes which are recorded. A copy of the answer sheet is archived and the orginals are sent to the psychometrician. The actual exams are also archived and securely stored at the BSDCG home office.

Scoring the exam is performed by the psychometrician to ensure proper separation of duties for exam delivery and scoring. The scoring activity itself is automated. Answer sheets are fed into a special reader (Scantron iNSIGHT 4ES, OpScan 4ES, or similar reader) which uses special software.

The raw answers for each scan are captured and are fed into additional software used for psychometric analysis. This is an important aspect of the scoring procedure. During this analysis, the psychometrician can compare the results from a single test to other recent exam scores. For example, the analysis can reveal whether an item is consistently being answered incorrectly – indicating a poorly worded item or answer set. It can also reveal if there are too many commonalities in answers among a group of candidates in one exam session – indicating possible cheating. This analysis is vital to the integrity of the Certification Program.

Once the items are scored it is time to communicate with the candidate whether they passed or failed. A PDF report indicating whether they have passed or failed is generated and sent to the candidate. If the candidate passed, a certificate is printed, an individual foil seal is applied to the certificate, the remaining third signature is added, an ID card and welcome letter are printed, and all are mailed to the candidate.

Assessing the Program's Effectiveness

How will you know if your Certification Program is successful? Money is one obvious answer, but not the only one. You'll be successful if you can demonstrate that your program is effective in supporting your most excellent software. Below are some metrics that may be useful in tracking effectiveness:

- *Registrations* – How many people are interested in your Certification Program enough to actually register to take the exam? While more registrations are better overall, there are several reasons why people will register and then not actually take the exam – lack of time, distance from a test center, a change in personal goals, and others. On the other hand, a continual stream of registrations shows that there is still continued interest.
- *Tests Given* – How many tests sessions have been scheduled and completed? This is a concrete number of people who have registered, paid, shown up and actually taken the exam and is a very useful metric.
- Candidates Pass / Fail – The number of people who pass and the number of people who fail the exam are very useful indicators about the quality and validity of the exam. Consider the extremes. If everyone failed, the exam wasn't focused on the objectives, or the items were somehow distorted. If everyone passed, the items were too easy. Your psychometrician will help determine the best pass / fail ration for your Program.
- *Usefulness* – How many candidates took the exam and were able to advance their careers with this new certificate? If candidates can show their certificate to a prospective employer who recognizes the value of the Certification Program, you are definitely on the right track.
- *Buzz* – Does your Certification Program have industry buzz? Are media pundits, bloggers, and technology journalists aware of your program? This is a very useful, if ephemeral metric. Enjoy it while it lasts.

Conclusion

You've built your software. You've built a community of dedicated People. You've set up your Procedures and your Technology as you've built your Certification Program. Now all you have to do is conquer the technology world. Carpe Diem!

JIM BROWN

Jim Brown has worked in the computer industry with continuous Unix involvement in development or administration since the early 1980s. His experience includes applications, systems and database programming, in a variety of languages. One of the founders of the BSD Certification Group, he is helping to develop the BSD Professional certification. He currently lives in Northwest Arkansas, USA.

In the next issue:

- ArabBSD
- Porting of libgtop2 to OpenBSD
- Dtrace on FreeBSD
- and Other !

Next issue is coming in September!

HACKTIVITY

The Largest Hacker Conference in Central and Eastern Europe

September 17-18, 2011. Millenáris / HUNGARY, BUDAPEST

keynote speakers:

PETER SZOR / USA

RAOUL CHIESA / ITALY

speakers:

VIVEK RAMACHANDRAN / INDIA wireless worm - the founder of securitytube.net

ERTUNGA ARSAL / GERMANY SAP security

JOSEPH MCCRAY / USA mobile phone security - Air Force veteran

ALEXANDER KORNBRUST / GERMANY Oracle Forensic

PAVOL LUPTAK / SLOVAKIA Cryptoanarchy

YANIV MIRON / ISRAEL SCADA security

MICHELE ORRU / ITALY BEeF - Browser Exploitation Framework

WIKILEAKS POST-MORTEM

HACK THE BRAIN – PSYCHO

STUXNET

DATABASE SECURITY

CRIPTOCHIPS' SECURITY

HARDWARE HACKING

SECURITY OF VIRTUALIZATION

CRIPTOGRAPHY

LAW AND SECURITY

FIRST EUROPEAN ONLINE CERTIFIED ETHICAL HACKER (CEH) COURSE WITH NetACADEMIA

EC-Council - The Global CyberLympics - CEE finals

LOCKPICKING (NON-DESTRUCTIVE LOCK-OPENING) LECTURE AND WORKSHOP

- this year we organize CTF game again with qualifying round on the web
- wargame putting emphasis on web-vulnerabilities
- hello workshops: jump from theory to practice
- old-timer computers brought back to life and you can see them under power
- hacker road, where you can learn about the history, present and future of hacking
- separate section for the history of Hungarian hacking

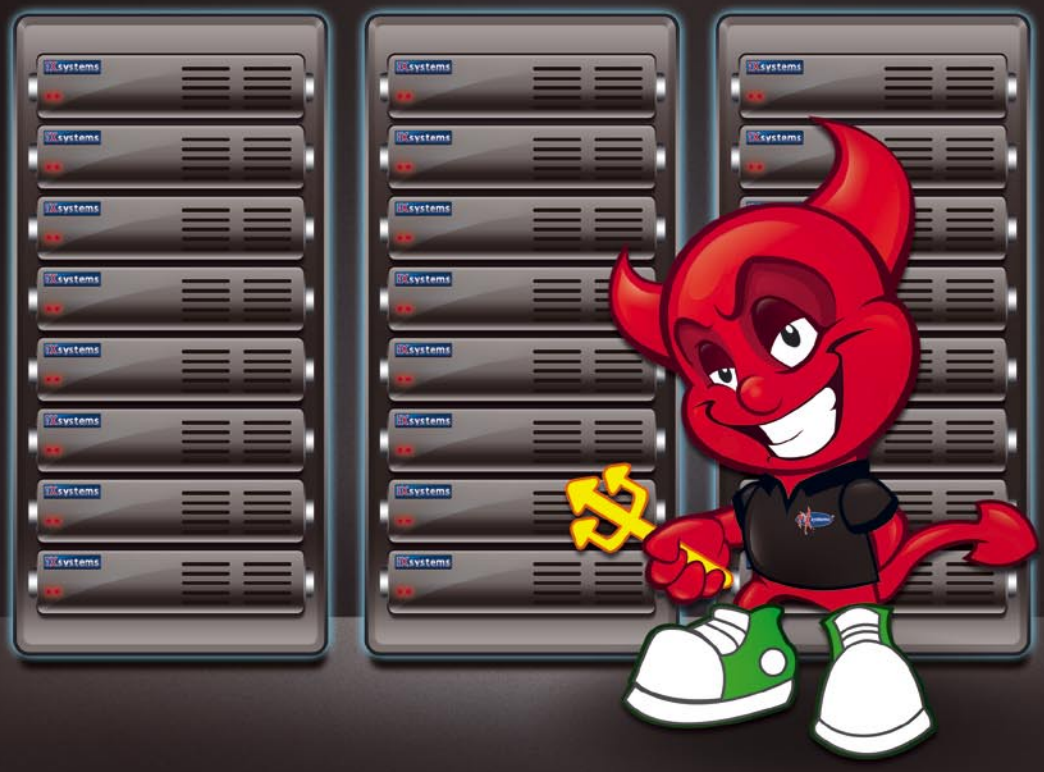
AND A BIG BIG SATURDAY NIGHT PARTY

Tickets are available until 10th of September with 10% discount on www.hacktivity.com

Full prices
for adults: 60 EUR
for companies: 120 EUR

further information and registration: www.hacktivity.com

What has your server vendor done for BSD lately? Probably, not much.



Work with a vendor that supports the operating system you love!

iX is the corporate sponsor of the PC-BSD® Project, a major corporate donor to the FreeBSD Foundation, and leads the FreeNAS™ development team -- all while employing some of the most brilliant minds in the FreeBSD® community. For BSD hardware and software expertise, look no further.

1-855-GREP-4-IX
http://www.iXsystems.com/community

iXsystems®