MAGAZINE

BSD

FOR NOVICE AND ADVANCED USERS

Adobe® PDF Magazine Version

ZFS AND FREEBSD

INSIDE

VOL.4 NO.2
ISSUE 02/2011(19)
1898-9144

10 Gig On Board

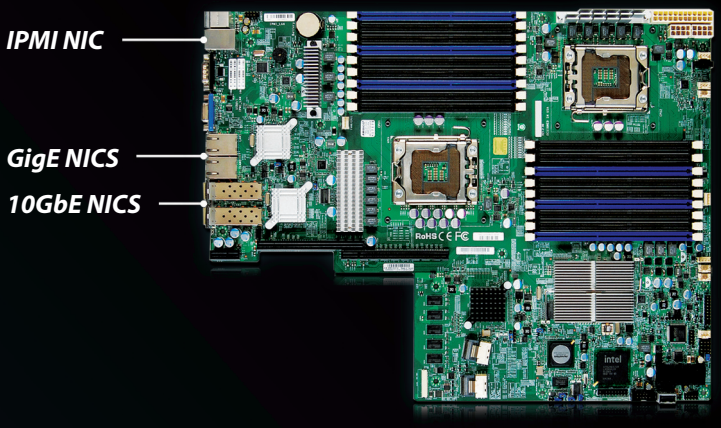
On-Board 10 Gigabit Ethernet Adapters leave your existing PCI-E slots available for other expansion devices.

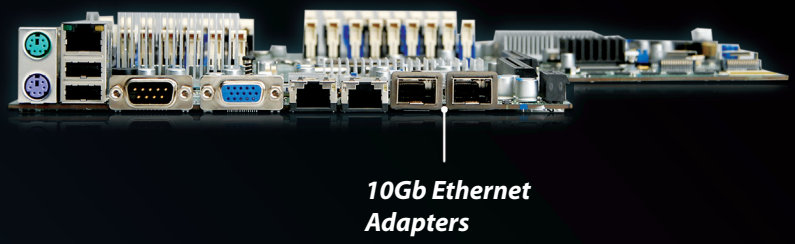
30% cost savings/port over equivalent Dual-Port 10 GB PCI Express add-on card solution

Blazing Fast, Embedded 10Gb Ethernet

10G Rackmount Servers in the iX-Neutron server line feature the Intel® Xeon® Processor 5600/5500 Series, and come with 10GbE networking integrated onto the motherboard. This eliminates the need to purchase an additional expansion card, and leaves the existing PCI-E slots available for other expansion devices, such as RAID controllers, video cards, and SAS controllers.

For more information on the iX-1204-10G, or to request a quote, visit: **http://www.iXsystems.com/neutron**





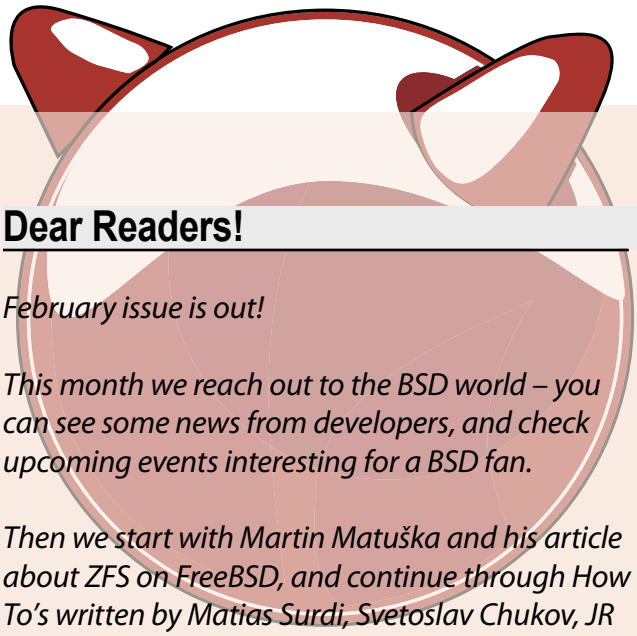
KEY FEATURES:

- Supports Dual 64-Bit Six-Core, Quad-Core or Dual-Core, Intel® Xeon® Processor 5600/5500 Series
- 1U Form Factor with 4 Hot-Swap SAS/SATA 3.5" Drive Bays
- Intel® 5520 chipset with QuickPath Interconnect (QPI)
- Up to 192GB DDR3 1333/1066/800 SDRAM ECC Registered Memory (18 DIMM Slots)
- 2 (x8) PCI-E 2.0 slots + 1 (x4) PCI-E 2.0 (in x8 slot -Low-Profile - 5.5" depth)
- Dual Port Intel® 82599EB 10 Gigabit SFP+ - Dual Port Intel® 82576 Gigabit Ethernet Controller
- Matrox G200eW Graphics
- Remote Management - IPMI 2.0 + IP-KVM with Dedicated LAN
- Slim DVD
- 700W/750W Redundant AC-DC 93%+ High-Efficiency Power Supply

Call iXsystems toll free or visit our website today!

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

Powerful.
Intelligent.



Dear Readers!

February issue is out!

This month we reach out to the BSD world – you can see some news from developers, and check upcoming events interesting for a BSD fan.

Then we start with Martin Matuška and his article about ZFS on FreeBSD, and continue through How To's written by Matias Surdi, Svetoslav Chukov, JR Aquino, and Michael Hernandez. As always you will also find interesting reflections in Let's Talk section.

Even though not everyone can feel it, spring is coming soon, and if there is one thing BSD fans should await it for – it surely is BSDCan Conference. Concidering this we have prepared a short interview with Dan Langille – member of BSDCan Comitee, and last years presentation by Kris Moore as a taste of what you can expect this year :)

I would also like to mention that we're looking for new Betatesters, so if you are interested in joining our team as one – please contact us.

Enjoy your reading!

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:
JR Aquino, Svetoslav Chukov, Mark Faust, Michael Hernandez, Martin Matuška, Matias Surdi, Sufyan bin Uzayr

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:
Karolina Lesińska
karolina.lesinska@bsdmag.org

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 AUPUS

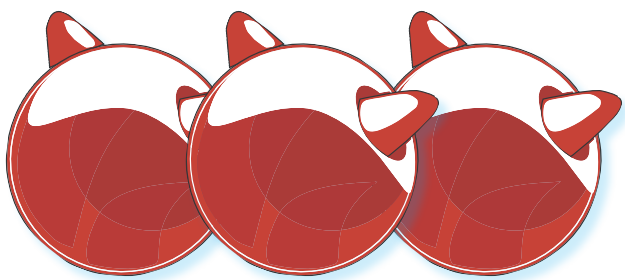
Mathematical formulas created by Design Science MathType™.

Get Started

How To's

Let's Talk

BSDCan 2011



MIROS SUPPORT ADDED TO PKGSRC

On January 23, an initial set of patches for MirOS support has been committed to pkgsrc by Alistair Crooks. As a part of these is still on hold for the moment, some patches are still necessary for successfully bootstrapping. You can bootstrap pkgsrc as root, installing for example into /usr/pkg, or as an unprivileged user. MirPorts and pkgsrc can be used in parallel.

Even if only a small number of packages has been tested on MirOS so far, the adoption of pkgsrc will give users access to vast, well-maintained archive of third-party software.

pkgsrc is a *ports tree* developed by the NetBSD project and portable to several other operating systems. Support for MirOS has existed since 2006 in the form of a private patch. Since January 2011, MirOS support is contained in the upstream pkgsrc repository.

pkgsrc allows the easy installation of third-party software not contained in the base system. It is very similar to MirPorts, the ports framework developed by the MirOS project, *native* on MirOS. However, pkgsrc generally contains more and newer software.

For more information on pkgsrc, see *http://www.pkgsrc.org/*.

MirPorts and pkgsrc can be installed and used in parallel. However, there is one caveat: the package tools are incompatible but they have the same names. Thus, when installing packages, you must assure that the right ones are used. Moreover, when bootstrapping pkgsrc on MirOS, the MirPorts directories (normally `/usr/mpkg/bin` and `/usr/mpkg/sbin`) must not be in your PATH.

Needed patches

At the moment, these patches must be manually applied.

- bmake.diff (*http://dl.dropbox.com/u/17617299/MirOS/bmake.diff*): this patch fixes the bmake test suite so that bootstrap can succeed.
- pkgsrc-libtool-miros-2.diff (*http://dl.dropbox.com/u/17617299/MirOS/pkgsrc-libtool-miros-2.diff*): work-in-progress version of MirOS support for libtool-2.2.6b. Without this, packages using libtool only build static libraries.

From MirOS BSD by Benny Siegert

PACKAGES FOR PKGSRC-2010Q4 BUILT

The uploads are finally complete for binary packages on Avalon. I think each of the package-building machines crashed at least once during the process, but thanks to Matt and Mike and others, they were restarted/fixed quickly.

I've changed the links on avalon, so `pkg_radd` for DragonFly 2.8 and DragonFly 2.9 will now download pkgsrc-2010Q4 packages. There's lots of packages:

```
i386/DragonFly-2.8/pkgsrc-2010Q4      9406
i386/DragonFly-2.9/pkgsrc-2010Q4      9406
x86_64/DragonFly-2.9/pkgsrc-2010Q4    8900
x86_64/DragonFly-2.8/pkgsrc-2010Q4    8917
```

If you get errors asking for a new `pkg_install`, see the *Update pkgsrc system packages* section on the pkgsrc page on the DragonFly BSD site: *http://www.dragonflybsd.org/docs/howtos/HowToPkgsrc/* (Even if you don't, it's still good information.)

I haven't tested this too heavily, but it may be possible to upgrade packages automagically with `pkg_radd -uv <packagename>`. This may work better with packages that have less dependencies. i.e. upgrading Vim may work, all of KDE won't.

Make sure that your `/usr/pkgsrc` is on the pkgsrc-2010Q4 branch so that everything matches. Check with `cd /usr/pkgsrc; git branch`. If you're on an earlier branch, switch with `git branch pkgsrc-2010Q4; git pull`. (I think; someone correct me if it's wrong.) If you're on pkgsrc master, stick with it unless it's from before 2011, in which case switching to pkgsrc-2010Q4 won't be any trouble.

About pkgsrc-current

I'm cobbling together a system to build pkgsrc-current on DragonFly-current. Max Rotvel kindly contributed a CPU, and the last item I need now is some DDR2 RAM. If you're willing to donate 2x 2G sticks, please mail me.

(I've been building pkgsrc-current on a VM very nicely contributed by Jan Lentfer. However, I'd like to have something I can physically reach when it has trouble, and has a bit more horsepower.)

From DragonflyBSD byJustin C. Sherrill

DRAGONFLY BSD / GOOGLE CODE-IN 2010 FINAL REPORT

During the Recent Google Code-In there were a total of 2167 tasks successfully completed by the 13-18 year old students. DragonFly's portion of these amounted to 72 successfully completed tasks, or around 3.3% of the total. Slightly lower than a perfect proportion considering there were 20 projects participating. In my estimation, however, we did quite well considering that tailoring tasks of the caliber required in order to benefit an operating system to 13-18 year old minds is quite challenging. That said, a number of students were able to tackle reasonably large

and complex tasks that many of us (mentors) would not have thought feasible, if even possible, at the start of the program. Overall, I believe the outcome of the program is as good as any of us could have hoped.

As mentioned in previous status emails, the documentation tasks received a wholly underwhelming response. When the program opened DragonFly had around 35-40 tasks, roughly half of these were documentation work, with the other half being code-related. Now, after the close of the program, there are 72 completed tasks, mostly code related, while 20 tasks went uncompleted or unclaimed. Nearly all of these unclaimed tasks are of the documentation variety, and many simply sat dormant the entire duration of the program.

Prior efforts invested in organizing and maintaining the various project pages on the DragonFly BSD wiki proved invaluable in the specification of a number of the tasks successfully completed during the program. I believe that more effort spent defining worthwhile tasks and specifying them in such a way that they may be broken down into bite-size units of work would easily pay dividends if the project were to participate in a program of this type in the future. Brief notes on the completed projects:

- EXAMPLES sections were written for the setitimer(2), getsockopt(2)/ setsockopt(2), socket(2)/accept(2)/bind(2)/ connect(2), sendfile(2), writev(2), select(2), poll(2), fork(2), send(2)/recv(2), mmap(2), setjmp(3)/longmp(3), dladdr(3)/ dlinfo(3)/dlopen(3), directory(3)/scandir(3), ucontext(3)/ makecontext(3)/getcontext(3)/setcontext(3), msgctl(3)/ msgget(3)/msgrcv(3)/msgsnd(3), glob(3), popen(3)/system(3), exec(3) and tree(3) manpages. *
- A patch was created to make the hammer(8) iostats command display humanized output. *
- A devattr tool was written.
- A libfsid was written.
- A `usage()` function/help output was added to vkernels.
- sysctl documentation strings were created for lwkt.*, p1003_1b.*, debug.*, net.inet6.*, net.inet.*, vfs.*, vfs.nfs.*, vfs.hammer.*, vm.stats.* and kern.ipc.* sysctl's.
- The default password hashing method was changed from md5 to sha2.
- Installation and vkernel setup screencasts were created and put on YouTube, *http://www.youtube.com/user/dragonflybsd*
- FTP server documentation was ported, *http://www.dragonflybsd.org/docs/newhandbook/FTP/*
- A document detailing hammer recovery was written, *http://www.dragonflybsd.org/docs/docs/howtos/howtorecoverdataonhammerfs/*
- 20+ pkgsrc packages were fixed and patches submitted to pkgsrc or upstream.
- Patches were submitted to convert various subsystems from zalloc to objcache, including: nfsnode, nfsmount, kqueue, dirhash, aio and crypto. *
- Most kernel usage of `m_get()` was converted to `m_getb()`. *

Those items with a * appended are not yet committed or only partly committed, most/all of the results of these tasks are committable and will hit the tree, but if you want to adopt something and get it in sooner than later feel free to let myself, alexh or another mentor know and we can fish out the patch for you. A big thanks to Google for the opportunity and the mentors and students for their time and effort.

From DragonflyBSD by Samuel J. Greear

ACTIVITY IN 0.4-CURRENT

Right after the release, I focused on getting some contrib software up-to-date. While writing the release notes for 0.3, it was apparent that we needed to get on the ball.

OpenSSL, OpenSSH, file, GNU sort, awk, sqlite, tcsh, BIND and sudo have all been updated in the last month. I've added it(4) and lm(4) to work with the sensors framework introduced in 0.3. eeemon(4) was recently added for hardware monitoring on some Asus eee PCs. alc(4) was introduced for atheros gigabit lan cards. ale, alc, ae were all added to GENERIC in current.

Intel coretemp(4) monitor was modified to work with the sensors framework. It's now possible to monitor the CPU temp with sensorsd on Intel CPUs. I'm working on adding similar functionality to amdtemp.

A locking fix was introduced today on the route code related to ICMP traffic.

From MidnightBSD by Lucas Holt

FREEBSD 7.4/8.2 RC3 AVAILABLE

The third (and probably last) Release Candidate builds for the FreeBSD-7.4/8.2 release cycles are now available. For 8.2-RC3 the amd64, i386, ia64, pc98, powerpc, and sparc64 architectures are available. For 7.4-RC3 the amd64, i386, pc98, and sparc64 architectures are available. ISO images for these architectures can be downloaded from most of the FreeBSD mirror sites (*http://www.freebsd.org/doc/en_US.ISO8859-1/books/handbook/mirrors-ftp.html*). Please see the official announcement (*http://lists.freebsd.org/pipermail/freebsd-stable/2011-February/061353.html*) for further details about these releases.

From FreeBSD

Conferences

DrupalCon Chicago 2011

March 7-10
Chicago, USA

http://chicago2011.drupal.org/

AsiaBSDCon 2011

March 17-20
Tokyo, Japan

http://2011.asiabsdcon.org/index.html

Indiana LinuxFest 2011

March 25-27
Indianapolis, USA

http://www.indianalinux.org/cms/

Flourish! 2011 Open Source Conference

April 1-3
Chicago, USA

http://www.flourishconf.com/2011/

Conferences

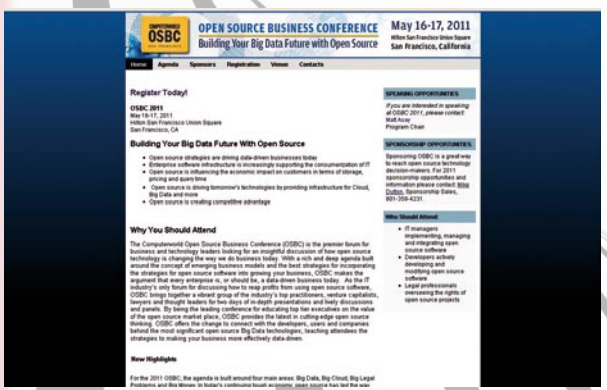
BSDCan 2011

May 13-14
Ottawa, Canada

http://www.bsdcan.org/2011/

Open Source Business Conference

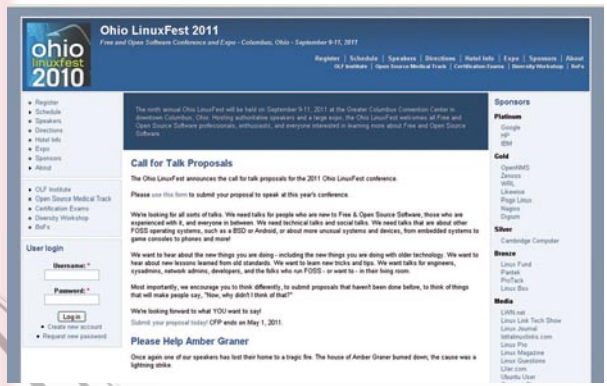
May 16-17
San Francisco, USA



http://www.eiseverywhere.com/

Ohio LinuxFest 2011

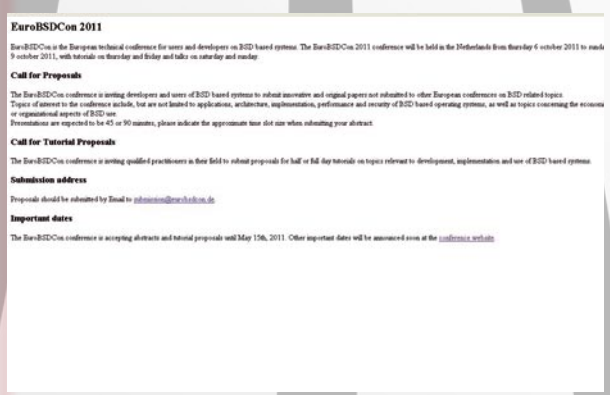
September 7-11
Columbus, Ohio, USA



http://ohiolinux.org/

EuroBSDCon 2011

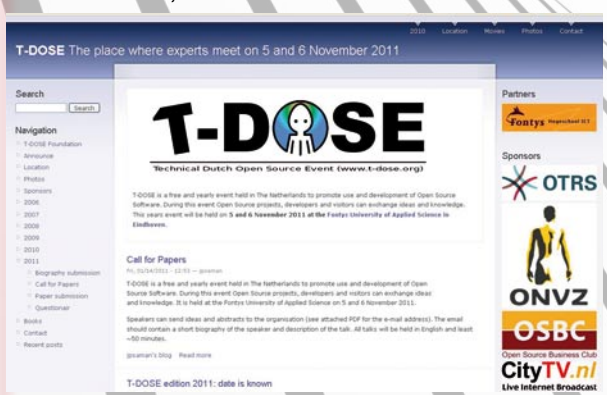
October 6-9
Netherlands



http://2011.eurobsdcon.org/CfP.html

T-DOSE 2011

November 5-6
Eindhoven, Netherlands



http://www.t-dose.org/

ZFS and FreeBSD

The Zettabyte Filesystem (ZFS) is one of the most advanced open source filesystems available today. Its design implements several revolutionary ideas with focus on data consistency, performance and ease of use.

What you will learn…

- What is ZFS and what are its features
- Which operating systems do ship ZFS
- What is the state of ZFS in FreeBSD

What you should know…

- Basic knowledge of computer data storage
- UNIX system administration basics

ZFS is a great filesystem and I am one of its happy users. My ZFS setups scale from standalone servers up to larger hybrid server farms with several operating systems. This article is going to give a brief overview of ZFS and the current progress of the implementation in FreeBSD.

Short history

ZFS was first publicly introduced in the OpenSolaris operating system in late 2005, followed by a first public release in Solaris Express. The port to FreeBSD was written by Pawel Jakub Dawidek in April 2007. Since then ZFS has undergone many substantial improvements and enhancements including level 2 cache, deduplication and encryption. In August 2010 the OpenSolaris project has been discontinued and the ZFS development continued in closed code. Therefore the latest available public ZFS pool version is 28 without encryption, which was introduced in closed source and made available for public testing in Oracle Solaris 11 express in late 2010.

Original ZFS features

The basic ZFS features (provided by all pool versions) are pooled storage, transactional semantics, checksums and self-healing, scalability, instant snapshots and clones, dataset compression and simplified delegable administration. I am going to give a brief introduction to these features in the following paragraphs.

Pooled storage

ZFS integrates a filesystem and a volume manager into one structure. Users create storage pools and on these pools filesystems and volumes. This feature is contended by many experts as it breaks the *traditional* file system layout. But on the other hand it provides features that are only possible if the volume manager knows more about the file system underneath. One of these features is data-based replication. If you are adding a new disk to a ZFS mirror (RAID-1), only data is replicated (*resilvered*) to the new disk device. The same counts for hot spare replacements in RAID-Z arrays (variations of RAID-5 and RAID-6). Another feature is that filesystems are not of fixed size, the only real limit is the available free space on the whole storage pool. Available space to a single filesystem (and its descendants) can be limited with the *quota* property and reserved with the *reservation* property (see Figure 1).

Transactional semantics

The writes in ZFS are performed using a transactional *copy-on-write model* (COW). New blocks are used to store

On-disk ZFS structures

The two main on-disk ZFS objects are `pool` and `dataset`. ZFS pool is the main storage object consisting of virtual devices (`vdevs`). These virtual devices may be:

- `disk` – drive, partition, FreeBSD GEOM object, etc.
- `file` – recommended for experimental purposes only
- `mirror` – group of two or more mirrored vdevs
- `raidz`, `raidz2`, `raidz3` – group of vdevs for single to triple parity RAID-Z
- `spare` – pseudo-vdev acting as a hot spare
- `log` – separate device for the ZIL (ZFS Intent Log), may not be raidz
- `cache` – separate device for the L2 (Level 2) cache, may not be mirror or raidz

ZFS pools are useless without datasets. Each pool contains one or more ZFS datasets. ZFS dataset is a generic name for:

- `filesystem` – this is the POSIX layer where the files and directories reside
- `volume` – virtual block device available to the operating system
- `snapshot` – read-only copy of a filesystem or a volume
- `clone` – filesystem with initial contents of a snapshot

What are the limits of ZFS?

Maximum pool size: 256 quadrillion zettabytes
Maximum filesystem/file/attribute size: 16 exabytes
Maximum pools/filesystems/snapshots: 2^64

Original ZFS features by design:

- pooled storage – filesystem and volume manager are integrated in one unit
- transactional semantics – the copy-on-write model ensures data consistency, there is no need for a fsck
- checksums and self-healing – all data blocks are checksummed; mirroring and variations of RAID-5 support automatic data correction
- scalability – ZFS is a 128-bit filesystem limited to 256 quadrillion zettabytes of storage
- instant snapshots and clones – old data can remain accessible in snapshots
- dataset compression – data can be automatically compressed by the filesystem using various algorithms
- simplified delegable administration – a system of unified management tools and filesystem properties eases administration and management tasks can be delegated to users

new writes and this is done in a transactional way, so until a transaction is completely finished it can be *rolled back* at any moment. This ensures constant data consistency so there is no need for a `fsck` command.

Checksums and self-healing

ZFS stores checksums of all data blocks. This enables the filesystem to continously verify if data has been altered (e.g. damaged in any way). Several checksum algorithms are supported and can be configured via the *checksum* filesystem property (checksums can be disabled at all). For data safety it is possible to store several copies of all blocks on a filesystem by appropriately setting the *copies* filesystem property. If using copies or a mirror device (RAID-1), ZFS automatically repairs defective data from a healthy copy. This is of course reported to the user in the `zpool status` command (see Figure 2).

Scalability

As ZFS is a128-bit filesystem, it has very high limit values. 256 quadrillion zettabytes (256 x 10^36 bytes) of data can be stored on a ZFS pool, a system may manage up to

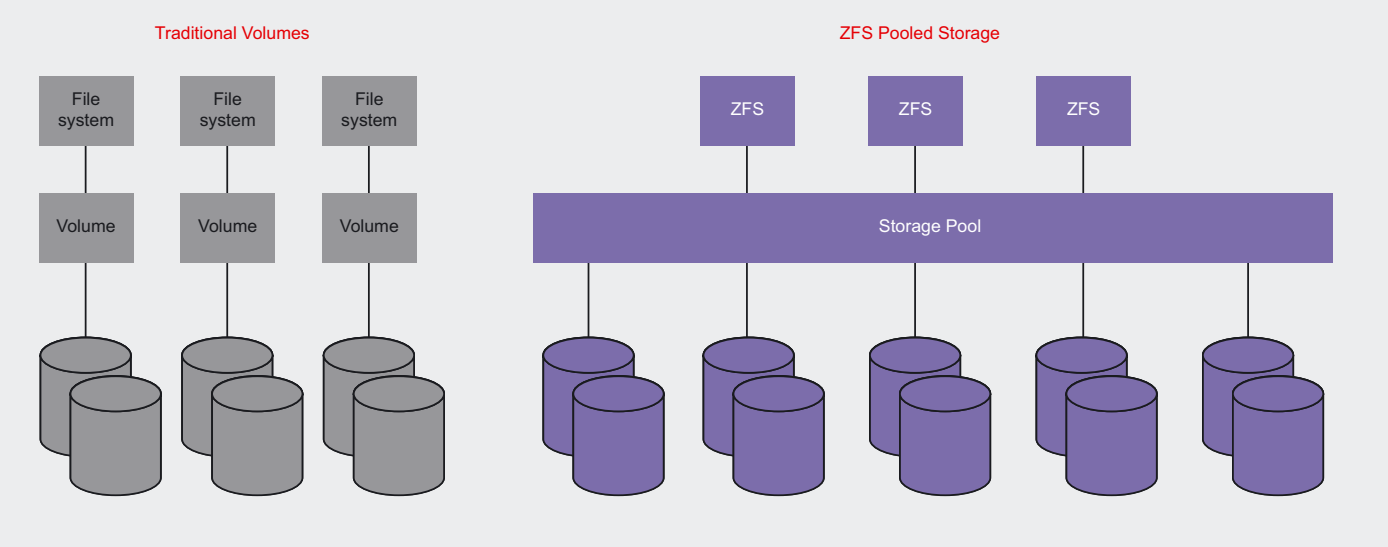


Figure 1. *Traditional storage model vs. ZFS pooled storage model*

2^64 pools, filesystems or snapshots and each filesystem, its attribute or a single file is limited with 16 exabytes of size.

Instant snapshots and clones

Due to the nature of the copy-on-write model, it is possible to instantly create read-only snapshots of filesystems and volumes. In other words, old data is not deleted and stays referenced and accessible to the user. This way if there is enough free space, past states of a filesystem can be saved and used as a backup source.

It is possible to create clones out of a snapshot – these are new writable filesystems with the initial contents of a snapshot (without the need to create a copy and saving space). The appropriate commands are *zfs snapshot* and *zfs clone*.

My personal favorite here is the possibility of streaming of ZFS snapshots (*zfs send*) and receiving these on other systems (*zfs receive*). This can be done incrementally (just sending a difference – between two or more snapshots) which enables consistent backups done in the background and also a kind of *cold-standby* on the filesystem level. This feature is very useful for systems requiring consistent backups and high availability at the same time.

Dataset compression

To save space, filesystems and volumes can be compressed with the lzjb compression algorithm invented by Jeff Bonwick. This algorithm has very low CPU overhead but is less effective than the gzip algorithm, which can be used since pool version 5. The compression applies only to newly written data, so this feature can be turned off and on at any time (old data remains untouched). Corresponding dataset properties are compression and compressratio.

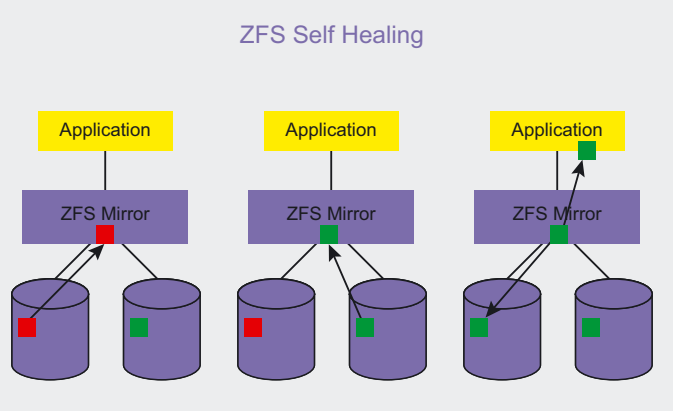


Figure 2. *ZFS self-healing example on a RAID-1 mirror*

Simplified delegable administration

Almost all ZFS administration is done with the subcommands the two main utilities *zpool* and *zfs*. Each pool and dataset conatin a set of properties that can be accessed with the *zpool get* and *zfs get* subcommands. Some of the properties are read-only and contain valuable information about the pool or filesystem (e.g. used space, compression ratio, creation date). It is posible to assign various administration rights to individual users with the *zfs allow* and *zfs unallow* subcommands.

Read and write caching

One of the main highlights of ZFS is its caching system. The main read cache in system's memory is called ARC (*Adaptive Replacement Cache*). It provides an algorithm effectively combining LRU (Last Recently Used) and LFU (Last Frequently Used) lists and keeps track of recently evicted entries. On top of this cache, since pool version 10 Level 2 cache on external devices is possible (L2ARC). This cache is recommended for read-intensive systems and should be placed on fast SSD drives. To speed up writes on write-intensive systems it is possible to dedicate external devices for the ZFS Intent Log (ZIL). Again, fast SSD SLC drives are recommended (see Figure 3).

ZFS Versioning

Some new features introduced in the ongoing the ZFS development require structural changes to the on-disk data structures. Every time such an incompatible structural change happens, the version number is increased. There are two version numbers in ZFS, the *pool* version and the *filesystem* version. The pool version is the major of these two and is usually referenced as the *ZFS version* itself. We are also distinguishing the *system* and the *on-disk* version. The first one is loaded in memory and represents the maximum supported on-disk version. ZFS provides backwards compatibility – this means a system can read, operate and upgrade all pools up to its highest supported version number. Forward compatibility is not provided, so newer pools

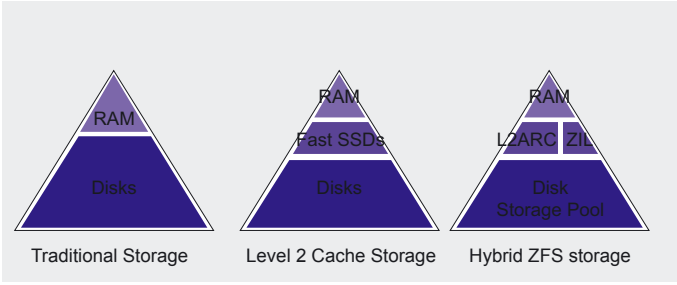


Figure 3. *ZFS hybrid storage concept*

are not readable on older systems. Version downgrade is not possible, too, so if upgrading a pool, make sure all systems working with this pool (for a bootable pool including the boot loader on the own system) support its version number.

New features

In the ongoing development of ZFS several new features have been added. Some of the features that required bumping the pool version number are double (pool v3) and triple (v17) parity RAID-Z (similar to RAID-5/6) with support for hot spares (v3), separate log devices (v7), user and group quotas (v15), *holdable* snapshots (v18), deduplication (v21) and encryption (v30, closed-source, only in Solaris yet). Other interesting new features without the need for the increase of the version number are device autoexpansion (post-v16), ZFS pool recovery (post-v19), deduplication of zfs send streams (post-v21), splitting mirror devices (RAID-1) into separate pools (post-v22) and displaying diffs between snapshots (post-v28).

Operating Systems

ZFS is currently available on OpenSolaris and related distributions (OpenIndiana, Belenix, SchilliX), Nexenta Core (OpenSolaris with Debian packages), FreeBSD, NetBSD (development), MacOS X (provided by a third party), Linux (via user-space fuse or recently as a native module by the company KQ Infotech from India, based on Brian Behlendorf's porting work) and of course on the commercial operating system Oracle Solaris. The Debian distribution using the FreeBSD kernel (Debian GNU/kFreeBSD) has native ZFS support, too.

License

The ZFS source code is distributed under the OSI approved *Common Development and Distribution License* (CDDL) which is based on the *Mozilla Public License* (MPL). It protects the licensee from patent claims by the author and contributors and includes other specific provisions and is to the disadvantage of Linux developers incompatible to GPL.

Open Source Distributions with ZFS support

- FreeBSD – *http://www.freebsd.org*
- OpenSolaris – *http://www.opensolaris.org/os/community/zfs/*
- Nexenta Core – *http://www.nexenta.org*
- Linux – *http://zfsonlinux.org/, http://zfs-fuse.net/, http://kqstor.com/*
- Debian GNU/kFreeBSD – *http://www.debian.org/ports/kfreebsd-gnu/*

On the 'Net

- *http://wiki.freebsd.org/ZFS* – FreeBSD ZFS wiki with many useful links and resources
- *http://blog.vx.sk* – author's blog with focus on ZFS and FreeBSD
- *http://mfsbsd.vx.sk* – memory-based FreeBSD image creation kit including bootable ISO images with automated ZFS install

ZFS on FreeBSD

ZFS is available as a FreeBSD kernel module since 7.0-RELEASE in February 2008 (pool version 6).

The current state can be classified as production and the upcoming 8.2-RELEASE will contain pool version 15. Pawel Jakub Dawidek, the head of ZFS porting at FreeBSD and some other developers including me have been working on the pool version 28 and released a public testing patch against 9-CURRENT and 8-STABLE in the *freebsd-fs@FreeBSD.org* mailing list. There are plans to import pool version 28 into 9-CURRENT after 8.2 is released. FreeBSD includes a ZFS boot loader capable of booting from single, mirror and raidz pools, so its easily possible to build a ZFS-only system. I recommend using the mfsBSD toolset (one of my projects, Links section) for simplified creation of bootable ZFS-only systems and for system recovery.

I have been using ZFS on several operating systems and as to my experience, for optimal operation tuning is required (on FreeBSD via the sysctl command). For a good overview of ZFS memory structures – ARC (*Adaptive Replacement Cache*), Level 2 ARC, DMU (*Data Management Unit*) prefetch and vdev cache I have provided a very practical statistics tool called *zfs-stats* which is available in the FreeBSD ports collection (sysutils/zfs-stats).

For novice users I recommend reading the freely downloadable *Oracle Solaris ZFS Administration Guide* and for advanced users reading the *ZFS Best Practices Guide* at *solarisinternals.com*. Only few sections in these documents are not relevant to FreeBSD. Other useful information about ZFS and FreeBSD is available on FreeBSD ZFS wiki and on my blog site (Links section).

MARTIN MATUŠKA

Martin Matuška (mm@FreeBSD.org) is a systems administrator and programmer. He has been an active FreeBSD developer since 2007, is part of the FreeBSD ZFS team and maintains several FreeBSD ports. He is running 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 the FreeBSD operating system. He writes at http://blog.vx.sk.

FreeNAS

FreeNAS is a very interesting project with a history spanning approximately 5 years. It's a fusion of FreeBSD with a webgui and embedded device framework, which creates a NAS device based on FreeBSD, fully manageable from a web-browser out of a PC with an x86 or AMD64 architecture.

The current production releases of FreeNAS are based on FreeBSD 7.x, with a PHP webgui and framework inherited from the popular m0n0wall project. FreeBSD 7 was the first branch of FreeBSD to include the popular ZFS filesystem from Sun Microsystems. Naturally, a feature like a new filesystem, especially one as rich in features as ZFS, is of great interest to a project with it's focus on storage. ZFS is an experimental filesystemin the FreeBSD 7 branch, although it quickly evolved from a highly unstable feature in FreeBSD 7.0 to a feature widely used in production in later releases, many of the features and innovations in ZFS were relegated to later branches such as FreeBSD 8.

Naturally, there was quite a bit of interest in upgrading FreeNAS to use FreeBSD 8. There was also several other issues in FreeNAS that were preventing it from making changes neccesary to meet the needs of the user community. The sum total of the issue was with all of the factors considered, FreeNAS really needed a complete overhaul in order to move forward.

By this time, the founder of the FreeNAS project, who was enthusiastic about FreeBSD, had moved on to a stage in his life that precluded spending much time on FreeNAS. The active contributors to the FreeNAS project were far more familiar with linux than FreeBSD, and they were of the mindset that if there was going to be a complete rewrite then one of the things that would be changed would be to switch the base platform from FreeBSD to linux.

In early 2010 an arrangement was reached that was satisfactory to all of the parties involved. FreeNAS would remain based on FreeBSD, and the current FreeNAS developers started a new project called openmediavault which would be based on linux.

The changes in FreeNAS have been widespread. The base system was switched to use FreeBSD 8. The ZFS implementation in FreeBSD 8 is based on version 15 and has had numerous performance and stability modifications. We are very excited about this technology, coupling it with a storage appliance is a natural fit. There are a number of observers of the storage industry and marketplace who feel that the viability of ZFS depends on how it fairs in FreeBSD. In a lot of ways FreeBSD has lagged behind in the area of filesystems for many years, while other systems enjoyed XFS, WAFL, and other technologies, FreeBSD used UFS2 with Softupdates. While this is a very stable filesystem, it lacked a number of features that more modern filesystems brought to the table. With ZFS FreeBSD is suddenly thrust to the forefront, with a filesystem that is cutting edge technology. ZFS is both a volume manager and a filesystem, designed from the ground up with a focus on data integrity. As storage sizes have increased, data corruption has turned from a highly improbable occurance to more of a "when not if" type of problem. ZFS maintains continual consistancy, thus eliminating the need for fsck type tools to repair filesystem inconsistancy after an unclean shutdown.

This is particularly useful in FreeBSD due to it's lack of a journelled mainstream filesystem. With modern disk drives it's getting very trivial to build disk arrays with sizes that takes hours to fsck on UFS2, and even midrange sized storage of several dozen terabytes simply isn't viable on UFS2. Because ZFS uses a copy on write transactional model, snapshots are very cheap to implement and use. Snapshotting a multi-terabyte ZFS filesystem is an operation that occurs nearly instantaniously. Many of the backup and replication strategies with ZFS are centered around sending and receiving snapshots and differences between snapshots to remote systems. Users of shared storage have come to love filesystems such as WAFL that implement snapshotting for managing their home directories. A risky change or delete operation can be easily reverted with virtually no system overhead. ZFS implements a varient of RAID 5 which is called RAIDZ. An issue in traditional RAID 5 or 6 arrays called "The RAID 5 write hole" doesn't exist in the ZFS RAIDZ implementation. RAIDz2 was introduced as a rough equivilent to RAID6, and exists in FreeBSD currently. ZFS has forged ahead with RAIDz3, which isn't currently available in FreeBSD, but will offer a level of redundancy not available in proprietary hardware RAID solutions.

The m0n0wall framework was abandoned. NanoBSD was brought in as the replacement build system and embedded framework.

The GUI has been rewritten as a django application, using modern web technologies like AJAX and dojo. FreeNAS traditionally used an XML file to store it's configuration. This was switched to use a sqlite database. Interaction between the GUI and the underlying FreeBSD system takes place through a middleware layer we call The Notifier, designed from the ground up with an eye towards abstracting system interactions from the GUI, but also with an eye towards being used as an abstraction layer that would expose the same API that the GUI uses internally to the command line.

A technology that we are somewhat excited about is the addition of what we are calling "The Treemenu". It was available as a technology preview in the last snapshot we released, and we are hard at work as of this writing getting it ready to turn on as the default web GUI interface. In many ways it takes the web GUI to the next level. It gives FreeNAS a level in technology unmatched by any NAS device on the market, at any price, for free.

One of the aspects of the old system that was particularly frustrating to the community was how static and hard wired the system was. It is relatively easy to add software to the base FreeBSD system in a FreeNAS 0.7x installation, but quite a bit trickier to change the GUI or backend to deal with that software. During the rewrite we've kept an eye on keeping the system modular and dynamic. While it's not documented at all yet, adding functionality to the GUI can be accomplished by adding django applications. We intend to release a spec and build system based on PC-BSD's PBI concept which will allow creation of FreeNAS specific binary packages that include the base system software as well as the GUI components to configure it.

Like any project, the results are tied to the people behind the work. Xin Li has done much of the python coding, as well as the design and implemetation of the notifier. James Nixon is responsible for much of the GUI design, as well as it's look and feel. Warner Losh converted FreeNAS to use NanoBSD, and did the majority of the work on the build system and early integration, as well as provided much guidance and direction for the project. Doug White has been involved in performance tsting, and his experience with many storage devices on the market have provided guidance for what should be expected behavior. John Hixson provided much of the installer, the CLI environment that you see when the system boots, as well as implementing the LDAP and AD integration. Matt Olander has also provided considerable input into the GUI, as well as doing a lot of the behind the scenes work to make this a viable project.

While we are by any measurement, not finished with FreeNAS yet, the current state of affairs is that the system is very usable as a new deployment for a storage device. We attempt to release snapshots fairly regularly, and the upgrade path for those snapshots is paid close attention. We encourage anyone who is interested to try out the latest beta snapshot. Let us know what you think, create a ticket if you find a bug, a missing feature, or something doesn't work quite the way you'd expect.

The download link is:
https://sourceforge.net/projects/freenas/files/FreeNAS-8-beta/

The support portal is:
http://support.freenas.org

JOSH PAETZEL

Josh Paetzel – A 37 year old advocate, user and developer of BSD UNIX based systems. he resides in Minneapolis, Minnesota, USA where he hacks on FreeBSD and PC-BSD, both as a volunteer and as part of his full time work as the Director of IT at iXsystems.

Bandwidth Control

with a transparent bridge

In this article I will explain how to setup a transparent bridge between your LAN and your Firewall/router. With "transparent" I mean that you won't need to do any change on your network in order to use it.

What you will learn…

- How to setup a transparent bridge between your LAN and your firewall.
- How to enable and use ipfw for rate limitation

What you should know…

- Basic networking concepts
- Basic FreeBSD administration

Even more, if at some point you want to plug it off your network just unplugging it and *tying* the cables from it will get you back to your original setup. See the Figure 1 and Figure 2.

As you can see in the figures above, unplugging the right cable from the bridge and plugging it to the gateway will remove it from the network in case you want to do some maintenance or debug any network issue.

Part 1 – Setting up the bridge

Let's suppose the two free interfaces on your server are em0 and em1. The idea is that first we will connect them together to act as a bridge and later apply some ipfw rules that will limit the flow per IP address to any value you decide.

On the bridge, run the following commands as root:

```
# ifconfig bridge1
# ifconfig bridge1 addm em0 addm em1
# ifconfing bridge1 up
```

At this point, if you connect the bridge as described on the introduction, you should be able to reach your gateway as usual, but now your traffic is going through our bridge.

If everything works, don't forget to add this to `/etc/rc.conf` so that when your system is restarted you have the bridge up again:

```
cloned_interfaces="bridge1"
ifconfig_bridge1="addm em0 addm em1 up"
```

Figure 1. *Before connecting our bridge*

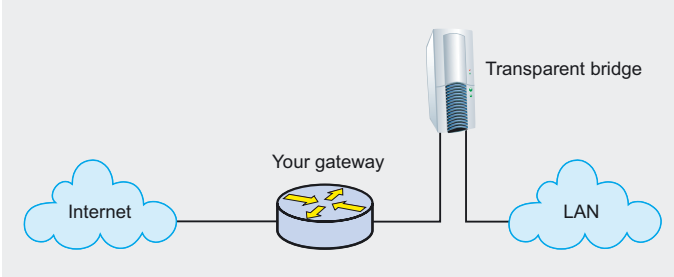


Figure 2. *After connection our bridge*

Hardware required

- FreeBSD server (latest stable release). Hardware specs depends on the traffic you pretend to manage. Anything better than a 1Ghz Pentium IV will do for most mid sized company.
- Two unused network cards on that server
- Two ethernet patch cords.

Part 2 – Setting up the rate limitation

Now, we will enable ipfw and setup a couple basic rules that will rate limit upload and download bandwidth (separately) by source (your LAN computers') IP address.

Add the following to `/etc/rc.conf`

```
firewall_enable="YES"
firewall_logging="YES"
firewall_script="/etc/ipfw.rules"
```

Run the following commands as root: see Listing 1.

You are encouraged to play with the values of the first and last sysctls, as I found these are the most appropriate for my network.

Also, don't forget to add them to `/etc/sysctl.conf` to have them configured after a system reboot.

Now, create `/etc/ipfw.rules` if it doesn't exist yet and add the following lines (I assume here that your LAN network is one of the designed by the RFC1918 see Listing 2).

This will limit any upload to 150KByte/s but after you send more than 500KBytes. This will give more performance for users browsing a web site for example, as they won't be rate limited but will instead penalize the download of a large file. Same happens with download limit, but this time it is 300KByte/s.

Now, start the *firewall* and try it by downloading any file from the internet on one of your LAN computers:

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

Listing 1. *Setting the sysctls required*

```
# This will improve the performance for large networks
sysctl -w net.inet.ip.dummynet.hash_size=512
# This will enable ipfw filtering on bridge interfaces
sysctl -w net.link.bridge.ipfw=1
#This will limit the traffic that will filter/rate control our bridge to IP traffic
sysctl -w net.link.bridge.pfil_onlyip=1
#This allows "fast mode operation" on dummynet, for more info read dummynet docs.
sysctl -w net.inet.ip.dummynet.io_fast=1
# This will improve performance for large networks
sysctl -w net.inet.ip.dummynet.dyn_buckets=1024
```

Listing 2. */etc/ipfw.rules*

```
IPF="ipfw -q"
ipfw -q -f flush
ipfw -q pipe flush
LAN="10.0.0.0/8,192.168.0.0/16,172.0.0.0/12"
INTERNET="not 10.0.0.0/8,127.0.0.0/8,192.168.0.0/16"
UPLOAD_LIMIT="150KByte/s"
UPLOAD_BURST="500KByte"
DOWNLOAD_LIMIT="300KByte/s"
DOWNLOAD_BURST="500KByte"

$IPF add pipe 1 all from ${LAN} to any
$IPF add pipe 2 all from any to ${LAN}
$IPF pipe 1 config mask src-ip 0xff  bw ${UPLOAD_LIMIT} burst ${UPLOAD_BURST}
$IPF pipe 2 config mask dst-ip 0xff  bw ${DOWNLOAD_LIMIT} burst ${DOWNLOAD_BURST}


$IPF add 10000 allow all from any to any
```

Listing 3. *Complex /etc/rc.conf*

```
firewall_enable="YES"
firewall_logging="YES"
firewall_script="/etc/ipfw.rules"
ejabberd_enable="YES"
pflog_enable="YES"
ifconfig_igb0="up"
ifconfig_igb1="up"
cloned_interfaces="vlan10 vlan910 vlan12 vlan912 vlan20
vlan920 vlan22 vlan922 vlan24 vlan924 vlan30 vlan930
vlan32 vlan932 vlan60 vlan62 vlan960 vlan962 bridge10
bridge12 bridge20 bridge22 bridge24 bridge30 bridge32
                    bridge60 bridge62"
ifconfig_vlan10="vlan 10 vlandev igb0"
ifconfig_vlan910="vlan 10 vlandev igb1"

ifconfig_vlan912="vlan 12 vlandev igb1"

ifconfig_vlan20="vlan 20 vlandev igb0"
ifconfig_vlan920="vlan 20 vlandev igb1"

ifconfig_vlan22="vlan 22 vlandev igb0"
ifconfig_vlan922="vlan 22 vlandev igb1"

ifconfig_vlan24="vlan 24 vlandev igb0"
ifconfig_vlan924="vlan 24 vlandev igb1"

ifconfig_vlan30="vlan 30 vlandev igb0"
ifconfig_vlan930="vlan 30 vlandev igb1"

ifconfig_vlan32="vlan 32 vlandev igb0"
ifconfig_vlan932="vlan 32 vlandev igb1"

ifconfig_vlan60="vlan 60 vlandev igb0"
ifconfig_vlan960="vlan 60 vlandev igb1"

ifconfig_vlan62="vlan 62 vlandev igb0"
ifconfig_vlan962="vlan 62 vlandev igb1"

ifconfig_bridge10="addm vlan10 addm vlan910 up"
ifconfig_bridge12="addm vlan12 addm vlan912 up"
ifconfig_bridge20="addm vlan20 addm vlan920 up"
ifconfig_bridge22="addm vlan22 addm vlan922 up"
ifconfig_bridge24="addm vlan24 addm vlan924 up"
ifconfig_bridge30="addm vlan30 addm vlan930 up"
ifconfig_bridge32="addm vlan32 addm vlan932 up"
ifconfig_bridge60="addm vlan60 addm vlan960 up"
ifconfig_bridge62="addm vlan62 addm vlan962 up"
```

Part 3 – Adding some complexity

Nowadays, there are not so many simple networks like the one I described here. Most companies have their divided into VLANs so that there is a VLAN for engineering, other for sales, other for IT servers, and so on.In fact, I have that scenario at work. My solution was to plug the bridge in a vlan Trunk and setup each vlan two times, one on each bridge network interface, then creating a bridge between both vlan interfaces and repeating this process for each VLAN I have. The final rc.conf will be something like this: see Listing 3.

Part 4 – Adding exceptions

Now, it will be very easy to add exceptions to the ipfw rules. For example, to not rate limit your IT server farm, the big boss laptop by its MAC address or to apply a 1kbyte/s limit that user you know is using some p2p software see Listing 4. etc...

Not just for internet access

Actually, I use this system not just for upload bandwidth but also to rate limit the traffic on our Wireless network to avoid (or at least mitigate) network congestion. In this scenario, the *Upload* bandwidth will be the traffic you send to wireless clients from your router and the *Download* bandwidth will be traffic your Wireless clients send to your router. It all depends on the exact setup you have on your network, but this is quite flexible to adapt to most scenarios.

MATIAS SURDI

Matias Surdi has been working as a unix systems administrator since his first job, currently working as Senior IT Engineer of a large Spanish Social Network is a passionate of unix based server operative systems, networks management and systems tools development.

matiassurdi@gmail.com

FreeBSD Mall

Your FreeBSD & PC-BSD Resource

www.FreeBSDMall.com

FreeBSD 8.1 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 8.1 CD $39.95
FreeBSD 8.1 DVD $39.95
FreeBSD 7.3 CDROM $39.95
FreeBSD 7.3 DVD $39.95

FreeBSD Subscriptions

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

FreeBSD Subscription , start with CD 8.1 ..... $29.95
FreeBSD Subscription, start with DVD 8.1 ..... $29.95
FreeBSD Subscription, CD 7.3 ..... $29.95
FreeBSD Subscription, DVD 7.3 ..... $29.95

PC-BSD 8 DVD (Hubble Edition)

PC-BSD 8 DVD $29.95
PC-BSD Subscription $19.95

BSD Magazine

BSD Magazine $11.99
BSD Magazine Subscription $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 8.1 ..... $79.95

The FreeBSD Bundle

Inside the Bundle, you'll find:

- FreeBSD Handbook, 3rd Edition, Users Guide
- FreeBSD Handbook, 3rd Edition, Admin Guide
- FreeBSD 8.1 4-disc set
- FreeBSD Toolkit DVD

☆ Special: The FreeBSD CD Bundle $89.95
☆ Special: The FreeBSD DVD Bundle $89.95

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!

Building

an iSCSI storage with BSD

Highly loaded databases need a fast and reliable storage solution, something like a big server with many hard drives, probably with 4, 8, or 16 drives. Also, many 1U servers do not have the necessary storage capacity to offer services that need it.

What you will learn…

- iSCSI basics
- How to run an iSCSI server under NetBSD.
- Benefits of iSCSI over other protocols like NFS. (There is an example of a real life issue with NFS and iSCSI.)

What you should know…

- You should know how to install and configure a NetBSD server.
- Basic networking.
- There are examples of that how to connect to the iSCSI target via a Linux system. So, you should know what Fedora and OpenSUSE are.

In these cases additional servers can be used to take the huge I/O load. These servers usually have as many drives is needed and they are suitable for an extra storage attached to the main server. There is a special protocol dedicated to attaching remote block devices as they are local. This protocol is called iSCSI. iSCSI is an abbreviation of Internet Small Computer System Interface. iSCSI can be used to transmit data over network, or the Internet and can enable location-independent data storage and retrieval. This protocol allows clients to send SCSI commands to SCSI storage devices on remote servers. Unlike traditional Fibre Channel, which requires special-purpose cabling, iSCSI can be run over long distances using existing network infrastructure. It is more effective as the network is faster. It is not suitable for a busy storage to be cabled with only 100/100MBit network. The I/O performance of the server is limited to the network throughput, so as more it is as more productive will be the server.

iSCSI Benefits

The potential benefits of iSCSI are many. Some of them are:

- iSCSI gives you the block-level access and storage I/O intelligence of SCSI. It transforms your backing unit (file or storage devices) from server-attached to network-attached. Network-attached devices can be managed, partitioned, and allocated to servers and applications.
- You get great reliability and availability. There is no need to take down a server to add storage to an iSCSI SAN. Also when it is once added, that storage can be made available to any SAN-attached server.
- You can use a single WAN or LAN connection to transport both file and block-based data. For high reliability it is better if the connection is backed up by another one.

The choice of an operating system – NetBSD.

There is one important point when we build such a storage server. What type of operating system to use? We need a stable and secure one, and we also need mature and good support of iSCSI. We have to use one of the BSD distributions. There is also another additional point when we build such a server. We need also good performance, we do not need the greatest of the greatest in the security area (the server is supposed to be a storage not a firewall) but we need good performance and high scalability. In order to answer the requirement of high I/O bandwidth and keep sufficient response time, the operating system should be fast, reliable and scalable in high network and disk load. But also the OS should survive when multiple threads work in parallel on files in the storage. I chose NetBSD primarily because it has good iSCSI support and

is also stable and mature operating system. FreeBSD was also a good choice but a big plus for NetBSD was the NetBSD iSCSI target project that is available. So, one can run an iSCSI server in NetBSD for about 10 minutes or so. Of course, a qualified professional can probably run iSCSI everywhere for about the same time.

A real example

Some time ago, in my practice as an administrator, I had to implement a storage server that holds huge amount of data and shares that data to another server. The usual solution was to do it via NFS but I found that it would be very problematic in my case. I need that storage server for a place to put mail, files and home folders. The server that accepts clients actually was the other server. So, that storage was attached to the main server via iSCSI. I decided to use iSCSI because of some issues with NFS. I had 2 options: 1.) To use the storage via NFS, and 2.) to use the storage via iSCSI.

At the first option, I experienced some problems with postfix and samba services. Generally speaking samba was fine with this option but in some rare cases, when shared excel files was used, a strange locking of the used file occurs. Also there were some issues with postfix. The troubles above were the main reason for my switching to iSCSI. Compared to the NFS, iSCSI was a little bit harder to configure but it gave me huge opportunities for performance, stability and easy of use. Once attached to the server, the iSCSI LUN can be mounted as a usual block device. The actual difference is that it is a network block device instead of a local one. All the issues with file locking and freezing are eliminated at once in this case.

What is an Initiator / Client?

In the iSCSI world, the client is called an initiator. An initiator typically serves the same purpose to a computer as a SCSI bus adapter would. The difference is that instead of physically cabling SCSI devices, it is connected to the target via a network. So, iSCSI initiator sends SCSI commands over the IP network.

What is a Target / Server?

ISCSI target is an instance of a storage resource located on the iSCSI server. So, the target is visible to the clients and then can connect to it. After that the storage of the target is accessible by the clients. In our example below, we define a target with name target0 and that target has its backing store in the `/iscsi/target0-lun0.iscsi`. As we can see from the Figure 4, the target name is seen in the client side with its full unique name. In our example it is `iqn.1994-04.org.netbsd.iscsi-target:target0`. After logging in to this target, we can use the storage from the client side (initiator). If we define other 2 targets with the names target1 and target2, we will see them in the client side as well.

Preparation and configuration of the iSCSI target on NetBSD

In order to set up the target server we have to make some preparations in the `/etc/rc.conf` file.

Listing 1. *Configuration of the iSCSI target. It is put in /etc/iscsi/targets*

```
# extent        file or device       start                length
extent0      /iscsi/target0-lun0.iscsi  0                  1000MB

# target        flags                storage              netmask
target0      rw              extent0      0.0.0.0/0
```

Listing 2. *Starting the iSCSI target service on the NetBSD server*

```
# /etc/rc.d/iscsi_target start
Starting iscsi_target.
Reading configuration from '/etc/iscsi/targets'
target0:rw:0.0.0.0/0
        extent0:/iscsi/target0-lun0.iscsi:0:1048576000
DISK: 1 logical unit (2048000 blocks, 512 bytes/block), type iscsi fs
DISK: LUN 0: 1000 MB disk storage for "target0"
TARGET: TargetName is iqn.1994-04.org.netbsd.iscsi-target
```

Listing 3. *Connecting from a Linux client to the server*

```
[root@fedora-0 ~]# iscsiadm -m discovery --portal 192.168.122.3 -t
                   sendtargets
192.168.122.3:3260,1 iqn.1994-04.org.netbsd.iscsi-target:target0

[root@fedora-0 ~]# iscsiadm -m discovery --portal 192.168.122.3 --
                   login
192.168.122.3:3260,1 iqn.1994-04.org.netbsd.iscsi-target:target0
Logging in to [iface: default, target: iqn.1994-04.org.netbsd.iscsi-
                   target:target0, portal: 192.168.122.3,3260]
Login to [iface: default, target: iqn.1994-04.org.netbsd.iscsi-target:
                   target0, portal: 192.168.122.3,3260] successful.
```

Open the file and add the following line:.

```
iscsi_target=YES
```

Then we define the basic characteristics of the target – file or block device that will be exported to the clients, the size of this file/block device, permissions, name, netmask. All of these options are specified in the file `/etc/iscsi/targets`.

So we must first edit `/etc/iscsi/targets`, so that it contains the following lines: see Listing 1.

The file `/iscsi/target0-lun0.iscsi` is a backing store and it is persistent. So, it will be available automatically after reboot. In our example, the backing store is a small one, it is just 1000MB. If you need to skip over MBR or some disklabel, you can specify an offset. It is very useful in such cases.

The extent is mounted read-write by `target0`, and is served up to any host in the network (the 0.0.0.0/0 netmask).

Let's start the iSCSI target service. Issue the following command:

```
/etc/rc.d/iscsi_target start
```

and you should see this message: see Listing 2.

How to attach the NetBSD iSCSI target to a Linux server? Is that difficult? Actually no, it is very easy to access the target from everywhere on the network. There is no limitation to a particular operating system. Let's see a simple example of using this storage on a linux client under 2 different distributions.

OK. Now we have a running iSCSI server. In order to see the work done, we will try to access the target from other servers. Let's take for an example 2 client machines, that could be linux servers that need to use that storage. The iSCSI target is running NetBSD but it is useful to see the benefit of it. Of course, it can serve clients that are running different operating systems.

Listing 4. *We are connected to the server and we see our storage as a usual hard drive. This is the output of „fdisk -l" command in the Linux client*

```
Disk /dev/sda: 1048 MB, 1048576000 bytes
33 heads, 61 sectors/track, 1017 cylinders
Units = cylinders of 2013 * 512 = 1030656 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x00000000

Disk /dev/sda doesn't contain a valid partition table
[root@fedora-0 ~]#
```

Let's take for example 2 of the most popular open source distributions of the GNU/Linux operating system. That are OpenSUSE and Fedora. Our first example will show how to attach OpenSUSE server to the NetBSD iSCSI target.

How to use the storage under Fedora Linux?

In order to attach a fedora server to our NetBSD iSCSI target, we have to issue the following commands as root. The IP address of the iSCSI server is 192.168.122.3 and the port for connection is 3260 (see Listing 3).

Voilla. Let's see if we can access our storage. The following is output of fdisk -l in Fedora Linux: see Listing 4.

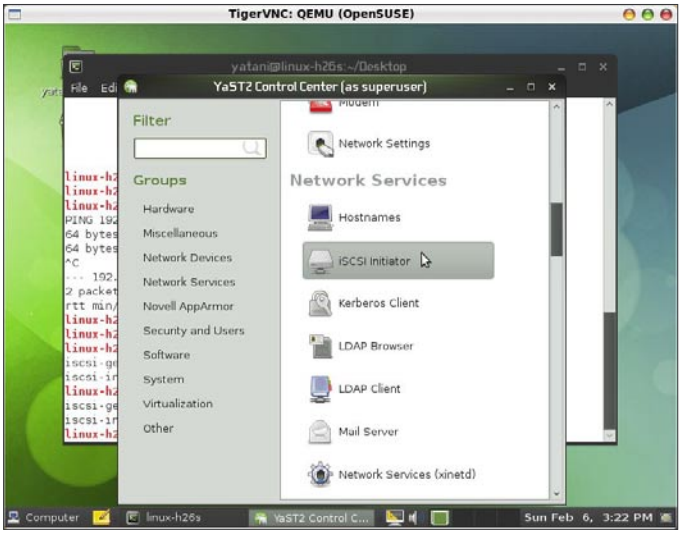


Figure 1. *iSCSI Initiator configuration via YaST*

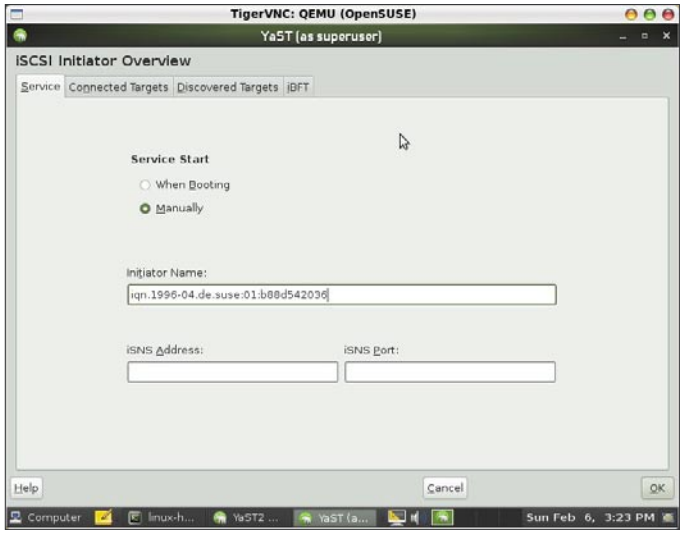


Figure 2. *Properties for our machine, the client side*

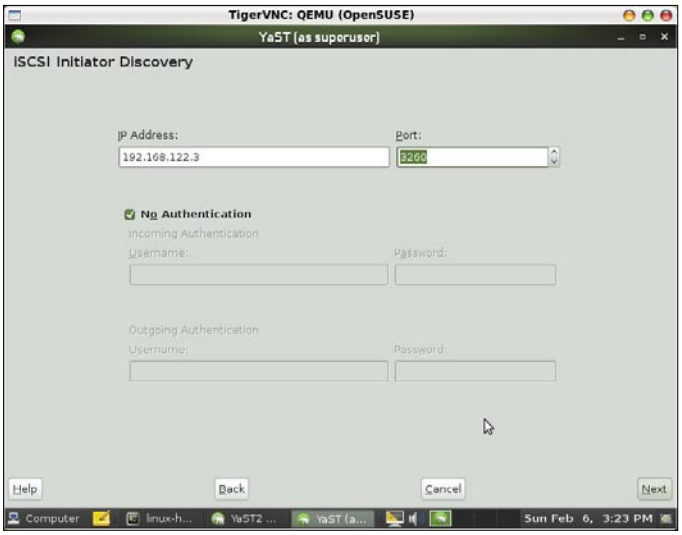


Figure 3. *We will ask the specified IP address to see if it is the appropriate iSCSI server*

How to use the storage under OpenSUSE Linux?

At first we have to start YaST (YaST is something like a control panel in OpenSUSE), then scroll to the *Network Services* section and find the option called *iSCSI Initiator*. This option will give us some configuration windows where we have the opportunity to specify the IP address, target name, and properties of our iSCSI server. Have a look at Figure 1, this is the main YaST window and the iSCSI section. After choosing it, a window with more connection specific options will appear. In Figure 2 we can see this window. There are 4 tabs, first of them is the *Service* tab. It is not that important, we can leave it with default values. The next 2 tabs are more important, they have the names *Connected Targets* and *Discovered Targets*. So, we specify an IP address and connection properties for particular target (Figure 3) then our client tries to connect to this target. After this procedure complete, we can see our target in the list of the connected targets in the Figure 4.

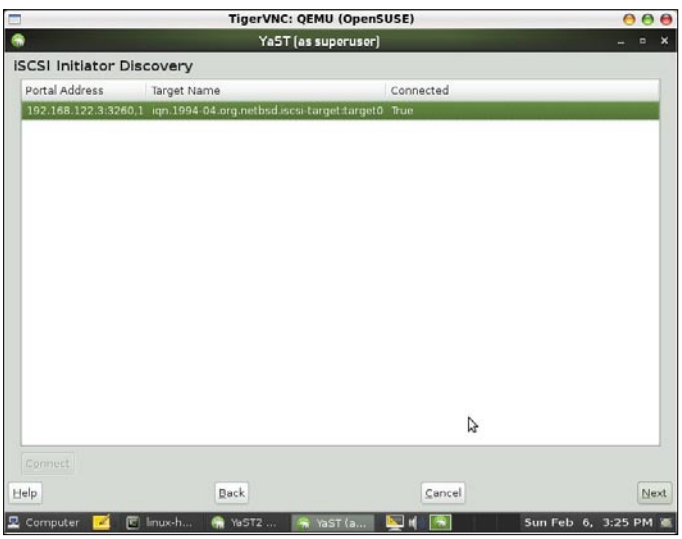


Figure 4. *OK. We found it, this is our iSCSI target server and we are connected to it*

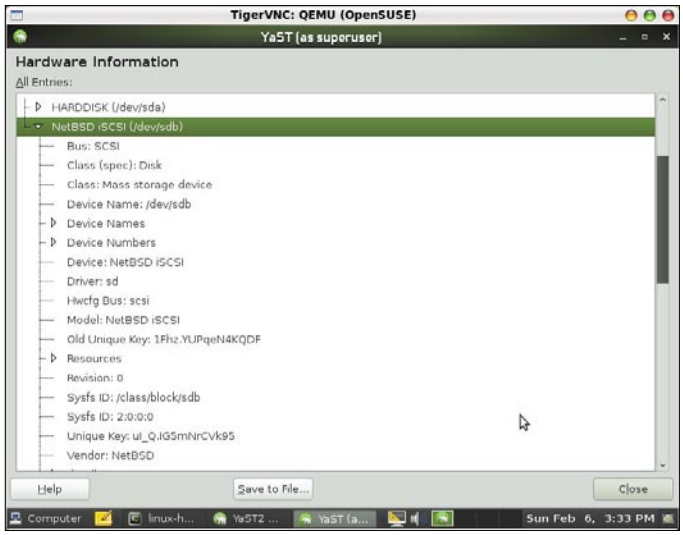


Figure 5. *Let's see the properties for the attached device. We have it as a usual block device under the name of sdX*

If the operation for attaching to the iSCSI server is successful we can see the iSCSI device as a block device in the list of devices. At Figure 5 we can see an example for this. As we can see, our device is connected and running and we can use it as we use other block devices under Linux. In our example, this device is listed with the name sdb.

Summary

iSCSI has its advantages and disadvantages over other protocols for linking data storage facilities. A particular storage server can offer its place to other servers in many ways, and everyone of them has its own benefits, problems and troubles. My personal issues with NFS took me the iSCSI variant because of that the client machine sees only a block device. This block device can be mounted in any directory on the system and can be formatted with any filesystem. Probably it is not the best choice but it worked for me.

SVETOSLAV CHUKOV

The author is a system administrator with experience in BSD and Linux. Some of the primary interests for him are: system security, firewalls, improving performance of the servers, filesystem optimizations, benchmarks, high availability and some others... He enjoys benchmarking huge storage servers, or if they aren't available, he also likes to play with „more simple" 2 nodes clusters.

How to setup a

USB Memory stick for installing a pfSense SoHo Firewall/ Router

This article covers the installation and initial configuration of a pfSense Firewall / Router on a small form factor PC.

What you will learn…

- How to install pfSense on a PC without a cdrom drive by using a bootable usb stick

What you should know…

- How to become root

What is pfSense and why would you want to use it?

pfSense is a free, open source customized distribution of FreeBSD tailored for use as a firewall and router. In addition to being a powerful, flexible firewalling and routing platform, it includes a long list of related features and a package system allowing further expandability without adding bloat and potential security vulnerabilities to the base distribution. 1. *http://pfsense.org/*

Feature List:

- OpenBSD's PF Firewall
- State Table
- Network Address Translation (NAT)
- Redundancy (OpenBSD's CARP)
- Load Balancing
- VPN (IPSec, OpenVPN, PPTP)
- PPPoE Server
- Reporting and Monitoring
- Real Time Information
- Dynamic DNS
- DHCP Server

Hardware choice

I should start by explaining my hardware choice, as there is already a market for tiny embedded firewall systems: Soekris Engineering, Hacom Embedded Systems and Appliances, etc.

The entry-level systems are often low power but also very low horsepower: 133Mhz – 500Mhz with 10/100 network interfaces.

I chose the Polywell ITX-525L2 for its small profile, performance, and expandability. It comes equipped with a 1.8Ghz Intel Atom D525 CPU and dual GigE network interfaces.

While most soho solutions have 10/100 Ethernet at the LAN and WAN ports, it was important for me to have GigE on both. I chose this design so that this firewall could be placed behind a separate WAN/DMZ device which would require full gigabit speeds between the protected clients and DMZ space.

Difference in managment interface

pfSense provides a headless installation image for use with embedded systems like Soekris appliances. These systems are lacking a SVGA port in favor of a DB9 Serial/ Console port. The install process is designed to take place over a serial cable using a terminal emulator such as MiniCom.

I wanted my solution to have the flexibility of having both SVGA and Console ports for management. -Hacom appliances do have SVGA, but come preinstalled with pfSense.

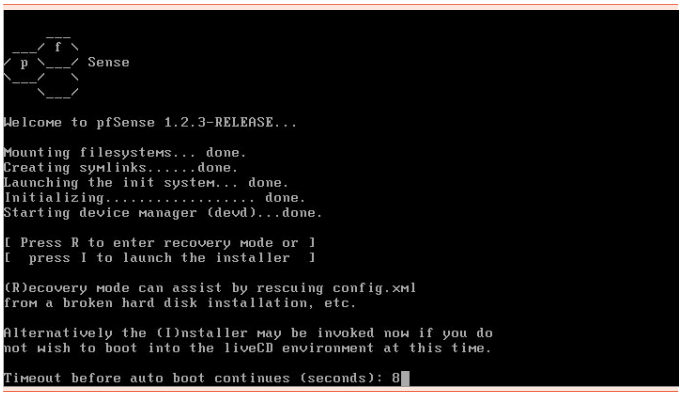
This article will detail how to modify the pfSense cdrom based install iso for use with a USB stick for installation

without the need of a console cable or terminal emulator.

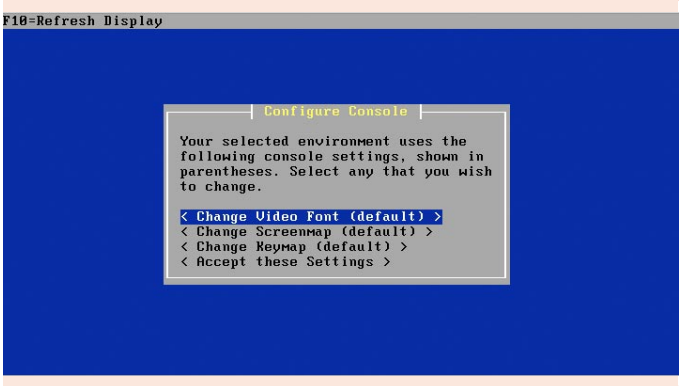
Setup installation medium from an existing FreeBSD Install:

1) Download pfSense ISO:
   a. *http://www.pfsense.org/mirror.php?section=downloads*
   b. I chose the Chicago Mirror:
      ftp *http://files.chi.pfsense.org/mirror/downloads/pfSense-1.2.3-RELEASE-LiveCD-Installer.iso.gz*
   c. Unzip the ISO image:
      `gunzip pfSense-1.2.3-RELEASE-LiveCD-Instal-ler.iso.gz`
   d. Mount the ISO:
      i. `mdconfig -a -t vnode -f pfSense-1.2.3-RELEASE-LiveCD-Installer.iso -u 0`
      ii. `mount -t /dev/md0 /mnt`
2) Prepare and format your USB stick:
   a. `fdisk -BI /dev/da0`
   b. `bsdlabel -B -w da0s1`
   c. `newfs -U -O1 /dev/da0s1a`
   d. `boot0cfg -v -B da0`
3) Mount USB stick:
   a. `mkdir /usb`
   b. `mount /dev/da0s1a /usb`
4) Copy the contents of the ISO over to the USB stick:
   a. `cp -r /mnt/.* /usb/`
5) Modify the /usb/etc/fstab on the USB stick to point to the USB Device:
   (Insert the following line and ensure it is the only entry)
   a. `/dev/da0s1a / ufs rw 0 0`
6) Unmount the USB stick, unplug it and plug it into your small form factor PC
7) Turn on the small form factor PC and take care to adjust your boot order to boot from the USB stick.

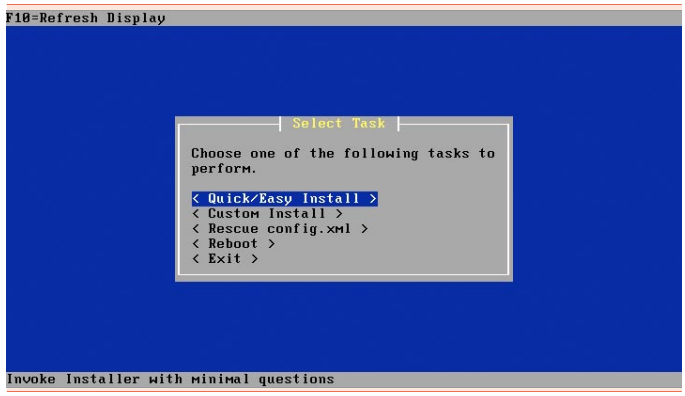
Installing pfSense from a USB Stick:



01 Press „I" at the selection screen to begin the install process

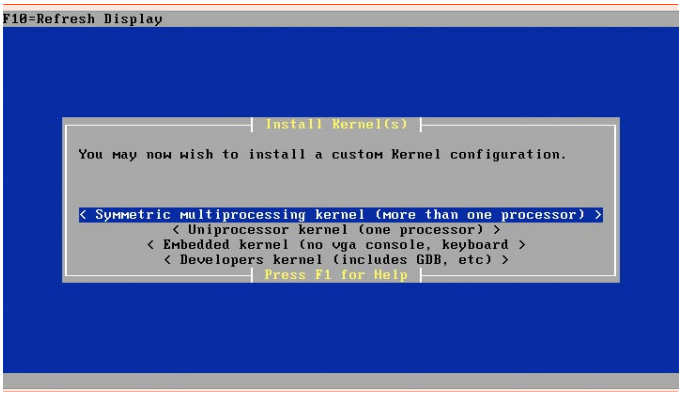


02 Confirm your console settings

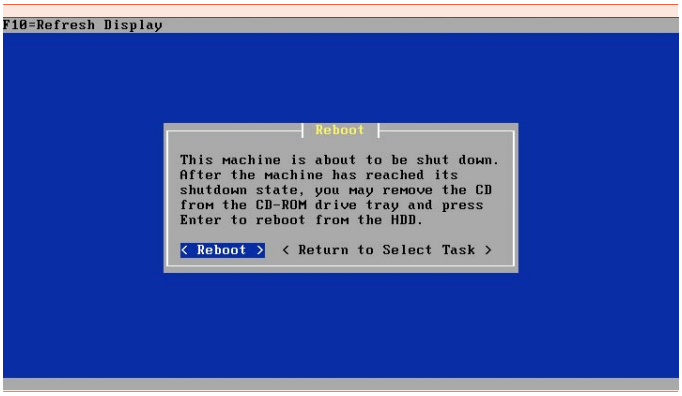


03 Since this is a dedicated device, select Quick/Easy Install

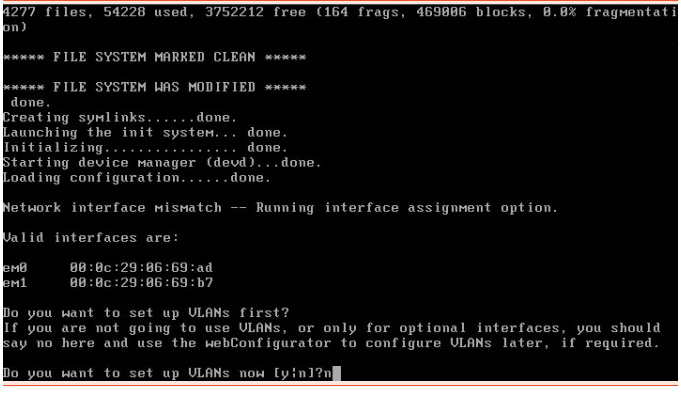
04 Confirm that you would like to dedicate the disk for pfSense



05 Select your kernel CPU configuration (Single Core/SMP)



06 Reboot (remember to unplug the USB Stick at this point)



07 Skip VLAN settings

08 Identify the LAN interface (If you aren't sure, you can disconnect all but the desired interface and choose „a" for auto-configuration.)

09 Identify the WAN interface (If you aren't sure, you can disconnect all but the desired interface and choose „a" for auto-configuration.)

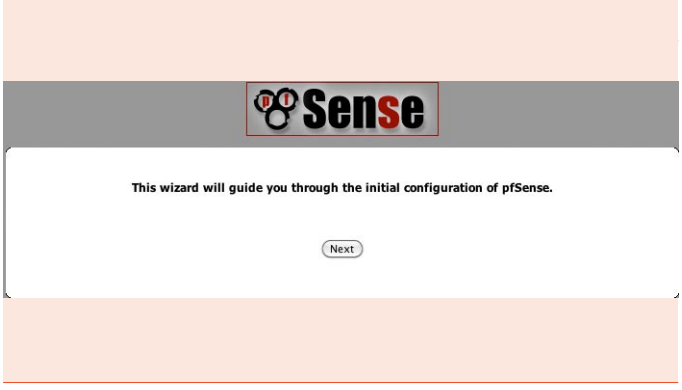
10 Skip past Optional interface

```
*NOTE*  pfSense requires *AT LEAST* 2 assigned interfaces to function.
        If you do not have two interfaces you CANNOT continue.

        If you do not have at least two *REAL* network interface cards
        or one interface with multiple VLANs then pfSense *WILL NOT*
        function correctly.

If you do not know the names of your interfaces, you may choose to use
auto-detection. In that case, disconnect all interfaces now before
hitting 'a' to initiate auto detection.

Enter the LAN interface name or 'a' for auto-detection: em0

Enter the WAN interface name or 'a' for auto-detection: em1

Enter the Optional 1 interface name or 'a' for auto-detection
(or nothing if finished):

The interfaces will be assigned as follows:

LAN  -> em0
WAN  -> em1

Do you want to proceed [y|n]?y
```

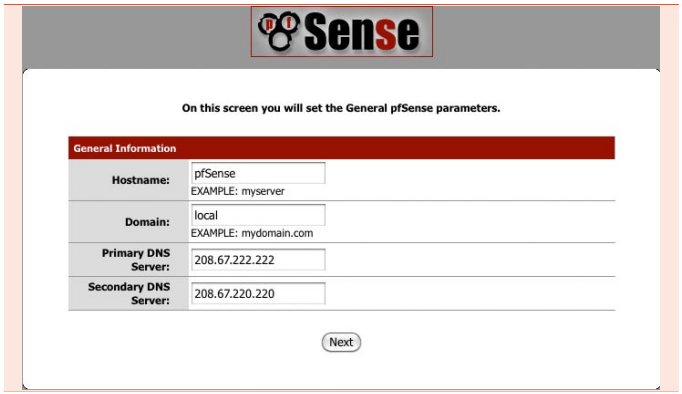
11 Confirm your settings and continue

```
*** Welcome to pfSense 1.2.3-RELEASE-pfSense on pfSense ***

 LAN*                  ->   em0     ->      192.168.1.1
 WAN*                  ->   em1     ->      10.22.53.92(DHCP)

 pfSense console setup
**************************
 0)  Logout (SSH only)
 1)  Assign Interfaces
 2)  Set LAN IP address
 3)  Reset webConfigurator password
 4)  Reset to factory defaults
 5)  Reboot system
 6)  Halt system
 7)  Ping host
 8)  Shell
 9)  PFtop
10)  Filter Logs
11)  Restart webConfigurator
12)  pfSense Developer Shell
13)  Upgrade from console
14)  Enable Secure Shell (sshd)

Enter an option:
```

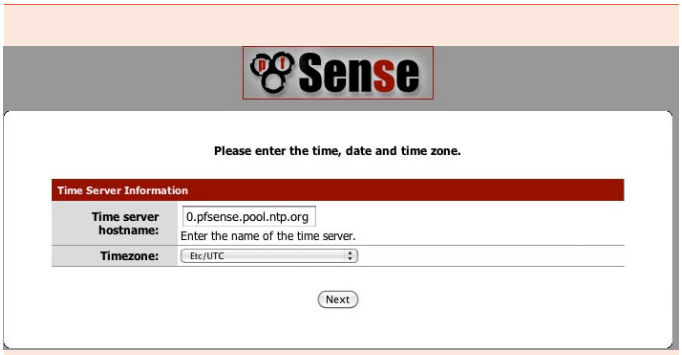
12 pfSense is now ready to administer



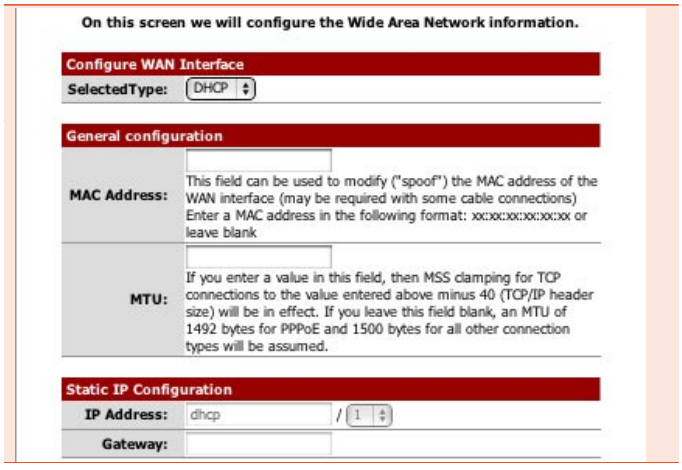
13 Make sure your worksation is plugged into the LAN side and go to: *http://192.168.1.1* to complete the configuration process using the WebConfigurator



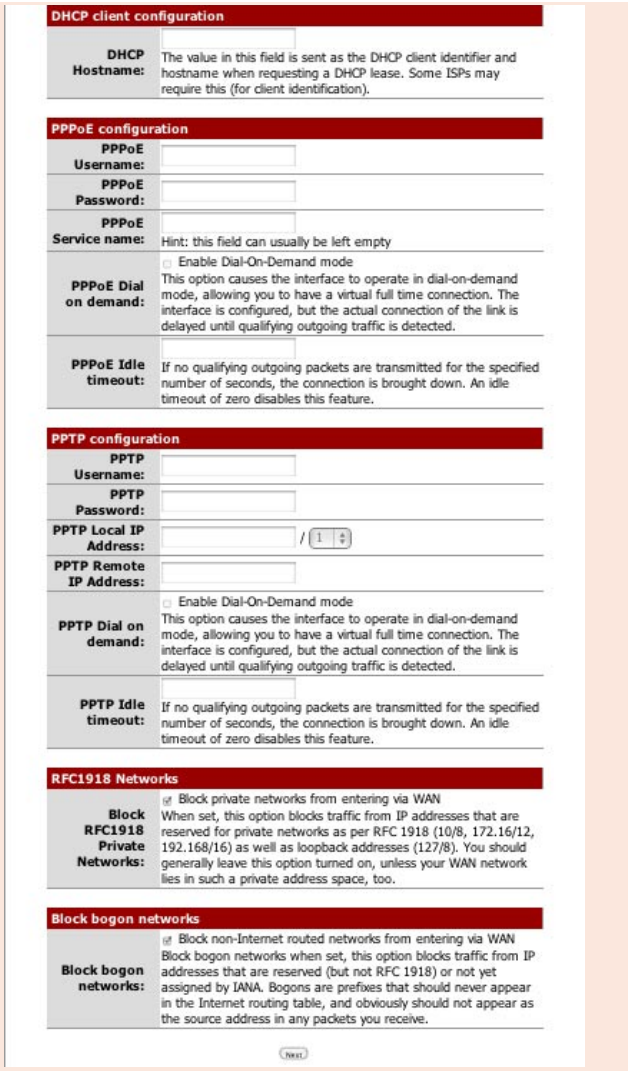
14 Set your hostname and DNS Servers (I choose to use OpenDNS, but you can input your ISP's DNS servers here)



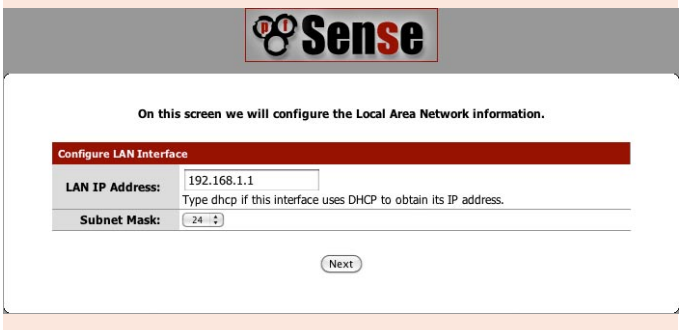
15 Set your Time Zone and NTP Source



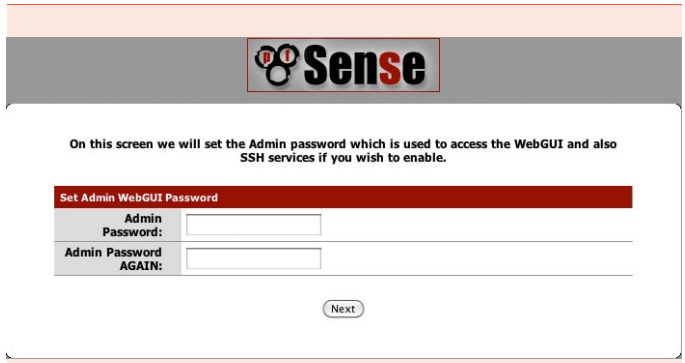
16 Configure IP settings for the WAN interface (If you have been assigned _some of the last_ public static IP Addresses, you can set them here)



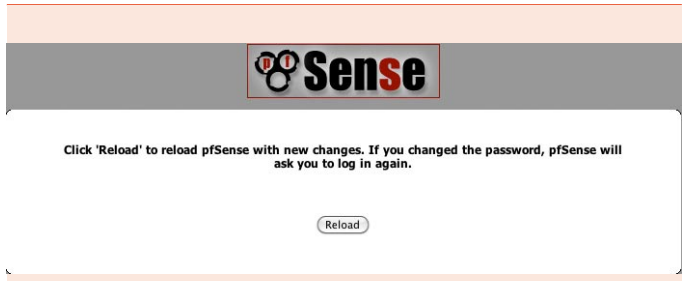
16 Configure IP settings for the WAN interface (If you have been assigned _some of the last_ public static IP Addresses, you can set them here)



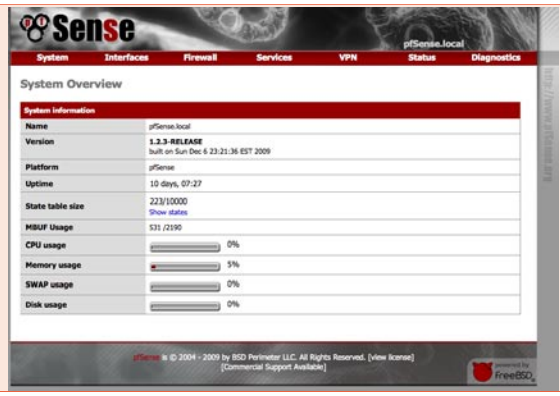
17 Configure IP settings for the LAN interface



18 Change the admin password (pfSense requires that you do, but it is good practice to always reset factory default passwords)



19 Reload the system and login with the new admin password



20 Congratulations you have now successfully installed and configured pfSense!!!

JR AQUINO

Jr Aquino is an Information Security Specialist working for Citrix Online. His contributions to open source efforts include projects such as: Yersinia, OSSEC, Python, and has recently been working on SUDO for the FreeIPA project. He has been a BSD advocate since first using FreeBSD 3.2 back in 1999. He resides in the beautiful county of Santa Barbara and owes his success to the support of his amazing wife and geek-in-training son.

Data CENTER

FOR IT PROFESSIONALS

MAGAZINE

Want to have all the issues of Data Center magazine?
Need to keep up with the latest IT news?
Think you've got what it takes to cooperate with our team?

Check out our website and subscribe to Data Center magazine's newsletter!

Visit: *http://datacentermag.com/newsletter/*

Mutt On OS X

Whenever my boss walks by my desk, he can't help but ask, „Why do you insist on using the command line for everything? Are you stuck in the 1970's or something?"

What you will learn…

- How to install Mutt and configure it to read and send mail on OS X

What you should know…

- How to use Terminal.app
- You'll need to be able to install Macports
- This is not going to replace Mail.app with Mutt (yet!)

I can't help but laugh – it's all too common for people to see a text-based, *command line interface* (CLI) and equate it with obsolete, primitive technology. In contrast to drab black and white text, Apple's OS X offers a beautiful *graphical user interface* (GUI), but luckily for us, *Beneath the appealing, easy-to-use interface of Mac OS X is a rock-solid UNIX-based foundation...* [1], and there's no better place to leverage the power of this BSD foundation than from the CLI. Indeed, the CLI is where the real power of any BSD is, and I'm so comfortable there that I rarely want to leave it.

Mutt [2] allows me to work with my email without having to leave my terminal emulator. Of course, sometimes I need to view an attachment or HTML mail (because some people insist on sending HTML mail...), and using Mutt on OS X does not stop me from doing that. With the help of some other CLI utilities, I can also search my address book for email addresses, query my keychain for mail server passwords, and other useful stuff. In this article I will introduce you to Mutt on OS X and help you get it installed and configured to read and send mail from a Gmail account.

I've found that the easiest way to get Mutt installed is to use Macports [3]. If you're not familiar with Macports, you should get acquainted with its use by reading the docs on the Macports website [4]. I'm going to assume you have Macports installed and you're ready to install Mutt. Of course, you'll also need a Gmail account [5] if you want to follow along with the sample config, although with some adjustments, you should be able to use any IMAP account. You will need to enable IMAP for your Gmail account, which is not difficult. For directions, see Gmail's help [6].

There are actually two versions of Mutt available via Macports, the stable version (which in the case of Mutt really means *legacy* version) and the development version. We'll be using the development version, which at the time of this writing is 1.5.21. Don't be afraid, the development version of Mutt is actually quite stable! I use it every day.

I use the following command to install Mutt (where the `%` is my zsh prompt):

```
% sudo port install mutt-devel \
+compress+gdbm+gnuregex+gpgme+headercache \
+imap+pop+sasl+smtp+ssl+trash
```

Macports offers a feature called *variants* which allows you to customize the installation of a given *port*. Each variant name, preceded by a + sign, represents an option that can be enabled during the compilation. I found that these options work for me, but feel free to experiment with the variants as you see fit. The variants I chose allow for IMAP and POP support,

SMTP support, SSL support and more. For the full list of the variants available for mutt-devel, you can run the following command:

```
% port variants mutt-devel
```

Be careful not to choose conflicting variants. It might be a good idea to try the options I've recommended above first, then experiment later. Macports will download and compile dependencies for you, if necessary, so you don't have to worry about that. There may be many, depending on what you have installed already on your system and what variants you've chosen to build with. When the build is done, you can test the installation with *mutt -v*, and you should see output telling you the version of Mutt along with the compile-time options that were enabled. Now for the fun part: configuration!

Open your favorite text editor (which is vim, of course?:) and create a file in your home directory named *.muttrc*. Leave the file empty and save it, and then run Mutt by entering *mutt* at the command line. When you first run Mutt, you'll most likely see nothing. In fact, you might get an error message telling you that */var/mail/yourUserName does not exist*. That's okay! Mutt tries by default to open the mail for your local shell user, and on a Mac it's highly unlikely that you'll ever get mail to your local shell account. You can safely ignore any error you receive regarding this, or you can create the missing file if the error message annoys you. Remember that unless you change the default configuration of your system, it's extremely unlikely that you'll ever get mail to your local user account, even if you've created the local mail spool file.

If you take a look at the top of your console, you'll see a list of some common commands. These should be self explanatory. One thing I'll mention though, is that the `g` command, which is noted as *Group*, is really *reply all*. It took me a little while to figure that out. At the bottom of the console you'll see a status line that tells you the current folder that is open and some other info. So where's the mail? We're getting to that! Mutt configuration can be complicated, so we're taking this step by step. As we progress we're going to use Mutt's own *command line* to check the values of variables, as well as reload our configuration file so that we don't have to quit and restart Mutt over and over.

The first thing we'll add is some comments to our .muttrc so we can leave notes for ourselves. It's easy to get lost in a configuration file, so leaving some clues about what we were thinking when we wrote a line or section will be helpful later on. Comments in Mutt configuration files start with a `#`. Here's the top of my *.muttrc* (the entire sample .muttrc from this article is available for download, see the resources at the end of the article [7]):

```
# Muttrc for OS X
# Sample .muttrc

# set realname to be your... real name
set realname="Mike Hernandez"
```

Add that to your *.muttrc* and save it. Then either quit and restart Mutt, or press colon `:`, which will bring you to the Mutt command line, and then type `source ~/.muttrc` and press enter. The realname variable will add your name to outgoing mail. To see that it was set, press `:` and enter: `set ?realname` at the Mutt command line. It should show the value you set in your configuration file. I recommend that you take time to get acquainted with the use of the Mutt command line – you'll want to reload your configuration file and check the value of variables quite frequently as you get things set up the first time. If you're lazy like me, you can press the up arrow after you've pressed `:` to scroll through the history of your previous Mutt command line entries. This way, you don't have to enter `source ~/.muttrc` repeatedly – you can just press *up* a few times and press enter. Don't forget to press `:` first though!

So where's the mail? In order to read our Gmail account, we need to tell Mutt where to connect, and who we are. Let's start with our IMAP username, which we'll set by adding the following to our *.muttrc*:

```
set imap_user="username"
```

The username should be the first portion of your email account, for example if your Gmail address is *janedoe@gmail.com* your IMAP user would be *janedoe*. As you may have guessed, the password for the IMAP account will be set via the value of the variable named `imap_pass`. Let's take a minute here to compare this to a more common email setup. If you've set up Gmail for use with *Mail.app*, your password will be stored in your system keychain, and when you open *Mail.app*, it will know the password for the account and could access your mail without asking you for your password. With Mutt, you generally have 2 options: you can store the password in the configuration file (which could be a security risk, especially on a shared machine, since the *.muttrc* file is plain text), or you can leave the password variables unset, which will cause Mutt to prompt you for the password when it needs it. If you're on a Mac, you

have a 3rd option, which I think is so amazing and useful that I'm astonished no other BSD based OS has such a feature available.

If you've been using your Mac for even a short time, you should be familiar with the keychain, which has a GUI application front-end named *Keychain Access*. The keychain is where your mac stores passwords for things such as wireless networks and email accounts. What you may not be aware of is that there is a command line utility that allows access to everything in your keychain! By using the power of the shell, we can query the keychain at runtime and retrieve the passwords that we need. Here is how I do it:

```
set imap_pass=`security find-internet-password -g \
-s imap.gmail.com 2>&1 >/dev/null | cut -d\" -f2`
```

There is a lot to explain here. First, you should note that I use back-ticks, NOT single-quotes. The back-tick (`) is special because it tells Mutt that it should execute a shell command and assign the output of the command to the variable. Inside the back-ticks, you'll see the command and options that allow me retrieve the password for my Gmail account. The security command-line utility is extremely powerful, and I highly recommend reading the documentation found in the man page [8]! It's well beyond the scope of this article to explain everything with regards to this utility, but even a casual glance can tell you that the command above will *find an internet password... for a server named imap.gmail.com...* The rest just edits the output of the security command so all that is returned is the password (the security command outputs quite a bit of info, so it's necessary to cut only what we need from it).

Now, to tell Mutt we have a mailbox we want it to check for us, and to let Mutt know that we would like to start with our Gmail inbox open by default, we add the following to our *.muttrc*:

```
# which mailboxes to acknowledge
mailboxes imaps://imap.gmail.com/INBOX

# start with our Gmail as the default mailbox to display
set spoolfile="imaps://imap.gmail.com/INBOX"

# how often to check for new mail in the mailbox
set mail_check=60
```

Note that the mailboxes command is different from setting a variable – there is no `set` and no `=`. This is in contrast to the other lines in our config, where we set `variablename="value"`.

At this point, I recommend quitting Mutt and restarting. As Mutt starts up, it will connect to your Gmail account and then query your keychain for the IMAP password. You might get a GUI dialog box requesting whether or not you want to allow Mutt to access your keychain: you may want to *always allow* this, to avoid having to click *allow* every time you open Mutt, but that's up to you. What you should see now is all the mail in your Gmail inbox! Congratulations.

Reading your mail is, of course, only half of the fun. Now, let's add to our configuration so we can also send

Listing 1. *SMTP related settings in Muttrc*

```
# set the From address and tell Mutt to use it
set from="yourgmail@gmail.com";
set use_from;
set envelope_from;

# set the SMTP server - note the use of the imap user variable here
# also note that the protocol in the URL is smtps://
set smtp_url="smtps://$imap_user@smtp.gmail.com";

# if you've already set up Mail.app for Gmail, you can use the security utility
# otherwise you can set smtp_pass to your password
# or you can leave it blank to have Mutt prompt you when it needs the password
set smtp_pass='security find-internet-password -g \
-s smtp.gmail.com 2>&1 >/dev/null | cut -d\" -f2''
```

On the 'Net

- *http://developer.apple.com/library/mac/#documentation/MacOSX/Conceptual/OSX_Technology_Overview/SystemTechnology/SystemTechnology.html* [1]
- *http://www.mutt.org/* [2]
- *http://www.macports.org/* [3]
- *http://guide.macports.org/* [4]
- *http://www.gmail.com* [5]
- *http://mail.google.com/support/bin/answer.py?hl=en&answer=77695* [6]
- *http://culmination.org/Mike/sample-muttrc* [7]
- *http://developer.apple.com/library/mac/#documentation/Darwin/Reference/ManPages/man1/security.1.html* [8]

mail. In order to send mail from *gmail.com*, we need to use Google's SMTP server, the only valid sending server for mail that is from the gmail.com domain. With the current version of Mutt, as long as SMTP support is enabled, we can add the following to our configuration file: see Listing 1.

Save the configuration, then either reload it or quit and restart Mutt. To test, compose a message by pressing 'm'. Mutt will ask you who you want to send the mail to, as well as the subject of your message. After you enter these, it will open your default editor and let you compose your message. Type a quick test message, save it, and quit your editor, and then you'll be returned to Mutt. Notice that the menu of available commands has changed: you can press *y* to send the message, *q* to abort sending the message, etc. If you send the message to yourself, you should see it appear in your inbox soon, otherwise go check the account to which you sent the test mail. That's it!

Next month we'll go deeper into Mutt configuration and learn how to search the Mac Address Book for email addresses, configure and use multiple email accounts, and more. See you then!

MICHAEL HERNANDEZ

Mike is an IT consultant and web programmer. He lives in Brooklyn, New York, and he and his wife are celebrating their one year anniversary on February 14th. He also loves electronic dance music and commuting on his fixed gear bike, appropriately named Constance.

The Missing Links

to Strategic Implementation

In regards to growth and strategy, the father of management and strategy, Peter Drucker was wont to say, "Everything must degenerate into work if anything is to happen."

How often have you experienced the *work* of your team's strategies, not coming to fruition, nor having efficacious follow up? Frequently delays occur with great plans, not because of a lack of discipline in the team, but rather because of shortcomings in processes and tools with which your team must work.

Regardless of your size, the devil is in the details of prompt follow-ups, and finding and executing on particular communications. You as a leader need to ensure your team has the best tools and processes available for successful implementation of your strategy.

Is the progress and next steps on your objectives the focus of your progress meetings or is there too much attention spent on the irrelevant details of past activities? Is the focus on the greatest constraining factors slowing progress and then on solutions or is your team focused on proving their value through verbally reporting out past accomplishments?

I have found that better managers tend to have the reporting out of past accomplishments already communicated before monthly progress meetings. They tend to have agenda slots for discussing, brainstorming and deciding on possible solutions to ameliorating the most constraining factor around progress on each of the top objectives. Every process has a top constraining factor, phase or step in its process, you as a manager must continually be looking for those constraints and opening them up.

Apropos to constraints, one of the most frequently noted is the inconsistency in follow up, and time wasted in communications management; i.e., finding and following up upon all of the hundreds of bits of communications that make up our work. It isn't about time management; we all have the same amount of time, its communications and priority management; doing the right things at the right times and having easy access to the communications of such accomplishment.

When you think about the communication venues in your work they will fall into a handful of buckets; meetings, discussions, letters and emails for example. Meetings are usually well documented both in pre and post emails. Discussions are often at risk of not being document and thus not followed up upon, letters have become quite rare, but emails hit us by the hundreds each day.

Now to handle email overload and empower your email client to actually become a part of your strategic execution and review toolset as well as to become a nimble instant recaller of all communication for the ultimate in communication management tools consider these options.

If you aren't nailed down to an email, contact and calendar client, consider the new alternative to Microsoft Exchange and Outlook; GoogleApps. Built in is the best spam filter, I've ever experienced. While not instant, searches for anything are easy, and the cloud-based aspect allows for the both the speed of a synched database on your PC's but also the worldwide accessibility from any web browser. Stability is also exemplary. But like MS Exchange, you'll need help, Google *gappsmasters* and you'll find some good solution providers for this approach.

If you are like 90 plus percent of companies you are on Outlook. This next tool is as valuable if not more so than the Outlook venue itself. The NEO Pro (*Nelson Email Organizer Pro*) solution at *www.emailorganizer.com* is hands down the best communication and priority management tool for making Outlook a solution to implementing strategic growth initiatives with faster follow

up and increases in productivity, thus making significant impact on your bottom line.

NEO Pro creates an index of every word in every email allowing for instant searches of any word in every email in a couple of seconds verses the latest Outlooks *improved* search approach that can still take minutes. All individuals with whom you corresponded are given their own folder so like a relational database instant access to every email with any individual can be retrieved with just a few clicks. No need to create organization in your email system, it is already done for you.

A variety of labeling options and alarms allow for emails to be prioritized into a workload and follow up that give you the control of multiple buckets for multiple projects vs. the constricting linear approach of Outlook's layout. Most every client of mine reports saving at least 10 minutes a day and many report an hour a day saved, but more importantly they have greater control over getting more of the right things done at the right time… thus getting their strategy executed upon because these email tools have become de rigor to their white collar toolbox.

Remember, your strategy is worthless if your team can't track and follow up upon communications effectively and efficiently. Strategy must degenerate into work accomplished in a timely and organized manner, these tools can help to make your growth strategy a reality sooner.

MARK FAUST

Mark Faust is the founder of Echelon Management and author of Growth or Bust available at www.Borders.com Since 1990 Mark has been a growth consultant and coach to CEO's, management and sales teams as well as a sought after keynote speaker. Contact Mark at Mark@em1990.com – www.EchelonManagement.com

Browser Wars

With the rise of the Internet, there has been a considerable increase in the number of web browsers available for BSD platforms.

Not so long ago, in this world of internet, there used to be a time when the term *web browser* signified only a few text-based browsers. Nobody needed to run Pop-up blockers and Flash may have been another comic book character, who cared! SImple, and minimalist. Right?

Move on to the present day. Things have changed. And they have changed big time. We have numerous Open Source browsers. BSD Magazine brings to you a round-up of (arguably) the best web browsers.

Getting to the point, which is the best choice for surfing the Web? No, I am not going to answer that here. But I will surely help you make that decision yourself. After all, freedom consists of helping and sharing, right?

Well, the choice for browsers depends on the end-users. With the ever increasing number of options available, one is surely bound to feel overwhealmed. Gone are the days when Mozilla Firefox used to be the de-facto standard web browser with every system, – Google's Chrome web browser (Chromium, for sake of clarity) has fast risen to challenge the Titan. So without further ado, let's plunge into the exploration of one web browser after the other.

Mozilla Firefox

You can't speak of the Internet without mentioning Firefox in the same breath (on second thought, yes you can actually, but then never mind). Firefox is the world's second most used web browser and boasts of features such as tabbed browsing, spell checking, embedded search, live bookmarks, a download manager, etc. Furthermore, the Firefox community continuously develops new and innovative addons to enhance browsing experience. No matter what your usage is, Firefox might be your first choice for browsing!

Google Chrome (Chromium)

Chrome is one of the fastest growing browsers in the world from the house of Google. Most of Chrome's source code is open source under the BSD license, so you will definitely not be violating the GPL in using it. Chrome aims to improve speed and overall performance, and it does quite well in achieving its goal. It has a minimalist interface which is customizable via themes and its Javascript processing speed is considered to be the fastest to date. Gaining popularity day by day, Chrome has established itself as the ideal web browser for numerous home users.

Opera

The browser-of-choice for the geek community, Opera has been around for a long time, though its presence on the desktop has rarely ever been noticeable. Opera Mini for mobile phones is popular with mobile internet users, but the same story is hardly applicable on the desktop. Nevertheless, Opera has few unique features up its sleeves, such as Speed Dial to access favorite pages (no longer unique though, Chrome also offers the same), Widgets, Opera Turbo to speed up slow connections (a real blessing if your are on GPRS or other slow connection), built-in phishing and malware protection (shame on you, Internet Explorer!), cross-platform compatibility (including BSD and Solaris) and strong encryption. The negative side? Opera is not *entirely* Open Source – it is a closed source freeware application with chunks of open source thrown in here and there.

Seamonkey

Yes, there is a web browser by that name. I am not joking. Actually, its more of an Internet Suite including a browser,

an email client and an HTML editor (wow! Talk about goodies). So if you opt for Seamonkey, you need not consider a separate browser or email client.

Coming from the same stable as Firefox (Mozilla), Seamonkey is an Open Source web browser available for both Windows and Linux/BSD. Overshadowed by Firefox, Seamonkey does have some useful features such as Password Manager and Form Manager.

Flock

Flock has established its reputation as being the Social Browser. It offers seamless integration of social networking sites such as Facebook, Twitter, MySpace, etc. On the downside, most of the software is proprietary. Nevertheless, Flock with its features such as the inclusion of a blog editor and a feed reader has gained tremendous popularity. It does not 'natively' support BSD yet, but the UNIX version can be installed via the GUI.

Konqueror

Konqueror is the default web browser bundled with KDE. Until KDE 3.5, it used to double up as a web browser as well as file manager, but with the release of KDE 4, the file managing part was separated into Dolphin in order to make Konqueror a stand-alone web browser in its own right. Being fully open source, Konqueror suffices as a well to do web browser for everyday usage, though it seriously lacks the prowess of the likes of Firefox or Chrome.

Epiphany

Epiphany is the default web browser for the GNOME desktop. Minimalist and simple, Epiphany provides a nimble browsing experience for the casual user with great support for add-ons.

GNU IceCat

GNU IceCat was earlier known as GNU IceWeasel. It includes several enhanced security measures such as the option to block third party cookies, warnings against URL redirection (which I found to be quite irritating: loading Gmail gives you 4 warnings), etc. IceCat natively supports Firefox add-ons and extensions as it is based on the Mozilla engine.

Midori

Midori has three factors that add to its USP: speed, stability and a nice name (Japanese word meaning *green*). By default, XFCE desktop environment bundles Midori in its Goodies' components. Midori offers seamless integration with GTK+, hassle-free bookmark management and adblocks (though not many ads were successfully blocked while I was testing it).

Swiftweasel

Swiftweasel is yet another browser based on Firefox, but it is available only for Linux. It is completely free and open source and does all that Firefox can, except for the fact that more often than not it struggles with Flash content.

Lynx

Ok, let's say you have a true console lover in you. Or perhaps you wish to be renamed *Keyboard Warrior*. In that case, a text-based browser is what you need (on a serious tone, text-based browsing is meant for character based terminals). Lynx is one such text-based web browser that is highly customizable and supports a good deal of HTML. Lynx cannot natively display images or videos, but it can launch third-party programs to do the same. Besides, Lynx does not support Javascript either. But it proves to be very handy in low bandwidth scenarios or in cases when only text-based browsing is available.

Well, that was a summary of some of the main browsers available for the BSD/UNIX architecture. There are plenty more, but the above ones dominate the scene.

So the simple question that would arise now is: which one to use? Again, that depends from user to user. I tend to use Opera and Seamonkey, but then the majority uses Firefox or Chromium. Either way, you would not make a bad choice. So go ahead, pick you weapon (oops... browser) of choice, and don't forget a *Too COOL for IE* badge!

SUFYAN BIN UZAYR

Sufyan is a 20-year old freelance writer, graphic artist, programmer and photographer based in India. He writes for several print magazines as well as technology blogs. He is also the Founder and Editor-in-Chief of http://www.bravenewworld.co.nr He can be reached at http://www.sufyan.co.nr.

BSDCAN 2011

THE BSD EVENT OF 2011
http://www.bsdcan.org/

Ottawa, Canada

BSDCan 2011 – The event to be at this year

BSDCAN 2011

There's only one major BSD Event in North America in 2011: BSDCan

WHERE

Ottawa, Canada

WHEN

Early May 2011, with two days of tutorials before the conference. Exact dates to follow.

WHO

Contributors, developers, and users

VENUE

University of Ottawa
http://www.uottawa.ca/

AT FEES YOU CAN AFFORD

We plan to keep costs to a minimum. As such, the conference will be held at University of Ottawa and accommodation is available within the University residences. Hotels are also within close walking distance of the conference venue.

WHAT DOES IT COST?

Type	CAD
Regular	$195
Corporate	587
Additional Corporate	$120
Student	57
Tutorial	57

Comfortable accommodation is available on campus at very reasonable rates. See our website for details.

Take the BSDA Certification exam.
For details see
http://bsdcertification.org/

SCHEDULE OVERVIEW

Wednesday

4:00 pm	Sign-in desk opens at a local pub. Get your registration pack and have a drink.
8:00 pm	Sign-in desk closes.

Thursday

9:30 am	Opening words
11:00 am	First set of talks
12:00 pm	lunch
1:00 pm	Second set of talks
2:00 pm	break
2:30 pm	Third set of talks
3:30 pm	break
4:00 pm	Fourth set of talks
5:00 pm	Key Signing Party

Friday

10:00 am	First set of talks
11:00 am	break
11:30 am	Second set of talks
12:30 pm	lunch
1:30 pm	Third set of talks
2:30 pm	break
3:00 pm	Fourth Set of Talks
4:00 pm	Fifth Set of Talks
5:00 pm	Closing words

Sat

8:30 am	Breakfast
9:30-4:00	Tourist fun

TALKS FROM 2010

Please see the website for complete details.

- ClangBSD - Replacing gcc with clang
- Consideration for the BSD Professional Exam
- Security Implications of the Internet Protocol version 6 (IPv6)
- Puffy At Work -- Getting Code Right And Secure, The OpenBSD Way
- Everything you need to know about cryptography in 1 hour
- Networking from the Bottom Up: IPv6
- Porting dummynet to Linux and Windows
- Journaled Soft-Updates
- Porting hwpmc to non x86 platforms
- Maintaining a Customized FreeBSD Distribution
- Debuggers - Architecture and Implementation
- pfSense 2.0
- Networking from the Bottom Up: Ipv6
- The New VVorld
- A new packet scheduling architecture for FreeBSD

SOCIAL ACTIVITIES

It's not all work. Social activities play a major role in project development.

Wednesday

4:00 pm Drinks + registration at a local pub

Thursday

4:30 pm BOFs
7:00 pm Gathering at local eateries for dinner

Friday

4:30 pm Key signing party
7:00 pm Gathering at local pubs for drinks

Saturday

8:30 am Breakfast

afterwards: various tourist-type things

To stay informed, please join our announcement mailing list. Details at http://www.bsdcan.org/

2010 PLATINUM SPONSOR

OUR 2010 GOLD SPONSORS

Interview with Dan Langille

Dan Langille is an organiser of BSDCan conference since its start in 2004, and is well known in BSD community. You can find more of his work at FreeBSD Diary (http://www.freebsddiary.org/about.php) and FreshPorts (http://www.freshports.org/about.php).

For those of our readers who are not familiar with it – can you tell what is your conference about? What is your mission?

Dan Langille: A key component of BSDCan is building relationships. Those relationships can be between developers, between projects, between users, or any combination of groups/people associated with BSD and related projects. We feel that knowing the people you work with is a important to building a good working relationship. To that end, BSDCan always has many opportunities for socializing and meeting people. For example, we have longer breaks between each talk to give plenty of time for people to set up, mingle, and chat.

Let us know who should attend to the conference?

Dan Langille: Anyone using or interested in BSD systems. :)

How did BSDCan start and how has it evolved to where it is today?

Dan Langille: I was living in Ottawa and unemployed at the time. I'd been to a number of BSDCon and USENIX conferences. I knew the main person behind the Ottawa Linux Symposium and talked to him about his conference. It seemed straight forward enough, so I decided to proceed. The start was very simple and cautious. Over the years, we've gained sponsors as well as having constant support from long-term sponsors. This has helped us tremendously in terms of what we're able to do. We progressed into bigger venues, starting providing lunch for the attendees, and we're able to bring in speakers from overseas. The support and enthusiasm of our sponsors, speakers, and attendees is the foundation that has allowed us to evolve.

You say you have found fantastic formula that appeals to wide range of people. Can you tell us what is it?

Dan Langille: I really don't know exactly what it is, but people keep telling us about how great we are. There are many things, all of which converge on Ottawa each May to result in a great conference. BSDCan is big enough to attract the great speakers and sponsors, but small enough and friendly enough that everyone is comfortable and sociable. We've always wanted to keep things inexpensive and the University of Ottawa campus provides us with facilities that meet that need. People can take a fast city bus from the airport to the venue. From there, everything is within walking distance; pubs, restaurants, entertainment.

How many companies are presenting at the conference?

Dan Langille: We have never allowed commercial presentations.

Who will be speaking at the BSDCan 2011?

Dan Langille: That will not be available until 19 February.

What kind of criteria do you use to select the topics for the conference agenda?

Dan Langille: Many things, including content: topic, speaker, interest in the topic, relevancy to our attendees.


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!

PC-SYSINSTALL

A new system installer backend for PC-BSD & FreeBSD

By Kris Moore

Since its very first beta many years ago, PC-BSD has been using a custom-built installer routine, which consisted of a graphical user interface, tied into some scripts which performed the actual installation process of the system. While the process worked reasonably well, it lacked many important features which would become desired and critical down the road. Features such as automated installation, an independent installer backend with interchangeable front-ends, support for advanced custom partitioning, and full error logging. In addition to these features, many times the idea had been brought up about enabling the new installer to also support traditional FreeBSD installations as well. This could be used as a way for users to bypass regular *sysinstall* and install using some new features such as ZFS, Encryption, Mirroring and more.

Late in the spring of 2009 as we began looking ahead to the eventual release of PC-BSD 8.0, based on FreeBSD 8.0, I decided that it was time to take these ideas and make them a reality. I evaluated several existing FreeBSD installation backends already in existence and found each of them to have strengths, but not be exactly what we had in mind for PC-BSD. Knowing this, work began shortly thereafter on *pc-sysinstall* and the new backend was merged into our subversion repository late that fall. This new backend supports a myriad of new features and abilities and will be the default installation system on PC-BSD 8.0 and future releases going forward.

Program flow design

As I began to approach the design of the new installer backend, there were several issues which needed to be resolved beforehand. In my analysis of other installer backends, I found that there were a few design options relating to how they interacted with a front-end. In the case of our existing installer, the front-end actually performed all the logic of the installation process itself, and as such was not able to be abstracted enough to allow automated installations with a supplied configuration file. This was the design we were trying to break free from, by creating a backend which could perform the entire installation independently.

On the other extreme, some of the backends I reviewed performed 100% of the logic, including program flow for the front-end during the data collection phases. The backend, once started would handle all user input, direct which dialogs to display to an end user, and decide where the user could go at any given time. This model also didn't seem to fit in with the flexibility we desired, especially when working with a powerful GUI toolkit such as QT.

After reviewing these options, I decided the approach we should take was to create a "hybrid". A backend, which once given a valid configuration file, would be able to perform a complete system installation unattended, however the task of actual program flow for the end user would be left up to the front-end, allowing a variety of front-ends to be developed or modified without requiring extensive changes to backend functionality. To assist the front-end in their program flow, the backend would supply different commands which could be used to query specifics about the system, such as the available disks and information on them, network card availability, locale data and more. In addition to these queries, the front-end could supply the backend with arguments to take specific actions prior to an installation, such as enabling networking, setting up ssh keys, querying for available installation servers, etc. This method ensured that when designing a frontend, the developer would be given maximum flexibility in creating the user-flow best suited for his program's needs, without having to delve into updating a separate backend code base.

Development language and toolset choices

With the program flow settled, the next major decision I was faced with was the choice of programming language to use, along with any tools which the backend would rely on. While there are a variety of languages and tools readily available for such a task, my main concern was trying to keep the new backend functional with little to no additional programs or libraries required than was already available on a standard FreeBSD installation. In addition to this requirement, I wanted to keep the backend in a format which could be easily modified *on-the-fly* on a booted installation disk, without requiring re-compiling tools or libraries which may not be present on a minimal installation image. This would greatly accelerate the development process, while providing an easy way to debug, and test potential fixes in a live environment. In the end I decided that all of this could be easily accomplished by creating the backend as a shell script (`/bin/sh`), and using only command-line utilities which are standard to FreeBSD such as fdisk, fetch, bsdlabel, glabel, zfs and others.

pc-sysinstall features and configuration

After making the decisions on program flow and format, development began immediately on the new installer and progressed quickly. On December 2 2009 the new installer backend and frontend (Illustration 1) were both committed as the defaults for PC-BSD 8.0, and have been in usage since then. The new backend improves upon our existing installer, while also offering many new features, such as:

- Able to choose between installing vanilla FreeBSD or PC-BSD
- Support for enabling gmirror across two drives
- Complex partition layouts, using variety of file systems:
  - UFS
  - UFS with softupdate
  - UFS with Journaling
  - ZFS
- GELI-based encryption
- File system labeling with glabel
- Installation logging and debugging output

- Timezone and localization configuration
- Install using source files from the Internet or local network
- Perform upgrades of existing PC-BSD systems
- Install using custom-rolled system images or backups

Illustration 1. *SysInstaller front-end in action*

From an end-user perspective using pc-sysinstall is fairly straight forward, and somewhat similar to creating a traditional FreeBSD sysinstall configuration script. A front-end is simply a tool which gathers user input on all the various installation options, then generates a working installation configuration file, which it then calls the backend to run with. By querying the backend independently, the front-end developer is able to fine-tune the workings and look their application, providing different and unique ways in which to allow the user to select their options, from disk management2 to user setup.

Let us take a look at some of the specifics of a pc-sysinstall configuration file, and how it can be used to install either a FreeBSD or PC-BSD system. After running the SysInstaller frontend, it will generate a pc-sysinstall based configuration file, and save it to `/tmp/sys-install.cfg`. Within this file will be all the options necessary to perform an installation or upgrade of a system. The beginning of the file sets some basic options common to all types of installations:

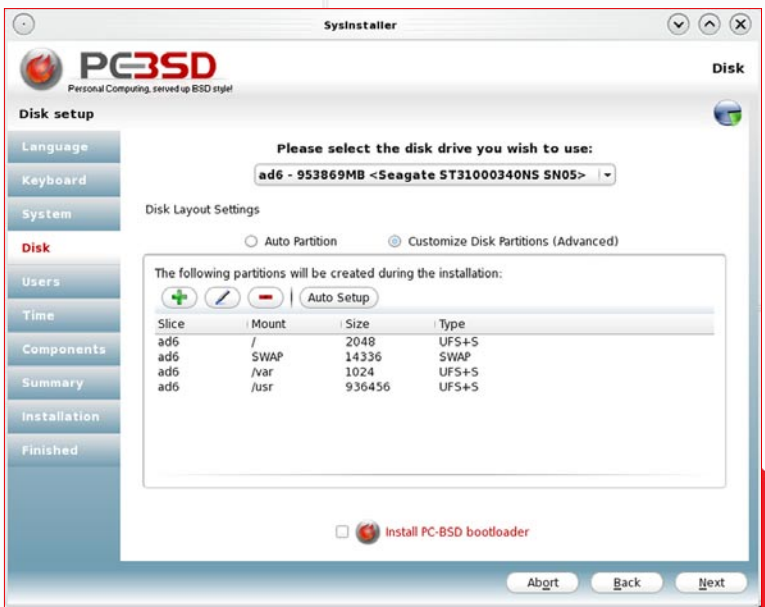


Illustration 2. *Disk Layout in SysInstaller*

```
# Auto-Generated pc-sysinstall configuration
installInteractive=no
installMode=fresh
installType=PCBSD
packageType=uzip
```

The configuration file syntax follows some traditional standards, such as placing comments with the `#` sign, and all options are specified in the format of `keyword=<setting>`. Settings can be placed in almost any order, with a few exceptions for disk-layout and user information blocks. In these first few lines, we are instructing pc-sysinstall to run completely non-interactively, no prompting via the command-line for anything, specifying that this is a *fresh* install as opposed to an *upgrade*. Also, with the `installType= and packageType=` we are specifying which type of system is being loaded (PCBSD or FreeBSD), and what format the installation archive is in (tar or uzip). While all of the

available options are documented in the examples/README file within pc-sysinstall, there are a few notable cases we will want to take a look at, specifically relating to disk management.

Disk configuration with pc-sysinstall

```
# Disk Setup for ad1
disk0=ad1
partition=ALL
bootManager=none
mirror=ad2
mirrorbal=round-robin
commitDiskPart
```

When specifying the disk(s) we want to format and install to there are a couple of specific blocks of instructions needed in order for pc-sysinstall to perform the tasks correctly. The first section is for specifying the target drive and partition (or slice). This section must begin with `disk0=`, with subsequent disks being labeled `disk1=`, `disk2=`, etc. The partition keyword indicates the where the installer will we be formatting and installing a label, either *all* for an entire disk, s1-s4 for an existing primary partition, or *free* which takes free disk space and creates a new primary partition for it. The bootManager keyword is used to specify if the FreeBSD MBR (boot0) should be installed onto the disk drive or not, by setting it to *bsd* or *none* respectively. Also in this example we provided information on mirroring, specifying that the disk `ad2` will configured as a gmirror of `ad1`, using the balance method of *round-robin*. Lastly the commitDiskPart command instructs pc-sysinstall that we are finished specifying disk setup options for this drive/slice. If the installer is to setup multiple disks, then a similar codeblock will be required for each disk with their respective options.

```
# Partition Setup for ad1(ALL)
# All sizes are expressed in MB
# Avail FS Types, UFS, UFS+S, UFS+J, ZFS, SWAP
# UFS.eli, UFS+S.eli, UFS+J.eli, ZFS.eli, SWAP.eli
disk0-part=UFS+S 2048 /
disk0-part=SWAP.eli 14336 none
disk0-part=UFS+S 1024 /var
disk0-part=UFS+J 250000 /usr
 disk0-part=ZFS 686456    /data
 commitDiskLabel
```

After the specifying of the disk slice/partition information, pc-sysinstall will next require a section detailing how the individual mount-points and file systems are to be setup on this disk. In the DiskLabel section section above, we can see a complete file-system layout for the target disk., In this example disk0 would correspond to the `disk0=ad1` specified in the previous *DiskPart* configuration block. The `disk0-part=` keyword is unique, in that it takes 3 arguments, separated by spaces, which are used to indicate the file-system type, size in megabytes, and mount points respectively. When specifying the filesystem, the *.eli* extension is special, indicating that geli encryption should be enabled for this partition. Partition letters will be automatically assigned and created from this configuration, such as ad0s1a for /, ad0s1b for SWAP and ad0s1d-h for additional filesystems. In addition to setting up the partitions, pc-sysinstall will also automatically generate labels for the devices using glabel, and reference

those in the auto-generated `/etc/fstab`. A mountpoint for `/` would be given a label such as `/dev/label/root[0-9]` or swap would become `/dev/label/swap[0-9]`. Other filesystems would take the directory name of the mount-point and use it as a label, so that `/data` would become `/dev/label/data[0-9]`. Lastly the commitDiskLabel command must be given, instructing pc-sysinstall that we are finished specifying file-systems and ready to apply the settings to disk. As with the DiskPart configuration block, any subsequent disks will each require their own DiskLabel section.

Advanced commands for pc-sysinstall

In addition to the disk configuration options available in pc-sysinstall, it also provides some additional commands which can be used to further customize an installed system. First among these are networking configuration options, which are used to customize the systems network interfaces, so they are available at first bootup:

```
netSaveDev=AUTO-DHCP

or

netSaveDev=em0
netSaveIP=192.168.0.49
netSaveMask=255.255.255.0
netSaveNameServer=208.67.222.222
netSaveDefaultRouter=192.168.0.1
```

The `netSaveDev=` keyword includes a special keyword AUTO-DHCP, which instructs pc-sysinstall to automatically locate and set DHCP mode on any detected network interfaces, also creating the appropriate wlan[0-9] devices for wireless nics. Should the user wish to assign their own networking configuration, the various netSave options also can specify a specific interface, IP, netmask, nameserver and default router.

If your system requires further customization during the installation, pc-sysinstall can also help, by offering various run commands:

```
runCommand=cp /root/rc.custom /etc/rc.conf
runScript=/root/post-install.sh
```

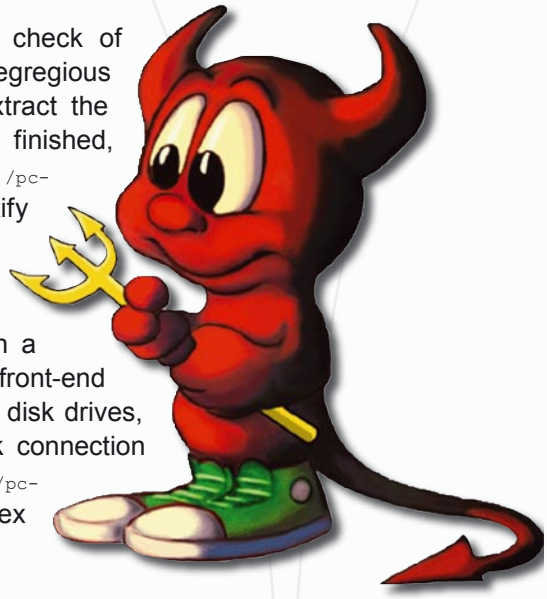
These options are checked for and executed after the initial extraction of the installation image, and each provide different functionality. First, the runCommand option allows the execution of the specified command within a chroot environment of the system. This can be useful when you only need to make minor adjustments to the system post-install. Should you require more advanced post-install configuration, it is also possible to supply a script you wish to run in the chroot environment using the runScript command. This command will take the specified script, copy it to the installed system, run it in chroot, and remove it afterwards.

Running pc-sysinstall and using the query interface

Once you have a complete configuration file for pc-sysinstall, starting the installation process is very straight-forward. While in the pc-sysinstall directory, we simply need to run the command with the `-c` flag to specify a working configuration:

```
# ./pc-sysinstall -c /tmp/sys-install.cfg
          ^^^^^^^^^^^^^^^^
          Path to configuration file
```

The installation process will start by doing a quick syntax check of the configuration file, in order to catch any of the more egregious configuration errors, before starting to configure the disk, extract the image, and perform post-install setup. After the installation is finished, a copy of the log file will be copied to the system disk at `/pc-sysinstall.log`. Should the installation fail, the installer will notify the user, and provide the location of the log file in memory for immediate inspection.



In addition to this usage, pc-sysinstall can also be run with a large variety of commands to provide information to the user, or front-end interface. Commands are available for tasks such as detecting disk drives, displaying available time-zones, testing for a working network connection and more. A full list of commands may be viewed by running `./pc-sysinstall` without any arguments or by viewing the doc/help-index file.

Future enhancements & goals

While the pc-sysinstall tool is already very powerful, there are still areas of improvement I would like to see worked on. First, is improving the encryption support, such as allowing pass-phrases to be specified for a partition. Currently the installer uses randomly generated keys, which are stored in `/boot/keys` on the installed system. Secondly I would like to work on improving the front-end interfacing, providing more common queries, and improving upon our existing ones, making it easier for more front-ends to be developed, including some which are text-based. Lastly I would like to improve the restore functionality, by expanding it to perform complete system restores from a wider variety of backup types, such as regular rsync, tar and more.

Getting pc-sysinstall and reporting bugs

pc-sysinstall is being used on PC-BSD media starting with version 8.0 and higher, and when booted may be located in the `/PCBSD/pc-sysinstall` directory. The code may also be accessed directly from our subversion repository under `/pcbsd/trunk/pc-sysinstall` or can be checked out anonymously with the following command:

```
# svn co svn://svn.pcbsd.org/pcbsd/trunk/pc-sysinstall
```

Should you run across a bug in pc-sysinstall or have ideas for improvement, the best place to reach me is either at the dev@lists.pcbsd.org mailing lists or directly at kris@pcbsd.org / kmoore@freebsd.org

Conclusion

We've only looked at a few small examples of the usage of pc-sysinstall, and some of the features it now offers. The interface is already quite mature, and able to support a variety of different installation configurations, as well as provide a complete backend for various front-ends to be developed. Through the release of PC-BSD 8.0 and future releases the interface will continue to be enhanced and made more stable, helping to make some of the latest cutting edge FreeBSD features available during install time, for novices and advanced users alike.

Support

Because sometimes, you just need some.

The iXsystems Professional Services Team is comprised of veteran FreeBSD® developers, administrators, and project committers.

Our expert staff provides unparalleled levels of support for FreeBSD®, PC-BSD® and FreeNAS™ ranging from problem resolution to consultation to custom development. For more information on our Professional Services Offering, visit our website at http://www.iXsystems.com/BSDsupport and complete the inquiry form.

iXsystems®

Call iXsystems toll free or visit our website today!

+1-800-820-BSDi | www.iXsystems.com