**FREE DVD INSIDE** NetBSD 4.0 + APPLICATIONS

MAGAZINE

# BSD

Vol.2 No.1 Price USD14.99 AUD14.99 Issue 1/2009 (3) 1898-9144

# Explore NetBSD

## a quick know-how

EXCLUSIVELY

▶ Interview with Simon Burge, Antti Kantee and Greg Oster

**INSIDE**



**USEFUL ARTICLES AND TUTORIALS**

# ROCK SOLID FIBONACCI SUPPORT

## For Your Server and Desktop Solution

iXsystems understands the need for an operating system that functions for both the casual user as well as the experienced professional. Based on a FreeBSD 7 core, PC-BSD 7.0 Fibonacci Edition brings stability, security, and ease of use to the desktop and the server. Fibonacci features a KDE 4.1 desktop and self-installing software packages, called PBIs (Push Button Installers), as well as graphical system administration tools. With PC-BSD Fibonacci Edition, your desktop will be running quicker, have enhanced wifi compatibility (including 802.11n support), improved stability for Wine (a compatibility layer for running Windows programs), dual-head monitor configuration support, and a series of server administration tools.

In order to meet your personal and professional needs, the iXsystems Professional Services Team is now providing phone and e-mail support for PC-BSD as well as FreeBSD. Here are just a few of the many reasons to purchase professional PC-BSD Support from iXsystems:

### Technical Expertise

You can rest assured that your PC-BSD support issue is being handled by the highest-level PC-BSD developers and contributors. iXsystems partners with major developers and long-time contributors from the FreeBSD community to offer custom development and advanced level FreeBSD and PC-BSD support and consulting services. Our Account Management Service Professionals will work with you to develop software solutions specific to your business operations.

### Custom PBI Creation and Deployment

PC-BSD uses graphical utilities known as PBIs to remove and install software. PBIs are self-contained and can be installed with their own libraries, eliminating the problem of shared dependencies. The experts at iXsystems are well versed in compiling software applications into PBIs for use on PC-BSD. Many popular programs are already available in PBI format, and iXsystems staff can create PBIs for your custom application. If needed, these applications can be deployed over multiple desktops by our technicians.

### Support for Large Rollouts

The Professional Services Team also provides installation support for large or small networks. Our technicians will work with you to determine your operational needs and set up any number of PC-BSD desktops and servers. Our experts can also provide specialized support for your system administrators.

### Escalation Management

iXsystems is the all-around FreeBSD company that builds FreeBSD-certified servers and storage solutions, runs the FreeBSD Mall, and is the corporate sponsor of the PC-BSD Project. When the iXsystems Service Support Team encounters a confirmed bug, we can escalate the bug to the FreeBSD engineering team. We can also work with The FreeBSD Project to create and submit patches to the FreeBSD community for possible inclusion in the latest release.

For more information contact iXsystems at (408)943-4100 or visit our website at http://www.ixsystems.com/support/professional-bsd-support.html and fill out the Inquiry form. We will pair you up with an Account Management Service Professional that can assess your needs and create a custom FreeBSD or PC-BSD support plan for your organization!

POWERED BY FreeBSD

MAGAZINE BSD

**Editor in Chief:** Ewa Dudzic
*ewa.dudzic@bsdmag.org*
**Executive Editor:** Karolina Lesińska
*karolina.lesinska@bsdmag.org*
**Editor Assistant:** Ilona Lepieszka
*ilona.lepieszka@bsdmag.org*

**Director:** Ewa Dudzic
*ewa.dudzic@bsdmag.org*

**Art Director:** Agnieszka Marchocka
**DTP Technician:** Przemysław Banasiewicz
**Prepress technician:** Ireneusz Pogroszewski

**Contributing:** Patrick Pippen, Benny Siegert, Thorsten Glaser, Jan Stedehouder, David Gurvich, Eric Schnoebelen, Michele Cranmer, Carlos Neira, Henrik Lund Kramshøj, Marko Milenovic, Edd Barrett, Donald T. Hayford, Peter N. M. Hansteen

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

**National Sales Manager:** Ewa Dudzic
*ewa.dudzic@bsdmag.org*
**Marketing Director:** Ewa Dudzic
*ewa.dudzic@bsdmag.org*
**Executive Ad Consultant:**
Karolina Lesińska
*karolina.lesinska@bsdmag.org*
**Advertising Sales:** Karolina Lesińska
*karolina.lesinska@bsdmag.org*
Ilona Lepieszka
*ilona.lepieszka@bsdmag.org*

**Production Director:** Marta Kurpiewska

**Publisher :**
**Software Wydawnictwo Sp.z.o.o**
**02-682 Warszawa, Bokserska 1**
worldwide publishing

**Postal addres:**
Software Media LLC
1521 Concord Pike, Suite 301
Brandywine Executive Center
Wilmington, DE 19803
USA
tel: 1 917 338 36 31
*www.bsdmag.org*

Software-Wydawnictwo Sp zo.o. is looking for partners from all over the World. If you are interested in cooperation with us, please contact us by e-mail: *editors@bsdmag.org*
Print: 101 Studio, Printed in Poland

Distributed in the USA by: Source Interlink Fulfillment Division, 27500 Riverview Centre Boulevard, Suite 400, Bonita Springs, FL 34134 Tel: 239-949-4450.



The editors use automatic DTP system AUPUS

Mathematical formulas created by Design Science MathType™.
DVDs tested by AntiVirenKit GDATA Software Sp. z o.o.

Subscription orders can be sent to
*subscription@software.com.pl*
Customer Service 1 917 338 3631

## *Dear Readers,*

First of all I would like to thank all people who are helping out with making BSD mag a reliable source of information and interesting reading for all of you, for devoting their free time and a great involvement in this project. This magazine is not only a work of the editorial team, but the work of the community of BSD users and professionals.

Since it is almost Christmas Time, and we won't have another chance to „talk" before it comes, I would like to wish all of you such passion and devotion you prove being the part of our team for the next year

and all following years as well. Hopefully some of you will find this issue as your Christmas present;)

This issue focuses on NetBSD. Since all distros are equally important for us, we try to share our pages fairly between all BSD distributions. I am sure each of you will find something interesting in this issue, independently on the distibution you use. All in all, it is still BSD!

Patrick Pippen shows you step-by-step how to install NetBSD 4.0 in a form of an easy to use tutorial. Benny and Thorsten introduce MirBSD -the „peaceful" operating system. Jan look at live CD's based on BSD and David prepared a ver interesting comparison of Opensolaris, FreeBSD and OpenSuse.

In a how-to's section, our regular contributors, Eric and Michele show you how to start up the conference room or chat server for your Jabber service, and Carlos start his series about GDB with debugging process. The security section is prepared by Henrik, who goes through the steps needed to implement Pelude IDS on NetBSD, and Marko shows some of the best solutions for encrypting in BSD family operating systems. For those more advanced, Edd demonstrates how to make your own packages for OpenBSD. In the multimedia section Donald shows how to teach our Slugs to play music.

As usual, Federico made a great interview with NetBSD developers: Simon Burge, Antti Kantee and Greg Oster. Last but nor least, is a great review of Dru Lavigne's The Best of FreeBSD Basics by Peter N.M. Hansteen.

*I hope you will enjoy this issue!*
*Merry Christmans and happy New Year to all of you!*

*all the best*

*Karolina Lesińska*
*Executive Editor*

## what's new



## dvd contents



## get started



## how-to's



## security corner



## advanced



## mms



## interview



## review



## Eldorado, Maximus and GSA

Firewall module for HP ProCurve switches Remember you can run a highly secure firewall based on OpenBSD directly in a switch, just a module you plug into an empty slot of an HP ProCurve chassis. No extra rack space, no cables, no extra power, multiple gigabit performance, and an integration into the switch environment. And it is something that has been designed for production use in enterprise environments with support and a suitable product portfolio. And it might even help to replace other vendors you're unhappy with by a real BSD-based solution. vantronix released the first firewall for ProCurve switches with the Intelligent EDGE Firewall module series. The internal ethernet ports are directly connected to the switch backplane. The modules can be configured as a bridging or routing firewall between the switch ports and VLANs. The typical use of this product is as an Inter-VLAN firewall, internet gateway, but also to extend the layer 3 networking capabilities. And of course, redundancy can be provided with multiple modules thanks to CARP, pf, and the .vantronix CLI cluster management software. Powered by OpenBSD 4.4 The engine driving the .vantronix security appliances is based on OpenBSD. The operating system is providing the rock-solid networking platform and it's well-known security history with only two remote holes in more than 10 years. .vantronix is integrating it in a firewall operating system with a powerful CLI, management software, and optimization for enterprise networks. The product upgrade cycle is following the OpenBSD release cycle with a new major version every sixth month and security and errata fixes in between. The appliances allow easy handling of major and minor binary upgrades of the system. .vantronix is proudly announcing that the latest version is based on the new OpenBSD 4.4 „Trial of the BSD Knights" release. The appliances will benefit from the improved networking with pf, loadbalancing, routing, but also many other changes in system performance and stability. About .vantronix .vantronix | secure systems GmbH is a vendor of enterprise security appliances for critical environments; firewalls, loadbalancers, IPv6 gateways, VPNs, and Anti SPAM solutions. The company is based in Hannover, Germany, and works with international partners and customers mostly located in North America, Europe, and Asia-Pacific.

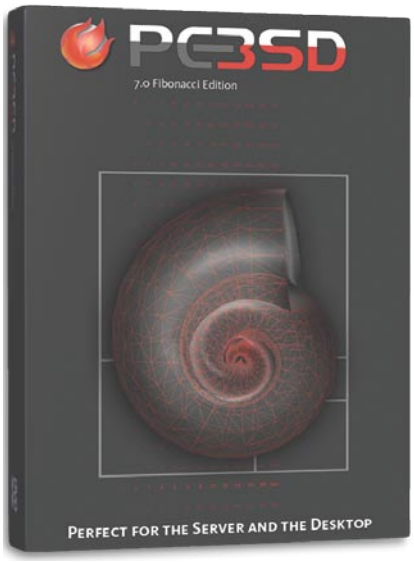
## PC-BSD Fibonacci Edition is here!

iXsystems has announced the release of PC-BSD 7.0 Fibonacci Edition. PC-BSD is a fully functional open source desktop operating system based on FreeBSD 7.0-STABLE. PC-BSD has a Push-Button Installer (PBI) wizard developed exclusively for PC-BSD that lets users download and install a wide range of available applications with a single-click graphical installer.

*PC-BSD 7.0 Fibonacci Edition is suitable for server as well as desktop use. The ULE scheduler greatly improves system performance and responsiveness, giving PC-BSD improved symmetric multi processing (SMP) performance over Linux. The Warden, a new administration utility, provides an easy to use framework for creating and managing FreeBSD Jails, with 3 control interfaces (GUI, dialog-based, and command-line). In addition, the improved GUI Installer allows users to set up a server with ZFS/UFS+J partitions, enabling setup of a server with ZFS in just a few minutes (including custom partitioning).*

*Highlights of the Fibonacci Edition include improved support and stability for Wine (a compatibility layer for running Windows programs), improved support with the Firewall Manager GUI, improved PBI installation support, and an improved PBI thumbnailer that displays embedded icons for large installers properly.*

*"Combining a usable desktop with the advancements of the FreeBSD 7.0 operating system results in a very fast and versatile OS", says Matt Olander, CTO of iXsystems. "Bringing PC-BSD up to FreeBSD 7.0 brings massive performance gains along with an easy to use graphical environment that makes server tools more widely available. The integration of the KDE 4.1 desktop window manager has brought a paradigm shift in productivity and useability."*



## NetBSD 5.0

NetBSD 5.0 includes many improvements and changes since NetBSD 4.0 was released in December 2007. The following briefly introduces some of the significant changes.

Rewritten threading based on 1:1 threading model replaced the Scheduler Activation model. New kernel synchronization primitives were introduced, and most of the core kernel was changed to use fine-grained locking or lock-less algorithms. A new scalable scheduler supporting real-time and time-sharing classes was added and support for POSIX real-time extensions, kernel preemption, processor-sets and thread affinity. The default memory allocator was replaced with 'jemalloc' designed to perform well in both single and multi-threaded processes. These changes dramatically improved performance and scalability on multiprocessor (SMP) systems.

Also support for POSIX asyn-chronous I/O and message queues was added, which al-lows to use modern and scalable APIs. Write support was added for the UDF file system. It can now read and write files and directories on CD-R, CD-RW, CD-MRW, DVD-R, DVD+R, DVD-RW, DVD+RW, DVD+MRW, (USB) flash media, and harddisc partitions.

The Automated Testing Framework (ATF) was added to NetBSD to easily define and run test cases for the operating system. It provides a common interface to easily run all the tests and reports the results in a consistent way. Many new tests have been added and most of the old regression tests were converted to use ATF.

Xen was improved with support for the i386 PAE extention to Xen3 domU (guest) domains and support for the EM64T/AMD64 architecture for Xen2 and Xen3 dom0 („host") and domU domains.

A new power management framework was added which includes suspend to RAM on x86 with ACPI-capable machines. The machine-independent framework also provides inter-driver messaging support for device drivers.

Metadata journaling for FFS file system was added. Contributed by Wasabi Systems, WAPBL (Write Ahead Physical Block Logging) provides rapid file system consistency checking after a system outage. And benchmarks show it faster than softdeps and significantly faster than default synchronous mounts. Support was added for a per-user /tmp and magic symlinks gained a real user ID magic string.

Bozohttpd was integrated in NetBSD based as „httpd". It provides a minimal HTTP daemon with indexing, ˜user translation, CGI, SSL, dynamic content encoding, basic authorization, and virtual hosting support.

An additional DHCP client was added: Roy Marples' dhcpcd. The minimal userland DHCP client daemon is much smaller and has been shown to use half of the memory while still supporting standard and advanced DHCP features.

The bootloader can now use an optional configuration file. It can be used to display menus to choose boot commands (such as selecting a kernel), define banner text, set timeouts, and select console devices.

The audit-pac kages tools were rewritten and added to the base. This is used to download the package vulnera-bility list and compare and report security iss-ues for installed packages.

The Runnable User-space Meta Program (rump) framework was added. It allows run-ning kernel code from userland by emulating portions of the kernel. It can be used for testing and debugging and supports various file systems using the Pass-to-Userspace Fra-mework File System development interface (puffs).

A Video4Linux2 compatible capture interface and support for USB video capture devices was added. Many Video4Linux2 applications are supported. The video device driver is divided into a high-level, machine independent layer and a low-level hardware dependent layer.

For more recent NetBSD news see:
*http://www.netbsd.org/changes/*.
For a complete list of changes for NetBSD 5.0 see:
*http://www.netbsd.org/changes/changes-5.0.html*.

by *NetBSD team*

## BSD Certification Group

More BSD Associate exams have been held in Augsburg, Sankt Augustin, San Francisco, Kiev, and Mont-de-Marsan. If you are interested in taking the BSD Associate exam just keep up to date with exam locations and events at: *https://register.bsdcertification.org//register/events*

Do you want to take the exam, but can't get to any of the upcoming events? If you know of other upcoming conferences in your area, let us know- we'll work with the conference organizers to get a BSDA exam scheduled.

Also, we are currently working on a college and university partnership program to make BSD Certification exams available on a regular basis. If you know of a university or college (including community colleges) you think might be interested, send us any contact information you have, and we will follow it up. Post your note in the „Contact Us" page on the BSD Certification Group website, *www.bsdcertification.org*, or send to *info@bsdcertification.org*.

# NetBSD 4.0

NetBSD is a free, secure, and highly portable Unix-like Open Source operating system available for many platforms, from 64-bit Opteron machines and desktop systems to handheld and embedded devices. Its clean design and advanced features make it excellent in both production and research environments, and it is user-supported with complete source. Many applications are easily available through pkgsrc, the NetBSD Packages Collection.

Major achievements in NetBSD 4.0 include support for version 3 of the Xen virtual machine monitor, Bluetooth, many new device drivers and embedded platforms based on ARM, PowerPC and MIPS CPUs. New network services include iSCSI target (server) code and an implementation of the Common Address Redundancy Protocol. Also, system security was further enhanced with restrictions of mprotect(2) to enforce W^X policies, the Kernel Authorization framework, and improvements of the Veriexec file integrity subsystem, which can be used to harden the system against trojan horses and virus attacks. Please read below for a list of changes in NetBSD 4.0.

NetBSD 4.0 runs on 54 different system architectures featuring 17 machine architectures across 17 distinct CPU families, and is being ported to more. The NetBSD 4.0 release contains complete binary releases for 51 different machine types, with the platforms amigappc, bebox and ews4800mips released in source form only. Complete source and binaries for NetBSD 4.0 are available for download at many sites around the world. A list of download sites providing FTP, AnonCVS, SUP, and other services may be found at http://www.NetBSD.org/mirrors/.

This is a partial list of changes between 3.0 and 4.0. The complete list of changes can be found in the CHANGES and CHANGES-4.0 files in the top level directory of the NetBSD 4.0 release tree. Some highlights include:

## Networking

- agr(4): new pseudo-device driver for link level aggregation.
- IPv6 support was extended with an RFC 3542-compliant API and added for gre(4) tunnels and the tun(4) device.
- An NDIS-wrapper was added to use Windows binary drivers on the i386 platform, see ndiscvt(8).
- The IPv4 source-address selection policy can be set from a number of algorithms. See „IPSRCSEL" in options(4) and in_getifa(9).
- Imported wpa_supplicant(8) and wpa_cli(8). Utilities to connect and handle aspects of 802.11 WPA networks.
- Imported hostapd(8). An authenticator for IEEE 802.11 networks.
- carp(4): imported Common Address Redundancy Protocol to allow multiple hosts to share a set of IP addresses for high availability / redundancy, from OpenBSD.
- ALTQ support for the PF packet filter.
- etherip(4): new EtherIP tunneling device. It's able to tunnel Ethernet traffic over IPv4 and IPv6 using the EtherIP protocol specified in RFC 3378.
- ftpd(8) can now run in standalone mode, instead of from inetd(8).
- tftp(1) now has support for multicast TFTP operation in open-loop mode, server is in progress.
- tcp(4): added support for RFC 3465 Appropriate Byte Counting (ABC) and Explicit Congestion Notification as defined in RFC 3168.

## File systems

- scan_ffs(8), scan_lfs(8): utilities to find FFSv1/v2 and LFS partitions to recover lost disklabels on disks and image files.
- tmpfs: added a new memory-based file system aimed at replacing mfs. Contrary to mfs, it is not based on a disk file system, so it is more efficient both in overall memory consumption and speed. See mount_tmpfs(8).
- Added UDF support for optical media and block devices, see mount_udf(8). Read-only for now.
- NFS export list handling was changed to be filesystem independent.

## Security

- The FAST_IPSEC IPsec implementation was extended to use hardware acceleration for IPv6, in addition to the hardware accelerated IPv4 that was available before. See fast_ipsec(4) for more information.
- mprotect(2) got restrictions to enforce W^X policies, from PaX. See options(4), sysctl(3), and paxctl(8).
- distribution or as third-party LKMs.

## This DVD contains the following items:

- NetBSD/i386 4.0 -- The DVD boots to the install program for NetBSD/4.0 on the i386 family.
- Source Code for NetBSD 4.0 (/source-4.0)
- A collection of binary packages for NetBSD/i386 4.0 as found on ftp.netbsd.org on September 9, 2008. (/packages). This does not contain all of the binary packages found on ftp.netbsd.org.

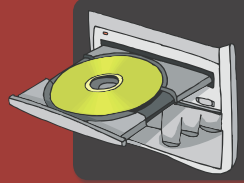
## See below on how to install packages to a running system.

- All of pkgsrc (Package Source) for the 2008Q2 branch. (/pkgsrc)
- CD images for NetBSD 4.0 (/iso-4.0)
  - i386cd-4.0.iso -- CD image for the install disk for NetBSD/i386.
  - amd64cd-4.0.iso -- CD image for the install disk for NetBSD/amd64.
  - i386pkg.iso -- CD image for the install with limited packages for NetBSD/i386.
- NetBSD-current as of September 7, 2008 (/current)
  - i386cd.iso -- CD image for NetBSD/i386 -current install.
  - amd64cd.iso -- CD image for NetBSD/amd64 -current install.
  - source -- the source for NetBSD-current

For more information visit *http://www.netbsd.org or ftp://ftp.netbsd.org*.

For more information on the binary packages visit *http://www.pkgsrc.org*.

There are more binary packages are available at *ftp://ftp.netbsd.org/pub/pkgsrc/packages*.

**If the DVD content cannot be accessed and the disc is not damaged, try to run it at least two DVD-ROMs.**

BSD MAGAZINE 1/2009

If you have encountered any problems with DVD, please write to: cd@software.com.pl

# NetBSD install

Patrick Pippen

NetBSD is one of the four major BSD systems available within the open source world today. It prides itself on being secure, and highly portable. This makes it a excellent multi-architecture system for uses as high-end servers to powerful desktops systems to handy handhelds and embedded devices.

Honestly, it is the only open source operating system with a portable package management system. It is designed to take advantage of the latest high end hardware systems from Alpha, Motorola PowerPC, Sparc, and PC computing platforms, yet it still maintains support for even those older computer systems. And it even supports more computer platforms than I wish to list in this article.

This is a concise description of NetBSD and since this article is about installation and configuration of NetBSD. Let's get started!

## Installation

NetBSD is not as difficult to install as you may think, it comes with an easy to use menu driven program called sysinst to make installation quick and simple. In reality it's not that difficult if you take the time to read through the NetBSD Guide and follow the instructions within it. After some time you'll be surprise how fast you can get it up and running.

### Getting the Media

NetBSD can be installed from a variety of media. Since this issue comes with a DVD of NetBSD 4.0, I will assume you are using installing from the DVD itself. If you do not have the DVD, NetBSD ISO images can be downloaded and burned to a CD from FTP, HTTP and even BitTorrent that are listed on the official website. Though I encourage you to purchase a CD as it is the project's main source of revenue.

### Basic system configuration

Log in using root as the user name and the password you set for root during the installation process.

```
NetBSD/i386 (Amnesiac) (ttyE0)
login: root
password:
We recommend creating a non-root account and using su(1)
for
root access.
#
```

Note: `su(1)`: ***http://netbsd.gw.com/cgi-bin/man-cgi?su+1+NetBSD-current***.

It good system management to not use root for your daily day-to-day tasks of browsing the web. So, lets create a regular user account for you to do such daily computing and make this account apart of the wheel group.

```
# useradd -m -G wheel username
```

After creating a regular user account for you to use. You need to set a password for this user as well. Note: ***Make sure that you change bitweiler to your desired user name instead***.

```
# passwd username
```

Next let's go ahead and setup system time.

```
# date 200809191610
```

Which sets the current date to August 19, 2008 4:10pm

Basic Network Configuration

NetBSD uses `/etc/rc.conf` for system configuration on system startup. Understanding this file is very important the `rc.conf (5)` manual page will give you a list of all available options that can be added to this file. The necessary information needed in your rc.conf file, the installation process should have all ready set some of the necessary values.

**Listing 1.** Getting an inital pkgsrc tree and setting up a caching DNS server. (note: after installation)

```
$ export CVSROOT="anoncvs@anoncvs.NetBSD.org:/cvsroot"
$ export CVS_RSH="ssh"
$ cd /usr
$ cvs checkout -r pkgsrc-2008Q2 -P pkgsrc
Now that you have it, available packages for NetBSD can be found in /usr/pkgsrc.
Let's install a few to give you a fill of it's simplistic usage.
 $ cd /usr/pkgsrc/net/maradns
 $ make install clean
 $ make clean-depends
Okay, I guess we should go ahead and set up a recursive(caching)DNS server for this machine as well.
This is the chroot directory maradns will run from.
$ mkdir -p /etc/maradns/logger
$ vi /etc/mararc
This is maradns configuration file, below are it's config options.
(Note: the documentation has more configuration options than are mention here.)
hide_disclaimer = "YES"
no_fingerprint = 1  # disable MaraDNS fingerprint
verbose_level = 3  # be verbose about query errors and log them.
ipv4_bind_addresses = "192.168.15.1, 127.0.0.1" # address to listen on
chroot_dir = "/etc/maradns"
recursive_acl = "192.168.15.0/24, 127.0.0.1" # addresses that can access this server.
upstream_servers["."] = "12.12.12.12"  # your ISP's name server
csv2 = {}
csv2["domain.com."] = "db.domain.com"  # our authoritative local zone file
debug_msg_level = 0  # so no info about maradns will be made public.
The default for debug_msg_level is 1 and in my opinion, it shouldn't give out information to
the public about itself when making queries. The default gives out maradns version number.
Consult the mararc man page for more variables that can be set.
Let's create our local zone file which will be in /etc/maradns directory.
vi /etc/maradns/db.domain.com
papa.%  192.168.15.100 # client machine
tango.% 192.168.15.120 # client machine
zulu.% 192.168.15.1  # our dns cache machine
The '%' will append domain.com to the above names when processing it's mara's shortcut for
the lazy at heart.To add MX records you just need to add to the zone file these lines for
each MX record serve.
domain.com.     MX  10  mail.domain.com.
mail.domain.com.        192.168.15.130
This sets up mail for domain.com being mailed to mail.domain.com which has the ip address of
192.168.15.130.  To be able to do reverse DNS lookups add to the zone file a PTR records.
Which would look like this for our example domain.
100.15.168.192.in-addr-arpa.    PTR     papa.domain.com.
120.15.168.192.in-addr-arpa.    PTR     tango.domain.com.
130.15.168.192.in-addr-arpa.    PTR     mail.domain.com.
1.15.168.192.in-addr-arpa.      PTR     zulu.domain.com.
To test it do: $ /usr/local/sbin/maradns
Switch to another termial or xterm to see if it running and do a netstat -an
(output edited for brevity)
Active Internet connections (including servers)
Proto Recv-Q Send-Q  Local Address          Foreign Address        State
udp        0      0  127.0.0.1.53           *.*
Look for this line, yes I see that the state isn't LISTEN udp services don't be in LISTEN
state.  To make this run on boot we need to edit the /etc/rc.local file and add this line.
/etc/rc.local
/usr/local/bin/duende /usr/local/sbin/maradns > /var/log/maradns.log 2>&1
The /var/log/maradns.log 2>&1 isn't really needed but I added it to catch any info that stdout
and stderr might give out that you might would like to see. Be sure to create the
/var/log/maradns.log file before rebooting.
$ touch /var/log/maradns.log
Administration
Contents and permissions of /etc/maradns
$ls -l /etc/maradns
total 10
-rw-r--r--  1 root  wheel    59 Feb 28 16:44 db.domain.com
drwxr-xr-x  2 root  wheel   512 Feb 19 08:23 logger
Next do some simple queries using askmara.
$ askmara Ngoogle.com.
$ askmara Nbsdnexus.com.
Maradns logs queries to /var/log/messages let's take a peek.(Again edited for brevity.)
$ more /var/log/messages|egrep dns
Jan  21 05:18:39 zulu /usr/local/sbin/maradns: Query from: 127.0.0.1 Ngoogle.com.
Jan  21 05:18:39 zulu /usr/local/sbin/maradns:  Log: Message received, processing
Jan  21 05:19:03 zulu /usr/local/sbin/maradns: Query from: 127.0.0.1 Nbsdforums.com.
Jan  21 05:19:03 zulu /usr/local/sbin/maradns:  Log: Message received, processing
If a weird crash or something unsuspected happens to your server check your
/var/log/maradns.log file which should be empty if everything is fine with your setup.
$ ls -l /var/log/maradns.log
-rw-r--r--  1 root  wheel  0 Mar  2 17:03 /var/log/maradns.log
```

To read the manual page on rc.conf.

```
# man rc.conf
```

The first modifications to your system should be to check make sure they exist within your /etc/rc.conf.

- Set "rc_configured=yes" (the installation process may have already did this.)
- Set "dhclient=yes"
- Set "hostname=yourhostname"
- If this computer is connected to a local network or a router. Set "defaultroute=192.168.1.1" for example. Note:***without the quotation marks though***.

From here you can edit the /etc/resolv.conf file.

You'll need to be able resolve host names and ip addresses of remote host systems to do so you just have access to a local or remote DNS server. Add it to this file using this syntax.

```
nameserver 164.253.3.75
```

## Mounting removable media

There's really nothing difficult about mounting a CD-ROM, floppy, or USB devices. It just takes reading the all so helpful mount manual page which even gives examples of how to use mount and it's options. Invoking mount by itself shows you want you the file systems that you have currently mounted.

```
# mkdir /cdrom
# mkdir /floppy
# mkdir /usb
```

Then add to your /etc/fstab file so that you can easily mount your removable devices at anytime.

```
/dev/cd0a /cdrom cd9660 ro,noauto 0 0
          /dev/fd0a /floppy msdos
rw,noauto 0 0
    /dev/sd0e /usb msdos rw,noauto 0 0
```

No need to reboot you can now mount your cdrom, floppy, an usb devices and start using it right away with these simple commands.

```
# mount /cdrom
# mount /floppy
# mount /usb
```

Just remember when your finished using these devices you have to unmount it or use eject to umount and eject the cdrom media to safely remove the cd.

```
# umount /cdrom
# umount /floppy
# umount /usb
(:optionally you can use the eject
command to umount and eject your cdrom
drive.)
# eject /dev/cd0a
```

Configuring the X server is easily done with the root user just running xf86cfg to configure X, but do not start X just yet.

```
    # xf86cfg
```

## Getting those extra packages

Even though NetBSD comes with plenty of tools and is usable with just the base system, you probably would like a working desktop system to really see what NetBSD is all about. And as I mentioned before, pkgsrc is the most portable package management system that I ever seen on any open source operating system. It currently works with FreeBSD, OpenBSD, DragonflyBSD, Mac OS X, Linux, Solaris, plus a few more operating systems. Pkgsrc originally came from FreeBSD's ports system and was developed for NetBSD only, then it gradually progressed to support more operating systems.

Okay, I know your ready to dive into this package management system, so let's get going.

Because binary packages aren't always available for different applications, it is best to get pkgsrc from a CVS repository . Using the sh or ksh shells we set it like so: see Listing 1. (Note: use setenv if your using the csh, "C shell".)

## Firewall configuration

Simply don't allow outside IP addresses to hit the server on port 53 UDP. Allow them to hit this server on ports between 15000 – 19095 UDP where the source port from remote server is 53 UDP. Allow UDP connections from your clients on the lan that use the server's cache to hit port 53 UDP.

Back to installing a couple more packages now. Go to /usr/pkgsrc/meta-pkgs/kde3 so you can have a familiar and comfortable desktop environment. No need to worry about dependency checking NetBSD's pkgsrc system takes care of all the worries for you and install everything that's needed.

```
$ cd /usr/pkgsrc/meta-pkgs/kde3
$ make install clean
$ make clean-depends
```

Afterwards you should add to your regular user account directory a .xinitrc file with the following contents.

```
        # su username
        # cd /home/username
 $ echo startkde > .xinitrc
# ln -s .xinitrc .xsession
```

All so make sure that the root user adds these line to your /etc/rc.conf file.

```
        xfs=yes
```

Also make sure to add this to your /etc/rc.local file, so kdm starts automatically on system startup.

$ echo /usr/X11r6/bin/kdm

Reboot and your NetBSD box is ready to rock hard. From here you can login through the kdm the X graphical login manager.

What I've always love about NetBSD, is that I don't have to buy a new computer every couples of years or throw away older computers.

I can keep the machines I've got and NetBSD works with it perfectly. I could not possibly cover everything in this article.

Now that you got your system up and running I suggest you read through the documentation. A Special thanks goes to the NetBSD & Pkgsrc

developers for making such a great open source operating system and package management tools.

## On the 'Net

- http://www.netbsd.org/docs/guide/en/
- http://www.netbsd.org/docs/pkgsrc/
- http://wiki.netbsd.se/Main_Page
- http://wiki.netbsd.se/Main_Page


01

## Language selection

To start the installation, boot your computer from the DVD. The installer will load the kernel and show you which devices were found and are supported.

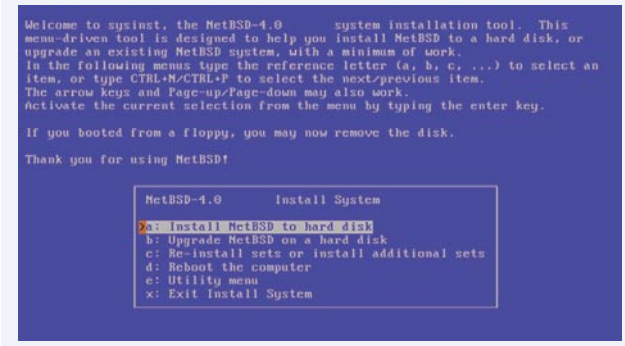
After that you'll find yourself within the NetBSD installation program sysinst.

To make a selection on the menu you can use the cursor keys or just press the letter displayed left of each choice and confirm your choice by pressing the Return (***Enter***) Key.


02

## Selection of Keyboard layout

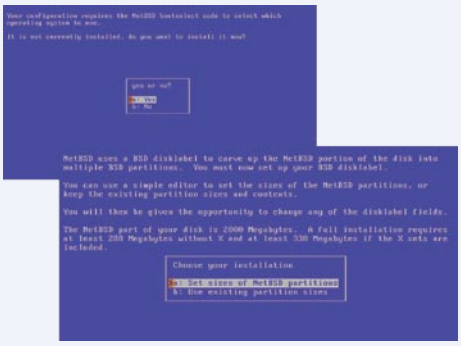
Start by selecting your language you prefer for the installation procedure naturally I choose US-English, next you will select your keyboard layout this will bring you to the main menu of the installation program.




03

## Choosing to install to a hard drive media

Since this is a fresh install we are going to choose to ***Install NetBSD to hard disk*** option on the menu. It will also warning you about backing up important data before taking this step. Seriously, you should do it, unless you don't care about the data on the hard disk. Which brings us to the next screen. To confirm installing the program to the hard disk select ***Yes***. Next, it shows a list of available hard drives we can install to. Since I only have one hard drive to install it only shows that disk named, `wd0`. If you have more than one disk it would be labeled wd1 as the second hard drive on your machine. If your using SCSI or external USB drives it will be labeled `sd0` as the first and sd1 as the second. Sysinst will ask if you want a full, minimal or custom installation. The default is full so we'll just go with that. NetBSD will now offer to install a boot selector to your hard drive so make sure you select yes to this option.



04

## Setting up the hard drive partitioning scheme

Now we can move on to creating the disklabel for this disk to create partitions or use the existing partition sizes, which NetBSD is okay with, so select ***use existing partition sizes***. If your experience tells you something different you can go ahead and set the partition sizes to your liking.


05

## Installing a bootblock

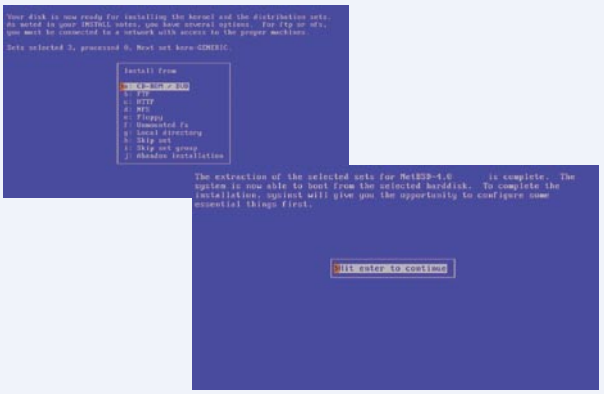
After defining the disklabel the menu prompts you to name your disk which I called ***NBSD***.

This our last chance to abort the installation before anything is written to hard drive. So let's continue by selecting yes. The next menu will ask us to select a bootblock, which the default ***Use BIOS Console*** is usually what we want.


06

## Install the base system

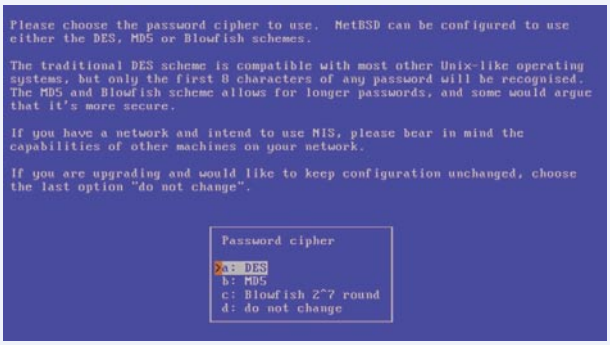
Give yourself a big pat on the shoulder, you have just completed the most difficult part of the installation. On to the second half of the installation process, which is extracting and installing operating system sets (kern, base, etc, comp....) Sysinst now asks which verbosity level you would like to use while installing the sets. Since it's your first time it is best to stick with the recommended verbosity level: ***Progress bar***. So we can see what is going on during the installation of the sets.




07

## Selecting the installation media

Now sysinst asks where it can find the installation media, since we are using the DVD media select the first option for the CD-ROM/DVD and press return on your keyboard. During this step sysinst will install all selected sets and create device nodes and will display a message letting you know that everything went well. Now hit enter to continue once we get to this screen:




08

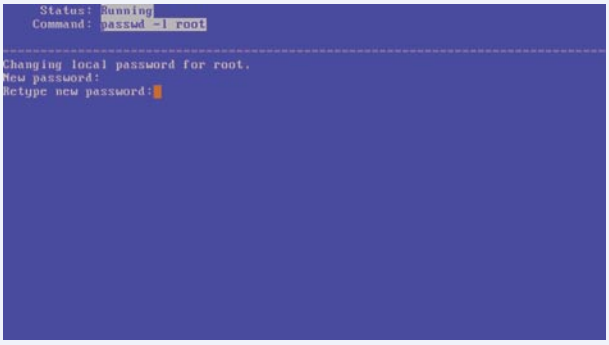
## Selection of system password encryption algorithm

Now lets get down to some of the basic system configuring, starting with setting up the timezone specific to your area. Using the cursor to scroll up and down the menu, then hitting enter to select your timezone, I chose US/Central for my timezone. The next thing you'll be asked is what encryption algorithm should be used for your password file. I recommend for you to use the Blowfish algorithm.


09

## Setting up the System manager password

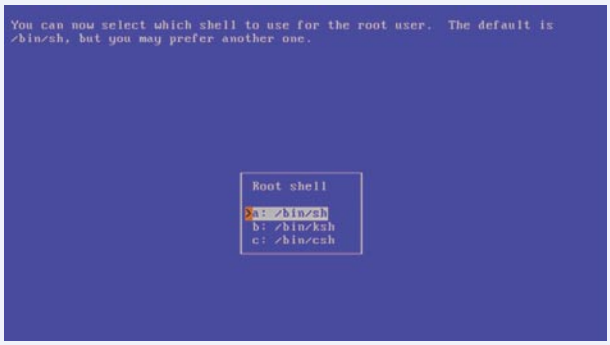
After choosing your encryption algorithm, it will ask you to set your root password. NetBSD does not start any services by default when booting up after installation.




10

## Choosing a password for the system manager

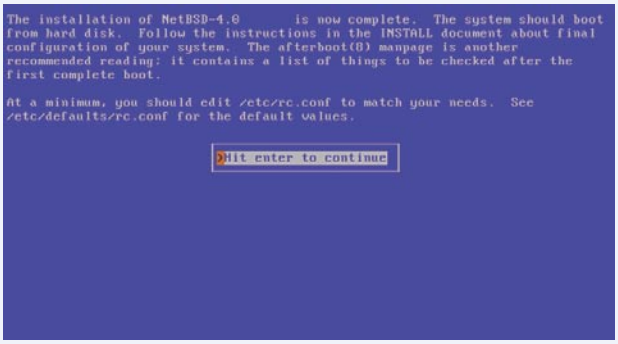
Still it is recommended that you go ahead and set your root password here for security reasons. So, enter it once then sysinst will ask you to re-enter it again. Next you are asked to choose a command shell. Traditionally BSD systems ship with the `csh` (`C shell`) as system manager shell. You can choose that one if you prefer. If you don't have any experience with Unix shells then select sh as your command interpreter, you can always change it later.




11

## Selection of a default command shell for system manage

I went ahead and choose ksh (Korn Shell) during this installation and you should as well, those more experienced with Unix shells are free to choose a root shell of your preference. After choosing your shell, you can now hit enter. Once you see this screen:




12

## Completing the installation

That's it! NetBSD is now installed and you will be able to log into it right after you remove the DVD and reboot your machine. Come on now that didn't take hours of time did it?

# MirOS BSD
## the peaceful operating system

Benny Siegert
Thorsten Glaser

Tired of manual pages that do not contain any information? Tired of having to run (hotplug) daemons with ever-breaking configuration files for your hardware to work? Well, we may have an OS for you.

The biggest advantage of the BSD operating systems is their separation between the base system and ports or packages for additional, third-party software. The base system contains the kernel, shell, tools, etc. These components are all tested with one another and just fit together. While modern GNU/Linux distributions pack a lot of complexity beneath their surface and try to hide it from the user, BSD systems try to leave you a chance to learn how the system works, by logically structuring the system internals. Even if you decide that BSD is not for you, this knowledge can help you in other OSes.

MirOS BSD is a secure computer operating system from the BSD family. It is a derivative of OpenBSD. A lot of code and ideas is taken from NetBSD and other sources. MirOS was started after some differences in opinion between Theo de Raadt, the OpenBSD project leader, and Thorsten Glaser, who is now the lead developer. The principal maintainer of MirPorts is Benny Siegert. There are several more persons working as contributors on the project.

The base system has been trimmed down. Seldomly used components like NIS, Kerberos, Bind and the BSD games have been removed. The latter two are installable as ports. Hence, the focus of MirOS BSD are small servers, appliances, routers, and developers' desktops.

## Using the DVD

The DVD contains a live CD as well as the installation files. Using the live CD allows you to run MirOS without any installation. It starts the IceWM window manager by default and includes a big number of already installed packages. For starting the live CD with X (the graphical user interface), the PC should have at least 128 MiB of RAM. For text mode only, 96 MiB are enough. Keep in mind that these requirements are only for the live CD, the installed system can run with less.

After booting your computer from the CD, a boot menu appears, asking for your choice. During the live CD boot, you will be asked to set a password for the user *live*, which is used for logging in. This is necessary because the live system can also be accessed over the network using `ssh(1)`.

## Before you install: partitioning

You should plan the organization of your hard disk now before you do the installation. The BSD operating systems don't use PC-style partition tables. Instead, they have their own scheme called a *disklabel*. Thus, you will normally create one `fdisk(8)` partition for MirOS which contains the disklabel and all the MirOS filesystems. This partition is then subdivided into so-called *slices* using the `disklabel(8)` editor. Partitions of other operating systems (e.g. `ext2fs`, `msdos`, or `ntfs` partitions) are usually automatically added to the disklabel. In fact, MirOS only uses the partition entry to find its disklabel and to show other operating systems that the space is used.

The MirOS partition needs to be big enough to hold all the file systems and the swap partition. At least 2 Gibibytes are recommended. It needs to be a primary or logical partition with type 0x27. If you are using a different operating system now and you are familiar with its partitioning tool (e.g. GNU fdisk), you can use it to create the partition before you start the installation. This way, you avoid having to learn a new partitioning tool and minimise the risk of data loss. In any case, do a full backup of your data NOW in case anything goes wrong during the installation or the change of the partition table.

In case you have only one big partition (for example, if you had only MS Windows(R) installed before), you need to either delete or shrink it. For the latter, tools such as fips, parted or Partition Magic (commercial) can be used.

In special cases, you can have more than one area on the disk for MirOS. However, you will have to calculate slice offsets yourself, so do this only if you know what you are doing.

In this case, the other partitions should be of a type otherwise unused in the system, such as 0xDB (CP/M-86).

The slices inside the disklabel are named using letters.

- `a` – This is always the root filesystem, i.e. the one from which you boot and which contains the kernel. For a full MirOS installation including XFree86(R), you will need at least 500 MiB of space in this partition – more if you want to install third-party applications using MirPorts or binary packages.
- `b` – This slice is always for swap space. As for its size, it used to be recommended to make it twice as big as the installed RAM, but on modern systems, you can use less (say 300 MiB) to save space. However, more swap space does not hurt, and it can be useful for high loads or compiling big software.
- `c` – As this slice represents the whole disk, you cannot change its size or put a filesystem on it.
- `i` – Auto-detected partitions from other operating systems are given names from i on.

## Starting the installation

If you want to install the system on your hard disk, choose the last option *Boot the MirOS installer* in the boot menu instead.

You will see lots of messages on a blue background scroll by. These are the normal kernel startup messages while your hardware is being probed and the appropriate drivers are loaded. Once the kernel has finished loading, you will be asked:

```
(I)nstall, (U)pgrade, or (S)hell?
```

Press the `I` key to start the installation procedure.

You will be greeted with a welcome message: see Listing 1.

Simply press Enter at this prompt. The next question is:

```
kbd(8) mapping? ('?' for list) [none]
```

If you want to use the default US keyboard table, press Enter. If not, enter the short code for your keyboard layout here. It is usually identical to your country code, for example `de` for Germany.

This is the point where you should stop if you do not really want to do the installation now (see Listing 2). If you are really sure you want to continue and if you have planned your disk layout (see above), then enter yes now to continue.

Enter the name of the hard disk you want to install MirOS on (Listing 3). The first IDE hard disk is `wd0`, while the first SCSI hard disk is `sd0`.

The next steps are partitioning with `fdisk(8)` and `disklabel(8)`. See the section above for advice. Enter the partition scheme you planned earlier here. If you want to use MirOS exclusively on the hard disk, say yes on the next question:

```
Do you want to use all of wd0 for
MirBSD? [no]
```

In most cases, you will say `no` here, so `fdisk(8)` will be started. You will get a prompt from fdisk that looks like this:

```
fdisk: 1>
```

The `print` command will show the current table. If you think you made a mistake, use the `exit` command to `quit` fdisk without saving any changes. `quit` saves the changes and exit. As an example of fdisk use, we will create one partition on an otherwise empty hard disk here. At the fdisk prompt, we edit the first entry in the partition table by entering `edit 0` (Listing 4).

It is very important that the first partition begins on head 1 and not on head 0 (i.e. at sector 63) to leave some space for the partition table. As you see, the prompt in fdisk is now marked with an asterisk. This means that the partition table was changed. Type `quit` now to save it and quit fdisk (Listing 5).

To see your current disklabel, use the `p` command. To add a slice in the disklabel editor, enter `a` followed by the slice name, for example `a a`. To create a slice, enter its offset (the starting point), size, and mount point. Offset and size values are in sectors, thus twice their size in kibibytes. Note that it is also possible to use a number

**Listing 1.** Installer welcome screen

```
Welcome to the MirOS BSD #10/i386 install program.

This program will help you install MirOS. At any prompt except
password prompts you can run a shell command by typing '!foo', or
escape to a shell by typing sometimes there is no default. At any
time you can exit this programme by pressing Control-C and then RE-
TURN, but quitting during an install can leave your system in an
inconsistent state.

Terminal type? [wsvtg]
```

**Listing 2.** Are you serious?

```
IS YOUR DATA BACKED UP? As with anything that modifies disk con-
tents, this program can cause SIGNIFICANT data loss.

It is often helpful to have the installation notes handy. For com-
plex disk configurations, relevant disk hardware manuals and a cal-
culator are useful.

Proceed with install? [no]
```

and a modifier, for example 256M for a partition of 256 MiB or 2G for a partition of 2 Gibibytes.

The default value for the offset is the beginning of free space, thus it is the right one if you add the partitions one after another. The default value for the size is the remaining space. Always leave the default when asked for the FS type. A very simple example with just two slices follows:

```
> a a
offset: [63]
size: [1023057] 896000
FS type: [4.2BSD]
mount point: [none] /
> a b
offset: [896063]
size: [127057]
FS type: [swap]
>
```

NOTE: If you are doing the installation on a *virgin* hard disk, you must use the `update` command to install a boot loader into the MBR (*Master Boot Record*) and initialise the magic number. Type `q` to quit and save your changes or `x` to quit without saving if you made a mistake and want to redo the disklabel or the partitioning.

After the disklabel is created, the new filesystems will be initialised (erased): see Listing 6

This is really your last chance to abort. To continue, enter `yes`.

## Initial network configuration

```
System hostname? (short form, e.g.
'foo')
```

After the creation of the filesystems, you will be asked for the host name of the system. This is the name that you give your computer, without the domain name. It should be unique on your local network. Many people use some kind of naming scheme for their machines, for example the last names of their favourite authors.

```
Configure the network? [yes]
```

If you say no here, you can skip the whole network configuration. This is useful if you do not have a local network or if you want to configure it by hand later.

```
Available interfaces are: ne3 plip0
irip0 irip1. Which one do you
wish to initialise? (or 'done') [ne3]
```

Now, you need to figure out the name of your local network interface. Under MirOS, network interfaces have a the name of their driver plus a number. `plip0`, `irip0` and `irip1` are `virtual` interfaces, thus the LAN interface in this example is 'ne3.

```
The media options for ne3 are
currently

media: Ethernet autoselect (10baseT)

Do you want to change the media
options? [no]
```

The default media type of *Ethernet autoselect* is sufficient in most cases. Say yes here if you want to fix the speed or the cable type manually. The latter might be necessary for cards with `10baseT` via `RJ-45` and `10base2` via coaxial cables, or if your switch is broken.

```
IPv4 address for ne3? (or 'none' or
'dhcp')
```

Enter the IPv4 address of the interface here. If you want to automatically configure the parameters using DHCP (*Dynamic Host Configuration Protocol*), enter `dhcp`. If you do not want to give the interface an IPv4 address, enter none. If you are not sure what to do, ask your network administrator or try `dhcp`.

```
Netmask? [255.255.255.0]
```

Enter the subnet mask here. In most cases, you can keep the default. Now you are brought back to the interface selector from before, where you can configure additional network interfaces if you want. Enter `done` after you finished configuring the last one.

```
DNS Domain name? (e.g. 'bar.com')
[my.domain]
```

**Listing 3.** Choose the hard disc

```
Cool! Let's get into it...

You will now initialise the disk(s) that MirBSD will use. To enable
all available security features you should configure the disk(s) to
allow the creation of separate filesystems for /, /tmp, /var, /usr,
and /home.

Available disks are: wd0. Which disk is the root disk (or 'done')
[wd0]
```

**Listing 4.** Partitioning the hard disc

```
Starting       Ending       LBA Info:

#: id    C   H  S -    C   H  S [       start:      size   ]
--------------------------------------------------------------------------
!0: 00    0   0  0 -    0   0  0 [           0:           0 ] unused
Partition id ('0' to disable)  [0 - FF]: [0] (? for help) 27
Do you wish to edit in CHS mode? [n] y
BIOS Starting cylinder [0 - 1014]: [0] 0
BIOS Starting head [0 - 15]: [0] 1
BIOS Starting sector [1 - 63]: [0] 1
BIOS Ending cylinder [0 - 1014]: [0] 1014
BIOS Ending head [0 - 15]: [0] 15
BIOS Ending sector [1 - 63]: [0] 63
fdisk:*1> flag 0
Partition 0 marked active.
fdisk:*1>
```

Enter the internet domain name of your computer here. If you do not have your own domain, then use something like `invalid`, but never enter a domain name that belongs to someone else.

```
DNS Nameserver? (IP address or 'none') [none]
```

Enter the name of your local domain name server here. If you used DHCP before, the nameserver has been configured automatically, and you can just leave the default. If you use a DSL router or something similar, enter the address your ISP gave you. If you do not want to use a nameserver now, enter `none`.

```
Edit hosts with ed? [no]
```

If you enter yes here, you can edit the `/etc/hosts` file with ed. This file contains a static table of host names and corresponding IP addresses. You will almost never need this.

## Installing the sets

```
Let's install the sets!
```

The installation sets are compressed archives that contain the different parts of MirOS itself.

In this case, the sets will be installed from the DVD.

**Listing 5.** Creating a disclabel

```
You will now create a MirBSD disklabel inside the MirBSD MBR parti-
tion. The disklabel defines how MirBSD splits up the MBR partition
(rather, the whole disk) into MirBSD slices in which filesystems
and swap space are created.

The offsets used in the disklabel are ABSOLUTE, i.e. relative to
the start of the disk, NOT the start of the MirBSD MBR partition.

If you have created a split space, i.e. one partition of type 27
and one or more partitions of type (e.g.) DB, use the command:


b<return>0<return>*<return> to enable using the entire disk for
MirBSD. Be sure to create slices mapping the filesystems of any
other operating systems in order to not overwrite them.

# Inside MBR partition 0: type 27 start 63 (0x3F) size 1023057
(0xF9C51).

Treating sectors 63-1023120 as the MirBSD portion of the disk. You
can use the 'b' command to change this.

Initial label editor (enter '?' for help at any prompt)

>
```

**Listing 6.** Last chance to cancel

```
The root filesystem will be mounted on wd0a. wd0b will be used for
swap space. No more disks to initialise.

MirBSD filesystems:

wd0a /

The next step DESTROYS all existing data on these partitions! Are
you really sure that you're ready to proceed? [no]
```

```
Location of sets? (cd disk ftp http
shttp nfs or 'done') [cd]
```

Just enter cd here and accept the defaults for the next questions. The next step is the install set selector. The available sets are:

- `base10.ngz` – As the name implies, this set contains the base files and directories. You want this.
- `bsd` – The operating system kernel. You need this.
- `bsd.rd` – A kernel image that boots into a "rescue system" that is contained within the image itself. A very handy tool for system recovery and later upgrades.
- `dev10.ngz` – The GNU Compiler Collection, binutils, system headers, static libraries and manual pages and associated documentation for developers. You will need this if you want to install additional software using the MirPorts Framework, or want to develop or compile yourself. For most normal systems, you will want this; however, in some cases (like when building a router), it might be wise not to install the compiler.
- `etc10.ngz` – This set installs the files in /etc as well as the httpd(8) manual and the default .profile files.
- `gnu10.ngz` – Contains those parts of the base system that are under less free licences, such as perl, sendmail, and lynx. You can choose to not install this set, but your system will not really be functional without.
- `pkgutl10.ngz` – The package tools, needed to install additional binary packages contained on the CD.
- `ports10.ngz` – This set contains the complete MirPorts framework (see below) used to install additional software from source.
- `xbase10.ngz` – Most of the files needed for XFree86(R), the graphical user interface.
- `xetc10.ngz` – Configuration files for XFree86(R).

If you are unsure which sets to install, just enter `all`. The installation of the sets is going to take a while. After it has finished, you will be asked a final set of questions.

```
Start sshd(8) by default? [yes]
```

`ssh` (*Secure Shell*) is a service that allows secure remote logins with encryption. It can be very handy for many uses, so say `yes` here.

```
Start ntpd(8) by default? [yes]
```

The ntp daemon synchronises your system clock from time servers over the internet or a local network. NFS and many other services rely on an exact time, so answer yes here if the machine has an internet connection.

```
Do you expect to run the X Window
System? [yes]
```

If you ever want to run the X Window System (the graphical user interface), answer yes to this question. This setting affects the machdep.allowaperture sysctl. If you respond negatively, you must enable it later in `/etc/sysctl.conf` in order to be able to run XFree86(R).

Next, you must select your local timezone, for example Europe/Berlin in Germany. Enter ? to get a list. Finally, the device nodes which reside in `/dev` are created by executing `MAKEDEV(8)`, and the bootloader is installed.

As the last step if the installation, a user account is created. This user will be used for logins after the reboot. In the default configuration, `sudo(1)` is used for administration tasks, root logins are not allowed.

At the end of the installation, the installer asks you to reboot. If you flagged the MirOS partition as bootable in the beginning, the computer will automatically boot into MirOS. If you use a boot manager, you will have to configure it accordingly to boot the partition (this is called a *chainloader* in LILO and GNU grub).

The default MirOS MBR also contains a boot manager. To boot the active partition, just press Return; select one of the four primary partitions with the keys 0 to 3, or boot from a floppy by pressing 5.

## Getting started

When you log in for the first time on your MirOS system, an e-mail will be waiting in your inbox, telling you about some of the next steps. The `afterboot(7)` manpage also contains some helpful information, type:

```
man afterboot
```

to read it. On many systems, XFree86 can be started without configuring it first, using the command `startx`. If this does not work or if you do want to change the settings, use the command:

```
sudo X -configure
```

Then edit the resulting `XF86Config.new` file and copy it to the right place using:

```
sudo cp XF86Config.new /etc/X11/
XF86Config
```

### Installing binary packages

If you installed the pkgutl10.ngz set, you can now directly install additional packages. First, mount the CD using:

```
sudo mount_cd9660 /dev/cd0c /mnt
```

Then change to the corresponding package directory `/mnt/packages/i386`. Information about a package can be obtained using the command:

```
pkg_info filename
```

To install a package and its dependencies, use the command:

```
pkg_add filename
```

The command `pkg_info` without arguments shows all packages installed on the system. Packages can be deleted using the command:

```
pkg_delete packagename
```

The name can be given with or without the version number.

## Using MirPorts

MirPorts allows to install third-party software from its source code, effectively creating your own binary packages. It is useful for software not contained on the DVD as a package, e.g. for licence reasons. If you installed the ports10.ngz set, all the necessary files have been unpacked in the directory `/usr/ports` on the hard drive. Before the first use, it has to be set up using the command:

```
cd /usr/ports ; make setup
```

This command installs the package tools and configures the MirPorts infrastructure. The ports themselves are in subdirectories, sorted by category (e.g. lang/python). Each of these directories contains a Makefile with the *recipe* for installation. The different targets supported by these Makefiles are explained in the `bsd.port.mk(5)` manpage. Attention, the make command for MirPorts (except for the command above) is called `mmake`.

Just executing `mmake` install in a port directory will download the source code, compile it, create a binary package and install it. Dependencies are automatically installed when necessary. Some ports exist in different ”flavours”, e.g. with or without X support. To show the available flavours for a port, execute the command:

```
mmake show=FLAVOURS
```

To select a specific flavour or a combination, execute a command like the following:

```
env FLAVOUR="perl no_x11" mmake
install
```

## About the Author

**Thorsten Glaser** initiated the MirOS Project in 2002, working on everything he is capable of and more. He admits being a DOS/GW-BASIC fan, finished an IT apprenticeship with best grades, decided university is nothing for him, and is currently working self-employed for a software/sysadmin company in Switzerland. He sometimes wishes he'd learned a real job instead of CS, and resides near Bonn, Germany, where the project idea was born.

**Benny Siegert** has been a MirOS developer for five years, working mostly on MirPorts. He lives in France and is currently doing his Ph.D. in chemistry.


# Trusted Proactive Security

The **.vantronix** Security Appliances
will take care of your network.

Highly secure high-end **firewalls**
developed for critical operation
with high requirements for
security, availability,
trusted systems and
comprehensive support.

Powered by

OpenBSD

The .vantronix | Intelligent EDGE Firewall
module for HP **ProCurve** chassis switches.

Firewall • Loadbalancer • IPv6 Gateway • VPN • Anti SPAM • vantronix.com/products/

Sign up as a reseller at
http://www.vantronix.com/partner/program/

Business Partner hp invent

.vantronix is a member of the **DSPP**
global alliance partnering program.

.vantronix®
secure systems

# BSD live

## cd's – an entry level acquaintance?

Jan Stedehouder

Linux used to be difficult to install and maintain for novice users. It required some skill to setup your partitions and go through the installation process. Of course, it was a challenge that attracted some, but also put off a lot of others that simply wanted 'to try it out'.

Add hardware issues to the mix and most interested users safely stuck to Windows. And then there was Knoppix, one of the first live CD's. Knoppix was (and is) Debian-based (arguably not the easiest of Linux distributions). It came with an unparalleled hardware detection and booted into a fully functional and usable graphical desktop environment. The live cd came loaded with KDE applications. Knoppix played an important role in promoting Linux as a viable desktop alternative.

The live cd has become all but the standard in Linux. The most popular Linux distributions offer a live desktop as a stage in the installation process or at least make a live cd/dvd available for testing purposes. In this article we will look into live cd's based on BSD. Which cd's are available and which live-BSD's exist that might point a novice BSD-user, albeit with some Linux experience, in the right direction?

### How to find BSD live cd's?

To try out BSD live cd's we first have to find them. For this various search engines were used with phrases like live cd BSD, *FreeBSD live*, *NetBSD live*, *OpenBSD live* and *Dragonfly live*. It revealed various attempts to offer live cd's, but only a few have sustained the test of time. Some projects were greeted with enthusiasm, but seem have turned defunct after the first few releases and updates. In some cases it isn't even clear whether the project is alive or not, or whether it is somewhere between releases.

#### Along the trail of history

One example is OliveBSD. Distrowatch paid attention to it in Distrowatch Weekly (issue 140, 27 February 2007) and was referred to alongside FreeSBIE, Frenzy and Anonym.OS. The review was not completely flattering as the author struggled with the rough edges of this OpenBSD 3.8-based disk. The idea was good however and the use of the IceWM Window manager was original enough. Sadly, OliveBSD wasn't maintained and updated.

Anonym.OS, also mentioned in the article, is another example. The concept, provide a live disk for secure and anonymous web browsing and computer use, was compelling enough for Wired to pay attention to it (*http://www.wired.com/science/discoveries/news/2006/01/70017*). It made use of Tor, a network for anonymous web surfing, to give an extra layer of security. Anonym.OS was based on the security-focused OpenBSD 3.8, thus it had a solid foundation to begin with. The disk was created by kaos.theory/security.research and there were plans to extend the scope of the project.

#### The Wired article said:

*But kaos.theory members say Anonym.OS is just the first step in making anonymity widely available. Future versions, they say, may run on a USB key chain. Additionally, they plan to implement Enigmail to allow encrypted e-mail for Thunderbird and Gaim Off-the-Record, which allows users to use instant messaging without their logs being tied to them.*

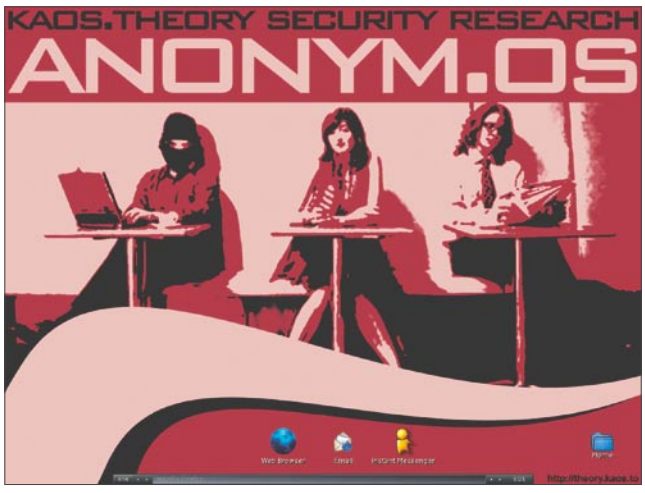



**Figure 1.** The Anonym.OS did have a nice appeal

The most recent release is dated January 14, 2006 and the original developers seem to have moved on to other projects. Anonym.OS is slowly drifting back into the anonymity history provides (Figure 1). FreeBSD Live still tops the list of the search engines, but it hasn't been active since 2003. This live cd was created by the Brazilian FreeBSD User Group, who released the scripts to build the disk in 2002. The next resource was The LiveCD List (*http://www.livecdlist.com/*) which has a separate entry for BSD disks. This list mentions 11 BSD-based disks, apart from the hundreds of Linux-based disks:

- FreeSBIE
- FreeBSD Live CD
- m0n0wall
- Frenzy
- Fuglta
- LiveBSD
- NetBoz
- Anonym.OS
- Ging
- NewBIE
- XORP Live CD

The problem with lists like The LiveCD is they aren't completely up to day. FreeBSD Live, LiveBSD, NetBoz and Anonym.OS are no longer active. The status of Ging – Debian with a FreeBSD kernel- is unclear. NewBIE seems to be replaced by a Linux-based live cd. XORP, the eXtensible Open Router Project, is still active, but one is hard pressed to find a clear reference to BSD on the project's website.

### The past and the present: an overview of BSD live cd's

Casting a wider net and digging somewhat deeper resulted in a list of past and present projects that offered BSD live cd's. The table is a reflections of our search. No doubt, it is far from complete, but it appears to be most complete list so far. (see document Table 1 of live distributions)

## Live cd's with a desktop focus

The main focus of this article is on desktop oriented BSD disks. This means that projects like FreeNAS, pfSense and m0n0wall (and perhaps also XORP) fall outside of our scope. They provide network-oriented services. Frenzy was not tested further, since it is aiming to be more sysadmin-oriented. The result was a somewhat shorter list of disks that might have the potential to introduce new users into the benefits of BSD:

- BSD Anywhere
- DragonFly Live
- Fugulta
- OpenBSD Live
- MarBSD
- NetBSD Live
- FreeSBIE
- DesktopBSD
- RoFreeSBIE

We will briefly discuss them here.

### BSDanywhere

BSDanywhere is still under development with a beta 2 version being released in July 2008. The aim is to release a final version on September 24 during the OpenExpo 2008 Zurich/Winterthur. BSDanywhere is based on OpenBSD 4.3. Development began in May 2008 and Stephan Rickauer, the developer, keeps a good pace in releasing intermediate versions.

**Tabela 1.** BSD live cd's

| Name release | Latest | Base | URL | Status |
|---|---|---|---|---|
| LiveBSD (1) | Unknown | Unknown | Unknown | Inactive(2) |
| NetBoz | 2003 (?) | FreeBSD | http://www.netboz.net/ | Inactive |
| NewBIE | Unknown | NetBSD (originally) | http://www.fosstools.org/ | Inactive |
| FreeBSD Live | November 2003 | FreeBSD | http://livecd.sourceforge.net/ | Inactive |
| Ging | 2005 | Debian/ FreeBSD kernel | http://glibc-bsd.alioth.debian.org/ging/ | Unknown |
| Anonym.OS | January 2006 | OpenBSD | http://sourceforge.net/projects/anonym-os/ | Inactive |
| FreeSBIE | February 2007 | FreeBSD | http://www.freesbie.org/ | Quiet |
| NetBSD Live | September 2007 | NetBSD | ftp://mirror.planetunix.net/pub/NetBSD/iso/livecd/netbsd-live-2007.iso | Quiet |
| RoFreeSBIE | November 2007 | FreeBSD | http://www.rofreesbie.org/ | Quiet |
| DesktopBSD | January 2008 | FreeBSD | http://www.desktopbsd.net | Active |
| TrueBSD | May 2008 | FreeBSD | http://www.truebsd.org | Active |
| OpenBSD Live | June 2008 | OpenBSD | http://jggimi.homeip.net/ | Active |
| BSDanywhere | July 2008 | OpenBSD | http://bsdanywhere.org | Active |
| Frenzy | July 2008 | FreeBSD | http://frenzy.org.ua/eng/ | Active |
| pfsense | July 2008 | FreeBSD | http://www.pfsense.org | Active |
| XORP | July 2008 | Unknown (4) | http://www.xorp.org | Active |
| FreeNAS | August 2008 | FreeBSD | http://www.freenas.org/ | Active |
| Fugulta | August 2008 | OpenBSD | http://kaw.ath.cx/openbsd/?en/LiveCD | Active |
| m0n0wall | August 2008 | FreeBSD | http://m0n0.ch/wall/ | Active |
| DragonFly Live | Recent (3) | DragonFly | http://www.lolaluci.se/gsoc/index.html | Active |
| MarBSD | Recent | OpenBSD | http://openbsd.maroufi.net | Active |

(1) There are reference to the LiveBSD project, though no further information is available
(2) The status *Inactive* means that the project is no longer active. There is no maintenance and no new release scheduled. The status *Quiet* means that the project appears to be inactive with very limited recent activity. It is unclear whether a new release is forthcoming.
(3) The release date *Recent* means that the release is based on a recent version of the the *mother BSD*. However, the available information is insufficient to pinpoint a more exact release date.
(4) The XORP website gives no clear indication on which BSD it is based.

The live desktop comes with the Enlightenment graphical desktop environment and some user-oriented software like Firefox, Thunderbird, The GIMP, Abiword and Audacious. Enlightenment provides an attractive and responsive desktop. When looking at novice users there are two downsides to BSDanywhere. It doesn't boot into a graphical environment, which needs to be launched from the command line. Installing BSDanywhere on the hard drive requires using 'bsd.rd' from the command line and pointing the installer to the OpenBSD file sets (Figure 2).

## Dragonfly Live

The DragonFly LiveDVD is to be considered *experimental*, but that shouldn't stop anyone from trying it out. The DVD boots into a graphical desktop environment (Fluxbox) with a small set of applications (*emacs*, *vim*, *pidgin*, *firefox 3*, *xpdf*, *xchat*).

A desktop icon provides easy access to the wizard that guides the installation on a hard drive It launches two terminal windows, one with the text-based installer and one with the log of the various steps. The screens for each step contain enough information to help novice users understand what is happening and what is required of them. It will be interesting to see how this project matures into a full and stable DragonFly Live desktop (Figure 3).

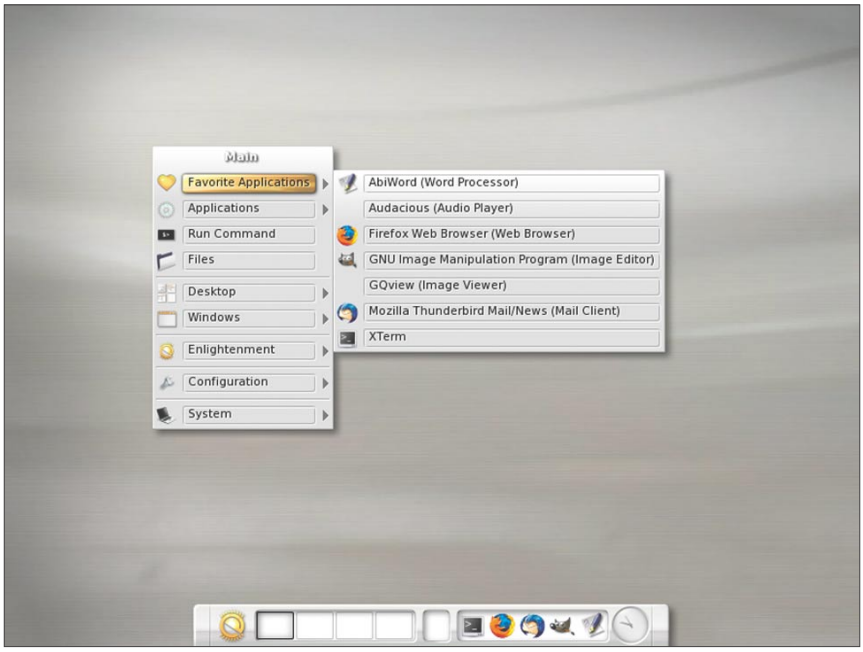



**Figure 2.** BSDanywhere combines OpenBSD with the Enlightenment desktop

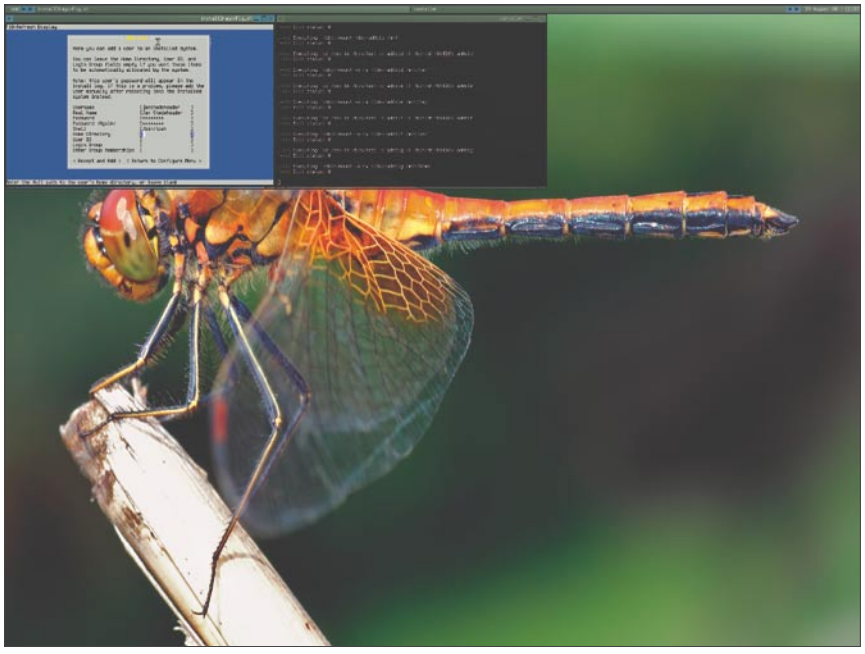


**Figure 3.** The DragonFly installer is easy to use and provides feedback for more experienced users

## Fugulta

Fugulta is based on OpenBSD and targets especially Japanese users. Fortunately, the website contains some English translation which makes it easier to understand the purpose of the project. As the site says, Fugu stands for *Blowfish* and Ita can mean both *Disk* and *Cook*.

Fugulta offers the user the IceWM Window Manager and applications like Emacs, the w3m text-based webbrowser and MPlayer. The choice of applications makes this a live disk for more experienced users and less so for novice users.

## OpenBSD Live

The OpenBSD Live project offers the widest choice of live disks. There are five different flavors:

- Basic LiveCD
- FluxBox LiveCD
- XFCE LiveCD
- KDE LiveDVD
- GNOME LiveDVD

The Basic LiveCD does have a graphical interface, though very lightweight with X and the Fvwm and cwn Window Managers. It also has no third party applications. The other flavors offer progressively *fatter* desktop enviroments.

Both the KDE and GNOME LiveDVD's could be interesting options for novice users, as most Linux users will be acquainted with either one of these desktops. The downside is that, although it does provide the OpenBSD file sets, it isn't easy to install on a hard drive for them. On the plus side, the OpenBSD Live website is one of the few that gives more than the basic information to it's visitors (Figure 4).

## MarBSD

René Maroufi is responsible for MarBSD. His work encompasses three different versions:

- MarBSD-light without a graphical interface and a limited set of console-based programs;
- MarBSD-serial which is similar to MarBSD-light, but uses a serial

console instead of a mouse and keyboard; and
- MarBSD-X, a basic OpenBSD-based system with a choice of the fvwm and the cwm Window managers.

Like the OpenBSD Live website, MarBSD's site provides quite a lot of information, mostly in German. One of the articles explains OpenBSD to Linux users and another one is very instructive for those who want to build there own OpenBSD-based live cd.

As an aside, the various builders of OpenBSD-based live cd's refer to Kevin Lo's article *Building an OpenBSD Live CD* on the OnLamp website (*http://www.onlamp.com/pub/a/bsd/2005/07/14/openbsd_live.html*). Another (more recent) resource is Andreas Bihlmaier's article on the OpenBSD wiki (*http://www.openbsd-wiki.org/index.php?title=LiveCD*).

### NetBSD Live! 2007

The NetBSD Live! 2007 cd is by far the fastest of the pack. The text-based wizard at boot requires the user to fill in some basic questions about language, keyboard, timezone and DHCP. After that, the login screen (KDM) bleeps into existence with the option to login as root or user.

The desktop is fully stocked with KDE 3.4.5 and accompanying applications. The team threw in extra applications like Abiword, The GIMP, Dia, Inkscape, Firefox, XMMS and a few games. It is a responsive disk. Please forgive the comparison, but NetBSD Live! 2007 had the most Knoppix-like experience.

The downside is, also here, that there is no clear possibility to install it on a hard drive. On the plus side, it can be used as a rescue disk for a wide range of systems. The inclusion of tools to access ext2/ext3-, Fat-, Macintosh HFS- and NTFS- file systems makes it easier to access partitions on systems that have problems (Figure 5).

### First conclusions

So far we have discussed live disks based on OpenBSD, NetBSD and DragonFly. The various OpenBSD-based live cd's range from the most basic systems to full-blown graphical environmens for the end-user. Sadly, only DragonFly Live is easy to install on a a hard drive, but it is still experimental and we will have to see how it matures over time.

## FreeBSD-based live disks

In the final part of this article we will focus on three FreeBSD-based live disks: FreeSBIE, DesktopBSD and RoFreeSBIE. As we will see in a moment, the latter can be seen as a fusion of the former two.

### FreeSBIE

FreeSBIE can be considered the continuation of the FreeBSD live project and was first released in 2003. The most recent release (2.0.1) is already a year and a half old and is based on FreeBSD 6.2. FreeSBIE uses the Xfce desktop environment with Fluxbox as an alternative.

The set of applications is relatively small, but complete. Abiword and GNumeric are available for office activities. Firefox, Thunderbird, xchat and GAIM take care of your connection to the outside world. The GIMP, Inkscape and GQView make up the graphics department and Beep Media Player, BMPx and MPlayer are there for your multimedia activities. With each live start you'll also get to see a new desktop background.

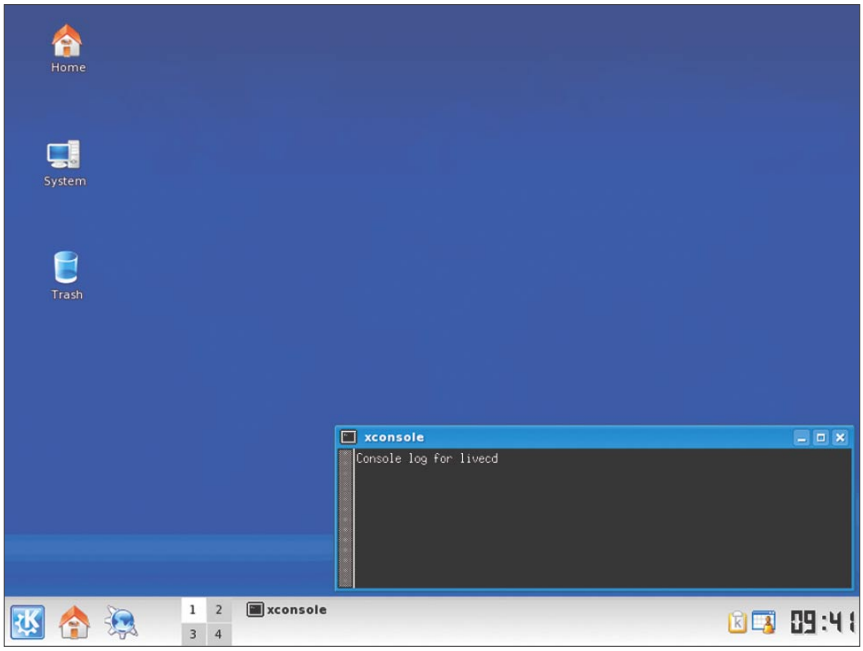



**Figure 4.** OpenBSD Live comes in various flavors, the KDE desktop being one of them

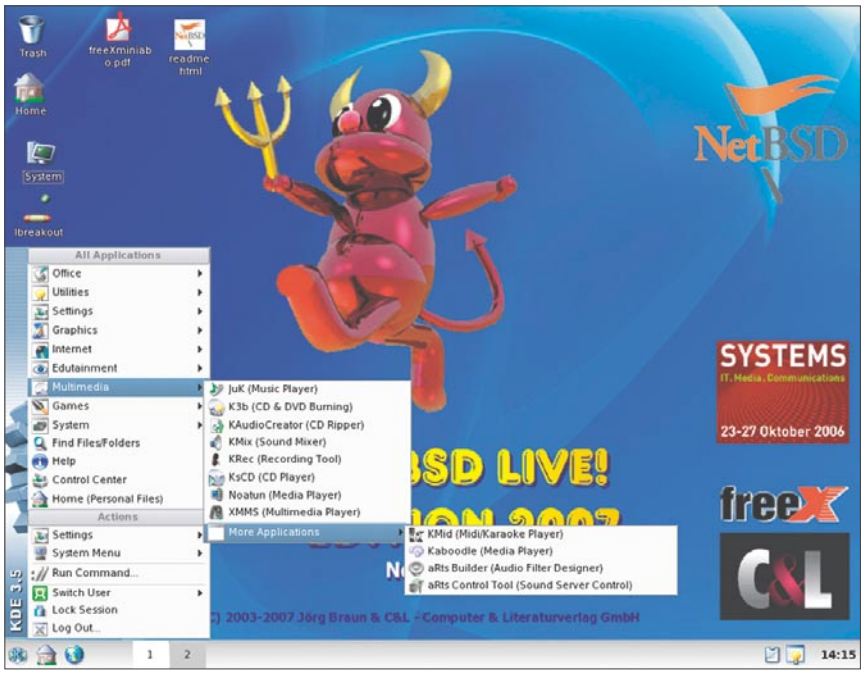



**Figure 5.** NetBSD Live! 2007 is one of the fastest and complete live disks

The live cd lacks a hard disk installer. Distrowatch Weekly (issue 186, 22 January 2007) had an interview with Matteo Rondato about FreeSBIE 2.0 that was released at the time. Rondato explained in the interview that the toolkit for FreeSBIE was completely rewritten and the hard disk installer wasn't ported at the time. It wasn't high on his list of priorities, since he prefers users to take FreeBSD proper if they want it installed on their systems (Figure 6).

## DesktopBSD

We discussed DesktopBSD extensively in BSD Magazine no. 1. Together with PC-BSD it has a sharp focus on the novice desktop user. The DesktopBSD tools make it easier, for instance, to manage software, take care of the network connections and mount partitions and devices on your system.

The KDE desktop environment has an attractive look and feel. With a 1.5 GB ISO image there is room for extra applications. Thus you can find *OpenOffice.org*, Firefox and Thunderbird apart from a large collection of KDE-based applications (Figure 7).

## RoFreeSBIE

RoFreeSBIE is an adaptation of - unsurprisingly- FreeSBIE. The most recent release followed FreeSBIE by eight months. This extra time was used to create a larger disk and more polished live desktop.

Where FreeSBIE uses the Xfce desktop as a default, RoFreeSBIE selected KDE 3.5. The Xfce desktop is available on the live disk, but definitely lacks the finish of either it's own KDE desktop or FreeSBIE's Xfce desktop (Figure 8).

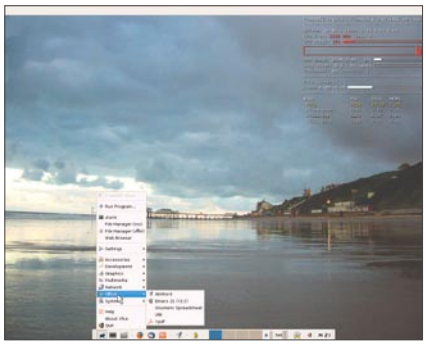


**Figure 6.** FreeSBIE uses the Xfce desktop environment

The KDE menu is changed into a very tidy and accessible menu tree. With Knoppix you can get overwhelmed because of a very crowded menu due to the large set of applications. RoFreeSBIE slashed the main entries back to but a few:

- Software, the applications
- RoFreeSBIE tools,
- DesktopBSD tools,
- Utilities
- Computer operations, like logging off and rebooting,
- Control Center (for KDE settings)
- etcetera

Under RoFreeSBIE tools we find a collection of interesting programs. The entry Configuration shows tools to setup your network, the firewall, manage users, get your TV card and wifi connections up and running. The entry Antivirus is one you won't find often on either a Linux or BSD-disk, but both ClamAV and F-Prot are there. Ubuntu users will be interested in the Add Remove Programs entry, but the *FreeBSD Ports Browser* (FPB) is unlike their tool to install software the easy way. Still, FPB is nice graphical front-end to either installing the port or the package of the software you are looking for.

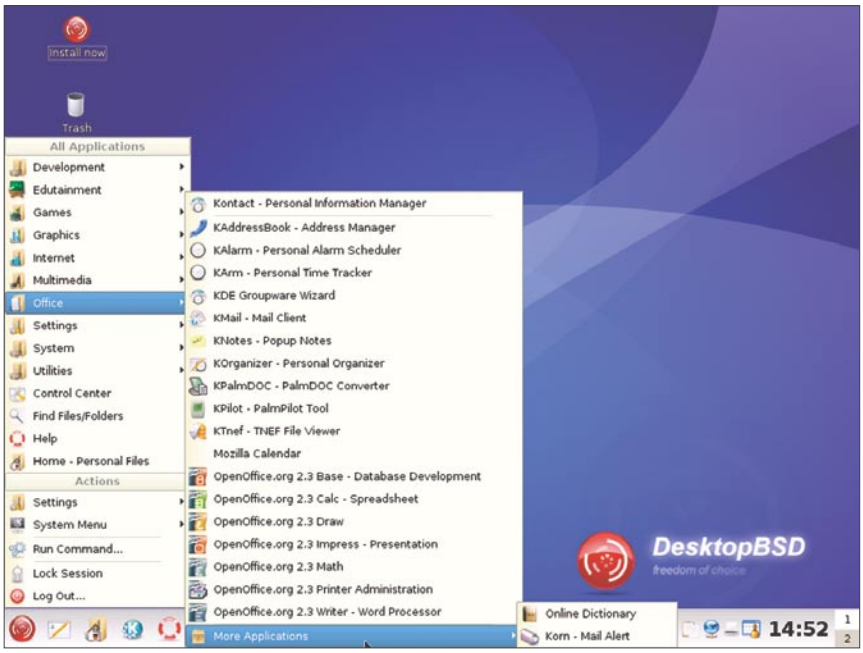



**Figure 7.** DesktopBSD offers a wide collection of software

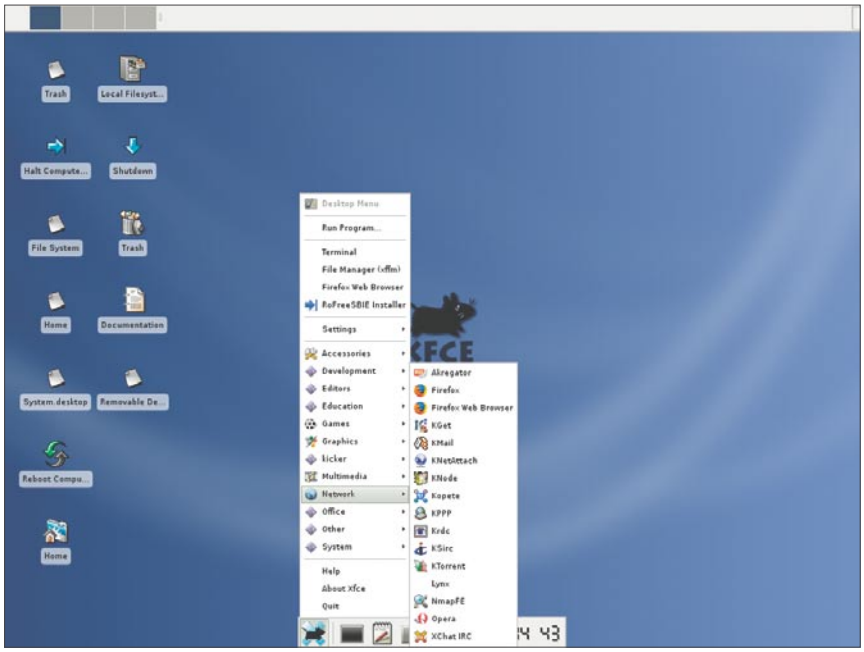



**Figure 8.** RoFreeSBIE's Xfce desktop won't win it many prices....

The main entry DesktopBSD tools shows that RoFreeSBIE incorporated the work of DesktopBSD in it's release as well. The entry Software Management launches the Package Manager, an alternative tot FPB. We have two graphical front ends for the same function.

RoFreeSBIE does that with other applications as well. You get three web browsers (Firefox, Opera and Konquerer), two office suits ( KOffice and *OpenOffice.org*), two IM clients (Pidgin and Kopete). Overall, it provides an extensive look into major applications.

It is possible to install RoFreeSBIE on your hard drive Under *RoFreeSBIE Tools -> Installations* tools we find the RoFreeSBIE installer. The installer is a graphical wizard guiding you through the steps of preparing your hard drive and actually installing the files. Once installed you'll find that root still doesn't need a password (Figure 10).

## Conclusions

The main question of this article was whether there were live cd's based on BSD that could lower the barrier for Linux and Windows users, so they can get acquainted with BSD in an easier way. Of course, one might wonder if providing a complete and functional graphical desktop environment with familiar applications and an easy installer is suitable as an introduction to BSD. Once loaded, it is hard to distinguish BSD live disks from it's Linux-based counterparts. But, if anything, they do show that BSD is well-suited for most day-to-day tasks. The benefits and strength of the underlying operating systems need to be explained later.

Looking at the various projects the conclusions is that most are not really focused on end-users, let alone novice desktop users. The majority caters to fellow experienced BSD. That might also explain why most project websites provide minimal information or documentation and why easy hard disk installers are absent, at least for those who lack the needed skills and experience. Your BSD friends already know how to care of themselves.

The development of Linux is sometimes frantic, to point of sometimes sacrificing stability for novelty. Compared to that, the development of the BSD-based live disks is almost glacial. When we look at desktop-oriented live disks, we notice that most of them were released in 2007, sometimes early 2007. There are some hints that the FreeBSD-based disks are moving towards the 7.0-RELEASE. The FreeSBIE mailing list briefly refers to work being done on it in November 2007, but no progress report since. DesktopBSD has released an early snapshot in March 2008. Of course, the choice between a stable and well-developed KDE 3 desktop and the innovative, sometimes controversial KDE 4 desktop, complicates matters somewhat.

NetBSD Live! 2007, RoFreeSBIE and DesktopBSD offer the best experience for novice users who want to get a taste of a BSD-based desktop. The work that is being done on BSDanywhere and DragonFly Live could provide interesting alternatives in the near future. For BSDanywhere I do hope that an easy to use hard disk installer will be added to the compelling combination of OpenBSD and the Enlightenment desktop.

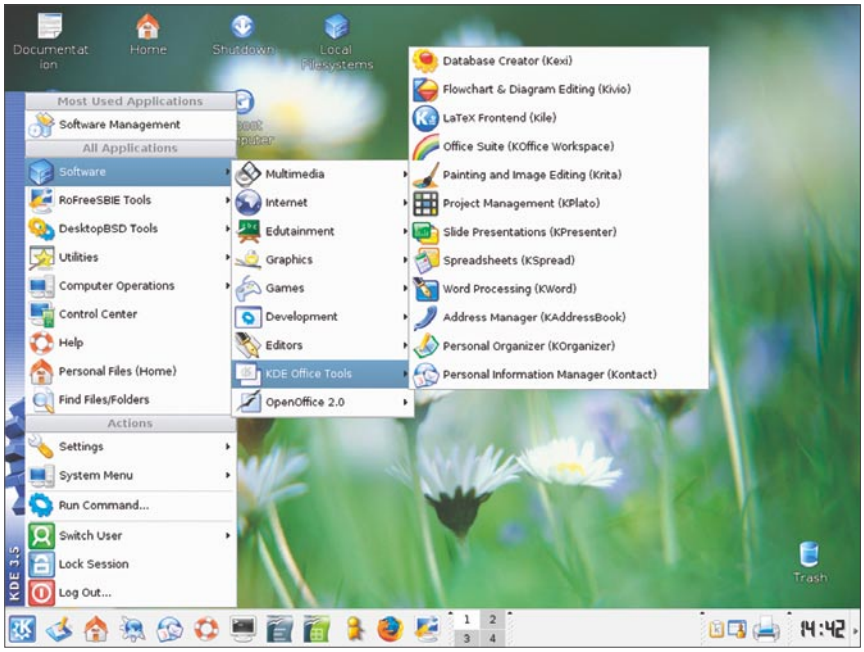



Figure 9. ... though the polish on the KDE desktop could be a source of inspiration for others

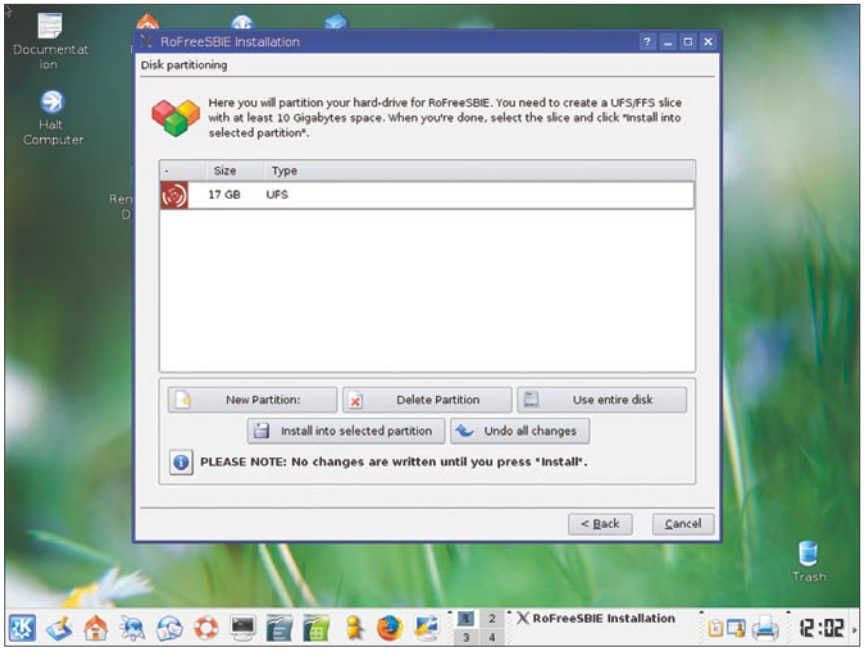



Figure 10. The RoFreeSBIE installer should be no problem for those who know DesktopBSD

# How it works?

## Opensolaris, FreeBSD, OpenSuSe

David Gurvich

This article is a comparison of Opensolaris-200805, FreeBSD7 and OpenSuSe11. The evaluation includes initial installation, device support, installing additional programs, and ease of use. The main considerations are ease, stability, and speed.

The install media consists of downloadable isos. The boot process may be discussed if a significant issue exists. When possible a livecd of the operating system is used to see how the systems react first. It is good technique/less risky to do so before installation. The two machines used have the following configuration: see Table 1.

All of the systems are setup with a root password and a non-root user. Opensolaris and OpenSuSE require a desktop installation, GNOME for Opensolaris and KDE4 for OpenSuSE. In fact, not installing a desktop would require some effort with both Opensolaris and OpenSuSE. FreeBSD requires manual installation and configuration for any desktop, the default is a console login. I used KDE3 in FreeBSD. KDE4 is also available on FreeBSD7 and a brief test was made.

### Opensolaris

The Opensolaris livecd boots up into a normal GNOME desktop. A graphical tool for detecting hardware and available drivers is on the desktop. Click on the *Device Driver Utility* and a window opens providing information on what hardware is detected and what driver supports that hardware. There has not been a better tool for the purpose. There were only a few issues with hardware support on these systems. Opensolaris does not come with drivers for many sound cards. You can solve this problem easily by using the open sound system drivers from 4front. This solution worked well with both systems. Another problem in Opensolaris with the desktop system is the lack of a driver for the ralink `rt2561/rt61` chipset. There are no plans for such a driver. On the plus side, those drivers that do exist work very well (Figure 1).

Network configuration uses nwam, a network wizard daemon. The wizard works well for ethernet and unencrypted wireless, but fails on encrypted wireless. The option offered for entering a wep key does not work. For encrypted wireless one must disable nwam and use the console tools, either dladm or wificonfig.

Opensolaris uses pkg for program maintenance. There is a graphical interface that is slow and buggy, rendering it unuseable. In fact, later versions crash immediately. The command line tool is quite good and works well. Package installation and updates are easy for any supported software. The main difficulty is installing programs from alternative repositories. Attempts to install a build environment for SFE, a set of spec files for various programs that are not supplied in the opensolaris repositories, all fail to build a single program. Installing prebuilt packages from the repository has worked. It cannot play commercial dvds, nor install the packages to do so. Perhaps the instructions for the build environment have not kept pace with the changes in Opensolaris since an earlier Solaris(b67) attempt to setup a build environment worked.


Figure 1. Drivers

No review with Opensolaris would be complete without a mention of `zfs`. You may have heard of zfs, the best new filesystem ever, repairs errors before they happen, allows infinite storage and cures cancer. Just kidding. In reality, `zfs` is very good and would certainly justify using Opensolaris for any server system. Backups, rollbacks, directory level compression, and more work well and are very fine-grained. The only place a normal desktop user might notice zfs is during a system update. System updates get written to a new pool and the bootloader is updated to use that. If there is some error, you can rollback the update very quickly and easily. Backups are handled similarly. Other uses would rarely be seen by normal desktop users.

Opensolaris works very well on supported hardware and not at all on unsupported hardware. Installing the nvidia driver, flash, Openoffice, Sun Studio, and a java environment were effortless and all worked with no problems. An updated wine, an important tool for those wishing to use Windows programs, was one of the few programs that installs properly from SFE. Multimedia codecs, commercial DVD playback, and other oft used packages were either not available as packages or did not work. This test makes no attempt to build directly from the source tarballs, only the spec files.

## FreeBSD7

The desktop FreeBSD7 installation requires updating from a FreeBSD6 iso. The FreeBSD7 iso results in instant reboots (DesktopBSD, PC-BSD, and Freesbie all have liveCD isos based on FreeBSD6 and alpha or beta isos on FreeBSD7. Freesbie is the most complete but does not offer a trivial install. PC-BSD and DesktopBSD make configuring a graphical environment slightly easier. All of them had problems booting on multiple systems) necessitating a more convoluted install. The laptop does not require any special means to install FreeBSD7. After updating, problems arise if the nvidia module and all the sound modules are added to loader.conf. Only after booting with a rescue cd and modifying the `/boot/loader.conf` file to NOT load nvidia and to load only the specific sound driver for the system would a console boot on both systems.

FreeBSD comes in two parts, the base system and everything else. The *ports* tree contains everything else, either as a prebuilt package or a set of recipes for building from source. The ports system makes source installation very simple, change to the directory of the port you wish to build and *make install clean*. If you wish to modify the default configuration then *make config* first. The sources will be downloaded and package configured, built, and installed with all required dependencies handled and an update to the package database for easy package management. If you do not wish to build from source (because, for instance, both of these systems can be quite slow for building *OpenOffice*) FreeBSD comes with `pkg_add` for fetching a binary package from a remote repository and installing it. There exist multiple tools for ports management.

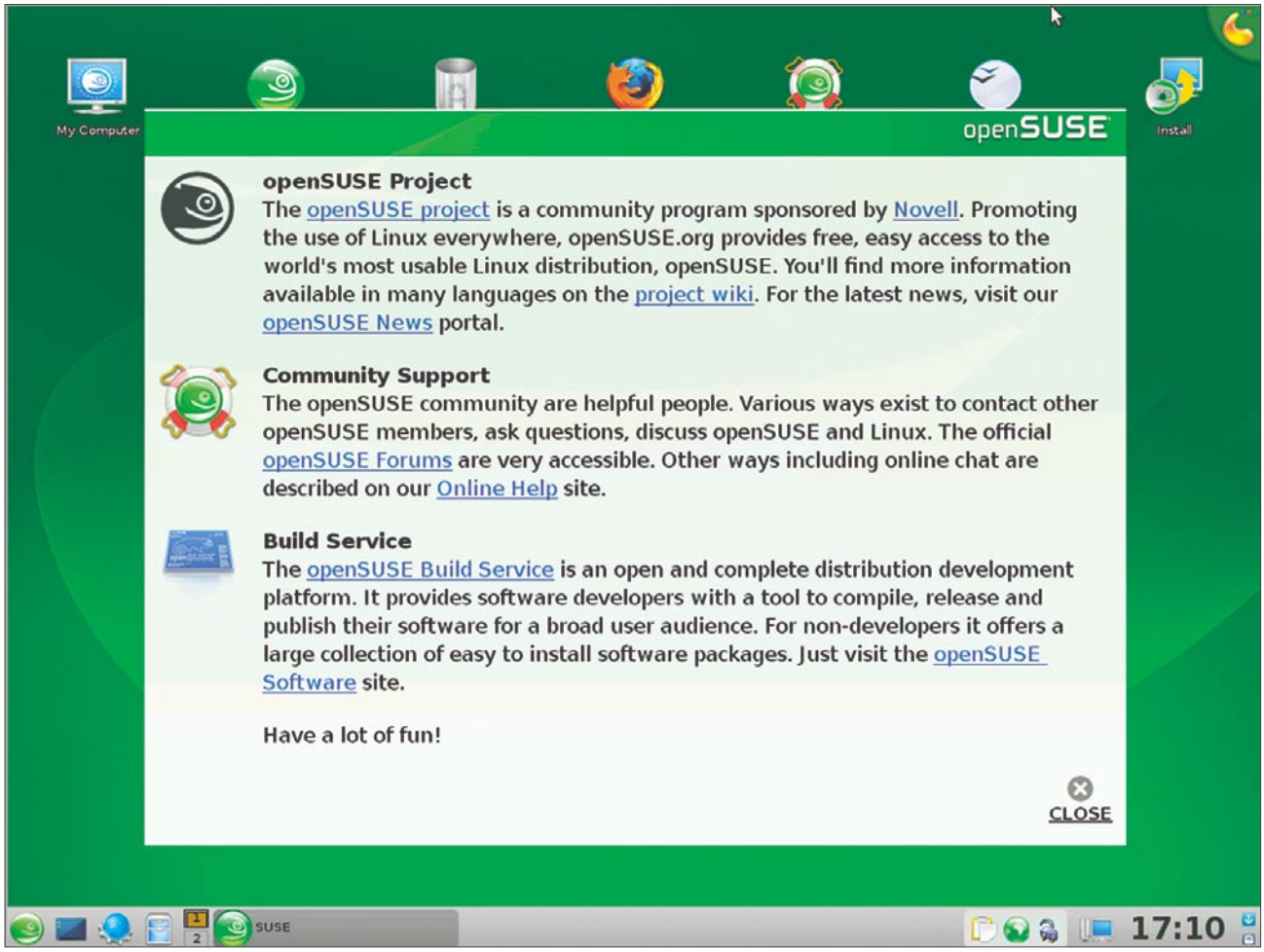



Figure 2. Screenshot from softpedia

Network configuration is done through a text config file in `/etc/` called rc.conf. There is no similar component to nwam in Opensolaris or networkmanager in OpenSuSE on FreeBSD7. For a trivial network configuration that does not change, including wep encryption, `rc.conf` is enough, just add a line for each interface you wish to configure. For wireless roaming or encrypted networks you need to write a `wpa_supplicant.conf` file and place that in `/etc`, adding *WPA* to the interface line in `rc.conf`. More complex situations require customized startup scripts. Changing from wireless to wired requires manual intervention and, just like wireless roaming, fails to reset the routing table. Switching from wireless to wired requires elaborate contortions. The only method that works for wireless roaming was to set `wpa_supplicant.conf` with multiple entries though that did not work particularly well. Rebooting in a new location works for wireless roaming and disabling the wireless interface in `rc.conf+restarting` network interfaces works for wired.

Although the hardware support on FreeBSD is greater than on Opensolaris, there are more problems. There are issues with the pcmcia slots on the laptop concerning the cardbus bridge and the nvidia driver on the desktop. There appears to be a bizarre bug in FreeBSD7 that causes connections on the laptop and on the desktop to disconnect at random intervals. There were no error messages and bringing the interfaces down and up restores the connections. The cause is unlikely to be bad hardware as Opensolaris and OpenSuSE have no issue.

The evaluation above uses KDE3.5.8. KDE4.1 is now available in FreeBSD7. Only the laptop is evaluated with KDE4,installed from packages, and manually configure to start from kdm. The KDE4 desktop looked good but consumed more resources over time, eventually making KDE4 unusable. There seems to be a bug in konqueror for KDE4 that causes processes to never quit on FreeBSD7. The same issue does not occur on OpenSuSE. Only individual programs crash and not KDE4 as a whole. KDE4 on OpenSuSE is much more polished, easier to use, and install. OpenSuSE clearly puts many more resources into making KDE4 better. If one wishes to test out KDE4, OpenSuSE is the clear choice.

Installing multimedia codecs and DVD playback requires minimal effort. To use a recent version of flash, wine needs to be installed. That allows using Windows *firefox+flash*. That combination works surprisingly well and uses the same amount of resources as the native *firefox+flash* combination on Opensolaris and OpenSuSE. Wine has to be built from the port as there is some incompatibility in the prebuilt package that causes wine to crash. A similar problem occurs with *firefox*. Some packages are not built with required flags and these need to be rebuilt. Installing a java jdk or jre is very annoying (I had to download 4 separate files from 3 different sites to build the native java version);the package does not work well.

The largest single issue facing FreeBSD is the lack of maintainers for large packages, as compared to Linux. You may have noted the version of KDE is 3.5.8 while the kde website shows that 3.5.10 is out and 3.5.9 has been out for quite some time. There is no official package for Openoffice. Either a matching package for your running system must be found, or you must build from source.

The most annoying aspect of FreeBSD deals with networking. Interfaces must be restarted or reconfigured constantly. All other aspects were bearable and either had no solution or a one-time solution. Most likely no alternative would need to be considered without that problem. The

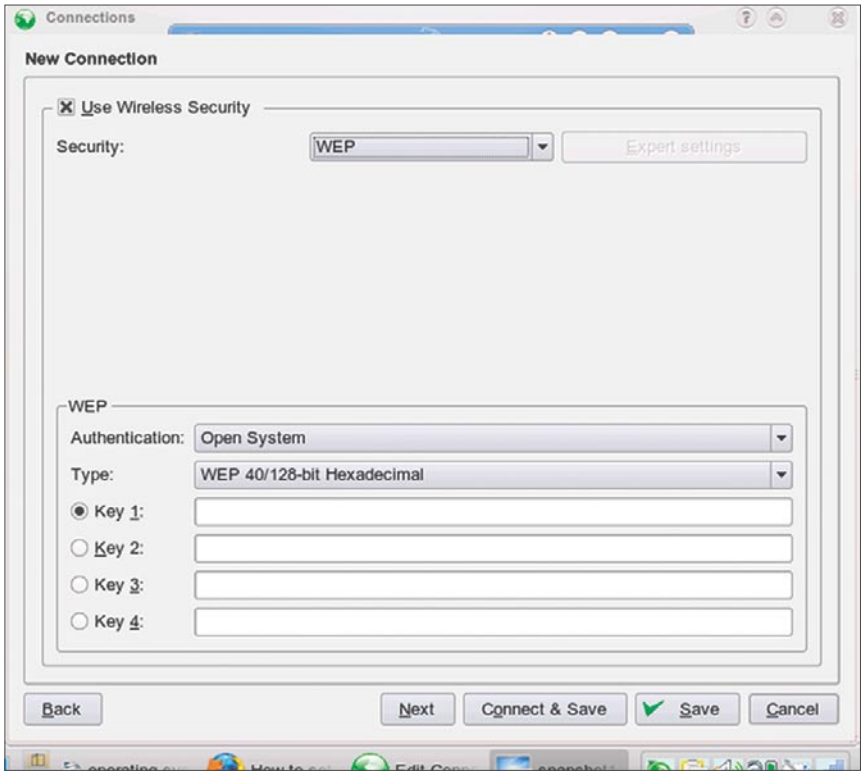



**Figure 3.** Knetworkmanager

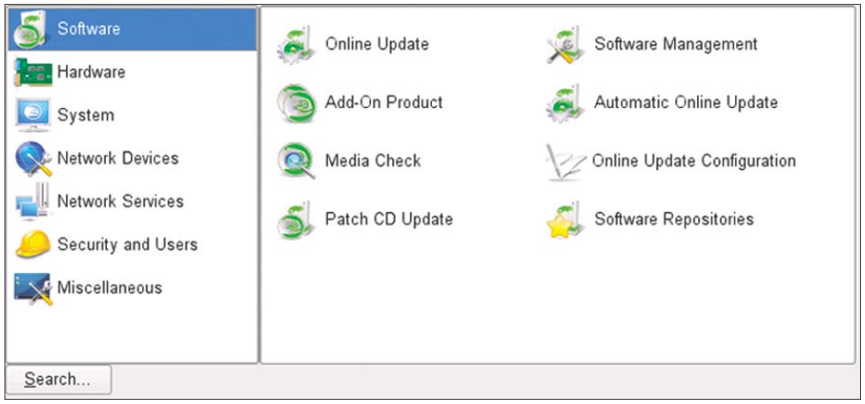



**Figure 4.** Main menu

networking problems create doubt about the viability of using FreeBSD7 as a server solution. Perhaps that is why most hosting companies still use FreeBSD6 and have not migrated to 7 or 8.

## OpenSuSE

The installation uses the most recent OpenSuSE-11.0 KDE4 liveCD. KDE4 was up after the shortest boot of the 3 operating systems. One interesting aspect of the liveCD environment is the ability to install new packages. Wine, flash, and firmware for the `rt2561` card are all easily installed. Windows applications, viewing flash sites, and browsing the internet, all work well (see Figure 2).

The actual KDE4 liveCD screen is similar in color and theme, but not the same as the screenshot, where an earlier version of KDE4 is used. There is an install icon that starts a very simple install using the liveCD environment as the base. The account name, password, root password, keyboard, and location are set during install. In addition, the partitioning can also be changed (for desktop use, a single large partition and a swap partition were created). As the boot is the shortest, so is the install. It took under 20 minutes from first boot to rebooting into the new system.

Networking is managed using a much improved networkmanager and knetworkmanager applet. In the past networkmanager was slow and buggy, with very limited functionality. Now one can set up most common encryption, save settings for individual networks, and change networks easily. There is no problem switching between wired and wireless networks and roaming is trivial. There are no problems with the networking that required disabling networkmanager and writing config scripts. Unbelievably, *It just works!* That cannot be said for FreeBSD, Opensolaris, or earlier Linux systems. There is no problem with networking on these two systems.

OpenSuSE uses YAST, yet another setup tool, for every configuration task. If you have not seen YAST, imagine a Windows Control Panel done right. YAST is wonderful with the only flaw being it's speed. That flaw is now gone. Every part of YAST is much faster than before. A package management system is included as one of the YAST modules. There is a system tray icon for updates and a separate software management frontend. Repositories are trivial to manage and supply many packages not available directly from OpenSuSE. The package management system can add repositories for KDE4 updates, the OpenSuSE build service, Wine, packman (a multimedia repository), and others.

Enabling dvd playback and installing codecs requires that one first uninstall the libraries provided by OpenSuSE. In particular, the xine library is built without support for most multimedia formats. If these libraries are not uninstalled, the package management system will not replace them with working versions from the alternative repositories. Flash does not work well on OpenSuSE. The only version that works is the beta version of `flash10` from adobe. That version requires `libcurl3`.

The KDE4 window manager on OpenSuSE includes many features that once required installing compiz. These work quite well and have not crashed the system. In fact, the laptop has far too little video memory to consider compiz. However, if you wish the rotating cube, compiz for kde4 is available.

Some of the programs that are included with KDE4 are simply not ready. Konqueror is a fine tool for everything else but crashes constantly when accessing websites. Install firefox3 and set that as the default web browser. Replace the beta of Koffice2 with OpenOffice. Koffice2 crashes far too often while OpenOffice has yet to crash.

## Conclusions

Opensolaris allows very fine-grained control of the entire system and only supports hardware that works well. These aspects make Opensolaris a wonderful server operating system, especially when one considers the existence of `zfs`, but necessitates a steep learning curve. Opensolaris on the desktop leaves much to be desired. Multimedia playback, random hardware support, suspend/resume, and the steep learning curve are just some of the problems facing a normal user.

FreeBSD7 is the only one of the 3 systems that could be installed on a 486 class machine and run with 16MB of ram. The ports system and installer allows for minimal installs, with only those packages installed that are required for a particular task. Multimedia playback, speed, and support for a large number of systems are all positives for FreeBSD7. Desktop users will have difficulty with configuration of a desktop environment, networking, and certain hardware.

OpenSuSE-11 has the easiest and fastest install, the easiest networking tool, the easiest system controller in YAST. In fact, easy might be the best description of OpenSuSE. The only flaws were in multimedia and flash. Both of these require one-time fixes. There are two hardware issues, both on the desktop. Firmware needed to be manually extracted from an archive, downloaded from the ralink website, and installed in `/lib/firmware`. Suspend/Resume fails miserably on the desktop, requiring a reboot. *Suspend/Resume* on the laptop requires creating a `/etc/pm/sleep.d/99sound` file to handle the sound modules.

**Table 1.** Thinkpad T23 & A7N266 motherboard with Nforce1 chipset

| | Thinkpad T23 | A7N266 motherboard with Nforce1 chipset |
|---|---|---|
| Processor | Intel P3 1Ghz | Athlon-XP 2Ghz |
| Ram | 1GB | 1GB |
| Video Card | Savage | Nvidia |
| Video Memory | 8MB | 128MB |
| Hard Drive | 120GB Western Digital(IDE) | 120GB Seagate (IDE) |
| Ethernet | Intel 100 | 3COM 905b |
| Wireless | Intel Pro 2100 (802.11b) | MSI (802.11g) ralink rt2561 |
| Sound | Intel 82801CA | Onboard |
| Other ports | 2 usb1, 2pcmcia, s-video, ps2 mouse, dock, vga, serial, parallel, modem, rj45, headphone jack | 4 usb1, serial, parallel, ps2 mouse + keyboard, vga, rj45, headphone |

# Multi-User Conferencing

Eric Schnoebelen
Michele Cranmer

So you've gone and done it, taken the plunge, and adopted Jabber as your instant messaging system. You've even built a couple of transports to keep connected with those less enlightened people who continue to use the 'walled garden' networks.

Now you want to be able to bring all your jabber using friends and family (don't laugh, on our server, we have a room just for the family) into a single (virtual) room, or maybe set up different rooms for different groups using your server. How are we going to do that?

Fortunately, the XMPP Foundation (and the Jabber Foundation before them) defined a standard for Multi-User Conferencing (XEP-0045). Jabber's Multi-User Conferencing is very powerful, and it's not restricted to the users of just one server. Assuming correct DNS entries, you can connect to a conference room on any Jabber server in the world, and any Jabber user can connect to your conference rooms to chat.

## Multi-User Conferencing

Most of the commercial or quasi-commercial XMPP servers (OpenFire, ejabberd, Jabber Inc) have built in conferencing packages. Unfortunately, the jabber server I run (and have been using in this series), the Open Source jabberd2 server, lacks a multi-user conference component (as does `jabberd1`.)

At the time of writing, we're aware of two component implementations of XEP-0045, usable with `jabberd2`. The first is mu-conference, originally written as an internal component for the orignal Jabber implementation, jabberd1 (now known as `jabberd16`). A component library was created from the jabberd1 source to allow it to operate as a component using jabberd2's component protocols.

The second XEP-0045 component implementation is palaver, written in python, using the twisted framework. (you should remember twisted from the transports article, where several of the transports are based on the twisted libraries.)

We're going to take a look at building, installing and configuring both mu-conference and palaver.

## Obligatory pkgsrc

mu-conference 0.6.0 is in `pkgsrc/chat/mu-conference`. I've packaged the updated 0.7.0 version in `pkgsrc-wip` on Source Forge, as wip/mu-conference palaver is in wip/py-jabber-palaver.

It uses/depends upon the editions of twisted in `wip/twisted-core` and `wip/twisted-words`.

To build from pkgsrc (assuming you've bootstrapped pkgsrc on your non-NetBSD platform), change the appropriate directory, and install the software by typing `"[b]make install"`.

FreeBSD has mu-conference in the ports collection as net-im/mu-conference.

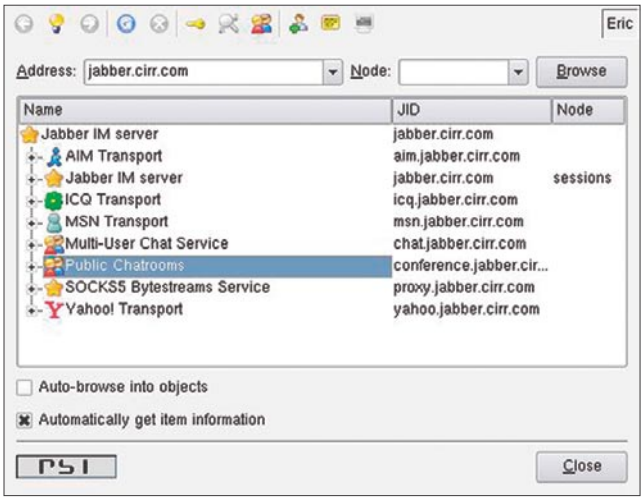



**Figure 1.** Service Discovery in Psi, note „Public Chatrooms" and „Multi-User Chat Service", implemented by mu-conference and palaver respectively

**Listing 1.** Working muc-jcr.xml configuration file for mu-conference

```
<jcr>
  <!--
      This is a config file for a copy of MU-
Conference, compiled against
      the Jabber Component Runtime (JCR). This is the
same file that I use
      to connect to my development server, running
jabberd2 beta2

     In order to connect to a jabberd v1.4 server,
simply change the
      <name> value to muclinker, and make sure the
muclinker section is in
      your main jabber.xml file, as per the MU-
Conference README file.
  -->

  <!-- the jid of your component -->
  <name>conference.jabber.cirr.com</name>

  <!-- this should be the same as above -->
  <host>conference.jabber.cirr.com</host>

  <!-- adress of the jabber server -->
  <ip>jabber.cirr.com</ip>

  <!-- port used to connect the service to the jabber
server -->
  <port>5347</port>

  <!-- secret shared with the jabber server -->
  <secret>***************</secret>

  <!-- directory containing the rooms data -->
  <spool>/var/spool/jabberd/muc</spool>

  <!-- directory containing the debug log
       (the file is called mu-conference.log) -->
  <logdir>/var/log/jabberd</logdir>

  <!-- file that will contain the PID of the process
-->
  <pidfile>/var/run/jabberd/mu-conference.pid</pidfile>

  <!-- uncomment to also send log to stderr -->
  <!--   <logstderr/> -->

  <!-- log verbosity, 255 for very verbose, 0 for
quiet -->
  <loglevel>124</loglevel>
    <conference xmlns="jabber:config:conference">
      <!-- rooms are public when created,
           comment to make them private by default -->
      <public/>
      <!-- the vCard section contains the vCard of the
service -->
      <vCard>
        <FN>Public Chatrooms</FN>
        <DESC>This service is for public chatrooms.</
DESC>
        <URL>http://jabber.cirr.com/conferences</URL>
      </vCard>
      <!-- maximum numbers of history lines send when
joining a room -->
      <history>40</history>

      <!-- where to store the room logs -->
      <logdir>/var/spool/jabberd/muc-logs/</logdir>
      <!--URL of the log stylesheet -->
      <stylesheet>../style.css</stylesheet>

      <!-- default text to send to legacy clients,
           will also be used in the logs -->
      <notice>
        <join>has become available</join>
        <leave>has left</leave>
        <rename>is now known as</rename>
      </notice>
      <!-- lists of admins of the service, add a
<user/> tag by admin -->
      <sadmin>
        <user>eric@jabber.cirr.com</user>
      </sadmin>

      <!-- when uncommented, only dynamic rooms can be
created -->
      <!-- <dynamic/> -->

      <!-- persistent rooms will be created, overide
<dynamic/> -->
      <!-- <persistent/> -->
      <!-- enforce the user nickname to the user part
of his jid -->
      <!-- <locknicks/> -->
      <!-- uncomment to allow only admins to create
rooms -->
      <!-- <roomlock/> -->
      <!-- configuration of MySQL,
           only used if the MySQL exports is
activated,
           see README.sql -->
      <!--
      <mysql>
        <user>root</user>
        <pass/>
        <database>chat</database>
        <host>localhost</host>
      </mysql>
      -->
    </conference>
</jcr>
```

## Building mu-conference the Hard Way

mu-conference can be downloaded from *http://download.gna.org/mu-conference /mu-conference_0.7.tgz*. mu-conference requires the libidn, expat and `glib2` libraries. Install as appropriate, either from source, or your OS's packaging system. (and of course, the compiler suite.)

Change directories into `mu-conference_0.7/src`, remove `mysql.o`

**Listing 2.** Twisted --help output, note „palaver" in the output

```
/usr/pkg/lib/python2.4/site-packages/twisted/plugins/
palaver.py:1: DeprecationWarning: mktap and related
support modules are deprecated as of Twisted 8.0.
Use Twisted Application Plugins with the 'twistd'
command directly, as described in 'Writing a Twisted
Application Plugin for twistd' chapter of the Developer
Guide.
  from twisted.scripts.mktap import _tapHelper
Usage: twistd [options]
Options:
      --savestats        save the Stats object rather
than the text output of
                         the profiler.
  -o, --no_save          do not save state on shutdown
  -e, --encrypted        The specified tap/aos/xml file
is encrypted.
      --nothotshot       DEPRECATED. Don't use the
'hotshot' profiler even if
                         it's available.
  -n, --nodaemon         don't daemonize
  -q, --quiet            No-op for backwards
compatibility.
      --originalname     Don't try to change the
process name
      --syslog           Log to syslog, not to file
      --euid             Set only effective user-id
rather than real user-id.
                         (This option has no effect
unless the server is running
                         as root, in which case it
means not to shed all
                         privileges after binding
ports, retaining the option to
                         regain privileges in cases
such as spawning processes.
                         Use with caution.)
  -l, --logfile=         log to a specified file, - for
stdout
  -p, --profile=         Run in profile mode, dumping
results to specified file
      --profiler=        Name of the profiler to use,
'hotshot' or 'profile'.
                         [default: hotshot]
  -f, --file=            read the given .tap file
[default: twistd.tap]
  -y, --python=          read an application from
within a Python file (implies
                         -o)
  -x, --xml=             Read an application from a
.tax file (Marmalade
                         format).
  -s, --source=          Read an application from a
.tas file (AOT format).
  -d, --rundir=          Change to a supplied directory
before running [default:
                         .]
      --report-profile=  DEPRECATED.
                         Manage --report-profile option,
which does nothing currently.
      --prefix=          use the given prefix when
syslogging [default: twisted]
      --pidfile=         Name of the pidfile [default:
twistd.pid]
      --chroot=          Chroot to a supplied directory
before running
  -u, --uid=             The uid to run as.
  -g, --gid=             The gid to run as.
      --help-reactors    Display a list of possibly
available reactor names.
      --version          Print version information and
exit.
      --spew             Print an insanely verbose log
of everything that
                         happens. Useful when debugging
freezes or locks in
                         complex code.
  -b, --debug            run the application in the
Python Debugger (implies
                         nodaemon), sending SIGUSR2
will drop into debugger
  -r, --reactor=         Which reactor to use (see --
help-reactors for a list of
                         possibilities)
      --help             Display this help and exit.
Commands:
    ftp              An FTP server.
    telnet           A simple, telnet-based remote
debugging service.
    socks            A SOCKSv4 proxy service.
    manhole-old      An interactive remote debugger
service.
    portforward      A simple port-forwarder.
    web              A general-purpose web server which
can serve from a
                     filesystem or application resource.
    inetd            An inetd(8) replacement.
    words            A modern words server
    toc              An AIM TOC service.
    palaver          A multi-user chat jabber
component.
```

from the list of `conference_OBJECTS`, remove `mysql_config --libs` from `LIBS`, and update the include and library search paths in `LIBS` and `CFLAGS` to make sure glib, expat, and libidn can be found. Similar changes need to be made to the Makefiles in `jabberd` and `jcomp`. Once those changes are made, type `make`.

After `make` completes, an executable named `mu-conference` should exist. There is no `install` target, so copy the binary to a suitable directory, `maybe ~jabber/bin`, or `/usr/local/bin`.

## Configuring mu-conference

In the `mu-conference_0.7` directory (above where we build the binary) is the file `muc-default.xml`. This is a prototype configuration file for `mu-conference`. Pick some directory to install the configuration file, perhaps `/usr/local/etc/jabberd`, as `muc.xml` (or something similar.)

Open the configuration file in your favorite editor. Nearly every tag set is going to need some attention.

`<name></name>` should be the Jabber ID (service hostname) for the service. The hostname needs to be in DNS if you wish off-site users to be able to use the conference server.

`<host></host>` (as the comment says) should match `<name></name>`.

`<ip></ip>` should be the hostname or IP address of the jabber server that's providing the parent service.

`<port></port>` should be set to the port required to connect to the router on `<ip/>`. The default is incorrect, at least for `jabberd2`. It should be set to 5347.

`<secret></secret>` needs to be set to the shared secret for `legacy component connections` in your `jabberd2 router.xml` configuration file.

`<spool></spool>` should be changed to an absolute path. Probably something like `/var/spool/jabberd/mu-conference`.

`<logdir></logdir>` should be changed to an absolute path. Probably something like `/var/log/jabberd/mu-conference.log`.

The final entry that `*must*` be changed is `<pidfile></pidfile>`. It should be changed to an absolute path, probably something like `/var/run/mu-conference.pid`.

The remainder of the configuration file is minor things that can be tweaked to control administrators, how rooms are created, and other default settings. As provide they are suitable for use.

Listing 1 is the mu-conference configuration running on *conference.jabber.cirr.com*.

A live production instance that you can reach out and touch with your Jabber client.

## Running mu-conference

Now that you've got the configuration file tweaked, it's time to start up the conference room/chat server for your Jabber server.

Start mu-conference by using the following command line (as your jabber user, or whoever you wish to own the spool files):

```
mu-conference -c /usr/local/etc/
jabberd/muc.xml &
```

**Listing 3.** Working palaver.xml configuration file for palaver

```
<muc>
  <!--
  This is a config file for palaver.

  This format is currectly backwards compatible with the JCR mu-conference
  config file format, as long as you surround the file with the <muc> tag.
  If you switching to palaver from JCR mu-conference, just add the
  surrounding <muc> and </muc> tags and you can use the same file.
  Note: several JCR fields may be unused in palaver.  This file
  includes only options that are actually used.
  -->

  <name>chat.jabber.cirr.com</name>  <!-- the name of our component -->
  <ip>jabber.cirr.com</ip>           <!-- the server to connect to -->
  <port>5347</port>            <!-- the port to connect to -->
  <secret>***************</secret>      <!-- the secret :) -->

  <!-- the storage backend configuration -->
  <backend>
   <type>dir</type> <!-- possibilities are: dir, memory -->
  </backend>

  <!-- postgresql storage mechanism -->
  <!--
  <backend>
    <type>pgsql</type>
    <dbuser>muc</dbuser>
    <dbname>muc</dbname>
    <dbpass>secret</dbpass>
    <dbhostname>localhost</dbhostname>
  </backend>
  -->

  <!-- spool is the directory where filesystem based backends store data
-->
  <spool>/var/spool/jabberd/chat</spool>

   <conference xmlns="jabber:config:conference">
    <sadmin>
      <user>eric@jabber.cirr.com</user>
    </sadmin>
  </conference>
</muc>
```

The prompt will return and mu-conference will be running in the background. (if you don't force it into the background with the ampersand, it will continue running in the foreground, effectively locking the terminal window.)

Picking your favorite Jabber client (mine is Psi), do service discovery on your server. You should now see a *Public Chatrooms* entry in your service discovery list.

Figure 1 shows the service discovery list on *jabber.cirr.com*. (via my favorite client Psi.) *Public Chatrooms* (the highlighted entry) is the mu-conference service.

## Building palaver the Hard Way

Palaver is an application written for the twistd framework. The needed components are Twistd-Core and Tistd-Words, along with pyOpenSSL. Assuming you've got twistd installed from last issues articles on transports, we'll forge a head.

Grab palaver from *http://onlinegamegroup.com/releases/palaver/palaver-0.5.tgz*, and extract it into a working directory.

Change into the palaver-0.5 directory, and do the standard python installation dance:

```
python setup.py build
sudo python setup.py install
```

You can verify that palaver has been properly installed by running the following command:

```
sudo twistd --help
```

At the bottom will be a list of commands, you should see palaver listed. (see Listing 2 for the full output of `twistd --help`).

## Configuring palaver

There are two ways to configure palaver. One is with an XML configuration file, and the other is via individual command line arguments to the palaver twisted component.

For this article, we're going to focus on the XML configuration file. palaver claims to be compatible with the mu-conference configuration file format, if it is wrapped in `<muc></muc>` tags. (`mu-conference` wraps the configuration in `<jcr></jcr>` tags.)

In the palaver directory, you'll find a file, `example-config.xml`. Copy it to your jabber configuration directory (`/usr/local/etc/jabber?`) as palaver.xml, and edit using your favorite editor.

`<name></name>` should be the Jabber ID (service hostname) for the service. The hostname needs to be in DNS if you wish off-site users to be able to use the conference server.

`<ip></ip>` should be the hostname or IP address of the jabber server that's providing the parent service.

`<port></port>` should be set to the port required to connect to the router on `<ip/>`. The default is correct for `jabberd2`.

`<secret></secret>` needs to be set to the shared secret for *legacy component connections* in your `jabberd2 router.xml` configuration file.

Delete the section for the pgsql backend, it's poorly formatted xml, and will cause palaver to complain loudly if not removed.

`<spool></spool>` should be changed to an absolute path. Probably something like `/var/spool/jabberd/palaver`.

Finally, set the `<sadmin></sadmin>` block to contain your Jabber ID as an administrator.

## Running palaver

Starting palaver is pretty straight forward. Start twistd specifying palaver as the subcommand as in:

```
twistd -u jabber \
    -l /var/log/jabber/palaver \
        --pidfile=/var/run/palaver.pid \
        palaver \
        --config /usr/local/etc/jabberd/
palaver.xml
```

Listing 3 is a working configuration file for palaver.

Figure 1 shows the service discovery list on *jabber.cirr.com*. (via my favorite client Psi.) *Multi-User Chat Service* is the palaver service.

## Using the conferencing service

How you access the conferencing service depends on your client. In most clients (I've checked Psi, Pidgen and JBother (a very nice client written in Java, and web-startable)) find the *Join Group Chat* item on their pulldown menu. Fill in the *Room* with a name (without spaces), a if the room doesn't exist, it will be created.

Then fill in the *Host* or *Server* field with the host name of your chat service (in my case, *chat.jabber.cirr.com*.) Fill in the *Nickname* field with the name you want to use in the room.

Finally, if you know the room doesn't exist, or the room is moderated or invitation only, enter a password. (The password is used when the room is made invitation only.)

Once you've created the room, there are a number of options that can be set, including a short description, the maximum number of room members, room moderation status, and room persistency.

Now that you've got your conferencing software all set up and tested, call the family around the (virtual?) fireplace, and have a nice family chat!

## About the Author

**Eric Schnoebelen** is a 25 year veteran of the UNIX wars, using both System V and BSD derived systems. He's spent more than 20 years working with and contributing to various open source projects, such as NetBSD, sendmail, tcsh, and jabberd2. He operates a UNIX consultancy, and a small, NetBSD powered ISP. His prefered OS is NetBSD, which he has running on Alpha, UltraSPARC, SPARC, amd64 and i386.

**Michele Cranmer** is a relativity new user to UNIX and Jabber, having been basically forced into learning it when she met Eric. After having been a loyal Windows and Yahoo Messenger user for many years, she finds that she prefers the *new* systems to the others because of ease of use and reliability. Being a college student, getting her degree in Special Education, she plans on using the *new* systems in her classroom as a way of teaching the children that there are many different ways to do things other then the *normal* ways and those ways are no more strange or unusual then they are.

# LONE-TAR®

PROTECT YOUR DATA™

Your data deserves better protection and disaster recovery capabilities then what freeware has to offer.

Free evals on the web

**www.CACTUS.com** **800.LONETAR**

# GDB and you
## Part 1

Carlos Neira

Segmentation fault, core dump, if you were lucky enough you would get one and work in the obvious or sometimes not so obvious problem, when you were not so lucky data would be silently corrupted and blood would be spilled (generally yours) if your application started exhibiting this erratious behaviour in a production setting.

This first part of the series asumes a basic knowledge of the c programming language which is necesary to follow the examples and because this article is centered in using gdb to debug C programs.

The function of a debugger is to allow you to see what is going on *inside* a program while it executes, or to see what was doing another program when it crashed, `gdb` can:

- Start your program, passing the arguments necessary for the execution or the program
- Make your program stop on specified conditions.
- Examine what has happened when your program has stopped.
- Change data in your program as it executes to reproduce an error condition.

## Enough for an introduction, let's get to work:

The first thing you should do to start debugging a c application (if you have the source code, if you don't is beyond the scope of this tutorial, you could at least use the disas command in gdb to see the code generated), is compiling with debug symbols if your compiler of choice is `gcc` the `-ggdb` flag does the trick, if your using another compiler `-g` works ok, the `-ggdb` flag is exclusive for `gcc` according to the manpage `gdb(1)`:

```
-ggdb
```

Produce debugging information for use by GDB. This means to use the most expressive format available (DWARF 2, stabs, or the native format if neither of those are supported), including GDB extensions if at all possible.

Make sure that in your makefile you are not using `STRIP(1)`, as the name says it would strip your binary of the debugging information generated by the compiler with the `-ggdb` or `-g flag`, lets see an example (Listing 1). There are fatal flaws in this code, but is *good* to test some features of `gdb`.

compile this with debugging symbols as explained earlier. Now execute the program you should have a segmentation

**Listing 1.** Buggy program

```
/* example for testing gdb */

#include<stdio.h>
#include<string.h>

void print_string(char* sz_string)
{
        char msg[16];
        sprintf(msg,"this is what you need printed
%.*s\n",strlen(sz_string),
         sz_string );
        printf("Yaay  i got :%s",msg);
}


int main(int argc, char** argv)
{

        char blah[16];
        char bleh[16];

        memset(bleh,'B',10);
        memset(blah,'A',17);
        print_string(blah);
}
```

fault, something like this: see Figure 1. To start a debugging session with gdb just do:

```
gdb <name_of_program>
```

in this case we have a core file so, after loading `gdb` we use the command:

```
core <name_of_core_file>.
Or all in just one line: gdb example2
example2.core
```

help tips refering a command are accessed through the help menu just type:

```
help <command you need reference>
```

This look pretty bad,the backtrace does not look so usefull,so we start setting some break points, you can set a breakpoint in a specific line in the source code or at a function: to set a breakpoint at a function just type :

```
break <name_of_function>, or just b
<name_of_function>
```

to set a breakpoint at a source code line number just type:

```
break source.c:<line>, example : break
example2.c:1
```

if we have the source in the same directory of the application being debuged, we can see a window with the source code just typing : `wi src`

when you are done setting breakpoints hit r to run the program.

## Let's get started

What we have just done? well first we set a breakpoint at main, then one at function `print_string`, then one at line 20 of `example2.c` (Listing 2), typing r begins execution of the program and stops in the first breakpoint (main), then you can resume execution by pressing n (next) or s (step) to advance one instruction at a time if you want to advance to the next breakpoint type c (continue).

Acording to the help menu :

next (n):

- Step program, proceeding through subroutine calls.
- Like the *step* command as long as subroutine calls do not happen; when they do, the call is treated as one instruction.
- Argument N means do this N times (or till program stops for another reason).

continue (c) :

- Continue program being debugged, after signal or breakpoint.
- If proceeding from breakpoint, a number N may be used as an argument, which means to set the ignore count of that breakpoint to N − 1 (so that the breakpoint

```
cneira@wallada /usr/home/cneira/Articles/Bsdmag]$ ./example2
Yaay  i got :this is what you need printed :AAAAAAAAAAAAAAAA@@@<@@@)@F
Segmentation fault: 11 (core dumped)
cneira@wallada /usr/home/cneira/Articles/Bsdmag]$
```

**Figure 1.** Your segmentation fault

```
[cneira@wallada /usr/home/cneira/Articles/Bsdmag]$ gdb example2
GNU gdb 6.1.1 [FreeBSD]
Copyright 2004 Free Software Foundation, Inc.
GDB is free software, covered by the GNU General Public License, and you are
welcome to change it and/or distribute copies of it under certain conditions.
Type "show copying" to see the conditions.
There is absolutely no warranty for GDB.  Type "show warranty" for details.
This GDB was configured as "i386-marcel-freebsd"...
(gdb) bt
No stack.
(gdb) core example2.core
Core was generated by `example2'.
Program terminated with signal 11, Segmentation fault.
Reading symbols from /lib/libc.so.7...done.
Loaded symbols for /lib/libc.so.7
Reading symbols from /libexec/ld-elf.so.1...done.
Loaded symbols for /libexec/ld-elf.so.1
#0  0x72702064 in ?? ()
(gdb) bt
#0  0x72702064 in ?? ()
#1  0x65746e69 in ?? ()
#2  0x413a2064 in ?? ()
#3  0x41414141 in ?? ()
#4  0x41414141 in ?? ()
#5  0x41414141 in ?? ()
#6  0x41414141 in ?? ()
#7  0x3cbfbfea in ?? ()
#8  0x29bfbfea in ?? ()
#9  0x0a080484 in ?? ()
#10 0x41414100 in ?? ()
#11 0x41414141 in ?? ()
#12 0x41414141 in ?? ()
#13 0x41414141 in ?? ()
#14 0xbfbfea41 in ?? ()
#15 0xbfbfea3c in ?? ()
#16 0x08048429 in _start ()
Previous frame inner to this frame (corrupt stack?)
(gdb)
```

**Figure 2.** gdb Backtrace with the bt or backtrace command

**Listing 2.** Buggy program fixed

```
#include<stdio.h>
#include<string.h>


void print_string(char* sz_string)
{
        char msg[64];
        sprintf(msg,"this is
what you need printed :%.*s\
n",strlen(sz_string),sz_string );
        printf("Yaay  i got :
%s",msg);
}


int main(int argc, char** argv)
{
        char blah[16];
        char bleh[16];

        memset(bleh,'B',16);
        memset(blah,'A',16);
        print_string(blah);
}
```

won't break until the Nth time it is reached).

### step (s):

- Step program until it reaches a different source line.
- Argument N means do this N times (or till program stops for another reason).

The next breakpoint we hit is at line 20, wait! look at the memset and the size of the blah array, we check the content of it with the print command (p), guess what is in the content of the byte at blah[16]!, C arrays start at 0 so the 16 position is just the content of the memory at that address that happens to be `A`, nice! one bug found, imagine having a variable which holds money at that memory address suddenly overwritten....

Let see that !, modify the source code to: see Listing 3

See what value a10 takes !!

Now back to the original program, we hit line 7 just in the sprintf instruction, let see what is the string length of the `sz_string`, just call strlen and print the result!: p `strlen(sz_string)` the result 28 ?? so you were passing not a NUL terminated array as a parameter see the manpage `STRLEN(3)`.

So to test this we just use the set command:

### (gdb) help set variable

Evaluate expression `EXP` and assign result to variable `VAR`, using assignment syntax appropriate for the current language (`VAR = EXP` or `VAR := EXP` for example). `VAR` may be a debugger `convenience` variable (names starting with `$`), a register (a few standard names starting with $), or an actual variable in the program being debugged. `EXP` is any valid expression.

This may usually be abbreviated to simply `set`.

So now we dont a have a segmentation fault and all is working ok (did you fixed the memset that we saw earlier?, if you did not is just a matter of "luck" if your program works or not ), the day (and maybe your job ) is saved !

```
(gdb) b main
Breakpoint 1 at 0x80484f0: file example2.c, line 13.
(gdb) b print_string
Breakpoint 2 at 0x80484a6: file example2.c, line 7.
(gdb) b example2.c:20
Breakpoint 3 at 0x8048537: file example2.c, line 20.
(gdb) r
Starting program: /usr/home/cneira/Articles/Bsdmag/example2

Breakpoint 1, main () at example2.c:13
13      {
(gdb) i lo
argc = Error accessing memory address 0x0: Bad address.
(gdb) n
main () at example2.c:18
18              memset(bleh,'B',10);
(gdb) s
19              memset(blah,'A',17);
(gdb) c
Continuing.

Breakpoint 3, main () at example2.c:20
20              print_string(blah);
(gdb) p blah
$1 = 'A' <repeats 16 times>
(gdb) p blah[17]
$2 = -23 '@'
(gdb) p blah[15]
$3 = 65 'A'
(gdb) p blah[16]
$4 = 65 'A'
(gdb) c
Continuing.

Breakpoint 2, print_string (
    sz_string=0xbfbfe9c4 'A' <repeats 17 times>, "@@@\024@@@)\204\004\b")
    at example2.c:7
7               sprintf(msg,"this is what you need printed :%.*s\n",strlen(sz_string),sz_string );
(gdb) p sz_string
$5 = 0xbfbfe9c4 'A' <repeats 17 times>, "@@@\024@@@)\204\004\b"
(gdb) p strlen(sz_string)
$6 = 28
(gdb)
```

**Figure 3.** Setting some breakpoints in your source code

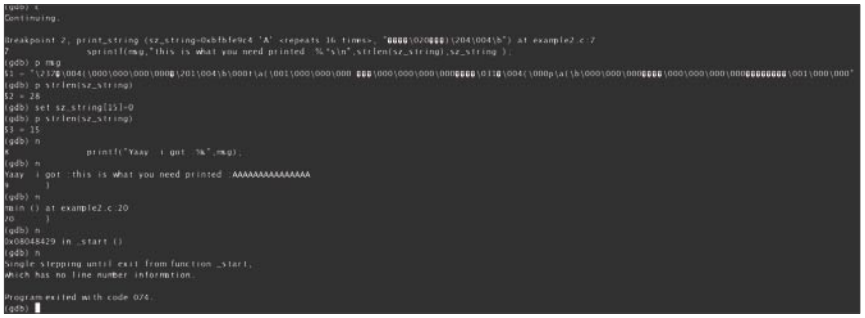


**Figure 4.** Setting some breakpoints in your source code

**Listing 3.** Example of a variable being overwritten

```
#include<stdio.h>
#include<string.h>

void print_string(char* sz_string)
{
        char msg[64];
        sprintf(msg,"this is
what you need printed :%.*s\
n",strlen(sz_string),sz_string );
        printf("Yaay  i got :
%s",msg);
}


int main(int argc, char** argv)
{
        int a=0 ;
        int a1=0 ;
        int a2=0 ;
        int a3=0 ;
        char blah[16];
        int a4=0 ;
                char bleh[16];
        int a5=0 ;
        int a6=0 ;
        int a7=0 ;
        int a8=0 ;
        int a9=0 ;
        int a10=0 ;

        memset(bleh,'B',16);
        memset(blah,'A',17);
            printf("%p %p %p %p %p
%p %p %p %p %p %p %p %p
%p\n",&a,&a1,&a2,&a3,&a4,&a5,&a6,&a
7,&a8,&a9,&a10,&blah[16]);
printf("a9 %d mydebts %d\
n",a9,a10);
        print_string(blah);
}
```

# HAKIN9

## PRACTICAL PROTECTION


# IT SECURITY MAGAZINE

## WWW.HAKIN9.ORG/EN

# Installing Prelude IDS

Henrik Lund Kramshøj

Protecting and securing an IT infrastructure takes a lot of time – and who knows if it is secure now and stays secure in the future. We cannot reach 100% security when connecting servers to the internet, running servers inside our company – or even protect our laptops when moving from one untrusted wireless LAN to another.

The best we can do is implement defense in depth and try our best to keep everything updated and configured sensibly. One things that helps though is operating systems that aim to be secure by default – like NetBSD. This foundation is great for a lot of systems and I will go through the steps needed to implement Prelude IDS on NetBSD as an example of the ease of use of NetBSD and also introduce a mature enterprise system for logging and detecting bad things. The operating system is NetBSD and the system I will be installing is Prelude which is a Universal *Security Information Management* (SIM) system. As described on the home page this system of applications collects, normalizes, sorts, aggregates, correlates and reports all security-related events. That might sound like it is promising too much, but in reality the structure and design of Prelude allows integration of existing software like snort, samhain, ossec, auditd, PF packet filter and includes a program for reading logs and creating events. Surely this is a great starting point and does much more than just adding a WEB UI to Snort (Figure 1).

## Prerequisites for Prelude

The Prelude system is based upon a database, so you need to install database software and configure the central database and the prelude-manager application – the central focal point for Prelude. There are also a few other things to do after installing the NetBSD operatingsystem.

So start by installing the NetBSD operating system (footnote 1) and then do the following:

- Add a user for yourself using the `adduser -m -G wheel joe`
- Add SSHD to `/etc/rc.conf` with the option `sshd=YES`
- Add your SSH public key to `.ssh/authorized_keys`
- Add sudo unless you use just `su`, I use sudo with special privileges for the wheel group
- Production setups might also want to add different file systems for database, data, logs etc.

## Adding Prelude

Building Prelude from scratch is not recommended for first time users, as there are a lot of components to a Prelude installation – but luckily there are Prelude packages for NetBSD which can be used.

You can thus install directly from the ftp repository as described in The NetBSD Guide using an export `PKG_PATH` with the right repository (Listing 1).

That `pkg_adds` commands alone should fetch prelude-manager with all dependencies including MySQL database software. Personally I mostly use PostgreSQL, and I won't get into a war about which database you should use. Choose MySQL or PostgreSQL for your installation as you please, Prelude can do use both.

| Project Website | Description |
|---|---|
| AuditD | Auditd provides user-space utilities for creating audit rules, as well as for storing and searching audit records generated by the audit subsystem in the Linux 2.6 kernel. It features an Intrusion Detection plugin that analyses the audit stream in realtime for suspicious events and alerts via IDMEF using Prelude. |
| Nepenthes | Nepenthes is a versatile tool to collect malware. It acts passively by emulating known vulnerabilities and downloading malware trying to exploit these vulnerabilities. |
| NuFW | NuFW adds user-based filtering to Netfilter, the state of the art IP filtering layer from the Linux kernel. Its exclusive algorithm allows authenticated filtering even on multiuser computers. NuFW can be seen as an Identity access management solution, at the network level. |
| OSSEC | OSSEC is an Open Source Host-based Intrusion Detection System. It performs log analysis, integrity checking, Windows registry monitoring, rootkit detection, real-time alerting and active response. |
| PAM | Linux-PAM is a system of libraries that handle the authentication tasks of applications on the system. The library provides a stable general interface that privilege granting programs (such as login and su) defer to perform standard authentication tasks. |
| Samhain | Samhain® is a multiplatform, open source host-based intrusion detection system (HIDS) for POSIX (Unix, Linux, Cygwin/Windows). Samhain provides file integrity checking, as well as rootkit detection, port monitoring, detection of rogue SUID executables, and hidden processes. |
| SanCP | SanCP is a network security tool designed to collect statistical information regarding network traffic, as well as, record the traffic itself to file in pcap format for the purpose of: auditing, historical analysis, and network activity discovery. |
| Snort | Snort® is a network intrusion prevention and detection system utilizing a rule-driven language, which combines the benefits of signature, protocol and anomaly based inspection methods. |

**Figure 1.** Prelude-ids.com-external-sensors

The other software installed above are the Prewikka WEB UI, Prelude Log Monitoring Lackey and Prelude PF logger – about 20 packages are installed by now.

## Configure Prelude databases

We are now going to go speeding through the configuration required as documented in the User Manual from Prelude available at: *https://trac.prelude-ids.org/wiki/ManualUser* – but first enable the rc.d scripts needed for starting and stopping the software: see Listing 2.

This will allow you to start the MySQL and add the database using the instructions in *https://trac.prelude-ids.org/wiki/InstallingPreludeDbLibrary*: see Listing 3.

Remember when upgrading that some database changes might be required and the scripts for upgrading the database are also located in the directory `/usr/pkg/share/libpreludedb/classic/` and named after the version being upgraded to such as mysql-update-14-6.sql which is the script to update to the layout in version 14.6

Prewikka also needs a database, so do another: see Listing 4.

Then edit the configuration file for Prewikka at `/usr/pkg/etc/prewikka/prewikka.conf` where you can add your company name and change database settings, if you didn't just use the defaults.

**Listing 1.** Installing packages for Prelude

```
# export PKG_PATH="ftp://ftp.NetBSD.org/pub/pkgsrc/packages/NetBSD-4.0/i386/All"
# pkg_add -v prelude-manager
# pkg_add -v prelude-lml
# pkg_add -v prelude-pflogger
# pkg_add -v mysql-server      (running mysql on the same host)
# pkg_add -v py24-prewikka     (py23 and py25 also exist, but preludedb
Python module was for 2.4)
# pkg_add -v apache    (running the apache and prewikka on the same host)
```

**Listing 2.** Add links for easy access

```
nutty# cd /etc/rc.d/
nutty# ln /usr/pkg/share/examples/rc.d/apache
nutty# ln /usr/pkg/share/examples/rc.d/preludemanager
nutty# ln /usr/pkg/share/examples/rc.d/prewikka
nutty# ln /usr/pkg/share/examples/rc.d/mysqld
nutty# ln /usr/pkg/share/examples/rc.d/pflogger
```

**Listing 3.** Create MySQL database

```
# mysql
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 1
mysql> CREATE database prelude;
Query OK, 1 row affected (0.00 sec)
mysql> GRANT ALL PRIVILEGES ON prelude.* TO prelude@'localhost'
IDENTIFIED BY 'prelude';
Query OK, 0 rows affected (0.00 sec)
mysql> exit
Bye
# mysql prelude < /usr/pkg/share/libpreludedb/classic/mysql.sql
```

**Listing 4.** Create tables in MySQL database

```
# mysql
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 6
mysql> CREATE database prewikka;
Query OK, 1 row affected (0.01 sec)
mysql> exit
Bye
# mysql prewikka < /usr/pkg/share/prewikka/database/mysql.sql
```

## Starting Prelude Manager

Configure the Prelude Manager using the default example configuration: see Listing 5.

## Adding a sensor

Next you need to register a sensor, lets use the `Prelude Log Monitoring Lackey` as an example, you need to do this for each new sensor program – but it is not as hard as it might seem. You need to start two programs that need to communicate, because the registration process requires the client-side to connect to the server-side. Begin by running something like this:

```
# prelude-admin register prelude-lml
"idmef:w admin:r"
```

and wait for generation of key material – takes forever, after which it will tell you to start another command – open another window and login into the server: see Listing 6.

Now you should go on and configure Prelude LML by copying the examples rulesets and start editing the configuration file:

```
# cp -r /usr/pkg/share/examples/
prelude-lml/ruleset /usr/pkg/etc/
prelude-lml
# vi /usr/pkg/etc/prelude-lml/prelude-
lml.conf
```


**Figure 2.** Prelude-ids-screenshot

**Listing 5.** Configure Prelude Manager

```
# mkdir /usr/pkg/etc/prelude-manager/
# cp /usr/pkg/share/examples/prelude-manager/prelude-manager.conf /usr/
pkg/etc/prelude-manager/
#  vi  /usr/pkg/etc/prelude-manager/prelude-manager.conf
// to have at least this content:
user = _prelude
group = _prelude
// also adding/changing database information as appropriate:
[db]
type = mysql
# Host the database is listening on.
host = localhost
port = 3306
# Name of the database.
name = prelude
user = prelude
pass = prelude

Before you can use the Prelude Manager you need to add a user (footno-
te 3)
# mkdir  /var/spool/prelude/prelude-manager

# prelude-admin add "prelude-manager" --uid _prelude --gid _prelude
Generating 1024 bits RSA private key... This might take a very long time.
[Increasing system activity will speed-up the process].
Generation in progress... X......+++++O+++++O

Created profile 'prelude-manager' with analyzerID '2037117093516026'.
# /etc/rc.d/preludemanager start
# 30 Aug 20:36:09 (process:15387) INFO: Subscribing Normalize to active
decoding plugins.
// prelude-manager should be running, check /var/log/messages if it is not
# ps auxw | grep prelude
_prelude 13838  0.0  1.7  2440  4540 ?     Isa   8:46PM 0:01.05 /usr/
pkg/bin/prelude-manager -d --pidfile /var/run/prelude-manager/prelude-
manager.pid
```

**Listing 6.** Register new sensor

```
# prelude-admin registration-server prelude-manager
The "65owe145" password will be requested by "prelude-admin register"
in order to connect. Please remove the quotes before using it.

Generating 1024 bits Diffie-Hellman key for anonymous authentication...
Waiting for peers install request on :::5553...
Waiting for peers install request on 0.0.0.0:5553...

// now enter this password in the 1st window and you will see the
registration on the server
Connection from 127.0.0.1:65345...
Registration request for analyzerID="3676201167722360" permission="idmef:
w admin:r".
Approve registration? [y/n]: y
127.0.0.1:65345 successfully registered.
```

When you edit make sure you add at least the following lines: see Listing 7.

If this is working you can kill the process and add it to the rc.local bootup as `prelude-lml -d`.

## Configure Prewikka

The fun begins when Prewikka is configured, so lets get started. This is a web application written in Python, so you need to add stuff to the httpd.conf, or as I recommend add an include like `Include conf/prewikka.conf` to `httpd.conf` which point to a configuration file containing the directives (see Listing 8).

This step can be a little troubling, unless you configure a proper DNS name for the server, add the virtual host AND update the rest of httpd.conf. Please se the advise in the old article Securing Apache 2: Step-by-Step by Artur Maj (footnote 5).

I had problems with Python missing the prelude and preludedb modules, which I fixed by deleting py25-prewikka and going back to py24-prewikka. Solving the issue of missing prelude python module by recompiling the pristine sources of libprelude-0.9.19.tar.gz from Prelude and doing `make install` – not clean, but the libprelude package was missing the module :-( YMMV.

## Testing the installation

An easy way to test the installation is to use the Prelude LML we configured – doing some logins with wrong password etc. This should show up in the Prewikka UI and everything is ready. Next step is to make sure that all programs start when the server is booted and add backup etc. to the environment. I will leave that as an exercise for the reader (see Figure 2).

The picture shown is from an installation where I also added the Snort NIDS which have support for Prelude built-in. The testing done was just doing portscans using the Nmap program by Fyodor.

## Cool stuff with Prelude

Once you get the basic Prelude installation up and running there are a number of ways you can enhance it.

The easiest is of course to add new sensors using the same software as you are running now, just register and configure the new sensor and Prelude

will pick up the new data. Even more fun is to start adding new sensor software which might give interesting data from your network.

- By just adding the Prelude-PFLogger you can make any firewall ruleset with PF create events by adding just log to the rules you want to monitor
- By adding some syslog software like SNARE (footnote 4) to your Microsoft Windows servers you can collect eventlogs and process them using Prelude Log Monitoring Lackey (Prelude LML) and centralize logging of successful and failed logins across your enterprise
- With Prelude LML running you can also use the existing rules that give you information about Sendmail, Postfix, IPFW, OpenSSH SSHD, `mod_security`, Cisco PIX and several others. You can even modify or change the rules using the PCRE syntax which is not hard.
- By adding Snort you can make use of the correlation in Prelude to detect network scans, from the same source – nicely presented in an easy to use UI

**Listing 7.** Prelude LML configuration

```
file = /var/log/messages
...
[Pcre]
ruleset=/usr/pkg/etc/prelude-lml/ruleset/pcre.rules

ruleset=/usr/pkg/etc/prelude-lml/ruleset/ssh.rules
ruleset=/usr/pkg/etc/prelude-lml/ruleset/sudo.rules

after which you can start the prelude-lml program:
# prelude-lml
30 Aug 21:12:32 (process:2747) INFO: PCRE plugin loaded 417 rules.
30 Aug 21:12:32 (process:2747) INFO: PCRE plugin loaded 432 rules.
30 Aug 21:12:32 (process:2747) INFO: PCRE plugin loaded 434 rules.
30 Aug 21:12:32 (process:2747) INFO: Monitoring /var/log/messages through
pcre[default]
30 Aug 21:12:32 (process:2747) WARNING: /var/log/everything/current does
not exist.
30 Aug 21:12:32 (process:2747) WARNING: /var/log/apache2/access_log does
not exist.
30 Aug 21:12:32 (process:2747) INFO: Connecting to 127.0.0.1:4690 prelude
Manager server.
30 Aug 21:12:33 (process:2747) INFO: TLS authentication succeed with
Prelude Manager.
30 Aug 21:12:33 (process:2747) INFO: /var/log/messages: No metadata
available, starting from tail.
```

**Listing 8.** httpd.conf virtual host settings

```
<VirtualHost *:80>
 ServerName prewikka.server.org
 Setenv PREWIKKA_CONFIG "/usr/pkg/etc/prewikka/prewikka.conf"

<Location "/">
        AllowOverride None
        Options ExecCGI

        <IfModule mod_mime.c>
                AddHandler cgi-script .cgi
        </IfModule>

        Order allow,deny
        Allow from all
</Location>

Alias /prewikka/ /usr/pkg/share/prewikka/htdocs/
ScriptAlias / /usr/pkg/share/prewikka/cgi-bin/prewikka.cgi
</VirtualHost>
```

## Conclusions

Prelude is way cool and pretty easy to install using the packages and ports on BSD, not just NetBSD, and can give a lot of useful information covering a large part of what modern enterprises or geeks need. I have used Prelude for more than a year personally and also use it for customer installations. I would recommend following the development of Prelude and perhaps build your own packages, since Prelude development moves pretty fast and getting updated packages may be hard. Also having a test installation to mess with is needed.

### Thanks

I only wrote this article not the software, so I am not the one doing the hard work. The great job is done by all the people from NetBSD, Prelude, MySQL, PostgresSQL, Snort, Samhain, PF, etc. So Thank you all :-)

### On the 'Net

- NetBSD installation is described in detail in The NetBSD Guide at *http://www.netbsd.org/docs/guide/en/netbsd.html*
- Prelude *http://www.prelude-ids.com*
- Prelude generation of keys can take a long time, try typing a lot on the keyboard and doing disk I/O and networking from the system
- SNARE *http://www.intersectalliance.com/projects/SnareWindows/*
- Securing Apache 2: Step-by-Step by Artur Maj *http://www.securityfocus.com/infocus/1786*

# If it moves! crypt it -
## hard drive encryption on BSD

Marko Milenovic

BSD operating systems are well known for their tough security and stability. You may surf the Web assured that your sensitive data is well protected by a firewall or some other advanced BSD protection.

Now let us, for the sake of this text, assume that your hard drive and/or the whole computer gets stolen. This is a very possible scenario considering that the most of us use laptops when we're not at home or at the office. So, your laptop got stolen. Thief might have a lot of problems cracking your password and accessing your files. Or would he? Remember, at this moment all your data is written on the hard drive in a readable form so our thief doesn't even need to boot up the operating system. All he needs is some *Live* BSD CD and he may mount hard drive and copy all your data.

This is one of many scenarios where our data gets stolen for being written in readable form. Let's now see the solution which every BSD system brings – hard drive encryption. Note: super user privileges will be needed to perform following actions.

### Encrypting NetBSD style - cryptographic device driver(cgd)

NetBSD brought hard drive encryption tool between versions 1.6 and 2.0 of the operating system. That's why you may find it starting from NetBSD 2.0. It brings a rather simple procedure for hard drive encryption. This doesn't mean that you should not do a backup of your data before this procedure.

First of all you need to enable cgd in your kernel. Check out kernel configuration file for this line:

```
pseudo-device    cgd      4        # cryptographic disk
driver
```

Number 4 indicates how many cgd devices may be configured at the same time. You may always change this later.

The cgd driver uses three different encryption algorithms:

- `aes-cbc` - AES uses a 128 bit blocksize and accepts 128, 192 or 256 bit keys.
- `blowfish-cbc` - Blowfish uses a 64 bit blocksize and accepts 128 bit keys.
- `3des-cbc` – 3DES uses a 64 bit blocksize and accepts 192 bit keys.

Another aspect of this driver that we need to have a look at before we begin is verification method. The cgd comes with three different ways of verifying the validity of password.

`none` – this method does no verification. This is a rather dangerous method for one particular reason. When a wrong pass phrase is entered cgdconfig does no verification and configures cgd device as normal but it destroys data which was on the volume. I bet you thought it would just configure the device giving access to your data, didn't you?

`disklabel` – cgdconfig checks for a valid disklabel. If a valid disklabel is found with the key that is provided authentication will succeed.

`ffs` - checking for a valid FFS is performed. If it's found with the key that is provided authentication will succeed.

### Encryption step-by-step

Let's go to the point of this story – hard drive encryption. Let's assume you've got a spare partition you'd like to use for storing sensitive data. First of all we want to make sure that there is no data what so ever on our partition. So let's do some scrubbing.

Let's first configure a temporary cgd device with a random key:

```
# cgdconfig -s cgd0 /dev/wd1e aes-cbc 128 < /dev/urandom
```

Now we may fill this partition with zeros:

```
# dd if=/dev/zero of=/dev/rcgd0d bs=32k
```

This will make these zeros look like random data on the device. Be aware that this process may last for some time depending on the size of your partition. Once done we may un-configure the device:

```
# cgdconfig -u cgd0
```

The cgdconfig program is used to manipulate information parameters such as encryption type, key length and random password salt. Before we do this basic configuration make sure that the default location for storing cgd information(`/etc/cgd`) exists and that its mode is 700.

```
# cgdconfig -g -V disklabel -o /etc/
cgd/wd1e aes-cbc 256
```

This creates a configuration file /etc/cgd/wd1e that should look something like this:

```
algorithm aes-cbc;
iv-method encblkno;
keylength 256;
verify_method disklabel;
keygen pkcs5_pbkdf2/sha1 {
        iterations 6275;
        salt AAAAgHTg/
jKCd2ZJiOSGrgnadGw=;
};
```

Let's now configure our cgd device and give it a pass phrase. Since we're doing this for the first time there is no valid disklabel so the validation mechanism we want to use won't work. We will bypass this only this time:

```
# cgdconfig -V re-enter cgd0 /dev/wd1e
```

This will ask you for a pass phrase twice just in case you make a typo.

Our cgd device is ready to be partitioned and activated with a new filesystem. Remember to use disklabel -I since you're creating an initial label for a new disk. After this you should use newfs to format all newly created partitions. Notice that new partition names will reflect cgd partition names:

```
# newfs /dev/rcgd0f
```

Now test this partition by mounting it:

```
# mount -t ffs /dev/rcgd0f /crypted
```

If all goes well you may add the following line to your fstab file:

```
/dev/cgd0f  /crypted ffs
rw,softdep    1 2
```

From now on each time you boot you'll need your cgd device configured earlier. So you should put the following line into `/etc/cgd/cgd.conf`:

```
cgd0    /dev/wd1e
```

This will use /etc/cgd/wd1e as configuration file for cgd0. And finally we need one more line in `/etc/rc.conf`:

```
cgd=YES
```

Each time system boots and starts `/etc/rc` you will be prompted for a pass phrase.

## Encryption OpenBSD style – vnconfig pseudo disk devices utility

OpenBSD is well known for it's proactive security as a router/firewall system. There is an urban myth that using OpenBSD as a desktop system is pretty hard. Well, it's just that – an urban myth. It works well on laptops and that's why we need to take a look at encryption solutions. We will use vnd disk driver that has been introduced to OpenBSD 2.1. Let's see how easy it is to create an encrypted partition.

Let's assume you have a partition with 10GB free space which you want to use for sensitive data. OpenBSD uses vnconfig tool to create a pseudo disk device utility. What does this mean? We won't be encrypting the partition itself. Instead we'll create a file which we'll then format and encrypt just like a regular partition.

First we create an empty file:

```
# touch /datacrypt/cryptfile
```

and then we need to fill it with… nothing.

```
# dd if=/dev/zero of=/datacrypt/
cryptfile bs=1024 count=10485760
```

This will create a file of 10GB size full of zeros. It is now time to use vnconfig to associate this file with a mount point and to do some encrypting:

```
# vnconfig -ck -v svnd0 /datacrypt/
cryptfile
```

You'll be asked for a pass phrase. Choose a good one you won't forget. Our disk is now ready to be initialized and partitioned:

```
# fdisk -i svnd0
# disklabel -E svnd0
# newfs /dev/rsvnd0a
```

This will create partition with 10GB of space. If you want to make more than one partition use disklabel to make modifications. Now let's mount this newly created partition:

```
# mount /dev/svnd0a /crypted
```

And that's it. If you want this partition to be activated at boot time create a script which would look something like this:

```
#!/bin/sh
/sbin/vnconfig -ck svnd0 /datacrypt/
cryptfile
sleep 1
/sbin/mount -f /dev/svnd0a /crypted
```

and put it in /etc/rc.local. Each time your system boots you will be asked for a pass phrase for the device, the system will then configure and mount it.

## Encryption FreeBSD style – geli me this, geli me that

In FreeBSD 6.0 a new encryption method was introduced – geli. The most important features of geli are:

- It utilizes crypto framework meaning that when cryptographic hardware is available, geli will use it automatically.
- Supports multiple cryptographic algorithms – Blowfish, 3DES and AES.
- Allows root partition to be encrypted.
- It is very fast - performs simple sector-to-sector encryption.

In order to use geli we must first enable it in kernel. Add the following to your kernel configuration file:

```
options GEOM_ELI
device crypto
```

After this you need to rebuild kernel and reboot your system. Or we may enable it through /boot/loader.conf by adding:

```
geom_eli_load="YES"
```

After this reboot your system and you should have geli enabled. After this we

need to generate a Master key for our hard drive:

```
# dd if=/dev/random of=/root/da2.key
bs=64 count=1
# geli init -s 4096 -K /root/da2.key
/dev/da2
Enter new passphrase:
Reenter new passphrase:
```

Now it's time to attach this key to a hard drive:

```
# geli attach -k /root/da2.key /dev/
da2
Enter passphrase:
```

When you are done you'll see a new device in your `/dev` directory called `da2.eli`.

You now have and empty partition than needs to go through the same process of labeling and partitioning as before:

```
# dd if=/dev/random of=/dev/da2.eli
bs=1m
# newfs /dev/da2.eli
# mount /dev/da2.eli /crypted
```

New partition is ready to be used. After you reboot your system this partition will not be mounted. You need to do it manually:

```
# geli attach -k /root/da2.key /dev/
da2
Enter passphrase:
# mount /dev/da2.eli /crypted
```

If you don't want to do this every time you reboot your system you may use geli's rc.d script. Just add following lines to /etc/rc.conf:

```
geli_devices="da2"
geli_da2_flags="-p -k /root/da2.key"
```

## Swap partition encryption

Your hard drive is now much safer and you don't have to worry about sensitive data any more, do you? What about that swap partition? All is well till the data is kept in memory. The moment system starts to swap we have problem since all that in there will not be encrypted. So, let's do something about it.

If you are using NetBSD you will want random-key cgd for swap space, regenerating the key each reboot. This is good cause each time the machine is rebooted, any sensitive memory contents that may have been paged out are completely unrecoverable, because you never knew the key.

So, you want to convert your existing swap partition. Let's just do the following:

```
# cgdconfig -g -o /etc/cgd/wd0b -V none
-k randomkey blowfish-cbc
```

When using the random-key generation method, only verification method `none` can be used, because the contents of the new `cgd` are effectively random each time.

In order to make labeling automatic you need to prepare valid labeling and put it in a file `/etc/cgd/wd0b.disklabel`. You ought to check disklabel(8) on how to use it and create a valid swap partition.

Now you need to create /etc/rc.conf.d/cgd so that saved label is restored to cgd:

```
swap_device="cgd1"
swap_disklabel="/etc/cgd/
wd0b.disklabel"
start_postcmd="cgd_swap"

cgd_swap()
{
        if [ -f $swap_disklabel ];
then
                disklabel -R -r $swap_
device $swap_disklabel
        fi
}
```

OpenBSD is doing swap partition encrypting by default. If you are unsure that your swap partition is actually encrypted just execute:

```
# sysctl vm.swapencrypt.enable
```

You should get the following as a result:

```
vm.swapencrypt.enable=1
```

Of the result is for any reason 0 just execute:

```
# sysctl vm.swapencrypt.enable=1
```

After that edit your /etc/sysctl.conf file and add:

```
vm.swapencrypt.enable=1
```

After rebooting your computer swap partition will be encrypted. FreeBSD brings a rather easy method for swap encryption. Let's assume that your swap has been used up to now so it is containing some sensitive data. Let's use random garbage to clear it:

```
# dd if=/dev/random of=/dev/ad0s1b
bs=1m
```

Now let's alter our fstab file a bit:

```
/dev/ad0s1b.eli    none    swap    sw
0    0
```

geli will use the AES algorithm with a key length of 256 bit by default. This may be changed through `/etc/rc.conf` file. First, let's enable geli swap encryption at boot by adding:

```
geli_swap_enable="YES"
```

Now we may alter the way that partition is created by adding:

```
geli_swap_flags="-e blowfish -l 128 -s
4096 -d"
```

This will create geli swap partition using Blowfish algorithm with a key length of 128 bit, a sectorsize of 4 kilobytes and the *detach on last close* option set. After rebooting you ought to check if everything went well. Just execute the following:

```
# swapinfo
Device          1K-blocks     Used
Avail Capacity
/dev/ad0s1b.eli     542720        0
542720     0%
```

## Word of warning

Using encrypted partitions is pointless if you backup your data unencrypted on DVD's or other media. The best solution for backup would be using GnuPG for file based encryption of data that will be stored away from your hard drive. The best solution would be to make a good security plan before even starting with data encryption. You have to know what is to be protected, be sure it deserves to be protected and then go all the way with the encryption.

## Summary

In a world where security has become the highest priority encryption has become very popular way of protecting sensitive data. In this article we have seen some of the best solutions for encrypting in BSD family of operating systems.

great subscriber offer

# SAVE $20!

## *Get your copy of BSD Magazine and save $20 of the shop price*

### *Three easy ways to order*

- *visit: www.buyitpress.com/en*
- *call: 001 917 338 3631*
- *fill in the form below and post it*

### *Why subscribe?*

- *save $20*
- *4 issues delivered directly to you*
- *never miss an issue*


## BSD Magazine ORDER FORM

☐ **Yes**, I'd like to subscribe to
BSD Magazine from issue ☐ ☐ ☐ ☐
1 2 3 4

### Order information
### (☐ individual user/ ☐ company)

Title
Name and surname
address

postcode
tel no.
email
Date

Company name
Tax Identification Number
Office position
Client's ID*
Signed**

### Payment details:

☐ USA $39.99
☐ Europe 29.99€
☐ World 29.99€

I understand that I will receive 4 issues over the next 12 months.
Credit card:
☐ Master Card ☐ Visa ☐ JCB ☐ POLCARD
☐ DINERS CLUB

Card no. ☐☐☐☐ ☐☐☐☐ ☐☐☐☐ ☐☐☐☐ ☐☐☐☐
Expiry date ☐☐☐☐ Issue number ☐☐
Security number ☐☐☐

☐ I pay by transfer: Nordea Bank
IBAN: PL 49144012990000000005233698
SWIFT: NDEAPLP2

Cheque:
☐ I enclose a cheque for $ _____
(made payable to Software-Wydawnictwo Sp. z o.o.)

Signed

Terms and conditions:
Your subscription will start with the next available issue.
You will receive 4 issues a year.

* if you already are Software-Wydawnictwo Sp. z o.o. client, write your client's ID number, if not, fill in the chart above
** I enable Software-Wydawnictwo Sp. z o.o. to make an invoice

# Packaging

## Software for OpenBSD – Part 1

Edd Barrett

The OpenBSD ports system offers developers a versatile way to make binary packages for OpenBSD. In this series of articles we demonstrate how you can make your own packages for OpenBSD.

Porting Fundamentals. In order to understand the ports system, you should first become familiar with the GNU auto-tools, in particular the configure, make, make install routine, which is pretty much standard practice for open-source software build systems now.

In a software tarball, a configure script is intended to inspect the the system environment and craft a Makefile. The software is then built by running make (or gmake), followed by installation with make install.

The ports system wraps this routine up and also provides a way to package the resulting files in a .tgz much like the ones you can install from the OpenBSD package servers. The ports system (which is written as a set of Makefiles) defines a set of targets, the ones of interest to us at this stage are listed in Table 1.

The above list is not exhaustive, but is sufficient as a means of introduction. Later you may wish to read the `bsd.port.mk(5)` manual page.

## Preparing the Ports System

The ports system is not a part of the base OpenBSD install, so you must obtain it either via CVS or an install mirror.

If you are planning on submitting your port to be included in OpenBSD, your port should be built on a recent `OpenBSD-current` build. The porting team will not accept ports for release and stable builds of OpenBSD. If however the port is simply for personal use, feel free to go ahead.

### Retrieving Ports from CVS

To retrieve the ports tree for `-current`: see Listing 1.

`Rt.fm` is one of the many anonymous CVS servers. A full list is available on the OpenBSD wesbite. If you are making a port for a release, you will need to tell cvs which branch you wish to check out, for example:

```
# cvs -danoncvs@rt.fm:/cvs co -rOPENBSD_4_3 ports
```

Also note, a tarball of ports for OpenBSD for releases is on the CDROM and install mirrors.

### Optional Configuration

Without going into too much depth, it is good practice to enable systrace and sudo port building. Systrace will stop a port from changing the file system outside it's work directory and the use of sudo to build ports can often capture some odd bugs in package builds. These procedures are documented in the OpenBSD FAQ (http://openbsd.org/faq/faq15.html#PortsConfig).

**Listing 1.** Obtaining the ports tree

```
# cd /usr
# cvs -danoncvs@rt.fm:/cvs co ports
cvs server: Updating ports
U ports/.cvsignore
U ports/INDEX
U ports/Makefile
U ports/README
cvs server: Updating ports/archivers
U ports/archivers/Makefile
cvs server: Updating ports/archivers/arc
...
```

**Listing 2.** Temporary directory

```
% mkdir -p /usr/ports/mystuff/sysutils/ncdu
% cd /usr/ports/mystuff/sysutils/ncdu
```

## A Simple Port

After rummaging through the ports tree for a while (and asking others for suggestions), I located a suitable port as an example. The `ncdu` port provides a program is similar to the `du(1)` utility which comes on all UNIX systems and is a perfect example of a super-clean and manageable port. I will step you through how this port would have been made, by starting from scratch. Let's make a temporary directory for this. See Listing 2.

`mystuff` is a special directory that ports recognizes as a work in progress area. You can't use any old name.

### The Makefile

The centre point of every port is a Makefile. You can find a sample port makefile at /usr/ports/infrastructure/templates/Makefile.template. The ncdu Makefile would start as follows: see Listing 3.

Having a space after an equals is the new porting convention.

- All ports start with the `$OpenBSD$` CVS macro, which will later be expanded by CVS (if it gets committed),
- `COMMENT` is a short one line description. It should not be enclosed in quotes and the first letter of this value should be lower case,
- `DISTNAME` is the name of the source archive without an extension. Ports will assume that the archive is a `tar.gz` unless you tell it otherwise with `EXTRACT_SUFFIX`,
- `CATEGORIES` is used to group ports via their functionality. A port may fall into several categories,
- `HOMEPAGE` is obvious,
- `MASTER_SITES` is a list of places where the source archive can be downloaded. This port's sources are available in two places: the author's web-page and also sourceforge. Ports knows all about sourceforge and it's mirrors via the `MASTER_SITE_SOURCEFORGE` macro,
- The next block defines which distribution methods the license allows, for both the source archive and the resulting binary package. You should always put a comment clearly stating the license and make sure it is correct. Most software has a `LICENSE` file, but also check the source files and make

**Listing 3.** The initial ncdu Makefile

```
# $OpenBSD$

COMMENT =               ncurses-based du(1)

DISTNAME =              ncdu-1.3

CATEGORIES =            sysutils

HOMEPAGE =              http://dev.yorhel.nl/ncdu/

MASTER_SITES =          http://dev.yorhel.nl/download/ \
                        ${MASTER_SITE_SOURCEFORGE:=ncdu/}

# MIT

PERMIT_DISTFILES_CDROM =Yes
PERMIT_DISTFILES_FTP =  Yes
PERMIT_PACKAGE_CDROM =  Yes
PERMIT_PACKAGE_FTP =    Yes

CONFIGURE_STYLE =       gnu

.include <bsd.port.mk>
```

**Listing 4.** Fetching the sources

```
% make fetch
===>  Checking files for ncdu-1.3
>> ncdu-1.3.tar.gz doesn't seem to exist on this system.
>> Fetch http://dev.yorhel.nl/download/ncdu-1.3.tar.gz.
100% |***************************************************| 98022       00:
00
>> Checksum file does not exist
```

**Table 1.** bsd.port.mk targets

| Ports Target | Description |
|---|---|
| fetch | Fetch the source code archive. |
| makesum | Record a checksum of the source code archive. |
| checksum | Check the checksum of the source file matches the recorded checksum. |
| extract | Extract the source archive. |
| update-patches | Record patches needed to build. |
| clean | Clean the port. Removes extracted sources unless told otherwise. |
| patch | Apply patches to the sources. |
| configure | Run the GNU configure script in the sources. |
| build | Build the sources using the *Makefile* generated at the *configure* stage. |
| fake | Run the *install* target of the sources *Makefile*. |
| plist | Generate the packing list. |
| package | Make a package. |
| port-lib-depends-check | Check library linkage recorded in package matches that of the binaries. |
| install | Install package. |

sure they are consistent. There is nothing worse than a license war. If in doubt email the software's author,

- CONFIGURE_STYLE defines the type of build system the software uses. Most open-source software uses the GNU build system, but others do exist,

The final line pulls in all the functionality required to process the port.

So now our work directory contains only this Makefile.

## Making the Port Build

The first thing to do is the fetch the source archive: see Listing 4.

Ports is quite right to state that no checksum exists. We are going to need one of them: see Listing 5.

Now the distinfo file contains checksums of ncdu-1.3.tar.gz. Next we extract the sources: see Listing 6.

The w-ncdu-1.3 work directory is created. This path is known as WRKDIR. Now we configure the sources. This runs the configure script in the root of the source code directory WRKSRC, which in this case is w-ncdu-1.3/ncdu-1.3: see Listing 7.

You may notice ports mentioning patching, this will be covered in the next installation of this article.

Now the ncdu build system knows a little about our system, we can build binaries with the build target. This will run make in WRKBUILD, which is the same as WRKSRC (in this case): see Listing 8.

## Package Creation

The next stage is to trick the GNU build tools into thinking they are installing binaries onto the system. Usually running make install would

## Further Reading

**RTFM! bsd.port.mk(5), pkg_add(1), pkg_create(1), pkg_delete(1), pkg_info(1), packages(7), ports(7).**
**The OpenBSD FAQ section 15: *http://www.openbsd.org/faq/faq15.html***
**General porting information: *http://www.openbsd.org/porting.html***
**Porting Checklist: *http://www.openbsd.org/checklist.html***

**Listing 5.** Generating checksums

```
% make makesum
===>  Checking files for ncdu-1.3
`/usr/ports/distfiles/ncdu-1.3.tar.gz' is up to date.
% ls
Makefile distinfo
```

**Listing 6.** Extracting the sources

```
% make extract
===>  Checking files for ncdu-1.3
`/usr/ports/distfiles/ncdu-1.3.tar.gz' is up to date.
>> (SHA256) ncdu-1.3.tar.gz: OK
===>  Extracting for ncdu-1.3
dec170% ls
Makefile   distinfo   w-ncdu-1.3
```

**Listing 7.** Configuring the sources

```
% make configure
===>  Patching for ncdu-1.3
===>  Configuring for ncdu-1.3
configure: loading site script /usr/ports/infrastructure/db/config.site
checking for a BSD-compatible install... /usr/bin/install -c -o root -g
bin

checking whether build environment is sane... yes
checking for a thread-safe mkdir -p... ./install-sh -c -d
...
```

**Listing 8.** Building the sources

```
% make build
===>  Building for ncdu-1.3
make  all-recursive
Making all in src
cc -DHAVE_CONFIG_H -I. -I..      -O2 -pipe -MT browser.o -MD -MP -MF
.deps/browser.Tpo -c -o browser.o browser.c
...
```

**Listing 9.** Installing files into the fake framework

```
% make fake
===>  Faking installation for ncdu-1.3
Making install in src
test -z "/usr/local/bin" || .././install-sh -c -d "/usr/ports/mystuff/
sysutils/ncdu/w-ncdu-1.3/fake-i386/usr/local/bin"

install -c -s -o root -g bin -m 555 'ncdu' '/usr/ports/mystuff/sysutils/
ncdu/w-ncdu-1.3/fake-i386/usr/local/bin/ncdu'
Making install in doc

test -z "/usr/local/man/man1" || .././install-sh -c -d "/usr/ports/
mystuff/sysutils/ncdu/w-ncdu-1.3/fake-i386/usr/local/man/man1"
install -c -o root -g bin -m 444 './ncdu.1' '/usr/ports/mystuff/sysutils/
ncdu/w-ncdu-1.3/fake-i386/usr/local/man/man1/ncdu.1'
```

put binaries in /usr/local, but the ports system sets the DESTDIR environment variable, which acts (or should act), as an over-ride install path. This is why this porting target is called fake: see Listing 9.

The software is installed into a fake scaffold of a root file system in WRKINST, which for this port is w-ncdu-1.3/fake-i386.

At this stage we are ready to start making some meta-data to be included in the package. We make a nicely formatted long description in the file pkg/DESCR: see Listing 10.

ncdu is an ncurses version of the famous old du unix command. It provides a fast and easy interface to your harddrive. Where is your disk space going? Why is your home directory that large? ncdu can answer those questions for you in just a matter of seconds!

The fmt -72 command wraps words so that they fit nicely on a standard 80 character wide terminal. This command will wait for input from standard input. You will probably want to paste a description from the project web page, followed by Enter then [CTRL]+[D]. Next we make a packing list. See Listing 11.

We have not yet checked the library configuration of the package. The package tools use library information to decide how to update packages when pkg_add -u is executed (amongst other things). We can check the library configuration like so: see Listing 12. This is ports telling us that we missed some library information in the port. To resolve this we add a line to our Makefile:

```
WANTLIB =                    c form ncurses
```

If you now re-run the check you should not receive any errors. Will talk more about library dependencies in the next part of this series. Now comes package creation see Listing 13.

**Listing 10.** Creating package meta-data

```
% mkdir pkg
% fmt -72 > pkg/DESCR
```

**Listing 11.** Creating plist

```
% make plist
===>  Updating plist for ncdu-1.3
/usr/ports/mystuff/sysutils/ncdu/pkg/PLIST is new
```

**Listing 12.** Checking the library linkage of the port

```
% make port-lib-depends-check
ncdu-1.3:

WANTLIB:        c.48 (/usr/local/bin/ncdu) (system lib)
WANTLIB:        form.3 (/usr/local/bin/ncdu) (system lib)
WANTLIB:        ncurses.10 (/usr/local/bin/ncdu) (system lib)
        WANTLIB += c form ncurses
*** Error code 1 (ignored)
```

**Listing 13.** Creating a package

```
% make package
`/usr/ports/mystuff/sysutils/ncdu/w-ncdu-1.3/fake-i386/.fake_done' is up
to date.

===>  Building package for ncdu-1.3
Create /usr/ports/packages/i386/all/ncdu-1.3.tgz
Link to /usr/ports/packages/i386/ftp/ncdu-1.3.tgz
Link to /usr/ports/packages/i386/cdrom/ncdu-1.3.tgz
```

**Listing 14.** Installing the package

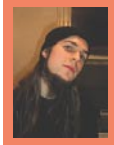
```
% make install
===>  Verifying specs:  c form ncurses
===>  found c.48.0 form.3.0 ncurses.10.0
===>  Installing ncdu-1.3 from /usr/ports/packages/i386/all/
ncdu-1.3: complete

dec170% which ncdu
/usr/local/bin/ncdu
% pkg_info | grep ncdu
ncdu-1.3            ncurses-based du(1)
```

## Conclusions

You can install the package with the install target, which simply calls pkg_add(1). You can now indulge in your new program, safe in the knowledge that the software is all accounted for in the package database: see Listing 14. Unfortunately it is very rare for packages to build and install so cleanly, so next time we will dissect less cooperative port and see what steps we must take to get them packaged to run on OpenBSD.

## About the Author



**Edd is a BSc Hons Computing student at Bournemouth University in the UK. He works mostly with C, C++, Java, Python, Ruby, /bin/sh and PHP. He is also a TeX user and member of the TeX user group. Edd was respobsible for bringing the TeX Live typesetter suite port to OpenBSD. He has just finished working for Bournemouth University doing UNIX system administration and a small amount of teaching. Edd is a bit of a metalhead and enjoys going to festivals.**

# Play Music
## on Your Slug With NetBSD

Donald T. Hayford

In an earlier issue of BSD magazine, we learned how to boot NetBSD on the Linksys NSLU2 (Slug), which I'm sure left a lot of your friends, spouses, or significant others wondering just why you spend so much time on this stuff.

So this time, let's do something a little more useful and teach our Slugs to play music. Though there are a number of ways to do this, we'll use a piece of software called *SqueezeCenter* (formerly known as *SlimServer*) that is available from Logitech at *www.slimdevices.com*. *SqueezeCenter* is an open source software package that will stream your music to a Squeezebox, a slick little device that connects to your network and outputs music to your stereo or powered speakers in either analog or digital (TOSLINK/SPDIF) form. Logitech has published the interface specifications for the Slim/Squeeze players; see *http://wiki.slimdevices.com/index.php/SLIMP3ClientProtocol*, for example. We'll use that same protocol to play music with our Slugs using a common USB audio device. Best of all, we won't have to do much in the the way of code writing, since others have already done the heavy lifting for us with software that runs on the Slug and emulates the Squeezebox. At the end of the article, we'll take a brief look the Slim data protocol so you can see how it works.

## What You'll Need

A Linksys NSLU2. Actually, what we'll do here will work on just about anything that runs NetBSD, but most of us won't want to cram our desktop computers into our stereo/TV cabinets when we're finished – that would make those friends/spouses/others start to wonder again. While Linksys no longer makes the Slug, the techniques we use here should work on any embedded processor with a USB and Ethernet port that has enough memory and can run NetBSD. However, you could still buy the Slug when this article was written.

A NetBSD-compatible USB device. How do you know if it is NetBSD-compatible? I don't know, but both of the ones I tried (one is from SIIG, the other from AOC) worked just fine. Though I'm no expert, USB audio devices seem to be pretty standard.

A desktop/laptop computer that has your mp3 music and that you can install *SqueezeCenter* on.

A computer (can be the same as in 3) that you can use to build NetBSD and also use as a NFS server for booting the Slug. A Linux, FreeBSD, or NetBSD system is recommended.

## Setting Up NetBSD

In the previous issue, we had to modify the Slug by adding an external serial port to use as the root console. This time, we'll work with a stock NSLU2, doing all of the setup through the network. The only problem with doing it this way is that NetBSD-current, which you will need to put NetBSD on your Slug, doesn't always reboot correctly. If you added the serial port, you just type *reboot* at the debug prompt, but if you didn't, you'll have to unplug then restart your Slug instead of rebooting. It still wouldn't hurt to remove the resistor that cuts your Slug's clock speed in half (if you have it – the more recent Slugs don't). If you want to do this, see *http://www.nslu2-linux.org/wiki/HowTo/OverClockTheSlug*.

To install NetBSD on the Slug, we'll follow the procedure from the NetBSD community wiki article *http://wiki.netbsd.se/How_to_install_NetBSD_on_the_Linksys_NSLU2_%28Slug%29_without_a_serial_port%2C_using_NFS_and_telnet*. Because we need to install audio, we'll have to change a few things in the kernel configuration file, but otherwise the steps are the same. For completeness, the steps required to retrieve and build NetBSD's world and kernel are shown in Listing 1. Note that most of the output from the build computer is suppressed for clarity. I have found that NetBSD builds pretty painlessly on a variety of different Linuxes, though I have had problems on a machine that runs Fedora 7 on an AMD64 processor. In that case, I was able to get the build to work by adding the three exports shown in Listing 1. If you experience problems during the build, refer to the file `~/net/src/BUILDING` for more hints.

Naturally, NetBSD builds pretty well on NetBSD machines, as well.

Adding USB audio to the kernel requires two additional lines in the kernel, one to add the driver for the USB hardware, and the other to link that driver to the standard audio driver. Both lines are shown in Listing 1. Additionally, we want to boot the Slug using NFS, so we add another line for that. Including the standard NSLU2 configuration file rounds out our new configuration file. While you can create the configuration file with a text editor such as vi, here I've created the file `NSLU2_AUDIO` using echo, as shown in lines 12-16.

After you've finished building NetBSD, you will need to set up your NFS server and TFTP servers following the instruction in NetBSD wiki article. Listing 2 shows the steps I used to setup up the NFS server and directory structure. Listings 3 through 11 show the actual files (in most cases) – you'll need to change your files to match these, making whatever changes that are appropriate for your setup for IP addresses and the like.

## Booting Up the Slug

Now it's time to boot up your slug. I highly recommend downloading and building the program from the nslu2-linux.org wiki *http://www.nslu2-linux.org/wiki/HowTo/TelnetIntoRedBoot* (Unfortunately, this program doesn't have a name on the *nslu2-linux.org* website, so we'll just call it `telnet_slug`. You can find it at *http://www.nslu2-linux.org/wiki/HowTo/TelnetIntoRedBoot* under the heading C program using Berkeley Sockets) that interrupts the Slug while it is booting from it's internal flash memory. That way, you can start that program running before you go off and reset the Slug without having to race back to your computer to enter [*Ctrl-C*] within two seconds. (Remember, this month it's all about style.) Plug the USB audio adapter (NetBSD can't run the USB audio device through a hub, so make sure you plug it directly into the Slug's USB port either one. If you add a disk drive later, that will work through a hub) in and start the `telnet_slug` program. Turn the Slug on and you should see the output at the start of Listing 12. The first time you boot, it will take a little time to check the disk drives, so be patient. You should be able to ping your Slug within several minutes if everything is going ok. A few minutes after ping starts working, you should be able to `telnet` into the Slug. Note that the Slug's IP address is set by the DHCP

**Listing 1.** Command line input to build NetBSD kernel and world

```
(1)$ mkdir ~/net
(2)$ export CVS_RSH="ssh"
(3)$ export CVSROOT="anoncvs@anoncvs.NetBSD.org:/cvsroot"
(4)$ cd ~/net
(5)$ cvs checkout -D 20080420-UTC src
(6)$ cd ~/net/src
# Add the next three lines if you get an error building NetBSD
(7)$ export HOST_SH=/bin/bash
(8)$ export HOST_CC=/usr/bin/gcc
(9)$ export HOST_CXX=/usr/bin/g++
(10)$ ./build.sh -m evbarm -a armeb tools
(11)$ cd ~/net/src/sys/arch/evbarm/conf
(12)$ echo 'include "arch/evbarm/conf/NSLU2"' >NSLU2_AUDIO
(13)$ echo 'uaudio* at uhub? port ? configuration ?' >>NSLU2_AUDIO
(14)$ echo 'audio* at uaudio?' >>NSLU2_AUDIO
(15)$ echo 'config netbsd-aud-npe0 root on npe0 type nfs' >>NSLU2_AUDIO
(16)$ cat NSLU2_AUDIO
include "arch/evbarm/conf/NSLU2"
uaudio* at uhub? port ? configuration ?
audio* at uaudio?
config netbsd-aud-npe0 root on npe0 type nfs
(17)$ cd ~/net/src
(18)$ ./build.sh -u -U -m evbarm -a armeb build
# follow the instructions in ~/net/src/sys/arch/arm/xscale/ipx425-
fw.README
# to get and build the NSLU2 Ethernet microcode
(19)$ ./build.sh -u -U -m evbarm -a armeb -V KERNEL_SETS=NSLU2_AUDIO
release
```

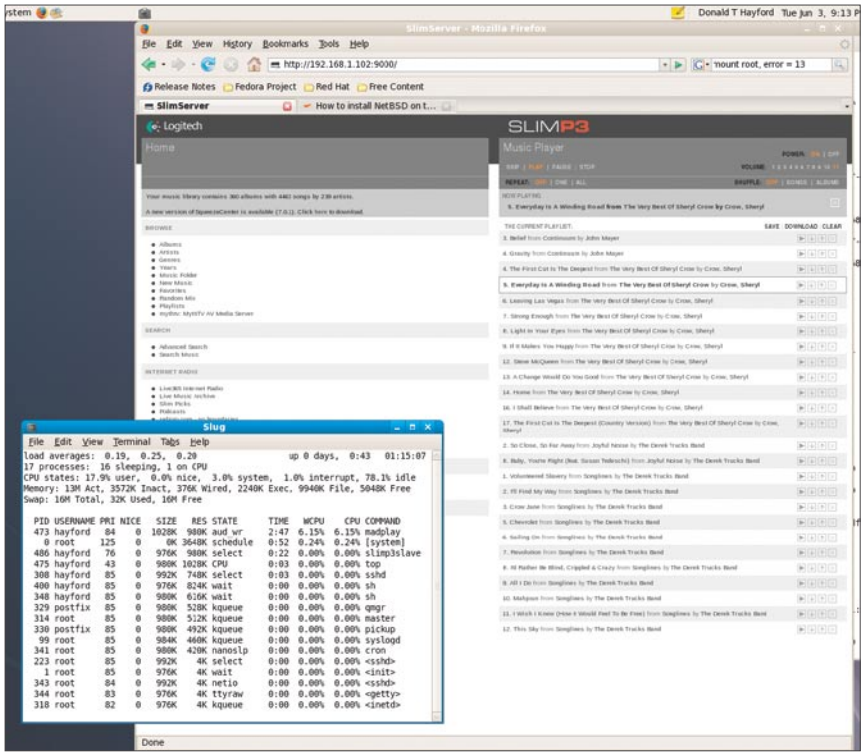



**Figure 1.** Screenshot of the Slug running top while playing music in the background

server. One common error is to have two DHCP servers on your network, the one that has the files that the Slug needs and your main router that connects to the Internet. If the router responds first, then the Slug won't boot up properly. Exit the copy of `telnet` that was started by the `telnet_slug` program and restart `telnet` with the Slug's new address. You should be able to log in as root (no password required), set up a user name, and add passwords for both your user and root.

**Listing 2.** Command line input to set up the NetBSD root drive and tftp/NFS server

```
(1)$ sudo mkdir -p /export/aud_client/root/dev
(2)$ sudo mkdir /export/aud_client/root/home
(3)$ sudo touch /export/aud_client/swap
(4)$ sudo dd if=/dev/zero of=/export/aud_client/swap
bs=4k count=4k
(5)$ sudo chmod 600 /export/aud_client/swap
(6)$ sudo mkdir /export/aud_client/root/swap
(7)$ sudo cp -r ~/net/src/obj/releasedir /export/aud_
client
(8)$ cd /export/aud_client/root
(9)$ sudo tar --numeric-owner -xvpzf /export/aud_
client/releasedir/evbarm/binary/sets/base.tgz
(10)$ sudo tar --numeric-owner -xvpzf /export/aud_
client/releasedir/evbarm/binary/sets/comp.tgz
(11)$ sudo tar --numeric-owner -xvpzf /export/aud_
client/releasedir/evbarm/binary/sets/etc.tgz
(12)$ sudo tar --numeric-owner -xvpzf /export/aud_
client/releasedir/evbarm/binary/sets/games.tgz
(13)$ sudo tar --numeric-owner -xvpzf /export/aud_
client/releasedir/evbarm/binary/sets/man.tgz
(14)$ sudo tar --numeric-owner -xvpzf /export/aud_
client/releasedir/evbarm/binary/sets/misc.tgz
(15)$ sudo tar --numeric-owner -xvpzf /export/aud_
client/releasedir/evbarm/binary/sets/tests.tgz
(16)$ sudo tar --numeric-owner -xvpzf /export/aud_
client/releasedir/evbarm/binary/sets/text.tgz
(17)$ cd dev
(18)$ sudo sh ./MAKEDEV -m ~/net/src/obj/
tooldir.YOUR.SYSTEM.HERE/bin/nbmknod all
(19)$ sudo sh ./MAKEDEV -m ~/net/src/obj/
tooldir.YOUR.SYSTEM.HERE/bin/nbmknod audio
(20)$ sudo cp -r /export/aud_client/root/etc /export/
aud_client/root/etc.orig
(21)$ cd /export/aud_client/root/etc
(22)$ sudo nano hosts
(23)$ sudo nano fstab
(24)$ sudo nano inetd.conf
(25)$ sudo nano rc.conf
(26)$ sudo nano ttys
(27)$ sudo nano ifconfig.npe0
(28)$ cd /export/aud_client/root
(29)$ sudo tar --numeric-owner -xvpzf \
/export/aud_client/releasedir/evbarm/binary/sets/kern-
NSLU2_AUDIO.tgz
(30)$ cp *.bin /tftpboot/
(31)$ sudo nano /etc/dhcpd.conf
(32)$ sudo nano /etc/exports
(33)$ sudo nano /etc/hosts
(34)$ sudo nano /etc/hosts.allow
(35)$ cd /export/aud_client/root/usr
(36)$ sudo ftp ftp.NetBSD.org
Trying xxx.xxx.xxx.xxx...
Connected to ftp.NetBSD.org (xxx.xxx.xxx.xxx).
220 ftp.NetBSD.org FTP server (NetBSD-ftpd 20060923)
ready.
Name (ftp.NetBSD.org:hayford): anonymous
331 Guest login ok, type your name as password.
Password:
ftp> cd pub/pkgsrc/current
250 CWD command successful.
ftp> get pkgsrc.tar.gz
local: pkgsrc.tar.gz remote: pkgsrc.tar.gz
227 Entering Passive Mode (xxx,xxx,xxx,xxx,xxx,xxx)
150 Opening BINARY mode data connection for
'pkgsrc.tar.gz' (35739284 bytes).
35739284 bytes received in 236 secs (1.5e+02 Kbytes/
sec)
ftp> bye
(37)$ sudo tar --numeric-owner -xvpzf pkgsrc.tar.gz
(38)$ sudo killall -HUP rpc.mountd
(39)$ sudo /etc/rc.d/init.d/nfs start
Starting NFS services:                          [  OK  ]
Starting NFS quotas:                            [  OK  ]
Starting NFS daemon:                            [  OK  ]
Starting NFS mountd:                            [  OK  ]
(40)$ sudo /sbin/service dhcpd restart
Shutting down dhcpd:                            [  OK  ]
Starting dhcpd:                                 [  OK  ]
```

**Listing 3.** NFS server and Slug file /etc/hosts

```
# Do not remove the following line, or various
programs
# that require network functionality will fail.
127.0.0.1               nfsserver
localhost.localdomain localhost
::1                 localhost6.localdomain6 localhost6
192.168.1.102           nfsserver
192.168.1.240           slug1
```

**Listing 4.** Slug file /etc/fstab, found at /export/aud_client/root/etc

```
nfsserver:/export/aud_client/swap   none  swap
sw,nfsmntpt=/swap
nfsserver:/export/aud_client/root   /     nfs   rw 0 0
```

**Listing 5.** Slug file /etc/ifconfig.npe0, found at /export/aud_client/root/etc

```
inet client netmask 255.255.255.0 broadcast
192.168.1.255
```

Let's make sure the USB sound device was recognized by NetBSD. Use `dmesg` as shown in Listing 12, line 10 to verify that the necessary drivers were installed. Notice that the format of the information written out by dmesg matches what we used in our configuration file, except that `*'s` and ?'s are replaced by numbers by the operating as it installs the software for each driver.

Open another terminal window at this point and make sure you can connect to the slug using `ssh`. Also, make sure you can su to root (see lines 12 and 13 in Listing 12). At this point, you can exit `telnet` and use `ssh` to login with your username. If you're the security-conscious type, you can disable telnet at this point by editing the Slug's `/etc/inetd.conf`, but don't do this until you're sure you can log in with `ssh`.

## Making Music

Next, we'll use the package system to build *madplay*, a program with a few libraries that will convert MP3 files to linear 16-bit audio and send it to your USB sound device (see lines 20 and 21 in Listing 12). It will take a while to build *madplay*. While it's building, add a music directory to the NetBSD root/usr directory on your NFS server and put a few of your favorite MP3 in there so we can test *madplay* when the build is finished. Test out *madplay* as shown in Lines 1 and 2 of Listing 13. You should hear your music playing on your Slug. If you don't want anybody to know what you're up to, you can hook a set of earphones to the USB dongle while you test things out.

Once we know *madplay* works, we need to download two files from the Internet to your desktop computer you can transfer them over to the Slug. The first file, `slimp3slave-0.4.tar.gz`, is available from *http://www.ex-parrot.com/~pdw/slimp3slave/*. The second file is a patch file for the source code in `slimp3slave.c`, called `slimp3slave.c-stinga-patch-01.txt`, and is available from *http://forums.slimdevices.com/attachment.php?attachmentid=321&d=1127457503*. Create a directory /home/(your-user-name)/slim and copy the files to the slug as shown in Listing 13. Untar the `slimp3slave` file and apply the patch as shown. There is another correction to the `slimp3slave.c` file that is shown in Listing 14 that will prevent some unnecessary printing from corrupting the display when you run `slimp3slave`. Build the program using the provided make file and move a copy of the executable to a location in your path (usually, `bin` in your home directory is included in your path).

Download *SqueezeCenter* from *http://www.slimdevices.com/* and install it on your desktop. You'll need to tell the program where your music is located, but the installation is fairly painless. A really nice feature of *SqueezeCenter* is the web-based interface that you can use to control what music is being played (note that the default port for the web interface is 9000, not 80 as for most web-based servers). Start the `slimp3slave` program on your slug, as shown in Line 13 of Listing 13, and open the *SqueezeCenter* web interface. Figure 1 shows the *SqueezeCenter* screen (I'm running version 6.5.4, so the screen may look slightly different from what you see) along with a overlaid window with the Slug running top. In it's default mode, `slimp3slave` invokes *madplay* to decode and play the mp3 stream coming from the *SqueezeCenter* server. Note that the *madplay* uses only about 20% of the available CPU to convert the mp3 files to the linear audio stream that is played by the USB sound dongle. Not bad for a little feller like the Slug.

The command line shown in Line 13 of the listing will run the `slimp3slave` player, but if you close the ssh session to your Slug, `slimp3slave` stops. However, the program will run in the background as shown in Line 14 (don't forget the trailing `&`). Once you start `slimp3slave` in this mode, you can close the ssh session and `slimp3slave` will continue to run on the Slug. Then use the *SqueezeCenter*

**Listing 6.** Slug file /etc/ttys, found at /export/aud_client/root/etc

```
#       $NetBSD: ttys,v 1.5 2004/06/20 21:30:27 christos Exp $
#
#       from: @(#)ttys  5.1 (Berkeley) 4/17/89
#
# name  getty                           type    status          comments
#
console "/usr/libexec/getty default"    vt100   on secure
ttyp0   "/usr/libexec/getty Pc"         vt100   off secure
ttyE0   "/usr/libexec/getty Pc"         vt220   off secure
ttyE1   "/usr/libexec/getty Pc"         vt220   off secure
ttyE2   "/usr/libexec/getty Pc"         vt220   off secure
ttyE3   "/usr/libexec/getty Pc"         vt220   off secure
tty00   "/usr/libexec/getty default"    unknown off secure
tty01   "/usr/libexec/getty default"    unknown off secure
tty02   "/usr/libexec/getty default"    unknown off secure
tty03   "/usr/libexec/getty default"    unknown off secure
tty04   "/usr/libexec/getty default"    unknown off secure
tty05   "/usr/libexec/getty default"    unknown off secure
tty06   "/usr/libexec/getty default"    unknown off secure
tty07   "/usr/libexec/getty default"    unknown off secure
```

**Listing 7.** First twelve lines of Slug file /etc/inetd.conf, located at /export/aud_client/root/etc

```
#       $NetBSD: inetd.conf,v 1.58 2007/10/16 02:47:14 tls Exp $
#
# Internet server configuration database
#
#       @(#)inetd.conf  8.2 (Berkeley) 3/18/94
#
#http   stream  tcp   nowait:600  _httpd  /usr/libexec/httpd  httpd /var/www
#http   stream  tcp6  nowait:600  _httpd  /usr/libexec/httpd  httpd /var/www
#ftp    stream  tcp   nowait      root    /usr/libexec/ftpd   ftpd -ll
#ftp    stream  tcp6  nowait      root    /usr/libexec/ftpd   ftpd -ll
telnet  stream  tcp   nowait      root    /usr/libexec/telnetd telnetd
telnet  stream  tcp6  nowait      root    /usr/libexec/telnetd telnetd
```

**Listing 8.** Slug file /etc/rc.conf, located at /export/aud_client/root/etc

```
#       $NetBSD: rc.conf,v 1.96 2000/10/14 17:01:29
wiz Exp $
#
# see rc.conf(5) for more information.
#
# Use program=YES to enable program, NO to disable it.
program_flags are
# passed to the program on the command line.
#
# Load the defaults in from /etc/defaults/rc.conf (if
it's readable).
# These can be overridden below.
#
if [ -r /etc/defaults/rc.conf ]; then
        . /etc/defaults/rc.conf
fi
# If this is not set to YES, the system will drop into
single-user mode.
#
rc_configured=YES
# Add local overrides below
#
sshd=YES
hostname="slug1"
defaultroute="192.168.1.1"
nfs_client=YES
auto_ifconfig=NO
net_interfaces=""
```

**Listing 9.** NFS server file /etc/hosts.allow

```
#
# hosts.allow   This file describes the names of the
hosts which are
#               allowed to use the local INET
services, as decided
#               by the '/usr/sbin/tcpd' server.
#
in.tftpd:       192.168.0.1
rpcbind:        192.168.1.240
lockd:          192.168.1.240
rquotad:        192.168.1.240
mountd:         192.168.1.240
statd:          192.168.1.240
```

**Listing 10.** TFTP server /etc/xinetd.d/tftp

```
$ cat /etc/xinetd.d/tftp
# default: off
# description: The tftp server serves files using the
trivial file transfer \
#   protocol.  The tftp protocol is often used to boot
diskless \
#   workstations, download configuration files to
network-aware printers, \
#   and to start the installation process for some
operating systems.
service tftp
{
        disable = no
        socket_type             = dgram
        protocol                = udp
        wait                    = yes
        user                    = root
        server                  = /usr/sbin/in.tftpd
        server_args             = -s /tftpboot
        per_source              = 11
        cps                     = 100 2
        flags                   = IPv4
}
```

**Listing 11.** DHCP server file /etc/dhcpd.conf

```
$ cat /etc/dhcpd.conf
#
# DHCP Server Configuration file.
#   see /usr/share/doc/dhcp*/dhcpd.conf.sample
#
ddns-update-style ad-hoc;
option subnet-mask 255.255.255.0;
option broadcast-address 192.168.1.255;
option domain-name-servers xxx.xxx.xxx.xxx,
yyy.yyy.yyy.yyy;
default-lease-time 2592000;
#max-lease-time 28800;
allow bootp;
allow booting;

#option ip-forwarding    false;  # No IP forwarding
#option mask-supplier    false;  # Don't respond to
ICMP Mask req

subnet 192.168.1.0 netmask 255.255.255.0 {
        option routers          192.168.1.1;
        range 192.168.1.110 192.168.1.189;
        }

group   {
      next-server 192.168.1.102;          # IP address
of your TFTP server
      option routers 192.168.1.1;
      default-lease-time 2592000;
      host slug1 {
                hardware ethernet 00:18:39:a2:26:7c;
                fixed-address 192.168.1.240;
                option root-path "/export/aud_client/
root";
                }
         }
```

web interface to start, stop, and select the music that plays on the Slug.

When you get it all working right, connect the dongle output to your home stereo, sit back and enjoy some tunes. When your spouse, significant other, or regular friends ask where you got the cool music player, act like it's no big deal. But tell your geek friends you read about it in *BSD Magazine*. Win-win.

The `slimp3slave.c` program has a small *feature* in that it doesn't look up the Slug's actual MAC address to send back to the server. What this means is that you can only have one copy of `slimp3slave` running on your network, since all devices will report a MAC address of 00:00:00:00:00:00 and the server will think that the same player is just changing it's network address. I leave it as an exercise for the reader to find and fix this feature. If you only have one Slug, don't worry about it.

## How It All Works

The data interface between `slimp3slave` and *SqueezeCenter* is captured in a single function inside `slimp3slave.c`, `read_packet()`, which can be found starting at Line 682 in the original file. After a connection between the machine running `slimp3slave` (the player) and the machine running *SqueezeCenter* (the server) has been established, the server sends packets to the player consisting of an 18-byte header and data. The contents of the data depend on the packet type, which is determined by the first byte of the header. If the first byte is an `l`, the contents of the packet data is alphanumeric information that should be displayed on the 2-line, 20 char player display. An `m` as the first byte indicates an MP3-encoded data packet that contains the music (or a portion of it) to be played. This data is buffered up and sent to a music player (*madplay* on the Slug) using a operating system pipe. Pipes are special files that are used by the operating system (NetBSD and other posix-compatible systems) to connect the input of one program to the output of another. As `slimp3slave` receives data from the server, it buffers it up and the operating system will

**Listing 12.** Slug first bootup and building madplay

```
(1)$telnet_slug
== Executing boot script in 1.220 seconds - enter ^C
to abort
Telnet escape character is '~'.
Trying 192.168.0.1...
Connected to 192.168.0.1.
Escape character is '~'.
(2)RedBoot> load -r -b 0x200000 -h 192.168.0.2 netbsd-
aud-npe0.bin
Using default protocol (TFTP)
Raw file loaded 0x00200000-0x004bad53, assumed entry at
0x00200000
(3)RedBoot> g
~
(4)telnet> q
Connection closed.
(5)$ telnet slug1
Trying 192.168.1.240...
Connected to slug1.
Escape character is '^]'.

NetBSD/evbarm (slug1) (ttyp0)
(6) login: root
Copyright (c) 1996, 1997, 1998, 1999, 2000, 2001,
2002, 2003, 2004, 2005,
    2006, 2007, 2008
    The NetBSD Foundation, Inc.  All rights reserved.
Copyright (c) 1982, 1986, 1989, 1991, 1993
    The Regents of the University of California.  All
rights reserved.
NetBSD 4.99.60 (NSLU2_AUDIO) #0: Wed Jun 4 21:32:16
EDT 2008
Welcome to NetBSD!
[snip...]

We recommend creating a non-root account and using
su(1) for root access.
(7)slug1# useradd -G wheel -m hayford
(8)slug1# passwd hayford
Changing password for hayford.
New Password:
Retype New Password:
(9)slug1# passwd root
Changing password for root.
New Password:
Retype New Password:
(10)slug1# dmesg | grep audio
uaudio0 at uhub2 port 1 configuration 1 interface 0: C-
Media INC. USB Sound Device, rev 1.10/0.10, addr 2
uaudio0: audio rev 1.00
audio0 at uaudio0: full duplex, independent
(11)slug1# echo 'nameserver xxx.xxx.xxx.xxx' >/etc/
resolv.conf
(12)slug1# su hayford
(13)$ su
Password:
(14)slug1# exit
(15)$ exit
(16)slug1# exit
slug1# logout
Connection closed by foreign host.
(17)$ ssh hayford@slug1
(18)Password:
NetBSD 4.99.60 (NSLU2_AUDIO) #0: Wed Jun 4 21:32:16
EDT 2008

Welcome to NetBSD!

(19)$ su
Password:
(20)slug1# cd /usr/pkgsrc/audio/madplay
(21)slug1# make install clean
```

periodically run *madplay* to consume the data generated by `slimp3slave`. Since *madplay* understands both the MP3 data format and how to play MP3 data on the audio device, the data sent by the server ends up coming out of speakers that you connected to your Slug's audio dongle. This is an excellent example of how *nix software uses smaller, special-purpose filters, like `slimp3slave` and *madplay*, to make software development easier.

## Conclusions

Using software components that are freely available on the web, you can build a sophisticated music player for your home using inexpensive components such as Linksys NSLU2 and a standard USB audio device. This device can be remotely controlled using any web browser, and the *SqueezeCenter* software allows you to start and stop the player, set up playlists, add music, and much more.

**Listing 13.** Testing madplay and building the slimp3slave player

```
(1)slug1# rehash
(2)slug1# madplay *mp3
MPEG Audio Decoder 0.15.2 (beta) - Copyright (C) 2000-
2004 Robert Leslie et al.
>> 01 - Volunteered Slavery.mp3
          Title: Volunteered Slavery
         Artist: The Derek Trucks Band
          Album: Songlines
          Track: 1
          Genre: Blues
error: frame 0: lost synchronization cd
^C

#execute the following commands on your desktop
computer after downloading slimp3slave and the patch
(3)$ scp slimp3slave-0.4.tar.gz slug1:/home/hayford/
slim
Password:
slimp3slave-0.4.tar.gz                        100%
8992     8.8KB/s   00:00
(4)$ scp slimp3slave.c-stinga-patch-01.txt slug1:
/home/hayford/slim
Password:
slimp3slave.c-stinga-patch-01.txt             100%
920      0.9KB/s   00:00
#execute the remainder of this listing on your Slug
(5)$ cd slim
(6)$ tar -xzf ./slimp3slave-0.4.tar.gz
(7)$ cd ./slimp3slave-0.4
/home/hayford/slim/slimp3slave-0.4
(8)$ patch <../slimp3slave.c-stinga-patch-01.txt
Hmm...  Looks like a unified diff to me...
The text leading up to this was:
--------------------------
|--- slimp3slave.c-01   2004-02-15 18:51:45.000000000
+0000
|+++ slimp3slave.c      2005-09-23 08:33:29.145997440
+0100
--------------------------
Patching file slimp3slave.c using Plan A...
Hunk #1 succeeded at 132.
Hunk #2 succeeded at 747.
Hunk #3 succeeded at 865.
Hunk #4 succeeded at 887.
done
(9)$ vi ./slimp3slave.c
(10)$ make
cc -O2 -g -Wall  -c slimp3slave.c
slimp3slave.c: In function 'loop':
slimp3slave.c:749: warning: 'p' may be used
uninitialized in this function
cc -O2 -g -Wall  -c util.c
cc  -g -o slimp3slave slimp3slave.o util.o -lcurses
(11)$ mkdir ~/bin
(12)$ cp slimp3slave ~/bin
#install and start Squeezecenter on your desktop
(13)$ slimp3slave -l -s 192.168.1.102
(14)$ slimp3slave -b -s 192.168.1.102 &
```

**Listing 14.** Add "if(debug)" after the else in function receive_mpeg_data() of slimp3slave.c

```
void receive_mpeg_data(int s, receive_mpeg_header*
data, int bytes_read) {

    if(debug)
        warn("Address: %d Control: %d Seq: %d \n",
ntohs(data->wptr), data->control, ntohs(data->seq));

    if(data->control == 3) {
        ring_buf_reset(outbuf);
    }

    playmode = data->control;

    if(playmode == 0 || playmode == 1) {
        if(output_pipe == NULL) {
            if(debug)
                warn("Opening pipe\n");
            output_pipe = output_pipe_open();
        }
    }
    else {
        if(debug)
            warn("Playmode: %d\n", playmode);
        if(output_pipe != NULL) {
            if(debug)
                warn("Closing pipe\n");
            output_pipe_close(output_pipe);
            output_pipe = NULL;
        }
    }
```

# In the next issue:

## A giude to PC BSD

Many interesting articles and tutorials about security, administration, multimedia and others.

On the DVD: PC-BSD Fibonacci  Edition

**Next issue of BSD magazine available in March !**

# Interview about NetBSD WAPBL

## With Simon Burge, Antti Kantee and Greg Oster

Federico Biancuzzi: Could you introduce yourself?

**Simon Burge:** I have been working for Wasabi Systems for about eight years now, and been involved with NetBSD for about 15 years.

I originally started with NetBSD to work on the PC532, and I was doing most of the recent maintenance on this port until the start of this year when unfortunately a lack of ELF binutils for ns32k and no ns32k support in gcc4 pretty much killed it off and it was removed from NetBSD.

I've also done a lot of work with some of the MIPS ports, especially the pmax port earlier on.

Now it is a pretty much a bit of anything when I get the chance.

**Antti Kantee:** I've been a NetBSD developer since the last millenium and have gotten my hands dirty with all the major kernel subsystems. Currently I work on my PhD thesis and misc. consulting jobs.

**Greg Oster:** I have been a NetBSD developer for 10 years and while I have poked around in many parts of the kernel my primary responsibility is RAIDframe (the software RAID driver). In my day job I'm a Laboratory Systems Analyst in the Department of Computer Science at the University of Saskatchewan.

FB: What is WAPBL?

**Simon Burge:** WAPBL stands for *Write Ahead Physical Block Logging*. WAPBL provides metadata journaling for file systems. In particular, it is used with the fast file system (FFS) to provide rapid file system recovery after a system outage. It also provides better general-use performance over regular FFS through less on-disk metadata updates – these are coalesced in the journal.

WAPBL was developed by Wasabi Systems, and recently Wasabi contributed that work back to NetBSD. Wasabi has been using WAPBL in its storage products for about four or five years now.

FB: How did you integrate it with FFS?

**Simon Burge:** Darrin Jewell did the original implementation of WAPBL – he might be in a better position to answer questions on the original implementation and integration.

In more recent times, Antti ported the WAPBL code from NetBSD 4.0 to NetBSD -current. Andrew helped tidy a few locking issues up with that as well. I added support for an in-filesystem journal (the original implementation used a log are between the end of the filesystem and the end of the partition) and Greg helped in the discussion about how that was done. Greg has also looked after the documentation and has done a lot of testing.

FB: What features does WAPBL provide on NetBSD-current right now[August 2008]?

**Simon Burge:** The two main features of WAPBL are fast file system recovery and in general increased metadata performance.

The fast file system recovery works when your system panics or loses power and doesn't shutdown cleanly. When your system restarts, any file systems with logging enabled will skip the potentially long fsck phase and WAPBL will replay any outstanding metadata transactions when the file system(s) are mounted. With the large disks of today an fsck can take half an hour or more – with WAPBL you skip this entirely!

The increased metadata performance comes from WAPBL aggregating metadata updates (any operations on directories and inodes like creating removing files) in the journal, whereas

normal FFS writes each of these operations out synchronously.

WAPBL is in the same ballpark as soft dependencies for most operations. The one known workload that WAPBL is slower is when the fsync(2) system call is used – this causes the journal to be flushed to disk each time.

### FB: Do you have any benchmark result?

**Simon Burge:** A reasonably common benchmark used by NetBSD people is extracting pkgsrc.tar.gz. Here is the time to extract that with various mount options on one system:

```
normal 1.489u 12.201s 18:29.87
log 1.296u 10.531s 0:37.78
softdep 1.555u 10.015s 0:33.00
async 1.426u 9.273s 0:20.66
```

and `rm -rf pkgsrc` times for removing that pkgsrc tree:

```
normal 0.115u 3.609s 9:46.81
log 0.075u 3.415s 0:14.70
softdep 0.084u 1.387s 0:15.32
async 0.125u 2.401s 0:12.29
```

### FB: In which contexts should WAPBL fit better? And when should we avoid it?

**Simon Burge:** Currently, file system snapshots (ffs(4)) do not work with WAPBL. This is being addressed and should be fixed before the next release.

In general, WAPBL should be relatively the equivalent to soft dependencies for most workloads. The one known area where it isn't is when the fsync(2) system call is involved. Most databases use this, as well as the CVS server (but not the client). Some mailers might use this as well, so WAPBL might not suite a high volume mailserver.

### FB: How can we use it?

**Simon Burge:** Currently WAPBL isn't available for NetBSD 4.0 but I'm hoping to make this available *soon*. Using WAPBL is as simple as making sure you have *options WAPBL* in your kernel config file (this is the default on most architectures now), and either using `mount -o log ...` or using `rw,log` in the mount options field of your `/etc/fstab`.

FB: Can we use some partitions with softupdates and some others with WAPBL?

**Simon Burge:** The only restriction is that you can't use both WAPBL and soft dependencies on any single file system. You are certainly free to have both active on different filesystems on the one machine.

FB: I saw on the mailing list that you are thinking at how to deal with the fact that fsck is not aware of WAPBL and might create problems. How do you plan to solve this?

**Simon Burge:** This is still under debate right now, so I don't have a simple answer. Note that -current fsck is aware of WAPBL. The situation we're trying to guard against is when you take a file system that has had WAPBL active on it and put it on an older system – think of say an external USB disk.

We're trying to guard against people unknowingly shooting themselves in the foot, but it is not a problem that you'll run in to in day-to-day running.

FB: How much space does WAPBL require to work?

**Simon Burge:** An in-filesystem log is sized according to the total file system size. 1MB of log is allocated per 1GB of disk space, up to a maximum of 64MB. This is the same way that Solaris uses to determine the log size. You can use a larger log than this either by specifying a log size with tunefs before you mount the file system, or by using an end-of-partition log after the filesystem. As far as limits, there might possibly be some 32-bit limits in the log size...

FB: Does WAPBL interefere with backup software such as dump(8)?

**Simon Burge:** WAPBL should be no different to soft dependencies in this respect – they both can delay writing out metadata so there is potential for dump(8) to not catch some files if they have been recently modified in some way.

Once file system snapshots work with WAPBL (and I saw a commit go by today that should enable this but haven't looked at the details), you will be able to use `dump -X` do make a consistent backup.

FB: How did you test and debug WAPBL?

**Simon Burge:** Ah, that reminds me RUMP. It allows you to run unmodified kernel code in userspace (taken straight from *http://www.NetBSD.org/docs/puffs/rump.html*. RUMP was really quite handy when writing the code that handles in-filesystem logs with WAPBL. Instead of rebooting with a new kernel to test new code, I was just able to run a simple program, and debug any issues with gdb.

It was also a lot safer working on a simple file system image in a file. I could have done this with a small file system on partition or vnd vnode disks, but again this was much simpler with RUMP.

**Greg Oster:** The final stress testing had a couple of phases. The first was to run multiple, n-way simultaneous extracts of src.tar.gz, with a spacing of 10 seconds between the start of one and the start of the next. So this started with a single src.tar.gz extract, followed by two src.tar.gz extracts, all the way up through 10 simultaneous src.tar.gz extracts. This phase was repeated a few times.

The second phase of final stress testing consisted of doing continuous `./build.sh -j 8 ...` builds on freshly extracted src.tar.gz source trees (extract, build, delete, repeat). I don't recall how manybuild cycles were done, but the machine spent about 56 hours in this phase, all without a single issue. At that point I felt we were ready to merge WAPBL into -current.

FB: What steps are needed to setup WAPBL via RUMP?

**Antti Kantee:** It requires a kernel with puffs support, either directly compiled into the kernel with *file-system PUFFS* or loaded as a module. Also, a system build with `MKPUFFS=yes` in `/etc/mk.conf` is required.

Then simply run `rump_ffs -o log device mountpoint` instead of `mount_ffs -o log device mountpoint`.

FB: Is there any plan to port WAPBL and/or RUMP to NetBSD 4.0?

**Antti Kantee:** The original patches supplied by Wasabi Systems were against NetBSD 4.0, so in theory WAPBL for NetBSD 4.0 exists already, although it lacks features such as in-fs log support (wasn't this already covered by the earlier questions). It took considerable effort to get WAPBL running on NetBSD-current because of the vast amount of SMP architectural changes done by Andrew Doran since NetBSD 4.0.

There are no plans to port rump to NetBSD 4.0. However, it should be noted, that since rump runs completely in userspace, it should be possible compile the rump code from NetBSD-current and run that on NetBSD 4.0. There may be some pitfalls, like such as libpthread on NetBSD 4.0 not supporting all the necessary routines. Most of them should be related to diagnostics and should therefore not be difficult to workaround for anyone interested in the task.

FB: What license covers the WAPBL code provided by Wasabi?

**Antti Kantee:** It is available under the standard 2-clause BSD license. The copyright has been assigned to The NetBSD Foundation.

FB: Is WAPBL suitable to small embedded systems too? Does it add too much additional work on the cpu or the disk?

**Greg Oster:** I haven't played with WAPBL on any embedded systems, but my understanding is that WAPBL was developed for use on Wasabi's storage products (which basically have an embedded system in them).

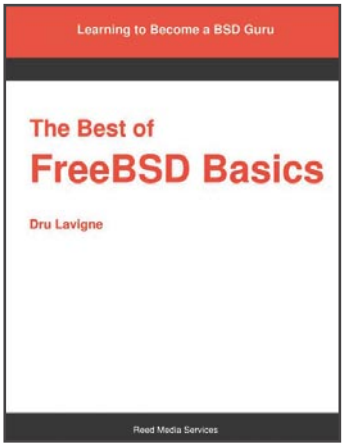
I've done some benchmarking, but none which would point out any overhead associated with WAPBL. I was pleasantly surprised to find that even with the journal located after the filesystem that performance issues related to seek times were basically non-existant. (at least on non-legacy hardware).

I think the short answer to the question is: Yes, WAPBL is suited for small, embedded systems, and no, it does not add significant overhead to the CPU or disk systems (at least on non-legacy hardware).

FB: Thank you for your time.

*Federico Biancuzzi*

# Dru Lavigne's The Best of FreeBSD Basics




Like probably most BSD oriented people, I first encountered Dru Lavigne's work via her onlamp.com online columns, dubbed *FreeBSD Basics*, which she started writing some time in 2000. I no longer remember which of her columns was the first I read or just how I found it. That's not important anyway; the Web and the Internet in general has become a very different place in the meantime.

I do however remember why I bookmarked FreeBSD Basics in my browser and kept coming back for more. The columns were short enough that you could get useful insights into a specific topic, about the software you either had installed already or something that was very easily within reach in a space of time that would be long enough for you to be feeling only a little guilty about extending the break. Well written too, and they would invariably contain something you would find useful later or something you just had not thought of earlier.

Those of us who wanted to see something by Dru in print were rewarded in 2004 when O'Reilly published *BSD Hacks*, a collection of short tips and tricks for things to do on your BSD system. The collection of online columns kept growing, too.

If you've been searching the Internet for FreeBSD related material, it is likely that you have found one or more of the columns online. Now they have been edited into this book, updated to be in sync with FreeBSD versions available in late 2007. From a writer's view, it must have been tempting to just put the original columns end to end and slap a cover around them, but fortunately Dru Lavigne instead chose to organize the material into logical groupings and edit the columns into nine chapters where the subtopics fit together quite pleasantly. The individual columns have been turned into subtopics in the chapters, and for each subtopic the URL to the original column is given along with other references. The columns started out as field notes taken from real-world situations, and the practical approach shines through here, helping to make the printed version very readable.

The book's chapters progress from the *basics*-oriented *Becoming Familiar with FreeBSD*, through gradually more advanced and varied topics under the chapter headings *Useful Unix Tricks*, *Ports, Packages, and PBIs*, *Bag of Tricks*, *Filesystems*, *Backups and RAID*, *Networking*, *Configuring Services*, *Security*, *Firewalls and VPNs*. With a foreword by Greg Lehey, the author's preface and a comprehensive index at the end, the book weighs in at just short of 600 numbered pages.

The book is slightly over the volume you could expect to get through in a single sitting, but this is a book that will stay with you for a long time, and the time you spend reading it (from cover to cover, *browsing or just zooming in on specific topics from the table of contents or index) is time well spent. The fact that the book started out as a collection of shorter pieces comes back as a distinct advantage; the various subtopics in each chapter have been edited together so the overall flow is very good, while at the same time every not-too-many pages you will get a feeling of tangible achievement as the author rounds off the current topic before moving you along to the next one. In each topic, you get enough of the reasoning behind the commands and options to get a good sense of the whys as well as the whats, and you get the references to look up in case you need a slightly different variation for your situation.*

But even if it is certainly a valuable resource for beginners, you do not need to be a Unix newbie to enjoy this book. If you are the more experienced, Unix-savvy kind of reader, you will still find yourself going *Ah! I hadn't thought of that!* every now and then. This is the kind of book where every reader will have their favorite sections. I have several, I particularly appreciate the filesystems chapter, which among other things offers an excellent description of the general Unix permissions system as well as a good discussion of FreeBSD's filesystem access control lists (ACLs). On the other hand, every chapter has one or more sections that likely will grab your attention.

For example, the *Useful Unix Tricks* chapter contains a very readable find tutorial as well as an overview of Unix processes, explores the init system, teaches you about how to use cron effectively as well as a number of other topics I'll let you discover for yourself.

Then there's the networking chapter, where you will find not only the 'how to get my box on the net' and a concise TCP/IP basics part, but also information on how to interface productively with Cisco equipment and a delightful wireshark section, along with other related and useful topics.

If you are looking for information on using a BSD as a desktop working environment, this book has several very useful sections for you, as well as the networking and server oriented parts that (for good reasons) tend to dominate in Unix books. No BSD book would be complete without a thorough exploration of the ports and packages system, and this one has a full (and very good) chapter on the topic as well as references to useful packages and ports where appropriate in other sections.

Your favorite part of this book may well be a different one than the ones I have pointed out specifically here. Every chapter contains a wealth of well-written, interesting and useful topics, and just about the only thing I miss at this point is some sort of electronic version. This is the kind of book you would very much like to have available on a portable device. I hope Reed Media Services will move quickly to make PDF or other versions available.

The Best of FreeBSD Basics covers a wide range of topics, and covers them well. The book does not pretend to cover anything besides FreeBSD (or in some parts PC-BSD, a FreeBSD derivative), but it is written with sufficient reasoning behind the specifics and large enough chunks of background information that it is in fact useful as a general Unix learning resource.

Summing up, if you are an instructor looking for an up to date Unix book or a FreeBSD text, this could very well be what you are looking for. Seasoned Unix pros and greybeards as well as users of other BSDs or Linux will delight in the wide range of topics covered and are likely to find new ideas or angles, and will enjoy having this volume within easy reach. For somebody learning FreeBSD, there is only one possible recommendation: Get this book. It deserves that spot right next to your keyboard. You will find some other spot for your coffee cup, easily.

*by Peter N. M. Hansteen*

# Worth reading

## NetBSD System Manager's Manual (SMM)

Two volume set containing a permuted index and the definitive and official NetBSD system operation and maintenance manuals from ac through zic.

## The BSD Associate Study Guide

The beginning BSD Unix administration book covers the objectives for the BSD Certification Group's BSDA (BSD Associate) Certification focused for BSD Unix system administrators with light to moderate skills. The community-written, open source book covers generic *BSD administration and specific skills as necessary for NetBSD, FreeBSD, OpenBSD, and DragonFly.

## Getting started with NetBSD

by Jeremy C. Reed

This book quickly introduces NetBSD -- from installation, standard setup, maintenance, standard system operations, and common use of NetBSD. It covers installation using sysinst, the first login, system startup, users and groups, networking setup and troubleshooting, DNS, system clock, software packages, pkgsrc (for easily installing software from source), email services, cron and scheduled tasks, log files, disks and file systems, custom kernels, updating NetBSD, security, performance monitoring and tuning, packet filtering, X11 and graphical interfaces, popular applications, multimedia, and printing.

## Building a high-performance computing cluster using FreeBSD

by Brooks Davis

This book covers architectural decisions, deployment, and experiences with development and maintenance of general purpose technical and scientific computing clusters running the FreeBSD operating system. It includes configuration management, network booting of nodes, scheduling, design issues, security, future expansion, and introduces parallel programming toolkits and applications.

Coming soon...

## The pfSense Handbook

by Christopher M. Buechler

The book covers the installation, configuration, and maintenance of pfSense, an open source, customized distribution of FreeBSD tailored for use as a firewall and router, entirely managed in an easy-to-use web interface. The book covers hardware, network designs, firewalling, packet filtering, network address translation, routing, WANs, traffic shaping, IPsec, PPTP, VPNs, load balancing, wireless access points, virtualization, CARP and redundancy, DHCP, DNS, and other various services and special purpose appliances. No FreeBSD knowledge is required to deploy and use pfSense.

Coming soon...

# BSD CERTIFICATION.ORG

# NEW CERTIFICATION EXAM

**The BSD Certification Group proudly announces the availability of the BSD Associate Exam (BSDA), the entry level exam for BSD System Administrators.**

The BSD Associate Exam is a written proctored certification exam in English only. The BSDCG has worked hard to make this psychometrically valid exam affordable worldwide. See the list of selected conferences and register for an exam seat for $75 USD at **www.bsdcertification.org**.

**Get Involved.**
**Get Certified.**
**Get Ahead.**

**www.iXsystems.com**

iXsystems is a proud sponsor of BSD Certification Group Inc.

NetBSD FreeBSD OpenBSD DragonFlyBSD

# GEMINI SERVERS

## THE POWER OF TWIN SERVERS

## The iXsystems Gemini Server Class

iXsystems offers an array of server configurations as part of our "Open Source Hardware Design" process. Open Source Hardware Design refers to our carefully crafted server product line, designed and manufactured to ensure operating system and cross-platform compatibility, while providing best in class performance and reliability for any business requirement.

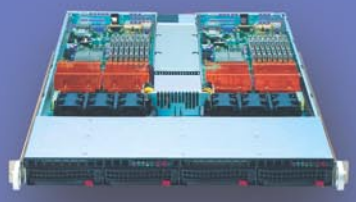
Our Gemini class of servers have superior processing power density and performance/Watt, making them ideal solutions for high-performance computing (HPC) cluster nodes, web servers, or rendering node clusters. They also offer the highest number of CPU cores per rack.

The versatility of these two systems, coupled with these energy saving features, makes them an excellent choice for HPCs, server farms, and other datacenters where space, cost, energy-efficiency, and density are high priorities.

For customers whose needs fall outside of our product line, we offer our custom server solution design process. This means the creation of a custom, open source hardware solution that addresses a company's technical and budgetary needs within their specific network architecture.

iXsystems is the all-around FreeBSD company that builds FreeBSD-certified servers and storage solutions, runs the FreeBSD Mall, and is the corporate sponsor of the PC-BSD Project. For more information about our Gemini class of servers contact iXsystems at (408)943-4100 or visit our website at http://www.ixsystems.com/gemini and fill out the inquiry form. One of our expert sales professionals will provide you with a customized quote that best meets your open source hardware solution needs.





## iX12X2

### ULTIMATE PERFORMANCE

- Two systems/nodes in a 1U form factor
- Each node supports up to two Intel Xeon Dual/Quad-Core CPUs and up to 1600 MHz Front Side Bus
- Total of four hot-swap SATA drives, 16 cores, and 16 FBDIMM, or up to 128 GB memory in a 1U form factor (Per node: Two hot-swap SATA drives and 8 FBDIMM slots provide up to 64 GB memory per node).
- 800 Mhz 1.5 volt FBDIMM memory
- A 980-watt high efficiency (90%+) power supply is shared by both nodes to optimize the utilization level and increase energy savings.
- Remote and Lights Out Management w/IP-KVM, SOL via IPMI 2.0

## iX12X2-10G

### INTEGRATED 10 GIGABIT ETHERNET & HIGHEST POWER EFFICIENCY

- Two systems/nodes in a 1U form factor
- Each node supports up to two Intel Xeon Dual/Quad-Core CPUs and up to 1333 MHz Front Side Bus
- Total of eight hot-swap drives, 16 cores, and 12 DIMM slots, or up to 96 GB memory in a 1U form factor (Per node: Four hot-swap 2.5" SATA drives and 6 DIMM slots provide up to 48GB memory per node).
- 10GigE Ethernet connection
- A 5100 Intel San Clemente chipset that takes registered memory, increasing power efficiency over systems with fully buffered memory.
- A 780-watt high efficiency (90%+) power supply is shared by both nodes to optimize the utilization level and increase energy savings.

iXsystems

Enterprise Servers for Open Source