FREE DVD INSIDE PC-BSD 7.0 FIBONACCI EDITION

MAGAZINE

BSD

INS&OUTS

PC-BSD UNCOVERED

EXCLUSIVELY

▶ **Interview with Matt Olander and Kris Moore**

INSIDE

PC-BSD and the PBI System
Personalizing Your PC-BSD Desktop
Using FreeBSD for Off-Site Backups
Building NetBSD for Embedded System Using Cygwin
Open Source Studio to Transmitter Link (OSSTL)
Django FreeBSD

Vol.2 No.2 USD14.99
2/2009(4) ISSN 1898-9144

60 PAGES OF PRACTICAL TUTORIALS

V7.1

PC-BSD Edition V 7.1 Codename Galileo

The PC-BSD Experience

PC-BSD is a stable and secure operating system with a smooth and seamless user experience. New features introduced in PC-BSD Version 7.1 Galileo Edition include updated versions of KDE (4.2), FreeBSD (7.1 stable), and Xorg (7.4). Furthermore, this latest version of PC-BSD features full proxy support for PBI and system updates, thin client server support, drastically improved nVIDIA performance and speed increases, and a new backup and recovery tool.

PC-BSD Version 7.1 Galileo Edition features everything current users love about PC-BSD, including a fully functional desktop operating system running FreeBSD under the hood, a graphical system installer to make the system installation process effortless, safety from viruses and spyware that plague other operating systems, and self-installing software packages that make loading programs a snap.

For current PC-BSD Fibonacci users, the upgrade path from Fibonacci to the Galileo Edition is relatively painless. With the PBI stand alone upgrade, double click and install the PBI file to upgrade from Fibonacci to Galileo.

Alternatively, the online update tool will notify you when an update from Fibonacci to Galileo is available. For new Galileo users, the installation process is only 8 to 12 minutes long and takes a couple of clicks for installation to be complete.

Professional Enterprise Grade PC-BSD Support is also available from iXsystems for those looking to maximize their PC-BSD systems' performance. This means guaranteed response time support with customized service plans for any OS-related problem. iXsystems also partners with some of the most brilliant minds in the FreeBSD community to offer custom development and advanced level FreeBSD and PC-BSD consulting solutions.

Experience the state-of-the-art, stable and secure open source operating system that is revolutionizing the desktop and server market. Download PC-BSD Galileo Edition today from http://www.pcbsd.org. Or, buy the box set or USB drive version from http://www.freebsdmall.com. For more information about PC-BSD technical support contact iXsystems at (408)943-4100 or visit http://www.ixsystems.com and fill out the inquiry form.

MAGAZINE BSD

Editor in Chief: Ewa Dudzic
ewa.dudzic@bsdmag.org
Executive Editor: Karolina Lesińska
karolina.lesinska@bsdmag.org
Editor Assistant: Marta Kobus
marta.kobus@bsdmag.org

Director: Ewa Dudzic
ewa.dudzic@bsdmag.org

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

Contributing: Jan Stedehouder, Donald T. Hayford, Jerry Dixon, James T. Nixon III, Eric Vintimilla, Amjith Ramanujam, Dan Fairs, Jason Ellison, Matt Olander, Federico Biancuzzi, Mikel King

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
Marta Kobus
marta.kobus@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,

At the very beginning, we had some doubts about the BSD mag project. We thought, well... there is no magazine devoted to this operating systems, but at the same time we had been wondering why nobody came up with this idea... first thougth – there is no interest in the market. But we decided to take the risk and see how it all works. We ended up being right. It turned out that you really need this mag and finally you have something prepared especially for you – not for open source software user – but you!

From the very beginning you gave us great support and help. Working with you is a real pleasure and in my opinion we became a great team!

Since you are all used to and support the idea of a main focus in each issue – this time we devoted the whole mag to one distribution – PC-BSD. When preparing each issue we try to give you the most useful tips and practical knowledge. Jerry Dixon guides you through the process of installation and configuration of PC-BSD and Jan Stedehouder shows the best way of installing software on a FreeBSD-based system. We also prepared many how-to's that will help you to develop your skills. James T. Nixon III discusses personalizing PC-BSD desktop. Eric Vintimilla guides you through using FreeBSD for the off-site backup. Donald T. Hayford demonstrates how to build NetBSD for embedded systems using Cygwin. Amjith Ramanujam presents ZFS – a state of the art filesystem on FreeBSD 7. Dan Fairs introduces Django: a web framework for perfectionists with deadlines.

In the multimedia section, Jason Ellison tells a story about the use of open source software in a local radio station.

For those who don't realy feel like getting more into details, BSD team and Federico Biancuzzi prepared an interview with PC-BSD folks: Kris Moore, the founder of the PC-BSD project, and Matt Olander, Chief Technology Officer at iXsystems. In the column Mikel King gives us his thoughts about BSD and its use in general.

Don't forget, I love hearing from you, so keep the mails coming in.

all the best

Karolina Lesińska
Executive Editor

what's new

dvd content

get started

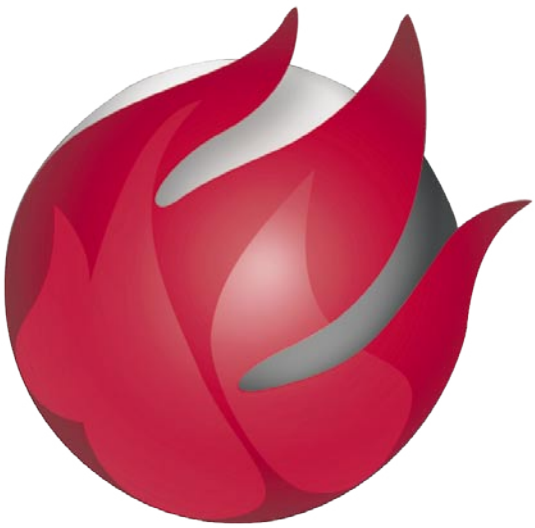
how-to's

mms

tips&tricks

interview

column



MeetBSD California 2008 Recap

MeetBSD California 2008 was a resounding success! Over 200 registered attendees participated in this unprecedented event, consisting of a 2-day conference with BSD-related talks, and culminating in the 15 year anniversary party for FreeBSD.

iXsystems was a primary sponsor and the organizer of the meetBSD California conference, which took place at the Google campus in Mountain View November 15-16, 2008. The 15 year FreeBSD anniversary party took place at the Zen Lounge, also in Mountain View, on the evening of Saturday November 15.

MeetBSD is a conference devoted to the BSD operating system, with a strong emphasis towards the FreeBSD family. MeetBSD was started in Krakow, Poland in 2004 and has continued every year since then. Last year the conference was held in Warsaw, Poland.

MeetBSD enlightened even the most sophisticated BSD user with lively discussion on a wide range of topics, including Xen Virtualization, the BSD Certification Process, Crypto Acceleration, and the ZFS File System. Other topics of discussion included Isolating Cluster Jobs for Performance and Predictability, Embedding FreeBSD, and FreeBSD Network Stack Performance Optimizations for Modern Hardware.

For more information and to see pictures from this year's MeetBSD conference visit the meetBSD California website at *http://www.meetbsd.com*.

The FreeBSD Foundation

The FreeBSD Foundation is now in its ninth of supporting the FreeBSD project and community. We had a banner year for supporting projects, sponsoring conferences, and providing travel grants.

The goal of the FreeBSD Project is to provide software that may be used for any purpose – and without strings attached. Our mission is to support the FreeBSD Project and community. Our funding comes from people like you – those who are determined to keep FreeBSD free!

How did we spend the money last year?

- Sponsored FreeBSD related conferences like BSDCan, EuroBSDCon, AsiaBSDCon, meetBSD, and NYCBSDCon. We also sponsored FreeBSD developer summits in Ottawa and Cambridge.
- Provided 23 travel grants and funding to individuals to attend these conferences this year.
- Provided legal support for the project on issues like understanding the GPLv3 impact on FreeBSD, providing a privacy policy, trademark ownership and permission, and other legal issues that come up.
- Provided grants for projects that improve FreeBSD, like Java binaries, Network Stack Virtualization, Improving Hardware Performance Counter Support, making improvements to the TCP stack, making FreeBSD tolerate the removal of active disk devices, and a couple of other projects that we will be announcing soon.
- Provided equipment for developers working to improve FreeBSD and projects like the NetPerf cluster. Facilitated donation of NetApp filer, 32-core hardware, and 10 Gigabit equipment for project continuity planning and the NetPerf Cluster.

How can you help?

Your financial support is critical for the FreeBSD Project. Please help us keep FreeBSD free. Go to *http://www.freebsdfoundation.org/donate/* to donate (any amount will help). And thank you for your continued support of the FreeBSD Foundation.

Upcoming conferences

BSDCan – The BSD Conference

BSDCan, a BSD conference held in Ottawa, Canada, has quickly established itself as the technical conference for people working on and with 4.4BSD based operating systems and related projects. The organizers have found a fantastic formula that appeals to a wide range of people from extreme novices to advanced developers.

BSDCan 2009 will be held on 8-9 May 2009 at University of Ottawa, and will be preceeded by two days of Tutorials on 6-7 May 2009.

Please subscribe to the announcement mailing list to be kept informed of changes as they are announced. To subscribe, please follow the instructions at *http://lists.bsdcan.org/mailman/listinfo/bsdcan-announce*.

AsiaBSDCon 2009

AsiaBSDCon is a conference for users and developers on BSD based systems. The next conference will be held in Tokyo, in 12-15 March, 2009. The conference is for anyone developing, deploying and using systems based on FreeBSD, NetBSD, OpenBSD, DragonFlyBSD, Darwin and MacOS X. AsiaBSDCon is a technical conference and aims to collect the best technical papers and presentations available to ensure that the latest developments in our open source community are shared with the widest possible audience.

Please visit *http://2009.asiabsdcon.org/ for more details.*

What happened at The MirOS Project

Since the release of MirOS #10 in March 2008, there has been quite some activity. Throughout the system, minor bugs were fixed and security patches applied; everything is now compiled as C99. Things are now much simpler yet more powerful: /root is gone (use sudo), shared code is now located in one instead of three places, Reiser CCCP is used by X11 and other applications now, mkisofs has been replaced by an overhauled makefs with machine-independent bootxx (MI installboot will follow), the SPARC port comes with an official boot floppy again, and the concept of DuaLive CDs was introduced: CD images that are bootable on both i386 and sparc, from CD, DVD, USB stick, hard disc, CF/SD card, network, and provide Live CD functionality on i386, Install CD functionality on both architectures. The entropy collecting and processing has been improved; VIA C3 Hardware AES encryption is used for UVM swapencrypt (and soon for vnd encryption); work has been invested in more reliable, 8-bit transparent Unicode support. The MirOS Licence is now OKFN and OSI approved; the main mirror and website now runs on a new server (a new CVS server is also planned). The MirPorts Framework has undergone a lot of fixes (even for use on MidnightBSD), and new ports (more recent Python; Fedora Linux emulation libs; kqemu, mcabber, Firefox 2 and 3, llvm/llvm-gcc/clang, pcc) have been introduced. There is even a patch for wchar_t and Unicode support courtesy of tg@ in pcc now. Over a Gibibyte of binary packages are available for MirOS #10/i386, courtesy of bsiegert@ with more to follow; automatic builds are more feasible now. The MirBSD Korn Shell (mksh) has been made more portable in addition to the bugfixes, and an upgrade to R36b at least is recommended, with plans for R37 to use a new memory allocator. Adam Hoka became a developer, working mostly on mksh and feeding back our patches (mostly makefs) to NetBSD for now. While the website has been improved (infrastructure wise, being PHP free and written in mksh and make; introduction of RSS feeds, tags, and SSL), plans are to improve the content (volunteers to help with this, as well as manual pages and install documentation, are always welcome). Long-term plans include importing feature (as opposed to security) improvements from the other BSDs, scheduled for before releasing MirOS BSD #11, delaying it as needed. Before the import will be started, a stabilised MirOS-current snapshot will however be issued, to be used in the meanwhile to get all recent improvements and fixes. Similarily, communication with the XFree86 developers about feeding back patches has started, and the codebase will be upgraded.

The MirOS Project representatives are available for discussion using our mailing lists, IRC (channels #MirBSD and #!/bin/mksh on Freenode PDPC), as well as several conferences throughout Europe: the Chaos Communication Congress (Berlin DE), FOSDEM (Bruxelles BE), Linuxtag (Berlin DE), FrOSCon (St. Augustin DE), and any others we manage to attend – sponsors welcome.

PC-BSD 7 Fibonacci Edition

PC-BSD 7 Fibonacci Edition is a free, easy to use Open Source desktop operating system available for 32bit (i386) and 64bit (amd64) platforms. It is based on FreeBSD 7-Stable, and is not a fork, which means it maintains 100% compatibility with FreeBSD. PC-BSD makes installing and running a FreeBSD desktop easy for both end users and for more experienced FreeBSD adopters.

Some of the ways PC-BSD makes BSD on the desktop easy include the following:

- Fully graphical system installer:
  - Provides simple-to-use initial setup of disk and users.
  - Supports advanced features such as custom partition layout, and file-system selection.
  - Allows selection of popular programs to be installed during setup (FireFox, OpenOffice and more).
- Graphical configuration utilities:
  - Networking and Wireless setup and monitoring.
  - System Administration and Common Tasks. (Portsnap, system source, etc).
  - User Manager with simple & advanced modes for management.
  - Basic Firewall tool, allows start / stop and creating exceptions to PF firewall rules.
  - Services Manager, start / stop various system services such as SSH and others.
  - Add / Remove Programs – Easily add desktop components or remove installed software.
- System Update service:
  - Allows users to easily upgrade to the latest version of PC-BSD via the internet
  - Notification and updates of older software packages
- KDE 4.1.x desktop environment.
- Includes support for Intel Wireless cards, Flash 9, and NVIDIA cards *out of box*.

In addition to these many features for the desktop, PC-BSD also introduces a new concept in open-source package management, with the PBI (Push Button Installer) system. (For the more hard-core users, traditional FreeBSD ports and packages can still be used) . The PBI format provides the backbone of package management in PC-BSD and is based upon a few core concepts:
Packages are *Self-Installing:*

- The PBI format doesn't require a massive package management tool to perform an install. Each package is a self-extracting binary which handles the entire installation process.
- Each package can create desktop and menu icons for the user, as well as register file associations as necessary.

Packages are *Self-Contained:*

- Contained within each PBI file are all the various libraries and binaries necessary for that program's operation.
- This helps eliminate the concept of numerous *dependencies* which can render a program unusable, or cause potential breakage with each library update.

A wide variety of programs in PBI format are available at *http://www.pbidir.com*

System Requirements

Minimum:

- Pentium II or higher
- 256MB Ram
- 6GB of free Hard Drive space (Either partition, or entire disk)

Recommended:

- Pentium 4 or higher
- 512MB of Ram
- 12GB of free Hard Drive space (Either partition, or entire disk)
- Network card
- Sound card
- 3D accelerated video card (NVIDIA or Intel)

Included Software

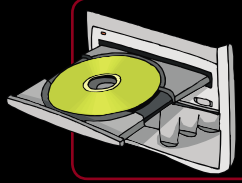
FreeBSD 7-Stable, KDE 4.1.1, Amarok 1.4.10, FireFox 3.0.0.1, K3B 1.0.5, OpenOffice 2.4.1, Opera 9.52, The Warden 0.9, Wine 1.1.3, NVIDIA Driver 177.70

Base System Updates

PC-BSD 7 Fibonacci Edition is the first release based on the new FreeBSD 7-Stable base, which brings with it a host of new features in the base system:

Dramatic improvements in performance and SMP scalability shown by various database and other benchmarks, in some cases showing peak performance improvements as high as 350% over FreeBSD 6.X under normal loads and 1500% at high loads. When compared with the best performing Linux kernel (2.6.22 or 2.6.24) performance is 15% better. Results are from benchmarks used to analyze and improve system performance; results with your specific work load may vary. Some of the changes are:

- The 1:1 libthr threading model is now the default.
- Finer-grained IPC, networking, and scheduler locking.
- A major focus on optimizing the SMP architecture that was put in place during the 5.x and 6.x branches.
- Some benchmarks show linear scaling up to 8 CPUs. Many workloads see a significant performance improvement with multicore systems.
- The ULE scheduler is vastly improved, providing improved performance and interactive response.
- Experimental support for Sun's ZFS filesystem.
- gjournal can be used to set up journaled filesystems, gvirstor can be used as a virtualized storage provider.
- Read-only support for the XFS filesystem.
- The unionfs filesystem has been fixed.
- iSCSI initiator.
- TSO and LRO support for some network drivers.
- Experimental SCTP (Stream Control Transmission Protocol) support (FreeBSD's being the reference implementation).
- Much improved wireless (802.11) support.
- Network link aggregation/trunking (lagg(4)) imported from OpenBSD.
- JIT compilation to turn BPF into native code, improving packet capture performance.
- jemalloc, a new and highly scalable user-level memory allocator.



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

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

Installing PC-BSD Fibonacci Edition

Jerry Dixon

Man pages? We don't need no stinkin' man pages. I don't need to show you any man pages. Well, that being said, at least review the hardware requirements prior to selecting your computer platform.

Checking the PC-BSD user guide, we find the minimum system requirements and the recommended system requirements.

Minimum Hardware Requirements

PC-BSD has moderate hardware requirements and commonly uses less resources than its commercial counterparts. Before installing PC-BSD on your computer, please make sure you meet the minimum requirements.

Minimum system requirements:

- Pentium II or higher
- 256MB RAM
- 6GB of free Hard Drive space (Either partition, or entire disk)
- Network card
- Sound card

Recommended system requirements:

- Pentium 4 or higher
- 512MB of RAM
- 12GB of free Hard Drive space (Either partition, or entire disk)
- Network card
- Sound card
- 3D accelerated video card (NVIDIA or Intel)

For our purposes, the recommended system requirements are now the minimum.

Leftovers: the best meals are leftovers

I selected a leftover Dell 4600C small form factor desktop system. Recently saved from the scrap yard, infested with over 600 Trojan down loader virus variants. (XP Pro) I formatted the 120 gig hard drive, thus removing all the pr0n, and changed the boot order to CD first.

Memory is your friend.. The more the merrier..

I used one gig ram (two 512 k chips) and all is well with the world. Do check the maximum capacity of your motherboard. Useless to buy a 2 gig chip if that capacity is not supported. And if the video card fails to measure up, the install will abort. Lucky for me, the Dell 4600C Intel video chip set was just right.

The installation DVD is included in this issue.

But wait, where is the installation DVD?

I think someone swiped my DVD.

Download the ISO files and burn to CD or DVD.

Free of course, *http://tinyurl.com/66to8f*

But can be purchased. *http://tinyurl.com/5wz83r*

This Dell didn't have a DVD, so I downloaded the first three ISO files and burned them to CD.

Now that the installation files have been downloaded, it's a good idea to check if the files are exactly the same as the one on the PC-BSD server, i.e. check the integrity of your *.iso* file. While downloading, some bits and bytes may get `damaged` or lost, so it's wise to check the integrity of the downloaded file.

Figure 1. pc-bsd install 1

Did you know that missing a few bits can make the CD unusable? One of the best (and free) data integrity check (also called MD5 checksum) programs for Windows is called Mat-MD5, but there's nothing wrong with WinMd5Sum (Figure 2).

Three CDs are needed if you wish to add extra programs during the install, Open Office, Firefox etc. You have now downloaded PC-BSD and checked if the download has gone OK. It's now time to burn the ISO to a CD or DVD. This section will assist you to burn PC-BSD onto a CD-R (Writable) or on a CD-RW (Rewritable). This tutorial assumes you already have:

- A CD-RW or DVD-RAM drive that allows you to burn media and not just read
- A writable medium in your DVD-RAM or CD-RW drive
- A file with a *.iso* extension containing PC-BSD
- A Windows or Unix computer to burn your ISO file.

At this point, you will have to choose how you want to burn the downloaded *.iso* file. If you use a Windows computer, you can choose from Nero, Burnatonce, ImgBurn or CDBurnerXP. If you use a Unix computer with KDE (such as PC-BSD), you can use K3B. Alternatively, if you use a Unix machine with Gnome, you can use Gnome Baker to burn your CD-ROM. And we also have instructions on how to burn the CD on the MAC. Nero worked for me (Figure 3).

PC-BSD doesn't come with a built-in partition manager. If you're planning to install PC-BSD on a partition, partitions have to be created before installing. Before creating or editing any partitions, make sure you back up your valuable data first.

PC-BSD can be installed on a PC as the sole operating system; this is the easiest way. Most people probably want to be able to run Windows or Linux (or both) and PC-BSD on the same computer. To accomplish this the hard drive need to be partioned, i.e. the disk has to be sectioned on parts for each of the operating systems (Disk partitioning is the creation of separate divisions of a hard disk drive using partition editors. Once a disk is divided into several partitions, directories and files of different categories may be stored in different partitions.)

So, there are two options for installing PC-BSD:

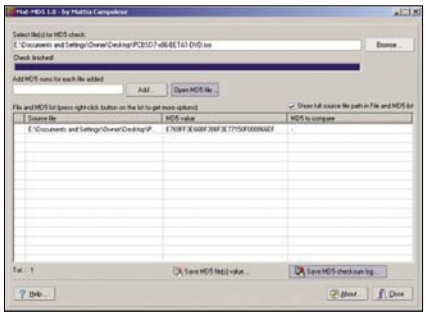


Figure 2. md5

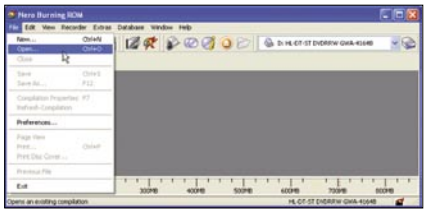


Figure 3. nero 1

Figure 4. Startup 1

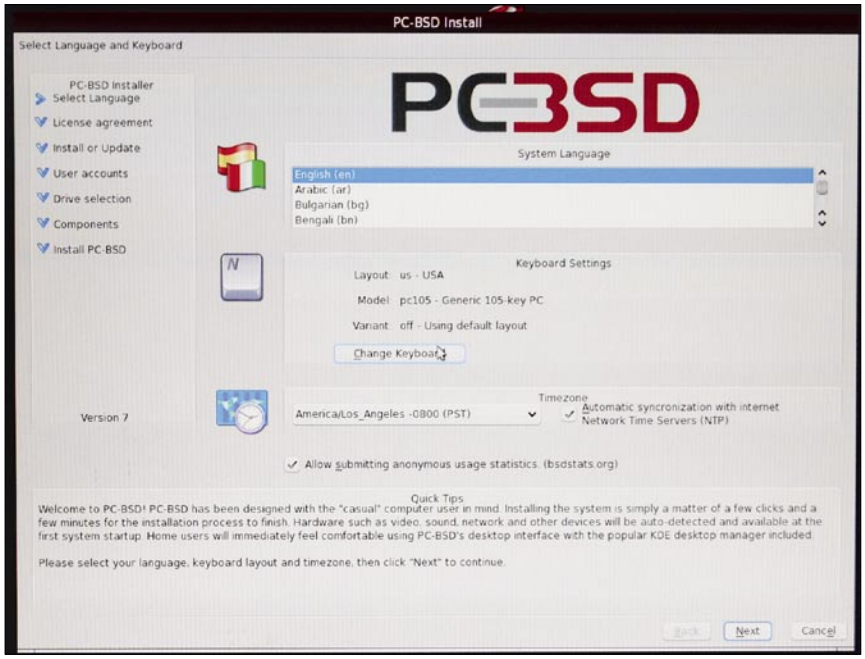


Figure 5. pc bsd install 2

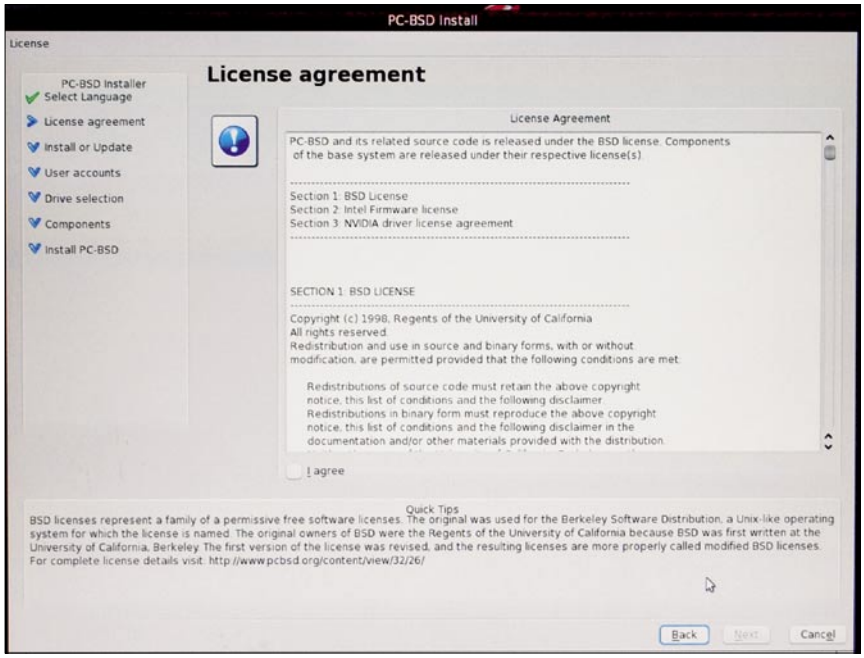


Figure 6. pc bsd install 4

- PC-BSD taking the whole hard disk drive
- PC-BSD installed on a partition of the hard disk drive with another operating system on another partition (dual boot, triple boot, multi boot)

Before you do any partitioning, ensure you have backed up all your valuable data, especially if this is the first time you install PC-BSD and you have saved data on another partition. Be aware that BSD operating systems, and hence PC-BSD, only recognize primary partitions and consider any logical partitions as a whole primary partition. Trying to install on a logical partition will convert your extended partition into a primary partition and erase all logical partitions of your system. PC-BSD can be installed on any primary partition; it doesn't necessarily have to be on the first one. Be careful and make sure you have a backup of your data.

PC-BSD can be installed from installation CD-ROMs and used on a *real* PC, or it can be downloaded as an VMware Player image and be run as a 'virtual computer' within Microsoft Windows. A Live CD is not available as yet.

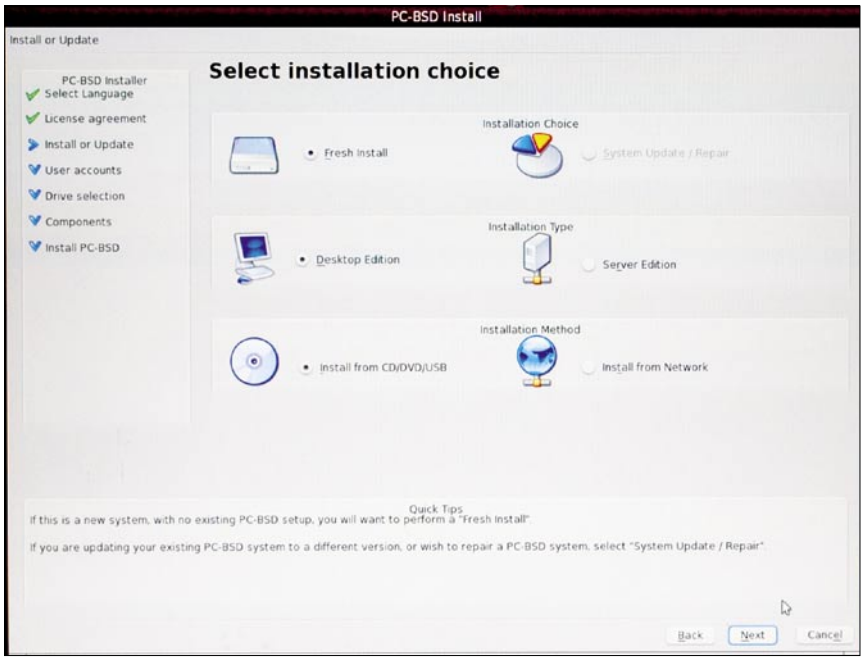


Figure 7. pc bsd install 5

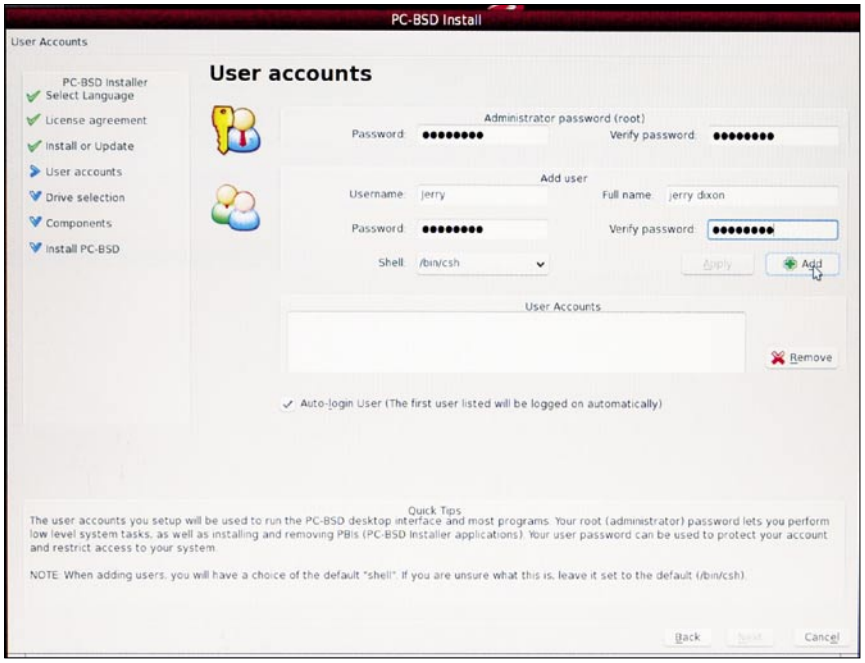


Figure 8. pc bsd install 5

The PC-BSD installer has made installing a Unix-like operating system as easy as installing Microsoft Windows. When installing PC-BSD you don't need to use the command line or text-based installers, neither do you have to manually edit configuration files. The installation of PC-BSD is a fast, easy and straight-forward process with a pretty looking Installer. An easy-to-use wizard will take you step-by-step through the whole process by asking a few simple questions and after a few minutes you will have your system up and running.

Though the installation process has been made as short and as easy as possible, there are still some advanced options available for power users. Yes, you can ignore the man pages. This distro is sweet.. boot it and forget it. Your first choice is a text screen.. (Figure 4). There are in total 9 options to choose from:

- *Boot PC-BSD [default]* – Normal bootup, start with all standard options enabled.
- *Boot PC-BSD with ACPI disabled* – This disables the ACPI device / power management, which may be useful for certain newer BIOS's and laptops `[advanced]`
- *Boot PC-BSD in Safe Mode* – Boot up PC-BSD with a forced PIO mode (disabling the use of DMA) and disables write caching for all IDE hard drives and CD ROM drives, disables the probing of EISA slots (as very few systems have them), and in i386 it also disables the use of ACPI and the APICs.
- *Boot PC-BSD in single user mode* – Boot to a shell prompt right after the kernel is finished loading.
- *Boot PC-BSD with verbose logging* – This option displays much, much more detail during the boot process.
- *Boot PC-BSD to emergency console* – Boot to a PC-BSD emergency text console, which can provide access to the hard drive, and allow fixing critical system failures.
- *Run installer in VESA mode* – Disable trying to auto-detect the video card driver, and instead default to VESA mode. Useful when option 1 fails to bring up the GUI installer.
- *Enable installer ZFS support* – This enable Sun's Zetabyte File System. This is for advanced users.

- *Escape to loader prompt* – Drop to the boot loader prompt to issue advanced commands, such as changing kernels, etc.

Option 1 (`"Boot PC-BSD [default]"`) is recommended for most users; Simply hit *Enter* or *1* to start the installation. The Graphical install is beyond easy.

As you can see, the installer is divided into four logical areas:

- The left pane with the different steps of the installer, showing the current and completed steps with a different arrow;
- The main area in the center where the installer expects user input;
- The quick tips at the bottom, always here to help you, to explain what each field means, and how to fill in the fields;
- The bottom navigation bar where you can go forward or backward.

While at the initial pane, you must select your language, keyboard layout, and time zone. You can synchronize your clock with the Internet if you want, or re-adjust your time manually for daylight savings, if needed in the future. If your keyboard is listed with the wrong details, click the *Change Keyboard* button to change the keyboard model, layout and variant.

First, select a language. How simple is that (Figure 5)? License agreement of course (Figure 6). And the type of install.. This install is a fresh install, a desktop edition, and from CD (Figure 7). Two choices interfere with your tea and crumpets or the frothy pint (Figure 8).

In this step, an Administrator account (or `root` account) has to be created, as well as at least one user account. Similar to other modern operating systems, PC-BSD has mainly two levels of administration:

- Administrator-level: Full control over the entire system, can manage files, software and users (`add/edit/remove`);
- User-level: Control limited to user's directory, cannot install applications system-wide, cannot edit files outside user directory

If you're security conscious or if you share the PC with others and you don't want others to access your files, unmark the checkbox for auto-login. You need to select the hard drive or "portion of" you plan on using. This step allows you to select which physical hard disk drive and which partition are going to be used for the installation. The *Detected Hard Drives* field lists the disks that PC-BSD has found during boot up, from which one needs to be selected. If you want PC-BSD to use the entire disk, for instance if you don't have any other operating systems on your PC, and no separate partition is used to store documents, you can check the *Use entire disk* option underneath to use all disk space of your hard disk drive. Be very careful when you select *Use entire disk* as it will overwrite all partitions!

In FreeBSD, and hence PC-BSD, each partition has a code in front of them, such as `/dev/ad0s1`, which is the path to the device (`dev`) file.

PC-BSD, starts counting disk drives from *0*, and starts counting partitions from *1*. Here are a few examples:

- `/dev/ad0s1` – First drive, first partition
- `/dev/ad1s1` – Second drive, first partition

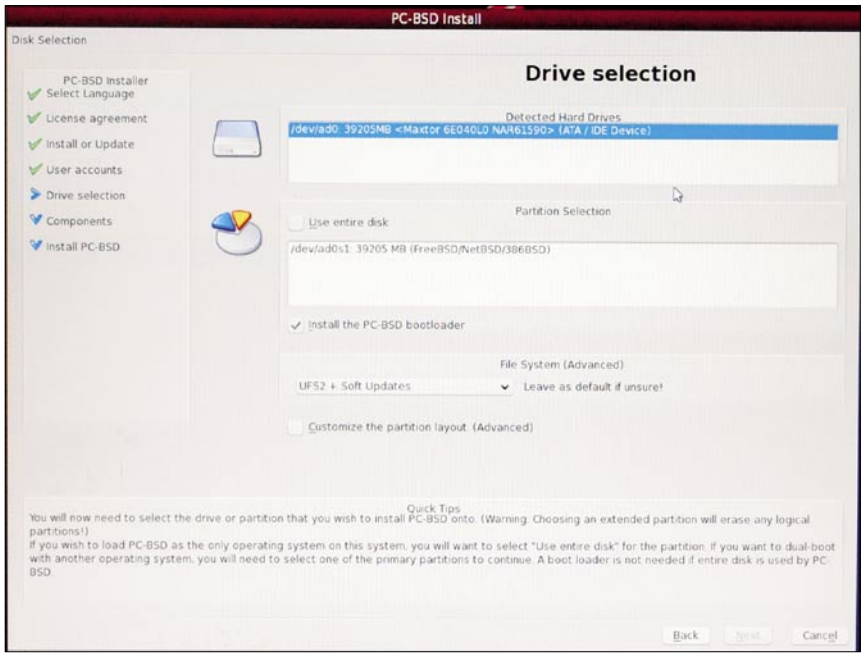


Figure 9. pc bsd install 9

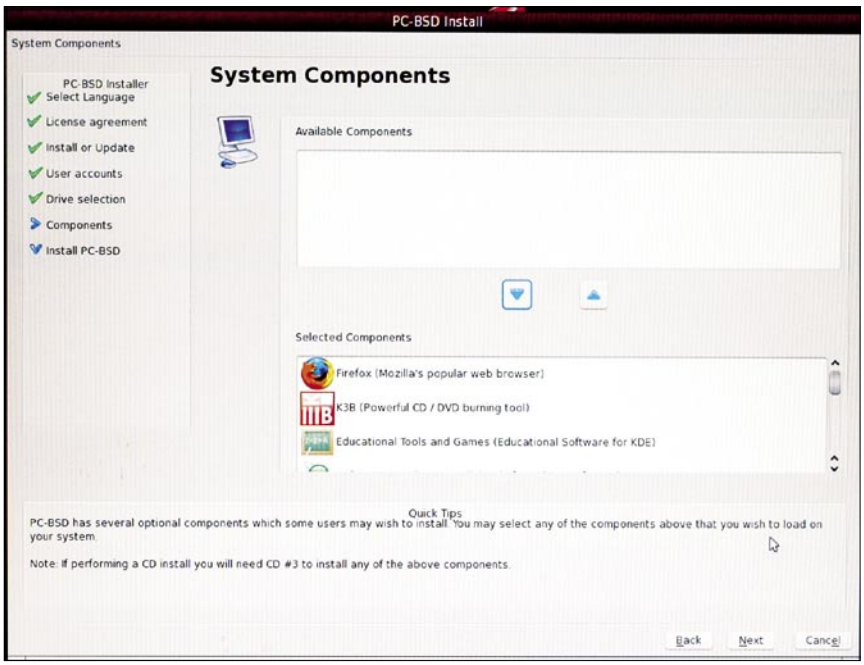


Figure 10. pc bsd install 10

- `/dev/da0s3` – First drive (SCSI), third partition
- `/dev/da1s1` – Second drive (SCSI), first partition

You also need to select the partition on which you want to install PC-BSD. If you already have more than one partition, all of them will be listed under the *Partition Selection* box. Select the one you want to use (Figure 9).

On the system components screen one or more software packages can be selected for installation (with the blue arrows). In the *Available Components* box the most common packages are listed, plus the FreeBSD ports directory and the FreeBSD source code. Selecting and installing packages at this point saves you from downloading and installing them after the installation PC-BSD (Figure 10).

The following packages are available as components:

- *Firefox* (Mozilla's popular web browser)
- *K3B* (Powerful CD /DVD burning tool)
- *Educational Games* (KDE Educational package)

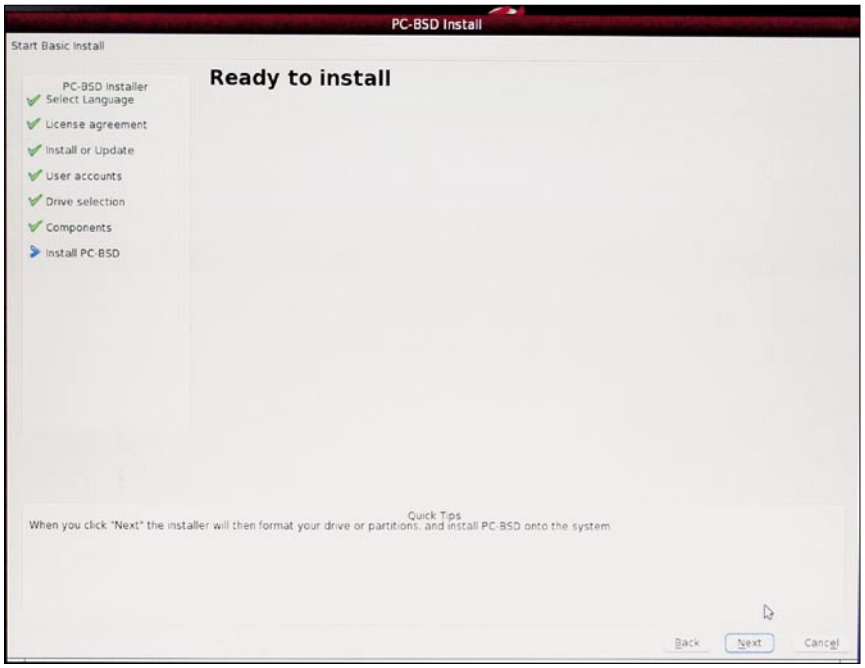


Figure 11. pc bsd install 12

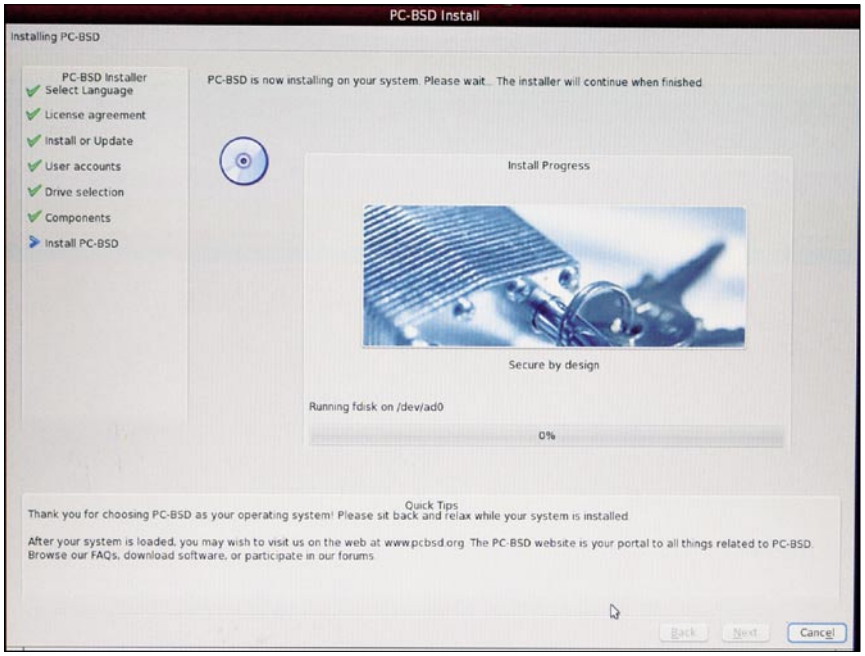


Figure 12. pc bsd install 13

Figure 13. TRW005

Figure 14. PA280001

- *Games* (Arcade, Board, and Card games)
- *Office Suite* (The KOffice Suite)
- *Software Development Kit* (tools for writing software)
- *Desktop Toys* (Misc programs for your desktop)
- *OpenOffice.org* (Full featured office suite)
- *Opera* (Web Browser with BitTorrent client)
- *Ports Tree* (FreeBSD ports collection)
- *Source Code* (FreeBSD system source)

To be able to install any of these packages, you will need installation CD #2 and CD #3 or the DVD (Figure 11). At this point, you are about to copy all files from the CD-ROM to your hard disk drive. Click *Next* to start the installation: see Figure 12. Now you may ask, Jerry, where did you find a leftover Dell computer?

Lucky for me, I live here in Hollywood California, and we have more computers here than anywhere near here. If you paid $150.00 US, you paid too much. The Dell 4600C was $40.00 US and a bargain (Figure 13). Monitor sold separately. `<grin>` Overall, the PC-BSD install was flawless, and I was able to surf the Internet in under an hour (Figure 14). The specs are: Dell 4600C 2.66 MHz 1 gig ram, 120 gig hard drive, CD reader, Dell/Intel motherboard/ chip set. Video, sound, and network setup were flawless. DHCP enabled by default.

I'm very impressed by this distro.

I recommend PC-BSD highly.

Software management simplfied: PC-BSD and the PBI system

Jan Stedehouder

What is the best way of installing software on a FreeBSD-based system? The FreeBSD handbook doesn't provide a single answer as both packages and ports have their benefits. All you have to do is open a terminal and...

At this point most novice users, either from Windows or – in a growing number of cases – from Linux, might decide that FreeBSD isn't for them. Using the command line to perform tasks isn't really considered 'easy' or 'the best way'. Linux distributions that aim at end-users and want to lower the threshold for adoption, develop a variety of graphical tools for software management. Of course, each distribution has it's own fanbase claiming their tools are the best, the easiest and the most powerful to use.

FreeBSD isn't without graphical tools either. The DesktopBSD Package Manager is one of them. Kris Moore decided to push it even further. In 2005 PC-BSD was born and along with it the PBI system for installing software. PBI stands for Push-Button Installer or PC-BSD Installer. In this article we will see how far the PBI system has progressed and whether it makes software management an easy task.

What are PBI's?

PC-BSD's main goal is to offer an easy-to-install, easy-to-use, easy-to-manage and easy-to-extend FreeBSD-based desktop system. The most recent incarnation is built on FreeBSD 7-STABLE and features a fully graphical installer, one that shouldn't pose too many problems. The new user is welcomed by the KDE 4.1.2 desktop. On it you will find a shortcut to the PBI website (see Figure 1). Thus, getting yourself a FreeBSD-based desktop system isn't a problem anymore.

You can extend your desktop by using PBI's. PBI's are self-contained packages to install new software. Each PBI comes with all the dependencies needed to run the program. Installing software is as imple as downloading a PBI, double-clicking it and following the steps of the graphical wizard. No dependency hell, no command line.

This self-containment isn't limited to offering packages with all dependencies solved. Each program is installed in it's own separate folder in the file hierarchy and -in principle- doesn't touch items in other folders.

It's *in principle* as there are (as always) exceptions to the rule. But, in general, you shouldn't be able to ruin your perfectly decent FreeBSD-based desktop by some malfunctioning program.

Where to get your PBI's?

A shortcut on the desktop brings you straight to the PBI website (*http://www.pbidir.com*). One of the stranger things on the website is the disclaimer on the frontpage that *the PBI directory is in no way associated with PC-BSD Software LLC.*

Weird, since the PBI system is pivotal for PC-BSD, but no doubt there are solid reasons for it. The PBI's are organized in

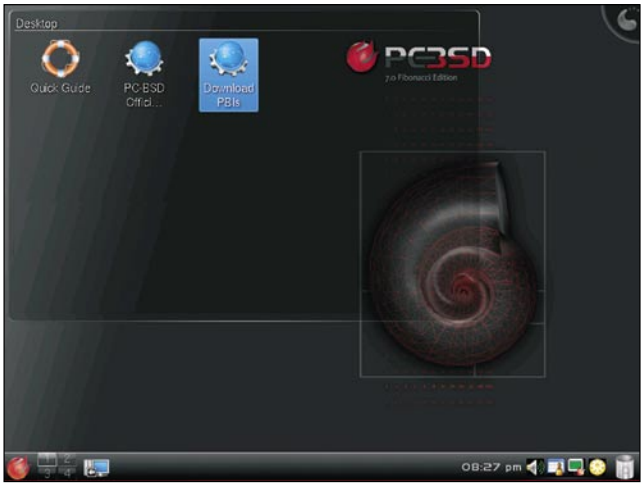


Figure 1. The KDE 4.1.2 website offers easy access to the PBI website

19 categories. In each category you can find a list of programs and from there you go to the specific page for the program you want.

The program pages are well laid-out. It gives a description of the program, a user-based rating, two tabs for the 32-bits and 64-bits versions of the program, a list of various releases of the program and an indication of which PC-BSD release it is meant for.

You can click on the specific item you wish to download. This brings you to a page where you should select a download server. Fortunately, you only need to do this once. The next step lands you on the download page where you are suggested to save the PBI file on your system.

Once downloaded it becomes a matter of double-clicking the PBI, providing your administrator password and following the steps. The program is added to the KDE menu in it's proper submenu and – if so desired- a shortcut is placed on the desktop.

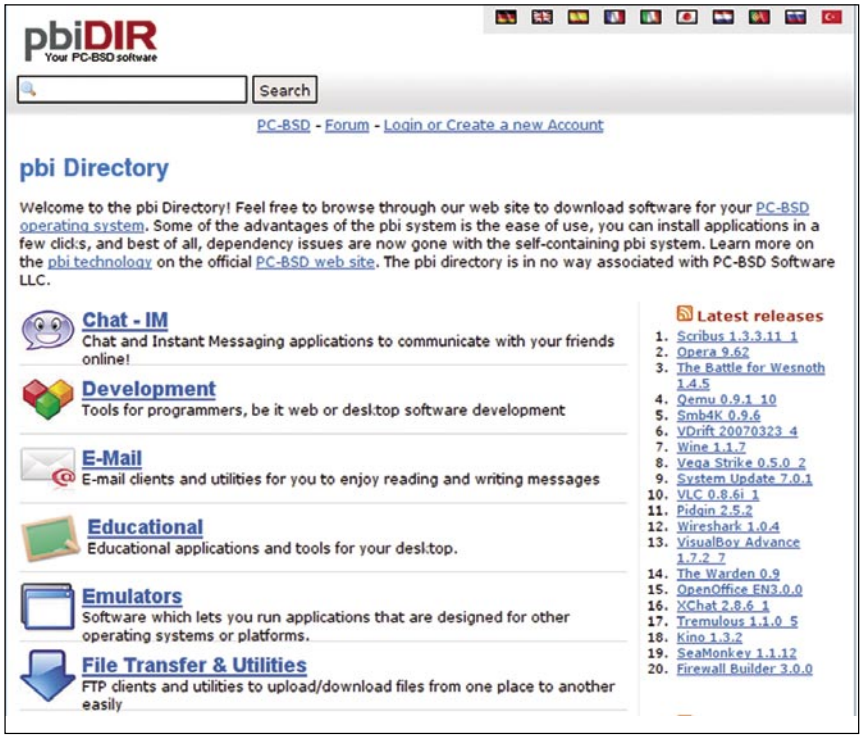


Figure 2. The PBI website is the repository for new PBI's

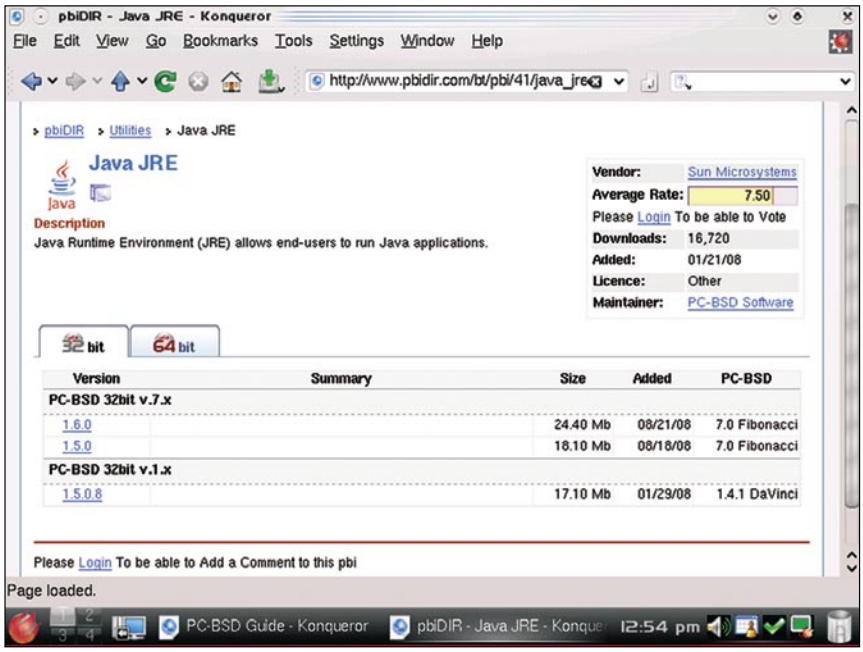


Figure 3. Each PBI program page contains sufficient information to select the proper package for your version of PC-BSD

Keeping your system up to date

One other task related to software is to keep it all up to date. Windows users need to check (or keep an eye on) update notifications for their operating system and all the various programs separately. Linux users are somewhat spoiled, since most package management systems take care of updating both the operating system as well the applications (provided they stick to the software repositories). PC-BSD needs to take care of two systems: the FreeBSD-based desktop and the PBI's. It does so via the PC-BSD Update Manager that is launched at startup. It checks for system updates and updates for the installed PBI's. Once updates are available, the user is notified and is able to select which updates he wants to apply (Figure 4).

Under Applications › System you can find the entry Add/Remove software (figure 5). Here you can remove installed system components and the installed PBI's. It doesn't allow for easy installation of PBI's from the PBI website.

Does it work?

In short, it does! The PBI system reduces software management to downloading a package, double-clicking it to have it installed and simply following the notifications to keep your programs up to date. It closely resembles the experience

Figure 4. The PC-BSD Update Manager keeps track of PBI's in need of updating

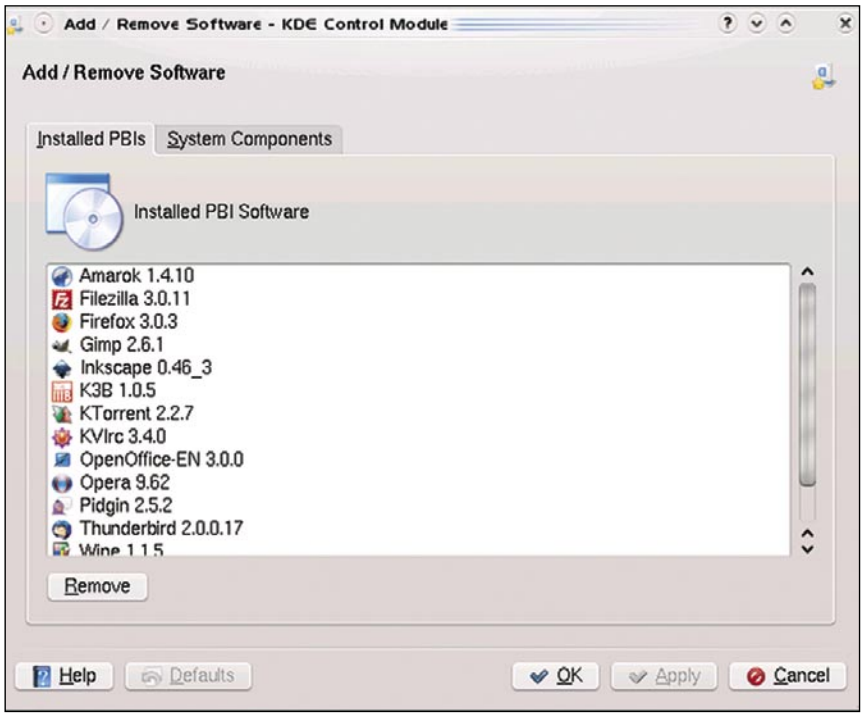


Figure 5. Add/Remove software isn't (yet?) the place to get new PBI's

of Windows users. With the easy installation of PC-BSD and this method of installing and maintaining your software, is easy to forget you are working on a FreeBSD-based desktop.

Last year I explored the PBI system for the first time. There were some glitches here and there and installing larger programs (like games) didn't always work. This time there were no such problems. Even the PBI's for *GNOME-based* programs worked. Should you be so inclined, via the PC-BSD forum you can get an experimental PBI to install a GNOME desktop.

Creating and building PBI's

The PBI directory has two programs to support (would-be) maintainers to create new PBI's: PBI Builder and PBI Creator. The PC-BSD website provides good information on how to use these programs.

If you believe a specific program should be added to the PBI website, you can take it upon yourself to make it happen. PBI Builder is a command line tool to convert a FreeBSD port into a PBI file. This creates a module that is added to the PBI Build Server.

This server checks the ports tree for updates and then rebuilds the corresponding PBI's. Once tested, they are available from the PBI website and show up via the PC-BSD Update Manager.

Similar initiatives in the Linux world

There are a few initiatives in the world of Linux that are similar to the PBI system. The rational is more focused on providing easy to use packages that can be installed regardless of the package management system the distribution uses. Whether it is `.deb-based`, `.rpm-based`, `.tar.gz-based` or any other system, the software should be able to install with a single click (Figure 6).

One of the initiatives is Klik (*http://klik.atekon.de/*). Linux users need to install a client first and then browse through the website to find their package of choice.

The number of categories is impressive, but quite a few of them have only one or two packages (or

Figure 6. Klik

Figure 7. Autopackage

Figure 8. Software management simplfied: PC-BSD and the PBI system

are empty). The site counter refers to somewhat over 400.000 packages delivered, which isn't much as it has existed for some years now.

Autopackage is another initiative (*http://autopackage.org*). With Autopackage you get a complete binary package that installs on your Linux desktop. This project isn't aimed at replacing the *normal* package management under Linux, but to provide developers a way to offer a single package for a variety of desktops.

The Dutch Tax Office uses autopackage for the annual tax forms (*http://www.belastingdienst.nl/download/1706.html*).

Two high school students in the Netherlands are extending autopackage to develop a game update platform called Starez (*http://starez.org/*) with a similar functionality as Valve's Steam system (Figure 7).

However, overall, the popularity of these initiatives is low. Distributions like Ubuntu have instead pushed for simplifying the package management system and offering simple and easy to use tools where the user ticks a box and clicks *apply* in order to install software.

Conclusions

The PBI system is quite innovative as it allows for very easy installation of new

Figure 9. Spread PC-BDS

software, even software that isn't yet available via ports or packages (or won't be available for other reasons). Compared to last year, enormous progress has been made in terms of speed, the amount of PBI's offered, keeping them up to date and providing the users with the needed information. The main draw back is having to go to the website and clicking your way through various webpages before being able to download the PBI. Extending the Add/Remove software tool with easy access to the PBI's would solve this issue.

Linux and FreeBSD power users might find the collection of PBI's lacking here and there, but we shouldn't forget these power users aren't the target audience for PC-BSD. It's aimed at what they call the *casual user*. For that user the PBI website covers most bases. Should additional software be needed, he/she now has a stable platform in order to explore new avenues. The user can install new software via the packages and ports. It would be nice to see a solid and easy to use graphical front end being offered on the default PC-BSD desktop as well.

Maybe it's time to put PC-BSD on one of those nifty netbooks and start spreading PC-BSD to the masses that way (Figure 8 and 9).

Personalizing Your PC-BSD Desktop

James T. Nixon III

What is the point of having a personal computer if it isn't personal? The vast amount of possibilities when customizing your PC-BSD desktop can be overwhelming. There are several aspects of PC-BSD that need to be understood before moving forward.

Think of each part of the desktop as a widget that can be removed, resized, and stylized. Let's begin by breaking down the panel.

Breaking down the Panel

There are several aspects that need to be understood before personalizing PC-BSD. Think of each part of the desktop as a widget that can be removed, resized, and stylized. First let us look at the *main panel1* and break it down.

- Application Launcher1a
- The Pager1b
- Show Dashboard1c
- Show Desktop1d
- New Device Notifier1e
- Task Manager1f
- Digital Clock1g
- System Tray1h
- Trashcan1i
- Panel Cashew1j
- Desktop Cashew2

1a. Application Launcher

The application launcher has changed since the last version of PC-BSD. It provides easy access to your favorite applications, recent documents, and system settings.

1b. New Device Notifier

This widget allows you to switch to 4 different Virtual Desktops. Imagine one large desktop broken up into a grid. Each section of the grid provides a separate space for your windows; making it much easier to use several programs at once without cluttering your screen. You may add up to 20 virtual desktops.

1c. Show Desktop

The *Show Desktop* widget minimizes every open window in order to show your desktop.

1d. Show Dashboard

Unlike the *Show Desktop* widget, the *Show Dashboard* widget places all widgets in front of open windows without moving, closing, or minimizing them. This becomes very useful when working with several applications at once.

1e. The Pager

This widget allows you to easily access, mount, and unmount any supported device you have plugged in or inserted, such as a CD, DVD, or thumbdrive.

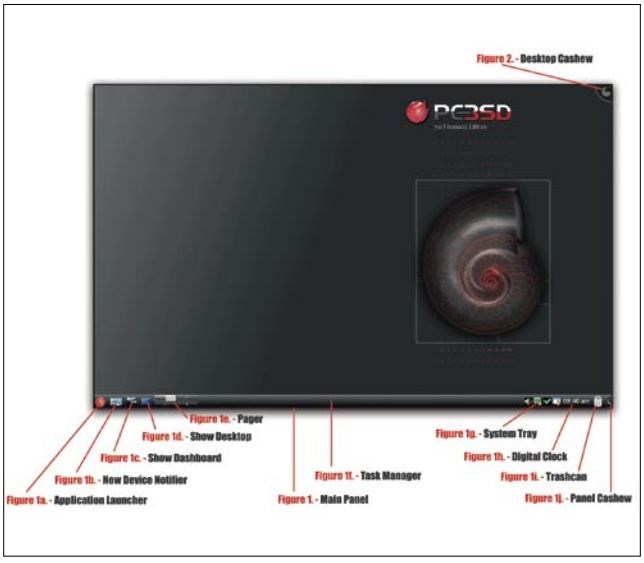


Figure 1. Desktop Overview

1f. Task Manager

This widget displays tasks from all desktops and also has the option to show tasks from the current desktop and/or screen. To do so, right click the taskbar and click *Task Manager Settings*.

1g. System Tray

This widget is pretty straightforward; you can customize it by right-clicking and choosing *Digital Clock Settings*.

1h. Digital Clock

The system tray widget provides *Volume Control*, a *System Update Manager*, a *Network Configuration Utility*, and also displays applications in use.

1i. Trashcan

With the trashcan located on your *main panel1* it is much easier to delete files and empty your trashcan.

1j. Panel Cashew

Unlike the other widgets on the panel, the *cashew* is embedded into every panel and can be used to resize and reposition the panel, as well as add or remove widgets. Panels can be *aligned left1ja*, *center1jb*, or *right1jc* and also have the option to *Add Widgets1jd*, *Lock Widgets1je*, *Remove the Panel1jf*, and *Close Configuration1jg*. You may also click and drag the *Configuration Window1jh* to move it to the top, bottom, left, or right hand side of your screen.

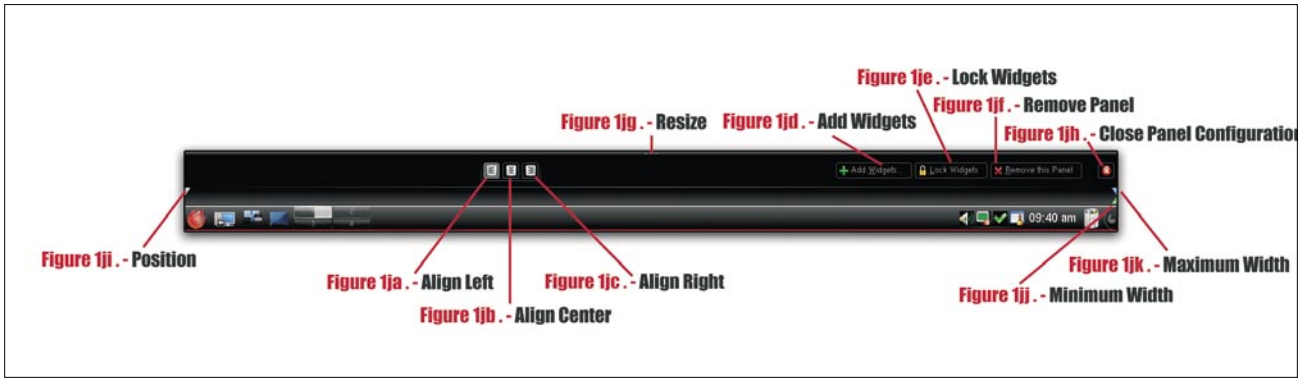


Figure 2. Main panel

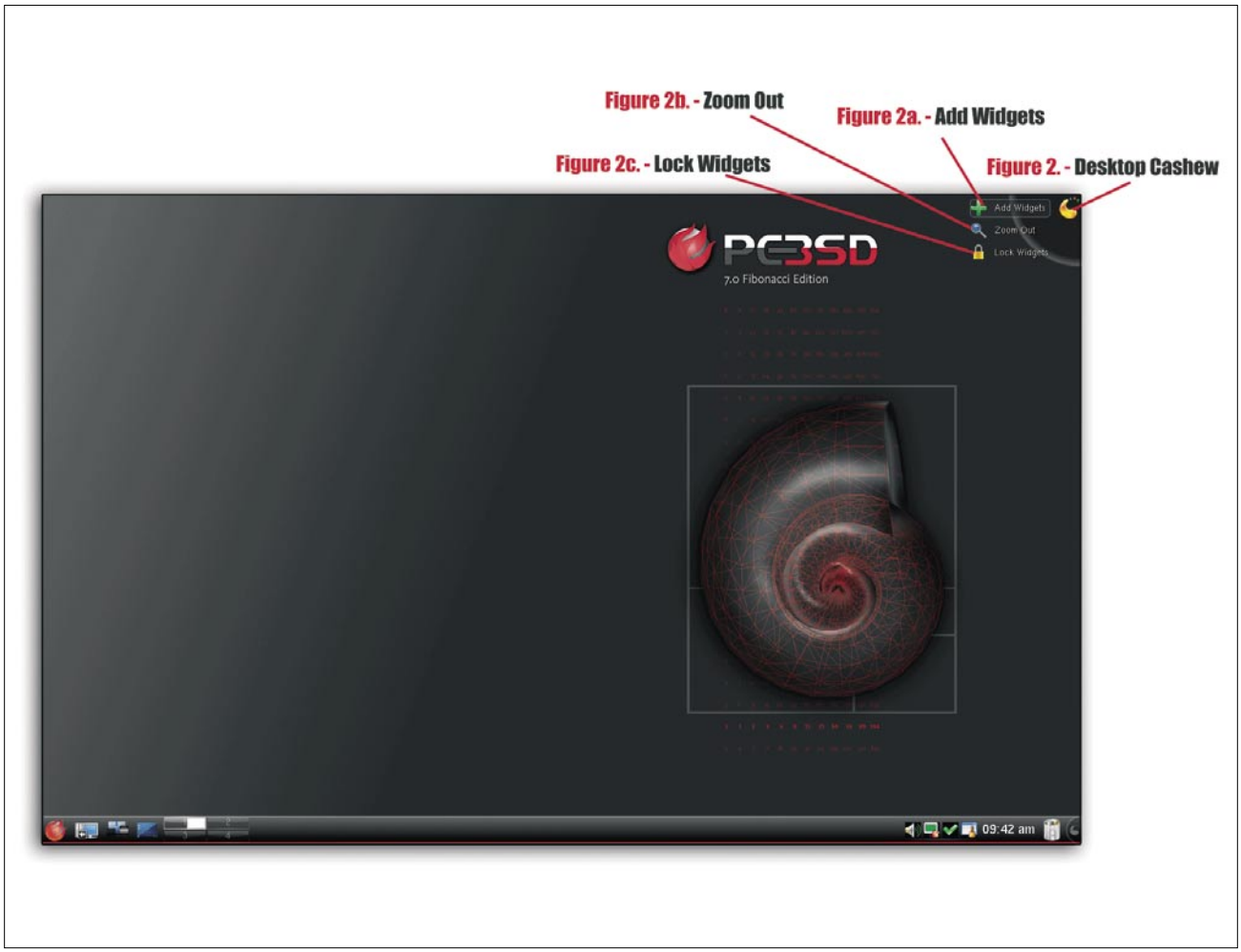


Figure 3. DesktopCashew

Zooming Through Your Desktop

The first thing to take note of is the ability to zoom out. When you click the *Desktop Cashew2* on the upper-right of your screen it lists the options: *Add Widgets2a*, *Zoom Out2b*, and *Lock Widgets2c*.

When you click *Add Widgets* it allows you to add a variety of different widgets to your desktop. The Zoom Out function does just what it says; it zooms out.

If you zoom out twice it opens your eyes to the wonders of a *ZUI* or *Zoom User Interface*, a term first coined by Franklin Servan-Schreiber when working for Sony Research Laboratories. *Lock Widgets* does exactly what it says; it locks each widget in place making them unmodifiable.

While zoomed out you should notice that the cashew on the upper-right of your screen now has two new options: *Zoom In2d*, and Add *Activity2e*.

Do not confuse *Virtual Desktops* used in the *Pager1b* with *Activties3* used in the ZUI. When switching from one virtual desktop to another your Activity never changes. You can, however, add more activities and easily switch between them by zooming in or out using the *cashew2*. Aside from using the cashew, you may also press [*CTRL*] plus [*+/-*] to zoom in or out.

When zoomed in the easiest way to switch between activities is by pressing [*CTRL*] plus [*N*] for the next activity or [*CTRL*] plus [*P*] for the *previous* activity. With the ability to assign a separate space for each activity one has an endless list of useful possibilities. Think of Work and Home as two separate activities; now assign different widgets for each activity.

Customizing Your PC-BSD Desktop

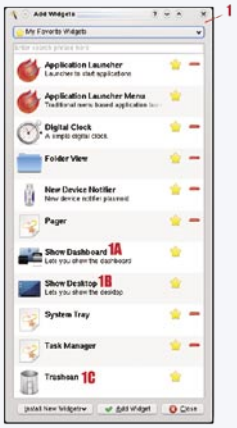
Now that you have the basic knowledge of the PC-BSD desktop, let's move forward and start tweaking it. This guide will aid you in coloring outside the lines of your desktop. Remember to have fun and experiment.

Figure 4. ZUI

01

Default Desktop

This is the desktop you start out with. Let's make it a tad more personal.



03

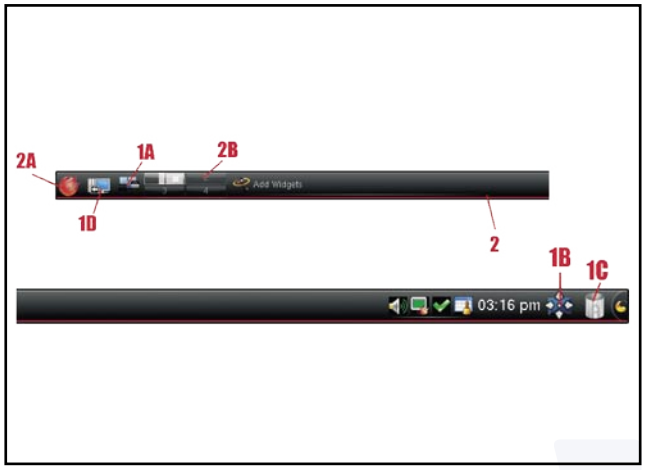
Add Widgets

Click the panel cashew and choose *Add Widgets*, a new window will pop up.

02

Adding Widgets

Widgets, also known as Plasmoids, are mini-applications that have all sorts of uses. From displaying the time in binary format to displaying the latest XKCD comic strip, you can find widgets for almost anything. Let's start by adding a couple of useful widgets to the panel.



04

Add Widgets

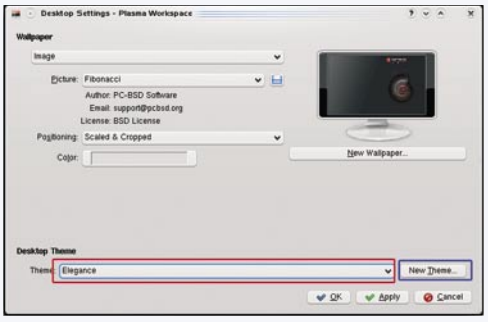
From the *Add Widgets1* window, add *Show Dashboard1A*, *Show Desktop1B*, and *Trashcan1C* to the *main panel2* if you haven't already done so. The *New Device Notifier1D* widget is already included on the main panel. If you would like to add a widget to the desktop, simply drag and drop it wherever you like.

Move the *Show Dashboard1A* and *Show Desktop1B* widgets you added between the *Application Launcher2A* and the *Pager2B*. You may also wish to move the *Trashcan1C*; I prefer to keep it on the right.

05

Changing Plasma Themes

There are a few ways to change your desktop's theme. Let's change the plasma theme first. The plasma theme defines the style of your panel and widgets, but does not affect the color scheme or style of your application windows.



06

Change Desktop Theme

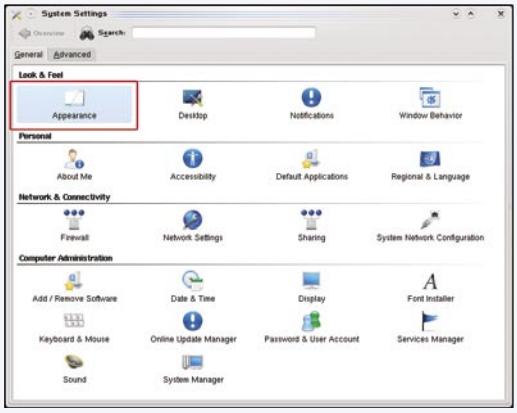
Right-click your desktop and choose *Desktop Settings*.

07

Download theme

Now choose a theme from the *dropdown menu*, I prefer *Elegance*. To download more themes click *New Themes*, there are several to choose from.

After the theme is done installing you may select it from the dropdown menu in *Desktop Settings*.



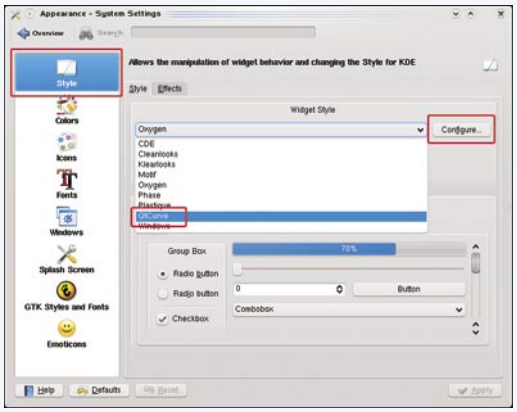
08

Changing System Themes

There is a lot more to personalize than just your plasma theme. Let's begin changing the color scheme, window decorations, and style of your desktop. Click the *Application Launcher* on the main panel (represented by the PC-BSD Flame) and choose *System Settings* from the *Favorites* tab.

Appearance style qtcurve

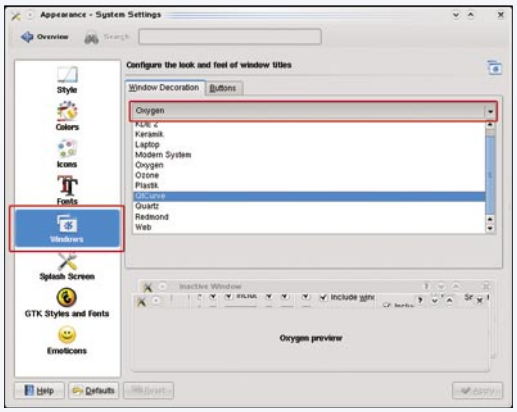
Now choose *Appearance* from the *System Settings* window.



09

Appearance style qtcurve

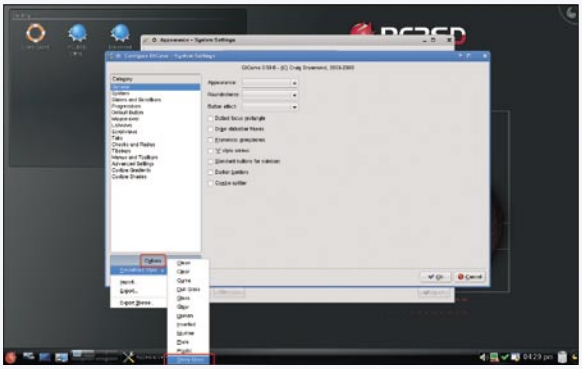
In the *Style* section, pick *QtCurve* from the dropdown menu and click *Configure*.



11

Appearance window qtcurve

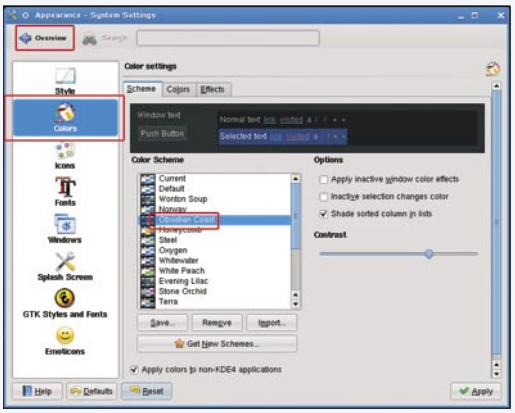
The *Windows* section will change the way Window Borders and the Title Bar look. Choose *QtCurve* and click *Apply*.



10

Appearance qtcurve glass

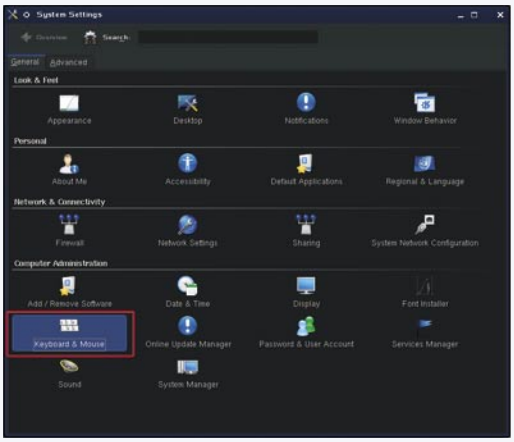
Now click *Options*, then *Predefined Style* and choose *Shiny Glass*. Click *OK* and hit *Apply* on the *System Settings* window.



12

Appearance colors obsidian-coast

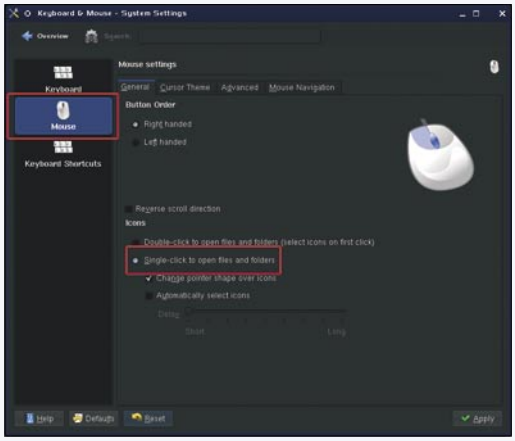
To change the color scheme go to the *Colors* section and choose a scheme. I chose *Obsidian Coast*. Once you are satisfied with a scheme click *Overview*, this will take you back to the *System Settings* window.



13

Mouse Settings

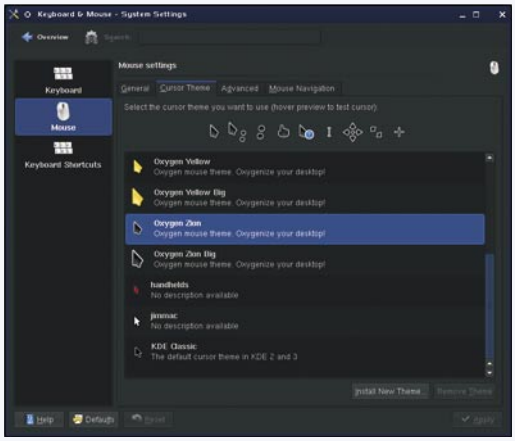
Choose *Keyboard & Mouse* from the *System Settings* window.



14

Mouse single-click

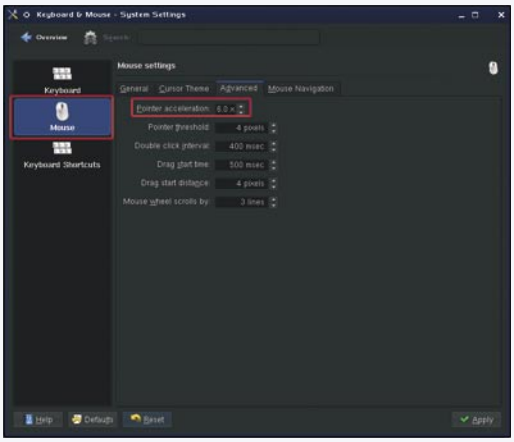
Go to the *Mouse* section and select whether you are *right* or *left* handed. You may also choose *single* or *double click* to open files and folders.



15

Mouse oxygen theme

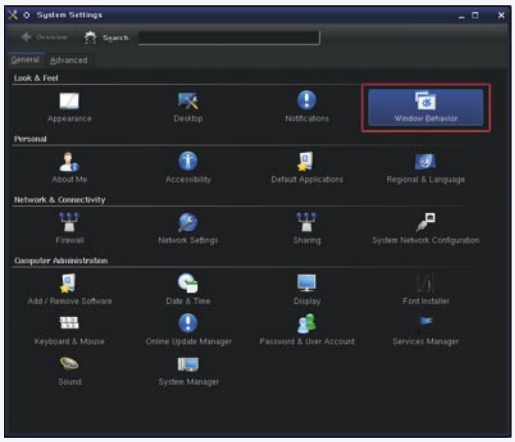
Click the *Cursor Theme* tab and select a theme.



16

Mouse speed

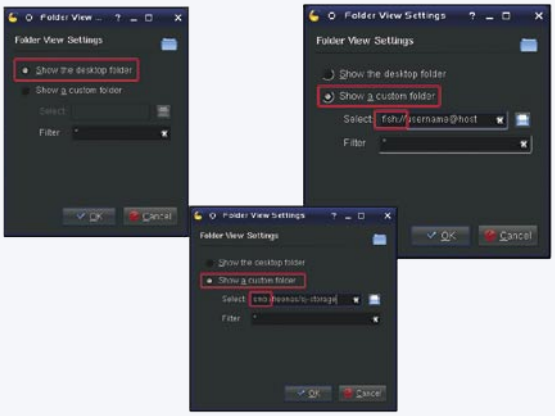
To make your mouse speed faster, click the *Advanced* tab and change *Pointer Acceleration* from 3.0x to 6.0x, or to the speed of your choice.



17

Window placement

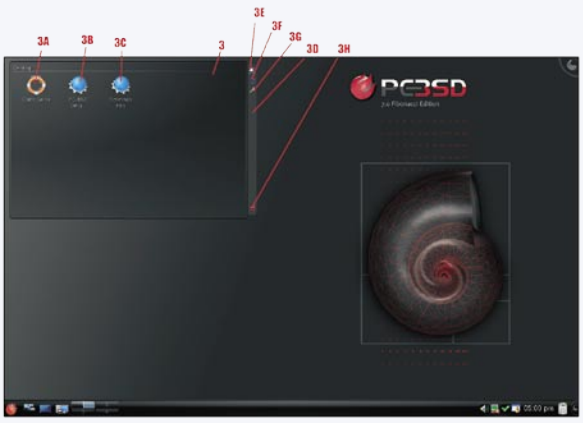
Go back to *Overview* and select *Window Behavior*, now click the *Moving* tab and select *Centered* from the *placement dropdown*. This will force applications and folders to appear in the center of your screen.



19

Folder view config

Let's configure the Folder View widget to display a different folder.



18

Folder View

The *Folder View3* widget that comes on the default PC-BSD install includes a *Quick Guide3A* to PC-BSD, a link to the *Official PC-BSD Website3B*, and a link to the *PBI Directory3C*. The *Folder View* widget displays the content from remote and local folders. While hovering over a widget, a *configuration bar3D* will appear. This is where a widget can be *resized3E*, *rotated3F*, *configured3G*, and *closed3H*.

20

Folder view size-position

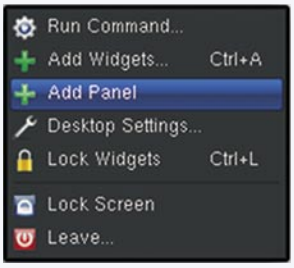
After clicking *configure3G* (the wrench icon) a settings window will appear. Select *Show the desktop folder* or *Show a custom folder*. You may browse for a custom folder or use a remote directory using the SMB or FISH protocols. The *SMB* and *FISH* protocols work the same way they do in Konqueror.

21

More widgets

You may use multiple instances of the *Folder View3* widget to display an assortment of local and remote folders. Drag icons from the *Application Launcher* to the *desktop* Folder View widget, then resize and position the widgets to fit your needs.

There are several more widgets to choose from, so have fun decorating your desktop with your favorites. The widgets shown above are, *Comic Strip4A*, *Analog Clock4B*, and *Calculator4C*.



22

Adding Panels

You may also use panels to display icons for your favorite applications and folders.

Right click your desktop and choose *Add Panel*.

23

New panel and new panel right

A *new panel5* will appear on the top of your screen, click the *panel cashew5A* to open the *configuration bar* and drag the bar from the top of the screen to the right hand side of your screen. Make sure you click and hold the inside the configuration bar and not the panel itself.

24

New panel right app-launch and new panel right icons

Now click the *Application Launcher* and drag your favorite applications onto the panel. An easy way to do this is by right-clicking an application and choosing *Add to Favorites*. Now drag all the applications from the *Favorites* section of the *Application Launcher* to the new panel.

25

New panel right resize

Once you've added all the icons you want, *resize the panel6A* and *center6B* it. If the right panel's *cashew6C* is hidden behind the *main panel6D*, click the *right panel6E* to bring it in front and then click the cashew to bring up the *panel configuration bar6F*. You can use this technique to place shortcuts to your favorite folders as well.

26

Wrapping it up

PC-BSD is geared towards the average Windows or Ubuntu desktop user, yet has the power of FreeBSD under the hood. Once you have familiarized yourself with your new PC-BSD desktop you will understand what *Personal Computing* is all about.

Sources

To learn more about PC-BSD go to *http://www.pcbsd.org*. To download your favorite applications go to *http://www.pbidir.com*. To chat on IRC about PC-BSD go to the `#pcbsd` room on Freenode. Also, remember to join the testing list and submit any bugs you find by signing up on *http://lists.pcbsd.org/mailman/listinfo/testing*.

Using FreeBSD for Off-Site Backups

Eric Vintimilla

It is becoming increasingly important for people to have backup systems in place. This is especially true for people who hoard multimedia content. What would happen if someone's hard drive failed and it contained their entire music library?

That would be cause for a panic attack at the very least. Worse yet, what would happen to their files if there was a fire or a flood? Having an NAS or an external hard drive will do no good in these situations.

This is why off-site backups are crucial, and with FreeBSD, it is very easy to automate this process.

Loading your media

The first step is to set up an easy way to copy your files to your FreeBSD machine. One of the best ways to do this is by setting it up as an NFS Server. Start by adding the folder you want to sync to /etc/exports:

```
echo "/home/backupuser/backups 192.168.1.10/24(rw,no_
root_squash,async,no_subtree_check) " >> /etc/exports
```

In this example, the /backups folder is going to be offered to an NFS client on 192.168.1.10. Now that this has been added, we can start nfs (or restart it if it's already running).

```
/etc/rc.d/nfsserver start
```

Once the NFS server has been started, it can be mounted on the client machine, and we can begin to copy over our multimedia files:

```
mkdir /mnt/backups
mount 192.168.1.41:/home/backupuser/backups /mnt/backups
cp -R /home/backupuser/MP3s /mnt/backups/
```

This will recursively copy everything within the MP3s directory (including the directory itself).

Accessing your off-site server

If you do not have your own off-site server, there is no need to worry. There are plenty of web sites out there that offer backup services. However, you will want one that allows SSH access.

The best tool for performing these backups will be the rsync command. It provides a lot of flexibility and is easy to use. Let's start by adding rsync to our backup system:

```
pkg_add -r rsync
```

Once rsync has been installed, you are ready to back up your data. Manually performing this tasks is a cinch, but we want to automate it. Whenever you try to upload your data, the responding server will ask you for your password, which is going to make automatic backups pretty difficult. We could write a script to do this for us, but then we would have to include a username and password for our backup server, which I recommend against. Luckily, we can get around this obstacle with RSA Private/Public key authentication.

Setting up RSA Private/Public key authentication is quick and painless. First, type the following command:

```
ssh-keygen -t rsa
```

Then, hit ENTER three times to accept the defaults. Once the public key file has been created, you have to copy it to your off-site server: see Figure 1.

```
scp ~/.ssh/id_rsa.pub backupuser@offsite.backup.com:~/
```

This command will copy it to the home directory of the user known as *backupuser*. Now, you have to log into your off-site server via SSH.

```
ssh -l backupuser offsite.backup.com
```

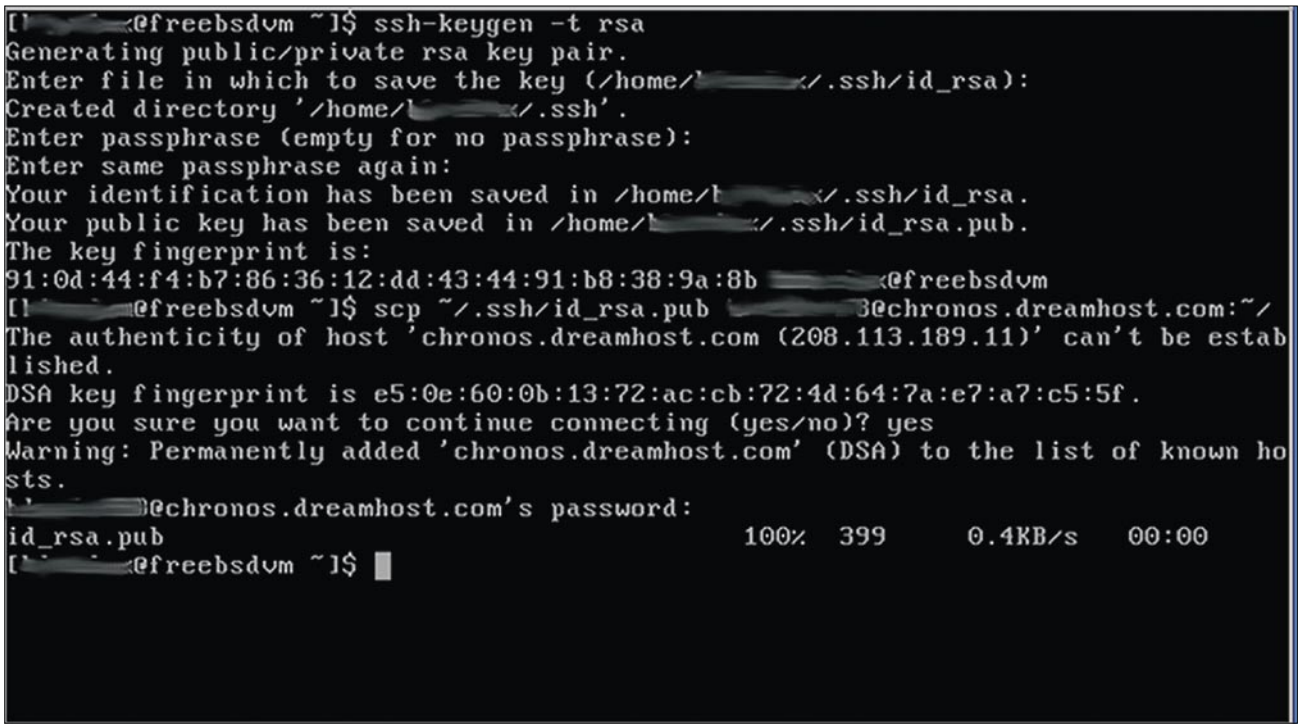


Figure 1. Keygen

Once you enter your password, you should see the file you transferred. From here, we have to append this file to our `authorized_keys` file. If there is no `.ssh` folder, create it.

```
mkdir .ssh
```

Then, add the data to the `authorized_keys` file and delete the file you uploaded.

```
cat id_rsa.pub >> .ssh/authorized_keys
rm id_rsa.pub
```

Now make sure permissions are set properly for all necessary files and directories:

```
chmod go-w ~
chmod 700 ~/.ssh
chmod 600 ~/.ssh/authorized_keys
```

That is it! Now, you can log into your off-site server without having to enter a password.

Automation

Now we are finally ready to backup our multimedia files! Since we want this to run overnight so it does not hog our bandwidth, we are going to write a small script and set up a cron job. We will create this script in our home directory for the sake of this example, but you should store it wherever you prefer to keep your scripts.

```
cd ~
nano remotebackup.sh
```

The script is going to be very small, just two lines:

```
#!/bin/sh
/usr/local/bin/rsync -ruqz /home/
backupuser/backups/* backupuser@offsi
te.backup.com:~/
```

What does this mean? The two most important switches that the rsync command is using here are the `-r` and `-u` switches. The `-r` tells it to recursively transfer files to the target location, so everything under the backups folder will be sent. The `-u` tells rsync to only copy files if the local versions are newer than the remote versions. Thus, you will not copy all our files every time you run this script, just the recently changed ones. The `-q` switch tells rsync to run in quiet mode (you do not want to receive a list of data in your inbox every time this runs) and the `-z` switch tells rsync to compress file data during transfer.

Now, we can just create a cron job to run this task for us.

```
crontab -e
```

Add the following lines:

```
# Backs up our files to our off-site
server.
0 2 * * * /bin/sh /home/backupuser/
remotebackup.sh
```

Now you are all set! This script will be run every night at 2 a.m.. Feel free to change the settings in your cron job as needed. Currently, the rsync command is set to run in quiet mode, so it will only output data if there is an error message, which will be mailed to your inbox.

Summary

Having a disaster recovery or backup plan is extremely important when storing large amounts of data. While many businesses have such safeguards in the place, the average user may not. Without measures to backup their multimedia and documents, they run the risk of losing their entire music libraries or important personal documents. Having an NAS or an external hard drive can be a great solution for local backups, but what would happen to a user's data in the event of a flood? They would face a total loss for certain. This is why anyone who wishes to protect their data should have an off-site backup server, and with FreeBSD, the whole process of uploading files to this server can be easily automated with only a few short commands.

Building
NetBSD for Embedded Systems Using Cygwin

Donald T. Hayford

You might think it is unusual that a magazine devoted to the *BSD operating systems would have an article about Cygwin, a Linux-like environment that runs on Windows.

The fact of the matter is, if you're a developer working with an embedded system, you are most likely using Windows as your development environment. And chances are, that environment is based on GNU-tools and Cygwin.

There are a number of reasons why this is true. Often, working with embedded systems will require specialized interface hardware that may only run under Windows. Your company's IT department may discourage the use of a non-Windows machine on your corporate network. Or, you may have development tools that you prefer using that don't run under Linux or BSD. As we all know, Windows-based machines easily outnumber all other operating systems combined and most embedded system providers will support Windows before any other operating system. To be considered a viable candidate for embedded systems, the operating system must support a Windows build environment. And NetBSD does just that.

NetBSD has a lot of potential as an embedded operating system, as it is small, fast, and powerful. Though not as popular as Linux or VxWorks, it is currently featured in some embedded products; for example, Wasabi (an American company) and IIJ (of Japan) both deliver products based on NetBSD. For those that would like to learn more about building and installing embedded systems using Cygwin (The same steps for getting and building NetBSD will work for most Unix-like environments.), this article will show you how to install NetBSD on the Kurobox Pro.

The Kurobox Pro

For several years, Buffalo has made a number of network addressable storage (NAS) devices known collectively as the Linkstation NAS. Hackers being what they are, they loved hacking these devices. According to web-rumors, Buffalo decided to sell a hacker version of the Linkstation, called the Kurobox (*Expert Box*) as a way to get rid of old hardware and to reduce the number of broken Linkstations that were being reported on various websites, preferring reports of broken Kuroboxes instead. The device proved to be very popular, and now three generations of Kurobox (or Linkstations) are available, sporting a PPC, a MIPS, and finally an ARM processor in the Pro version. If you want to hack something and you want some processor variety, consider the Kurobox (available from Revolution by Buffalo at *http://www.revogear.com/*) or the equivalent Linkstation. You can do almost everything with a Linkstation that you can do with a Kurobox, and they tend to be available at local computer stores. Recently, a computer store in my area was offering the Linkstation Pro with a 500 GB hard-drive for $20 less than the web-only Kurobox Pro which doesn't include a hard-drive. So, look around – just don't say anything if you break it.

The Kurobox Pro, as shown in Table 1, is based on the Marvell Orion System-On-Chip 88F5182. The Orion SOC is another member of the popular ARM family and is well supported by open source compilers and operating systems. The K-Pro comes with 128 MB RAM, two USB ports, a Gigabit Ethernet port, one PCI-Express connector, an internal SATA connector and a power supply big enough to power the hard-drive and the processor boards. The K-Pro also has 256 MB of NAND Flash memory that holds a Busybox version of Linux, one of the two main differences between the Linkstation and Kurobox versions (the other being the need for a user-supplied hard-drive). Out of the box, it is very easy to add a full version of Debian. For more information, see the community-generated wiki at *http://buffalo.nas-central.org/index.php/Main_Page* for information on the Kurobox Pro. Lately, there has been some activity in both the FreeBSD and

NetBSD communities to add support for the Marvell SOCs.

While there are several options for hacking the K-Pro, we are going to build both an install version and a standard version of NetBSD which we'll load into the device using TFTP (*Trivial File Transfer Protocol*) and FTP (the not-so-trivial version). To tell the K-Pro where and what to load on boot-up, you will need to add a serial port to the K-Pro. Fortunately, this is very easy to do and is well-documented at the community web site; for more information, see *http://buffalo.nas-central.org/index.php/Add_a_Serial_port_to_the_ARM9_Linkstation*. Be careful using information from the web site – all generations of Kurobox/Linkstation are covered there. Make sure the instructions you're following are for the K-Pro and not one of the other variants.

Getting and Installing Cygwin and TFTP

Cygwin is not the only choice for a Linux-like environment that can run under Windows. Given the speeds of modern processors and the capabilities of emulators like VMware or QEMU, actually running Linux on Windows is certainly practical. Microsoft also has some support for Unix-like commands in it's *Services for Unix* (SFU) package. SFU will run under Win2K and XP but has been replaced by the Subsystem for Unix Applications in Vista. Both SFU and SUA are incomplete and need to be supplemented by the Interix add-ons from Interop Systems. See the references section for links to more information on SFU/SUA/Interix. All-in-all, though, Cygwin is still the simplest to install and use.

Cygwin is available from *http://www.cygwin.com/* and can be installed using the instructions on the website. Download the `setup.exe` file and run it on your Windows machine; you should see a screen much like that shown in Figure 1. Cygwin normally installs a minimal set of files, so you'll need to add others if you want to build NetBSD. These packages are grouped in sets and you can either install an entire set (if you're lazy) or drill down into the set and pick and choose. To build NetBSD, you'll need to install the `Devel` set of packages. For those

Table 1. Kurobox Pro Hardware Features

Component	Value
Processor	Marvell Orion 88F5182 A2
System Memory	128 MB, Address: 0x00000000
Flash Memory (Boot)	256 KB NOR, Address: 0xfffc0000
Flash Memory (Code)	256 MB NAND, Address: 0xfa000000
Hard Drive	3.5" SATA 1.0, internal (user supplied) SATA 1.0 (not eSATA) external connector
Ethernet	1x 10/100/1000 Mbps, RJ-45 connector
Serial	115 KBaud, 3.3 V, two-wire, available at J6
USB	2x, USB 2.0/1.1, Type A connectors
Expansion Headers	UART (Serial), GPIO, I2C

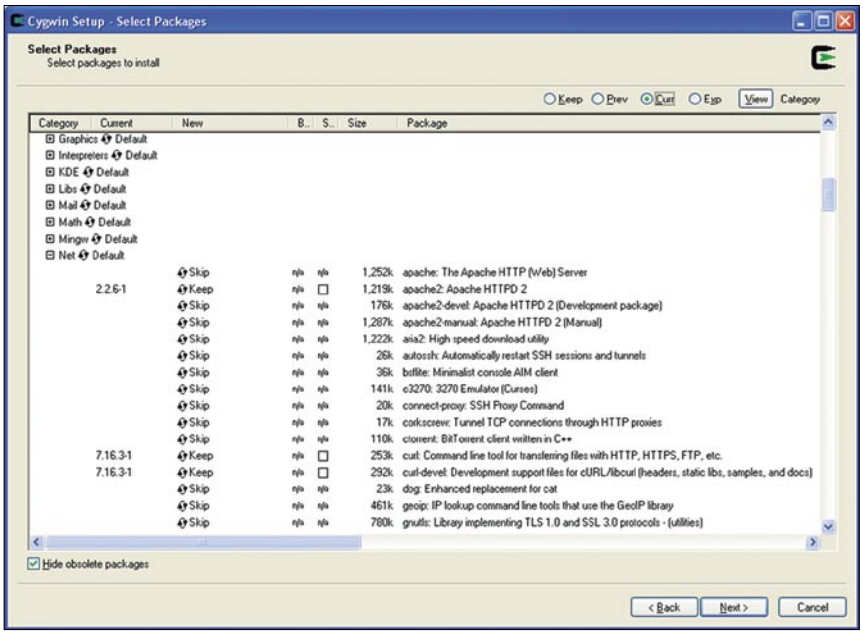


Figure 1. Selecting Packages With Cygwin Setup

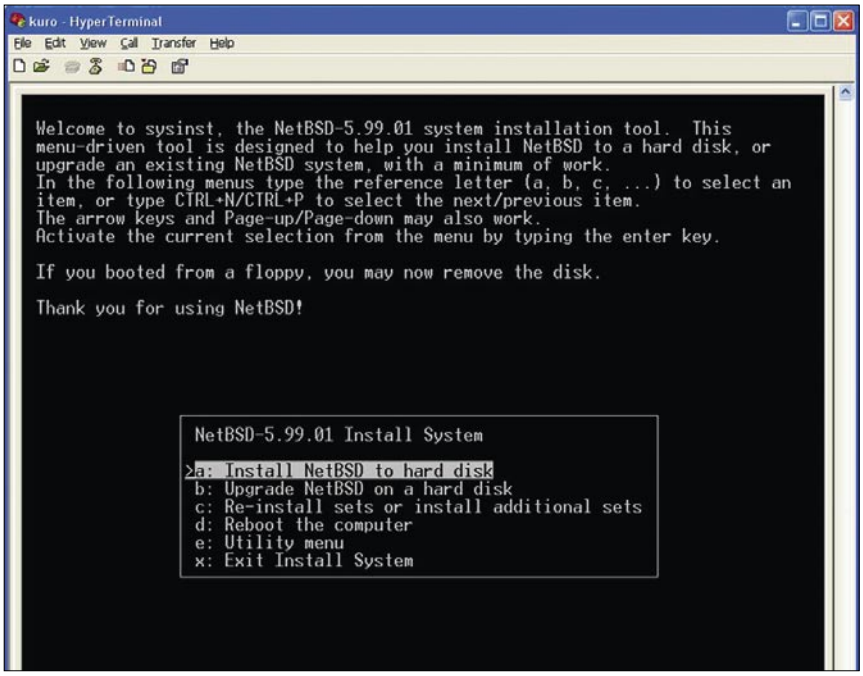


Figure 2. Start up screen for the NetBSD installation kernel, with the console running on HyperTerminal

that prefer a windows environment, there is an XWindows version that you can install, as well – choose the `X11` package set. You'll need CVS, but that comes automatically if you install all of the `Devel` packages. You'll also need the latest proftpd package in the `Net` package set. Other packages that are handy include `inetutils` and `openssh` in `Net` and *nano* in the `Editors` group of packages. Installing *Cygwin* can take an hour or more, mostly for downloading files. Once installed, you should see a Cygwin group in your Windows program files which will allow you to run a copy of the `bash` shell, giving you a command line interface to your new Linux-like environment. If you're a Linux guru already, there's a lot of other good stuff from Cygwin you might want, so look around. If you're not familiar with Linux, check out some of the documentation that is available on the Cygwin or other Linux-oriented sites.

Listing 1. Cygwin proFTPD configuration file snippet

```
# This is a basic ProFTPD configuration file (rename it to
# 'proftpd.conf' for actual use. It establishes a single server
# and a single anonymous login. It assumes that you have a user/group
# "nobody" and "ftp" for normal operation and anon.
ServerName                      "ProFTPD Default Installation"
ServerType                      standalone
DefaultServer                   on

# Port 21 is the standard FTP port.
Port                            21
# Umask 022 is a good standard umask to prevent new dirs and files
# from being group and world writable.
Umask                           022


# To prevent DoS attacks, set the maximum number of child processes
# to 30. If you need to allow more than 30 concurrent connections
# at once, simply increase this value. Note that this ONLY works
# in standalone mode, in inetd mode you should use an inetd server
# that allows you to limit maximum number of processes per service
# (such as xinetd).
MaxInstances                    30
# Set the user and group under which the server will run.
User                            SYSTEM
Group                           Administrators

<snip....The rest of the file is unchanged>
```

Listing 2. Cygwin shell script to install proFTPD as a Windows service

```
#!/bin/sh
# File: proftpd-config.sh
# Purpose: Installs proftpd daemon as a Windows service

cygrunsrv --install proftpd \
          --path /usr/sbin/proftpd.exe \
          --args "--nodaemon" \
          --type manual \
          --disp "Cygwin proftpd" \
          --desc "ProFTPD FTP daemon"
```

Listing 3. Command lines to start and stop proFTPD in Cygwin

```
# To start:
net start proftpd
# To stop:
net stop proftpd
```

If you're running a later version of Windows such as XP Pro, you'll probably need to open up several ports in your firewall to allow your machine to accept incoming requests for TFTP and FTP. To do this on XP, go to the *Control Panel*, select *Windows Firewall* (or *Security Center*) and add port 21 (the default FTP port) to the *Exception* panel. If you follow the suggestion below, Tftpd32 will open the default TFTP port (69) for you, but if you use (or try to use) the built-in Windows or Cygwin tftp, you'll need to open this port yourself. When adding a port to the exception list, you'll find that you have a choice of opening either a TCP or a UDP port. Since I never know which one is going to be used, I always open both, which means you'll have to add two entries for each port.

Once you've installed Cygwin and opened up the firewall, you'll need to configure the FTP server. The instructions that come with the software (in `C:\cygwin\usr\share\doc\proftpd-1.2.10\README.cygwin`) are mostly correct. The shell script I used to install FTP as a Windows service is shown in Listing 1. To configure FTP, I had to edit the configuration file found at `C:\cygwin\etc\proftpd.conf` and remove the two comment symbols (`#`) at the beginning of the lines that start with *User* and *Group*. These are shown in Listing 2. Finally, to start or stop the ftp server, enter one of the two command lines in Listing 3.

Both Cygwin and Windows (depending on the version you have) come with a TFTP server. I had trouble getting either to work with WinXP SP2. Fortunately, there is an excellent alternative for Windows called `Tftpd32` that is available from *http://tftpd32.jounin.net/*. Installation is straightforward and it should open up the default port automatically, but check your firewall to make sure. Incidentally, the program also includes some services you may not want, particularly if you already have a DHCP server on your network. Simply select the options you want to be active

from the options menu and restart the server. The TFTP server will only accept request when it is running, so make sure you have it running when you download the NetBSD kernel to the K-Pro.

Getting and Building NetBSD

We'll use the standard CVS method to get the NetBSD source tree. Because the K-Pro kernel is still experimental, we'll need to add or modify a few files once we get the source code. Some of the files we need were kindly provided by Mr. Kiyohara and are available on the NetBSD web site. By the time you read this, support for the K-Pro will probably be included in the -current version of the kernel. Check the mailing list at *http://mail-index.netbsd.org/port-arm/* for more details. The steps required to

Listing 4. Setting up the NetBSD source for the Kurobox Pro

```
~/ $ mkdir kuronet
~/ $ cd kuronet
~/kuronet $ export CVS_RSH="ssh"
~/kuronet $ export CVSROOT="anoncvs@anoncvs.NetBSD.org:/cvsroot"
~/kuronet $ cvs checkout -D 20081107-UTC src
~/kuronet $ cd src
~/kuronet/src $ wget ftp://ftp.netbsd.org/pub/NetBSD/misc/kiyohara/orion_
nas/orion_nas-20081107.diff
~/kuronet/src $ wget \
        ftp://ftp.netbsd.org/pub/NetBSD/misc/kiyohara/orion_nas/orion_nas-
20081107.tar.gz
~/kuronet/src $ patch -u -p0 <orion_nas-20081107.diff
~/kuronet/src $ tar -xzv <orion_nas-20081107.tar.gz
~/kuronet/src $ cd sys/arch/arm/marvell/
~/kuronet/src/sys/arch/arm/marvell $ nano if_oriongbe.c
        ...change the line:
#include "rnd.h"
        ...to:
//#include "rnd.h"
        ...save the file
~/kuronet/src $ cd ../../evbarm/conf
~/kuronet/src/sys/arch/evbarm/conf $ echo 'include "arch/evbarm/conf/
KUROBOX_PRO"' \
        > KUROBOX_PRO_INSTALL
~/kuronet/src/sys/arch/evbarm/conf $ echo 'no pseudo-device md' >>
KUROBOX_PRO_INSTALL
~/kuronet/src/sys/arch/evbarm/conf $ echo 'include "arch/evbarm/conf/
INSTALL"' \
        >> KUROBOX_PRO_INSTALL
~/kuronet/src/sys/arch/evbarm/conf $ cd ../../../../distrib/evbarm/
instkernel/instkernel
~/kuronet/src/distrib/evbarm/instkernel/instkernel $ nano Makefile
        ...add the line
                KUROBOX_PRO_INSTALL       ${RAMDISK}        -        \
        ...after the line
                IQ80321_INSTALL           ${RAMDISK}        -        \
        ...save the file
~/kuronet/src $ cd ../../../../etc/etc.evbarm
~/kuronet/src/etc/etc.evbarm $ nano Makefile.inc
        ...change the line:
                        IQ80310 IQ80321 SMDK2410 SMDK2800 \
        ...to:
                        IQ80310 IQ80321 KUROBOX_PRO SMDK2410 SMDK2800 \
        ...save the file
```

download and patch the necessary files are shown in Listing 4. Note that we are keying our CVS pull to a certain date (11/07/2008) to insure that the patch/tar files will be consistent with the NetBSD source code.

Once the source code has been retrieved and patched, we need to make the changes shown in the bottom part of Listing 4. In the directory `src/sys/arch/arm/orion`, you'll find all the files specific to this particular processor that are needed for the kernel. It is instructive to poke around in here and the evbarm directory a bit, particularly if you're thinking about porting NetBSD to a new processor or a new configuration of an already supported processor. One file in this directory, `if_oriongbe.c`, needs to be edited to remove an included file that prevents the source from building properly. The file `KUROBOX_PRO_INSTALL` is a new kernel configuration file that is used to build the install kernel, and

Listing 5. Command line steps to build NetBSD, with abbreviated output

```
~/kuronet/src/sys/arch/arm/marvell $ cd ~/kuronet/src/
~/kuronet/src $ ./build.sh -O ../obj -T ../tools -m
evbarm-el tools
<...snip...>
===> Tools built to /home/bigdon/kuronet/src/../tools
===> Summary of results:
        build.sh command: ./build.sh -O ../obj -T
../tools -m evbarm-el tools
        build.sh started: Sun Nov 23 11:03:13 EST
2008
        NetBSD version:   5.99.01
        MACHINE:          evbarm
        MACHINE_ARCH:     arm
        Build platform:   CYGWIN_NT-5.1
1.5.25(0.156/4/2) i686
        HOST_SH:          /usr/bin/sh
        No /home/bigdon/kuronet/src/../tools/bin/
nbmake, needs building.
        Bootstrapping nbmake
        TOOLDIR path:     /home/bigdon/kuronet/src/
../tools
        DESTDIR path:     /home/bigdon/kuronet/src/
../obj/destdir.evbarm
        RELEASEDIR path:  /home/bigdon/kuronet/src/
../obj/releasedir
        Created /home/bigdon/kuronet/src/../tools/
bin/nbmake
        makewrapper:      /home/bigdon/kuronet/src/
../tools/bin/nbmake-evbarm-el
        Updated /home/bigdon/kuronet/src/../tools/
bin/nbmake-evbarm-el
        Tools built to /home/bigdon/kuronet/src/../
tools
        build.sh ended:   Sun Nov 23 11:47:36 EST
2008
 ~/kuronet/src $ ./build.sh -O ../obj -T ../tools -U
-u -m evbarm-el distribution
<...snip...>
===> Successful make distribution
===> Summary of results:
        build.sh command: ./build.sh -O ../obj -T
../tools -U -u -m evbarm-el distribution
        build.sh started: Sun Nov 23 12:25:58 EST
2008
        NetBSD version:   5.99.01
        MACHINE:          evbarm
        MACHINE_ARCH:     arm
        Build platform:   CYGWIN_NT-5.1
1.5.25(0.156/4/2) i686
        HOST_SH:          /usr/bin/sh
        TOOLDIR path:     /home/bigdon/kuronet/src/
../tools
        DESTDIR path:     /home/bigdon/kuronet/src/
../obj/destdir.evbarm
        RELEASEDIR path:  /home/bigdon/kuronet/src/
../obj/releasedir
        makewrapper:      /home/bigdon/kuronet/src/
../tools/bin/nbmake-evbarm-el
        Updated /home/bigdon/kuronet/src/../tools/
bin/nbmake-evbarm-el
        Successful make distribution
        build.sh ended:   Sun Nov 23 22:44:17 EST
2008
~/kuronet/src $  ./build.sh -O ../obj -T ../tools -U
-u -m evbarm-el -V KERNEL_SETS=KUROBOX_PRO release
<...snip...>
===> Successful make release
===> Summary of results:
        build.sh command: ./build.sh -O ../obj -T
../tools -U -u -m evbarm-el -V KERNEL_SETS=KUROBOX_PRO
release
        build.sh started: Mon Nov 24 16:36:11 EST
2008
        NetBSD version:   5.99.01
        MACHINE:          evbarm
        MACHINE_ARCH:     arm
        Build platform:   CYGWIN_NT-5.1
1.5.25(0.156/4/2) i686
        HOST_SH:          /usr/bin/sh
        TOOLDIR path:     /home/bigdon/kuronet/src/
../tools
        DESTDIR path:     /home/bigdon/kuronet/src/
../obj/destdir.evbarm
        RELEASEDIR path:  /home/bigdon/kuronet/src/
../obj/releasedir
        makewrapper:      /home/bigdon/kuronet/src/
../tools/bin/nbmake-evbarm-el
        Updated /home/bigdon/kuronet/src/../tools/
bin/nbmake-evbarm-el
        Successful make release
        build.sh ended:   Mon Nov 24 22:57:47 EST
2008
```

should be put in the `src/sys/arch/evbarm/conf` directory. Two other files that are part of the standard NetBSD build process also need to modified to build the install kernel. In general, modifying NetBSD-supplied source files isn't recommended since any changes you make will be overwritten when you use CVS to update your source tree. Unfortunately, there doesn't seem to be any other easy way to do this for a board that isn't fully supported yet.

We'll use the standard NetBSD build process as shown in Listing 5. Since the Marvell processor is ARM-based and the code is designed to run in little-endian mode, you need to use the `-m evbarm-el` option for `build.sh`. It will take a little time to build the entire thing. Be patient. After all, Rome wasn't built in a day and most likely, NetBSD won't be either (well, maybe a long day). Part of the reason for the time it takes is the large number of files that will be built. Though convenient for building the entire set of packages, we'll also be building kernels for boards and/or processors that we won't be using because there is no easy way to shut them off. Another part of the reason is that Cygwin, since it is built on top of Windows, is not as fast as native version of Linux. But for the most part, the time it takes won't matter since it goes pretty quickly once you've built it the first time, assuming you're making minor changes from that point.

When the build is finished, you'll need to extract the installation and regular kernels and put them someplace that the TFTP server can get to, as shown in Listing 6. Generally, leaving them in the same directory is fine. We'll also need FTP access to the distribution files that can be found at `~/kuronet/obj/release/evbarm/binary/sets`. But, we won't need to extract these files – that is part of what the installation kernel does to install NetBSD.

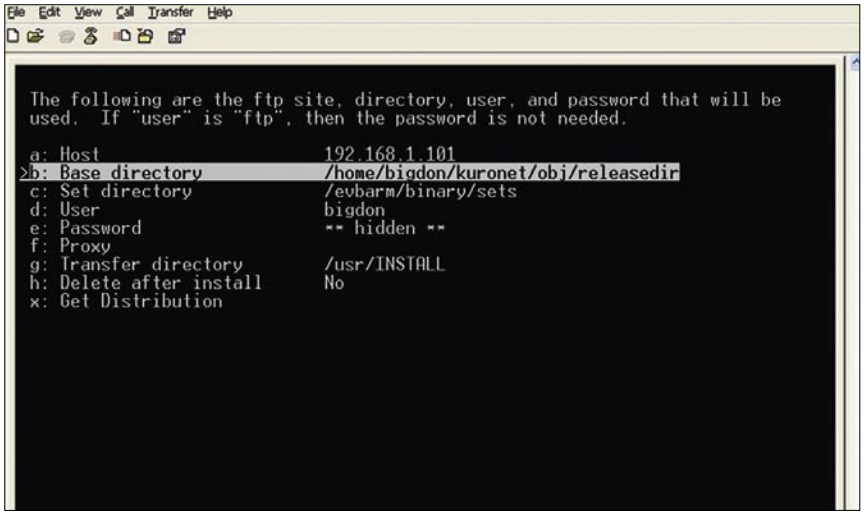


Figure 3. Setting up NetBSD to download the installation files from the local Windows machine

Booting and Installing NetBSD

Once the kernel is built, we'll use the install kernel to install NetBSD to a USB memory stick. Start the `tftpd32` program on your windows machine and set the current directory to `~/kuronet/obj/releasedir/evbarm/installation/instkernel` and boot up your Kurobox, as shown in Listing 7. If, like me, you've already installed Debian to your K-Pro, you would normally boot into Debian unless you stop the process by hitting a key on the keyboard. The default addresses for the K-Pro and TFTP server are 192.168.11.150 and 192.168.11.1, respectively. If your network is set up differently, you can change the default addresses as shown in the listing. If you change either of the addresses, but don't save the changes with the command `saveenv`, the changes will only last until the next time you boot up. This is probably the behavior you want for now. Then, have the K-Pro load the installation kernel by using the command line as shown in the listing.

Listing 6. Finding and expanding the installation and standard kernels for the Kurobox Pro

```
~/kuronet $ cd ~/kuronet/obj/releasedir/evbarm/installation/instkernel/
~/kuronet/obj/releasedir/evbarm/installation/instkernel $ ls -la | grep KURO
-r--r--r--  1 bigdon None 3471901 Nov 29 08:15 netbsd-KUROBOX_PRO_INSTALL.bin.gz
-r--r--r--  1 bigdon None 3472682 Nov 29 08:15 netbsd-KUROBOX_PRO_INSTALL.gz
-r--r--r--  1 bigdon None 6478285 Nov 29 08:15 netbsd-KUROBOX_PRO_INSTALL.srec.gz
-r--r--r--  1 bigdon None  129842 Nov 29 08:15 netbsd-KUROBOX_PRO_INSTALL.symbols.gz
~/kuronet/obj/releasedir/evbarm/installation/instkernel $ gunzip \
> netbsd-KUROBOX_PRO_INSTALL.bin.gz
~/kuronet/obj/releasedir/evbarm/installation/instkernel $ ls -la | grep KURO
-r--r--r--  1 bigdon None 8210384 Nov 29 08:15 netbsd-KUROBOX_PRO_INSTALL.bin
-r--r--r--  1 bigdon None 3472682 Nov 29 08:15 netbsd-KUROBOX_PRO_INSTALL.gz
-r--r--r--  1 bigdon None 6478285 Nov 29 08:15 netbsd-KUROBOX_PRO_INSTALL.srec.gz
-r--r--r--  1 bigdon None  129842 Nov 29 08:15 netbsd-KUROBOX_PRO_INSTALL.symbols.gz
~/kuronet/obj/releasedir/evbarm/installation/instkernel $ cd ../../binary/sets/
~/kuronet/obj/releasedir/evbarm/binary/sets $ tar -xzf kern-KUROBOX_PRO.tgz
~/kuronet/obj/releasedir/evbarm/binary/sets $ ls -la | grep netbsd
-rwxr-xr-x  1 bigdon None  5693079 Nov 29 03:22 netbsd
-rwxr-xr-x  1 bigdon None  5061572 Nov 29 03:22 netbsd.bin
```

Note that the load address is `0x8000`. Once the download is finished, use the go command to start the kernel and you should see the NetBSD installation startup screen. Select your language preference, hit return, and you should see the screen shown in Figure 2. At this point, you're ready to install NetBSD to your USB drive. Note that the USB drive is `sd0`.

Before you can install NetBSD, you'll need to start up the Cygwin FTP server as shown in Listing 3. Then, you'll need to select FTP as the installation method and provide the following settings for locations, addresses, and the like. My setup is shown in Figure 3, and if you've been following along during this build, yours should be similar except for the network addresses. From this point on, just follow the installation instructions and you should have no problems. Incidentally, when you set up your network for FTP, NetBSD will try to ping your name server to see if the network is setup correctly. If your nameserver is like mine, it won't respond to the ping, so NetBSD thinks it has failed. However, you can choose to ignore the failure and attempt to install NetBSD anyway, which is the right choice. If you haven't set up the network correctly, it will error out of the installation and you can try again. If worst comes to worst, you can reset the `K-Pro` and restart the installation.

At the end of the installation, you will get a message that your installation doesn't pass a sanity check. This is because the installation program is assuming a `i386` installation with a kernel and boot loader on the disk – we don't have that with the `K-Pro`. Ignore the message and reset the `K-Pro`. This time, use TFTP to load your standard kernel, `netbsd.bin`, from `~/kuronet/obj/releasedir/evbarm/binary/sets`. You should be able to log in as root, set up users, and use the K-Pro like any standard NetBSD machine. Have fun!

What's Next?

While NetBSD runs on the Kurobox Pro, one can't say that the installation is trivial or that NetBSD fully supports the K-Pro, though it probably will soon. Currently, NetBSD doesn't have a boot loader that will work with the Arm-based devices. Moreover, NetBSD doesn't have support for the K-Pro's flash memory so that you can install the kernel to flash instead of having to load it with TFTP. If you've installed Debian on your K-Pro, you can get your K-Pro to boot NetBSD on power-up by loading the kernel onto the /boot directory and changing the U-Boot environment to boot NetBSD instead of Linux. But, we still have some work ahead of us before all of that can be done natively with NetBSD.

Regardless, there are still lots of cool things that can be done with NetBSD and the Kurobox Pro. The machine is powerful enough to serve music, video, web pages, or a variety of other functions including, somewhat ironically, a NAS device. With a maximum power draw of only 25 watts, the Kurobox Pro would make an excellent green system for those that want to surf the net, but don't need those power-hungry graphics boards to play video games with. Hmmm…car computer, anyone?

Acknowledgments

Many thanks to KIYOHARA Takashi and the people at the NetBSD port-arm mailing list for getting NetBSD to run on the Kurobox/Pro and all of those other fun little processors.

References

- For information on NetBSD products, see *http://www.netbsd.org/gallery/products.html*.
- Windows Services for Unix, *http://technet.microsoft.com/en-us/library/bb496506.aspx*.
- Subsystem for Unix Applications Interix Add-ons, *http://www.suacommunity.com/*.
- A good source for help with NetBSD is the NetBSD wiki, *http://wiki.netbsd.se/Main_Page*.
- The NetBSD web site can be found at *http://www.netbsd.org/*.

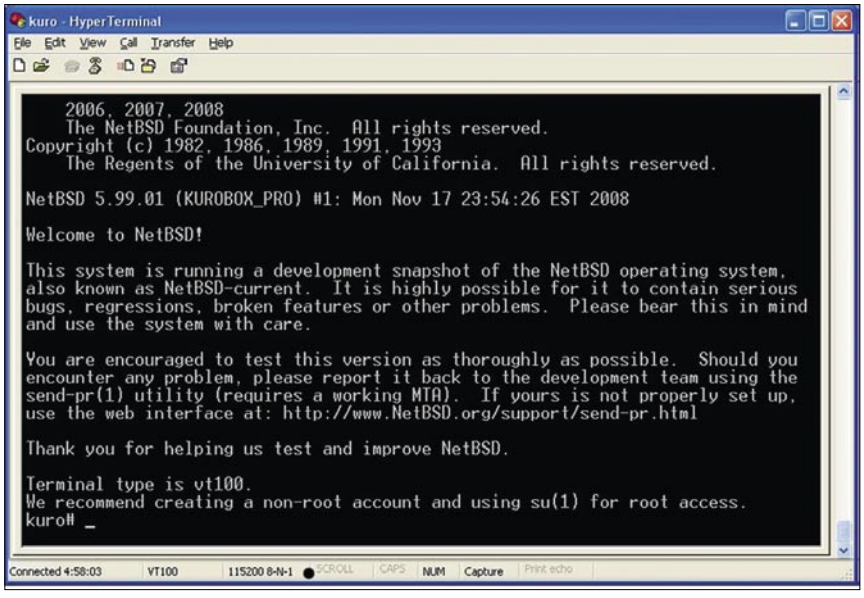


Figure 4. Running NetBSD on the Kurobox Pro for the first time

About the Author

Don Hayford is a Research Leader at Battelle Memorial Institute, where he specializes in the development of data acquisition, analysis, and control systems for customers. Don has been involved with microprocessors from the time they doubled in size from four bits to eight, and once knew how to boot up a PDP-11 using the front panel switches. In his career, he has written software for the CP/M, RT-11, MS-DOS, Apple DOS, Windows, Linux, and BSD operating systems using assembler, Basic, C, C++, C#, and Fortran (which would have been a much more interesting sentence if C++ and C# were called D and E, instead). Married with three children, Don and his wife like to spend their free time cooking, traveling, and playing WII sports.

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

ABC's of ZFS

Amjith Ramanujam

ZFS is a state of the art filesystem developed by Sun Microsystems. ZFS was first introduced in the OpenSolaris operating system and was later ported to FreeBSD 7.

Since PC-BSD 7 is based on the FreeBSD 7 kernel, ZFS comes bundled with PC-BSD 7. ZFS is a new and radical approach that deviates from the traditional assumptions of exsiting filesystems. ZFS is hailed as the Swiss Army Knife by Unix system administrators – it is efficient, powerful, and user-friendly.

Prominent Features of ZFS:

- Unlimited constant time snapshots – A snapshot is a read-only copy of a filesystem at the current time. Snapshots are very efficient for backups.
- Constant time filesystem cloning – A clone of a filesystem is a writable copy of a filesystem. Clones facilitate easier replication of software installation and common data.
- Raid-Z replication model is built into the filesystem. Raid-Z is similar to Raid-5, with some improvements, and it does not require the expensive NVRAM hardware.
- Built-in compression.
- Checksums for every block to detect data corruption
- Online disk-scrubbing – This mechanism traverses every block in the filesystem and validates it against the checksum while the storage system is live and in use.
- Unlimited files, directories, links and snapshots
- I/O pipelining – This is similar to CPU pipelining. In layman's terms, an efficient I/O load balancing.

The Three Levels of ZFS Architecture:

- zpool – This is similar to the concept of volumes.
- ZFS filesystem – Multiple zfs filesystems can exist inside a zpool. A zfs filesystem can span over one or more memory devices attached to the zpool.
- Memory devices – These are actual hard disks that belong in a zpool.

Setup:

Before we start experimenting with ZFS, lets make sure we have the required tools.

- Enable the ZFS option during the installation of PC-BSD 7.
- You must have super user privileges.
- If you have spare hard disks, you can use them. Alternately, you can use files to create virtual disks for experimentation.

Since I don't have any spare disks to experiment with, I decided to use 'mkfile' to create files to be used as disks.

Check if mkfile is installed in your system, if not, it can be installed from FreeBSD ports (Listing 1).

Now that we've created files that are 128MB in size, we need to turn them into memory devices and attach it to the system.

Please use full path to the file while issuing the following commands (Listing 2).

Executing the `mdconfig` command will create a memory device out of the file and return the name of the device (eg: `md1`, .. `md5`). These new devices can be found under `/dev/md1` to `/dev/md5`.

zpool:

Let's use one of the new memory devices to create a zpool called `test_p1` (creative huh?)

```
# zpool create test_p1 /dev/md1
```

The pool is automatically mounted under `/test_p1`.

Listing 1. Create Files

```
# pkg_info -a | grep mkfile
# pkg_add -r mkfile

# mkfile 128m /home/amjith/disk1
# mkfile 128m /home/amjith/disk2
# mkfile 128m /home/amjith/disk3
# mkfile 128m /home/amjith/disk4
# mkfile 128m /home/amjith/disk5
```

Listing 2. Create Memorey Devices

```
# mdconfig -a -t vnode -f /home/amjith/disk1
# mdconfig -a -t vnode -f /home/amjith/disk2
# mdconfig -a -t vnode -f /home/amjith/disk3
# mdconfig -a -t vnode -f /home/amjith/disk4
# mdconfig -a -t vnode -f /home/amjith/disk5
```

Listing 3. Add Mirror Device

```
# zpool attach test_p1 /dev/md1 /dev/md2
# zpool status
 pool: test_p1
 state: ONLINE
 scrub: resilver completed with 0 errors on Sun Dec 7
18:30:14 2008
 config:

 NAME STATE READ WRITE CKSUM
 test_p1 ONLINE 0 0 0
 mirror ONLINE 0 0 0
 md1 ONLINE 0 0 0
 md2 ONLINE 0 0 0

errors: No known data errors
```

Listing 4. Checksum Validation

```
# zpool scrub test_p1
# zpool status test_p1
 pool: test_p1
 state: ONLINE
status: One or more devices could not be used because
the label is missing or
 invalid. Sufficient replicas exist for the pool to
continue
 functioning in a degraded state.
action: Replace the device using 'zpool replace'.
 see: http://www.sun.com/msg/ZFS-8000-4J
 scrub: scrub completed with 0 errors on Sun Dec 7 18:
40:15 2008
config:

 NAME STATE READ WRITE CKSUM
 test_p1 ONLINE 0 0 0
 mirror ONLINE 0 0 0
 md2 ONLINE 0 0 0
 md1 UNAVAIL 0 0 0 corrupted data

errors: No known data errors
```

Listing 5. Replace Disk in zpoo

```
# zpool replace test_p1 /dev/md1 /dev/md3
# zpool status test_p1
 pool: test_p1
 state: ONLINE
 scrub: resilver completed with 0 errors on Sun Dec 7
19:47:11 2008
config:

 NAME STATE READ WRITE CKSUM
 test_p1 ONLINE 0 0 0
 mirror ONLINE 0 0 0
 md3 ONLINE 0 0 0
 md2 ONLINE 0 0 0

errors: No known data errors
```

Listing 6. List of Filesystems

```
# zfs list
NAME USED AVAIL REFER MOUNTPOINT
test_p1 250K 214M 21K /test_p1
test_p1/misc 65.5K 214M 21K /test_p1/misc
test_p1/misc/doc 18K 214M 18K /test_p1/misc/doc
test_p1/misc/important 26.5K 214M 26.5K /test_p1/misc/
important
test_p1/music 18K 214M 18K /test_p1/music
```

Listing 7. Compression Enabled Filesystem

```
# zfs list
NAME USED AVAIL REFER MOUNTPOINT
test_p1 1.01M 213M 21K /test_p1
test_p1/misc 844K 213M 21K /test_p1/misc
test_p1/misc/doc 797K 213M 797K /test_p1/misc/doc
test_p1/misc/important 26.5K 213M 26.5K /test_p1/misc/
important
test_p1/music 18K 213M 18K /test_p1/music
```

Listing 8. Setting Quota for Filesystem

```
# zfs set quota=50m test_p1/music
# zfs get quota
NAME PROPERTY VALUE SOURCE
test_p1 quota none default
test_p1/misc quota none default
test_p1/misc/doc quota none default
test_p1/misc/important quota none default
test_p1/music quota 50M local
```

Listing 9. Setting Reservation for Filesystem

```
 # zfs set reservation=20m test_p1/misc/doc
 # zfs get reservation
NAME PROPERTY VALUE SOURCE
test_p1 reservation none default
test_p1/misc reservation none default
test_p1/misc/doc reservation 20M local
test_p1/misc/important reservation none default
test_p1/music reservation none default
```

Listing 10. Quota and Reservation in ZFS Lisiting

```
 # zfs list
NAME USED AVAIL REFER MOUNTPOINT
test_p1 20.2M 194M 21K /test_p1
test_p1/misc 20.0M 194M 21K /test_p1/misc
test_p1/misc/doc 797K 213M 797K /test_p1/misc/doc
test_p1/misc/important 26.5K 194M 26.5K /test_p1/misc/important
test_p1/music 18K 50.0M 18K /test_p1/music
```

Listing 11. Creating Files in 'music' Filesystem

```
 # cd /test_p1/music
 # mkfile 2m music1 music2 music3
 # zfs list
NAME USED AVAIL REFER MOUNTPOINT
test_p1 26.2M 188M 21K /test_p1
test_p1/misc 20.0M 188M 21K /test_p1/misc
test_p1/misc/doc 797K 207M 797K /test_p1/misc/doc
test_p1/misc/important 26.5K 188M 26.5K /test_p1/misc/important
test_p1/music 6.02M 44.0M 6.02M /test_p1/music
```

Listing 12. Snapshot of 'music' Filesystem

```
 # zfs snapshot test_p1/music@20081107
 # zfs list
NAME USED AVAIL REFER MOUNTPOINT
test_p1 26.2M 188M 21K /test_p1
test_p1/misc 20.0M 188M 21K /test_p1/misc
test_p1/misc/doc 797K 207M 797K /test_p1/misc/doc
test_p1/misc/important 26.5K 188M 26.5K /test_p1/misc/important
test_p1/music 6.02M 44.0M 6.02M /test_p1/music
test_p1/music@20081107 0 - 6.02M -
```

Listing 13. Status of 'music' Filesystem Snapshot

```
 # rm -f music3
 # zfs list
NAME USED AVAIL REFER MOUNTPOINT
test_p1 26.3M 188M 21K /test_p1
test_p1/misc 20.0M 188M 21K /test_p1/misc
test_p1/misc/doc 797K 207M 797K /test_p1/misc/doc
test_p1/misc/important 26.5K 188M 26.5K /test_p1/misc/important
test_p1/music 6.04M 44.0M 4.02M /test_p1/music
test_p1/music@20081107 2.02M - 6.02M -
```

```
 # zpool list
 NAME SIZE USED AVAIL CAP HEALTH
ALTROOT
 test_p1 123M 110K 123M 0% ONLINE -
```

Alternatively, you can choose a mount point for the zpool.

```
 # zpool create -m /pools test_p1
/dev/md1
```

This is particularly useful if you are trying to create a zpool by the name `tmp` but there is already a folder named `/tmp`.

It is time to add some redundancy to the system. Lets add `/dev/md2` to the existing pool and use it for mirroring (Listing 3).

Resilvering is the process of synching up the disks to the same state.

Now even if one of the disks fail, we'll be able to continue operation since the disks are mirrored.

Let's try to corrupt one of the disks and see how the mirrors hold up.

```
 # dd if=/dev/random of=/home/amjith/
disk1 bs=512 count=1
```

We can force zfs to do a checksum validation using the zfs scrub to detect the corruption (Listing 4).

This immediately identifies which of the mirrored copies is corrupted and recommends a corrective action. However, the zpool is still functional since one of the mirrors is valid.

The corrupted disk can be detached from the zpool or it can be replaced by a different disk. I decided to replace it with a new device (Listing 5).

Now the corrupted device is replaced in the zpool and resilvered.

What if we are running out of space in the zpool and we'd like to increase the size? We can add more devices to the zpool without taking the zpool offline.

```
 # zpool list
NAME SIZE USED AVAIL CAP HEALTH
ALTROOT
test_p1 123M 110K 123M 0% ONLINE -
```

Let's add md4 to the zpool to see how that affects the size. Since the existing zpool has a mirrored configuration, we must add a mirror to the device we are

adding to the pool. `/dev/md5` will be used as the mirror for `md4`.

```
# zpool add test_p1 mirror /dev/md4
/dev/md5
```

Here is the proof that size of the zpool has doubled.

```
# zpool list
NAME SIZE USED AVAIL CAP HEALTH
ALTROOT
test_p1 246M 114K 246M 0% ONLINE -
```

Filesystems:

Now that we've understood the concept of zfs pools, we are ready to create some filesystems inside the zpool.

```
# zfs create test_p1/music
# zfs create test_p1/misc
```

Create a compression-enabled file-system inside the misc to hold the docs.

```
# zfs create test_p1/misc/doc
# zfs set compression=gzip test_p1/
misc/doc
```

Create a filesystem with extra redundancy built-in for important stuff.

```
# zfs create test_p1/misc/important
# zfs set copies=2 test_p1/misc/
important
```

This will ensure that there are two copies for every new write into the `test_p1/misc/important` filesystem (Listing 6).

You'll notice that all the filesystems have the full capacity of the zpool. This is because the filesystem will expand to occupy the space as needed. All the resources in the zpool are common to all the filesystems unless we reserve some space or set quota for certain filesystems.

Let's verify the compression feature of ZFS. Create a ascii text file in the `test_p1/misc/doc` filesystem that is 1Mb in size.

```
# < /dev/urandom tr -dc A-Z-a-z-0-9 |
  head -c 1048576 > /test_p1/misc/
doc/doc1.txt
# ls -lh
-rw-r--r-- 1 root wheel 1.0M Dec 7
22:59 doc1.txt
```

This validates that the file is actually 1MB in size, but a zfs listing shown below denotes that only 797KB of the disk space is used by the 'doc' filesystem (Listing 7).

Alternately, any file created in the `test_p1/misc/important` filesystem will take up twice as much disk space, since it makes copies of every file.

If we need to ensure that a filesystem doesn't exceed a certain storage limit, we can set quota for each filesystem (Listing 8).

If we want to set aside some disk space exclusively for one filesystem, we can reserve it. This will ensure that the reserved space will be available for use in that filesystem (Listing 9).

The quota and reservations will be reflected in the zfs listing (Listing 10).

My favorite feature of the ZFS filesystem is the constant time snapshots. Snapshots are a great tool for creating backups of your filesystem (Listing 11).

Let's take a snapshot of the music filesystem, then we can start messing with the files. The format of the snapshot name must be *filesystem@snapshotname* (Listing 12).

Take note that 0 bytes is used by the snapshot, since the original filesystem hasn't changed from the state of the snapshot. Let's make some modifications to the music directory and observe how the snapshot is affected (Listing 13).

Since we deleted one of the 2MB files from the music filesystem, that change was updated in the snapshot.

We can always rollback to the state of a previous snapshot, thus restoring the deleted file.

```
# zfs rollback test_p1/music@20081107
# ls /test_p1/music/
music1 music2 music3
```

The deleted file music3 is restored as a part of the rollback operation (Listing 14).

Conclusion

ZFS is a ground breaking paradigm shift from the traditional concept of a filesystem. It eliminates the need for volume management tools and opens up the possibility of using cheap disks to build reliable storage systems. The features showcased in this article are only a starting point to ZFS. Give it a try, and discover the filesystem that will make you wonder how you ever lived without it.

On the 'Net

- *http://flux.org.uk/howto/solaris/zfs_tutorial_01*
- *http://flux.org.uk/howto/solaris/zfs_tutorial_02*
- *http://opensolaris.org/os/community/zfs/*

Listing 14. Rollback operation

```
# zfs list
NAME USED AVAIL REFER MOUNTPOINT
test_p1 26.2M 188M 21K /test_p1
test_p1/misc 20.0M 188M 21K /test_p1/misc
test_p1/misc/doc 797K 207M 797K /test_p1/misc/doc
test_p1/misc/important 26.5K 188M 26.5K /test_p1/misc/important
test_p1/music 6.02M 44.0M 6.02M /test_p1/music
test_p1/music@20081107 0 - 6.02M -
```

About the Author

Amjidanutpan Ramanujam
I am a software programmer and an open source enthusiast from Salt Lake City, Utah. My favorite hobby is to try different operating systems and explore the new features and technology that it brings to the table. I write for OSNews.com in my spare time.

Django on FreeBSD

Dan Fairs

Dan Fairs, Director of Fez Consulting Ltd., a UK-based software development consultancy, introduces Django: a web framework for perfectionists with deadlines.

What does it look like? Django is a high-level web framework, written in the Python programming language. It fits neatly into the standard web stack, using a web server (commonly Apache or nginx) and a relational database. It can use MySQL, PostgreSQL or Oracle; or even SQLite, if your project is small.

We're going to use SQLite. Moving to another database is straightforward, though, as Django takes care of the differences between the databases for you.

Being pure Python, Django is as happy on FreeBSD as it is on Linux or Mac OS X, or even Microsoft Windows – should you feel that way inclined!

Django Application Architecture

Django application architecture is very simple, broadly following the Model-View-Controller pattern dominant in today's web frameworks, with a small change: Django views would be called 'controllers' in other frameworks, and other frameworks' views translate into Django templates. Django's authors often refer to Django as an *MTV* – Model-Template-View – framework.

Some web frameworks generate their data models by introspecting a database. Django works the other way around: the developer defines Django models in Python, and Django can then generate the SQL required to create the database, and then query and update it. (Django can in general work with an existing database although there are some restrictions on the schemas that it can handle.) Again, common to other frameworks, Django uses an *Object-Relational Mapper* (ORM) based on the ActiveRecord pattern.

Django views are responsible for translating an HTTP request from a browser into an HTTP response. In doing this, they will most likely use the Django ORM to obtain and modify data, and then use Django's templating system to create an HTML body for the response. Note though that the view really does work at the request/response level.

There's no reason why the response has to be HTML (it could be a JSON document), or why the Django ORM has to be used to fetch data (perhaps you're talking to a CouchDB database, or an LDAP directory.) You have complete control.

The final piece of the puzzle is how incoming requests are matched to a view. The Django developer simply sets up a mapping of regular expressions to views, and the URLs of incoming requests are simply matched to the list of regular expressions. The request is then dispatched to the view corresponding to the matched regular expression.

To summarise, let's take a quick look at how an HTTP request is processed through the Django stack:

- An HTTP request arrives at the Django web server
- Django compares the URL to a set of regular expressions, each of which is mapped to a view
- When a regular expression is matched to the URL, the corresponding view is invoked
- The view is responsible for returning an HTTP response. It usually does this by using the Django ORM and template system to build an HTML document

Let's get a development environment up and running.

Prerequisites

You are going to need some software installed before you can show off your first web site. Fortunately, FreeBSD helps us out here, as everything you need is in the ports tree. So go update your ports, and install the following software (I just accepted all the default options):

- lang/python25
- databases/py-sqlite3
- dev/py-virtualenv
- graphics/py-imaging
- dev/py-django

You might want to uncheck py-imaging's Tkinter support, else you'll pull in X11 too.

I'm using FreeBSD 7.0-RELEASE.

Before we begin

This article will deviate a little from the official Django tutorial. It will make use of a couple of third-party packages which will make life easier, and will also lay out your code in a slightly different way, giving greater opportunities for future reuse.

Creating a development environment

I confess: I sneaked something into that prerequisites which didn't really need to be there. That something is `virtualenv`. However, it's so useful, I'd recommend you use it any time that you're doing Python development. What `virtualenv` lets you do is to create a *sandbox* Python environment. This lets you write and install modules that don't affect the system Python, without requiring superuser privileges and without interfering with any other Python projects you're working on.

Let's go ahead and create and activate a sandbox for your first Django application: see Listing 1.

Note how the activate script changed your command prompt. This is a visual clue to show that you're working in a virtual Python environment called `babble_env`.

Any Python modules that you install now using the standard `easy_install` command will be installed into this virtual environment. Of course – if you install a module using the FreeBSD ports collection, it'll be installed into the system Python.

We have one more Python module to install now:

```
(babble_env)[dan@freebsd-vm ~]$ easy_
install fez.djangoskel
… lots of output …
```

Finished processing dependencies for fez.djangoskel

```
(babble_env)[dan@freebsd-vm ~/babble_
env]$
```

The package `fez.djangoskel` will be downloaded and installed into your virtual Python environment, together with its dependencies.

Projects and Applications

Django web sites are made up of two kinds of component: projects and applications. Applications are the meat of a site, and contain all of the logic and templates that make the site work. Projects are little more than a settings file (containing configuration information such as database connection details, time zone information, a list of installed applications, and so forth), and a `root` URL configuration (more on this later). A Django-powered site will typically have one project, and many applications.

Well-written Django applications should be self-contained, and reusable in other web sites aside from the one for which it was originally developed. Getting into the habit of writing reusable applications will save you huge amounts of time in the long run.

Babble is just going to have one application, and of course one project to store its settings. The official Django tutorial refers to using the `django-admin.py` command to create these. You can do that, but it makes life a little more complicated down the line, so we're going to spend a some time now to save time later. We're going to use the `fez.djangoskel` packaged we just installed, which provides some useful templates for Django applications and projects.

Create a project:

```
(babble_env)[dan@freebsd-vm ~/babble_
env]$ paster create -t django_project
```

Selected and implied templates:

Listing 1. Creating a Python virtualenv

```
[dan@freebsd-vm ~]$ virtualenv babble_env
New python executable in babble/bin/python2.5
Also creating executable in babble/bin/python
Installing setuptools.............done.
[dan@freebsd-vm ~]$ cd babble_env/
[dan@freebsd-vm ~/babble_env]$ source bin/activate
(babble_env)[dan@freebsd-vm ~/babble_env]$
```

Listing 2. Enabling the babble module

```
(babble_env)[dan@freebsd-vm ~/babble_env/babbleproj]$ cd ../babble
(babble_env)[dan@freebsd-vm ~/babble_env/babble]$ python setup.py develop
running develop
running egg_info
[… lots more output …]
Finished processing dependencies for babble==0.1dev
(babble_env)[dan@freebsd-vm ~/babble_env/babble]$
```

Listing 3. Running the Django development server

```
(babble_env)[dan@freebsd-vm ~/babble_env/babbleproj]$ cd ~/babble_env/
babbleproj/babbleproj/
(babble_env)[dan@freebsd-vm ~/babble_env/babbleproj/babbleproj]$ python
manage.py runserver
Validating models...
0 errors found


Django version 1.0.2 final, using settings 'babbleproj.settings'
Development server is running at http://127.0.0.1:8000/
Quit the server with CONTROL-C.
```

```
  fez.djangoskel#django_project
Template for a Django project
Enter project name:
```

Use babbleproj as your project name, and then accept the defaults for all the other settings by just pressing Enter.

Now let's create an application:

```
(babble_env)[dan@freebsd-vm ~/babble_
env]$ paster create -t django_app
```

Selected and implied templates:

```
  fez.djangoskel#django_app  Template
for a basic Django reusable
application
Enter project name:
```

Use babble as the name, and press Enter to accept the defaults for all the other questions. Once this is done, you should have one directory called `babbleproj` and another called `babble`. These top-level directories each contain the skeleton of a Python package, or `egg`. Eggs provide a way to reuse and distribute Python applications – any time you use `easy_install`, you are downloading and installing an egg from PyPI. Since we're going to be developing with these two eggs, we have to ensure that Python will be able to load them.

Let's do the babbleproj egg first. Don't be alarmed at all the output, it's fairly verbose:

```
(babble_env)[dan@freebsd-vm ~/babble_
env/babbleproj]$ python setup.py
develop
running develop
[… lots of output …]
Finished processing dependencies for
babbleproj==0.1dev
```

You can now confirm that Python will be able to load the module:

```
(babble_env)[dan@freebsd-vm ~/babble_
env/babbleproj]$ python
Python 2.5.2 (r252:60911, Dec 12 2008,
14:43:12)
[GCC 4.2.1 20070719  [FreeBSD]] on
freebsd7
Type "help", "copyright", "credits"
or "license" for more information.
>>> import babbleproj
>>> ^D
```

Listing 4. babbleproj/babbleproj/urls.py: enabling the admin interface

```
from django.conf.urls.defaults import *

# Uncomment the next two lines to enable the admin:

from django.contrib import admin

admin.autodiscover()

urlpatterns = patterns('',
    # Uncomment the admin/doc line below and add 'django.contrib.admindoc
s'
    # to INSTALLED_APPS to enable admin documentation:

    # (r'^admin/doc/', include('django.contrib.admindocs.urls')),

    # Uncomment the next line to enable the admin:
    (r'^admin/(.*)', admin.site.root),
)
```

Listing 5. babble/babble/models.py

```
from django.db import models
from datetime import datetime

# Create your models here.
class Message(models.Model):
    body = models.CharField(max_length=140)
    created_on = models.DateTimeField(default=datetime.now)
```

Listing 6. babble/babble/views.py

```
from django.shortcuts import render_to_response
from babble.models import Message

def view_messages(request):
    all_messages = Message.objects.all().order_by('-created_on')
    context = {
        'messages': all_messages
    }
    return render_to_response('babble/messages.html', context)
```

Listing 7. babble/babble/templates/babble/messages.html

```
<html>
  <head />
  <body>
    <dl>
      {% for message in messages %}
        <dt>Posted on: {{ message.created_on }}</dt>
        <dd>{{ message.body }}</dd>
      {% endfor %}
    </dl>
  </body>
</html>
```

You should not get any errors with the import line.

Press [*Ctrl*]+[*D*] to exit the Python interpreter, and do the same with the babble module: see Listing 2.

If you like, you can check that Python can now import the babble module using the same method we used before.

Running Django for the first time

When your application is running in production, you will use a 'real' web server, like Apache, nginx or lighttpd. Setting this up in a development is a little cumbersome. Django supplies its own development web server, which you can just start from the command-line: see Listing 3.

If you now visit *http://127.0.0.1:8000/* in your web browser, you should see a *Welcome to Django* page. Congratulations – you have a working Django development environment!

Configuration

Before we go any further, let's configure the SQLite database that we're going to use, and while we are at it, install Django's administrative application.

Stop the development server by pressing [*Ctrl*]+[*C*], and edit the settings.py file in the same directory. Look for the lines near the top which contain database configuration. Modify the `DATABASE_ENGINE` and `DATABASE_NAME` settings as follows:

```
DATABASE_ENGINE = 'sqlite3'
DATABASE_NAME = 'babble.db'
```

Scroll down to the bottom of the file and look for the `INSTALLED_APPS` setting. Let's add the Django administrative application. Modify the setting so that it looks like this (you should just need to add the last two lines):

```
INSTALLED_APPS = (
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.sites',
    'django.contrib.admin',
    'babble',
)
```

The admin interface, which is suitable for end-user use, will give us a simple way to add and edit both users and our application data.

The admin interface also keeps a simple audit trail of all data changes. As you'd expect, user and audit data are stored in database tables. Let's tell Django to create these tables in the database:

```
(babble_env)[dan@freebsd-vm
   ~/babble_env/babbleproj/babbleproj]
      $ python manage.py syncdb
```

You should see a series of *Creating table…* lines, as tables for the admin interface are created. You'll also see tables being created for Django's authentication system.

As part of this, you'll be prompted to create a superuser – follow the prompts, and keep a note of the credentials: you'll need them to log in.

Finally, you'll need to edit the urls.py file in the babbleproj/babbleproj directory. This will tell Django how to route URLs for the admin interface to the admin views. Open up the file, and uncomment the lines as directed so you end up with this: see Listing 4.

Fire up the Django development server again, and visit *http://127.0.0.1:8000/admin/*. Enter the username and password you just created, and explore the Django admin application.

Creating our models

We're going to create a simple Twitter clone, storing small messages in our database. Let's create a simple Python model for that. Open up the *babble_env/babble/babble/models.py* file, and amend it so it looks like this: see Listing 5.

That's all we need for our model: just a CharField of maximum length 140 for the

Listing 8. ~/babble_env/babble/babble/urls.py

```
from django.conf.urls.defaults import *
# URL patterns for babble

urlpatterns = patterns('babble.views',
  (r'messages/$', 'view_messages'),
)
```

Listing 9. babbleproj/babbleproj/urls.py: enabling the Babble application

```
from django.conf.urls.defaults import *

# Uncomment the next two lines to enable the admin:

from django.contrib import admin
admin.autodiscover()

urlpatterns = patterns('',
    # Uncomment the admin/doc line below and add 'django.contrib.admindoc
s'

    # to INSTALLED_APPS to enable admin documentation:

    # (r'^admin/doc/', include('django.contrib.admindocs.urls')),




    # Uncomment the next line to enable the admin:

    (r'^admin/(.*)', admin.site.root),
    (r'^babble/', include('babble.urls')),
)
```

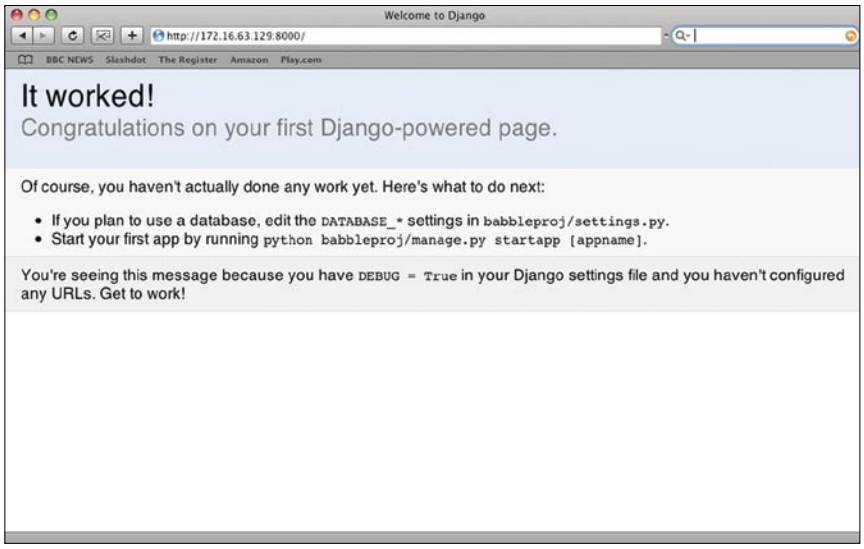


Figure 1. It worked

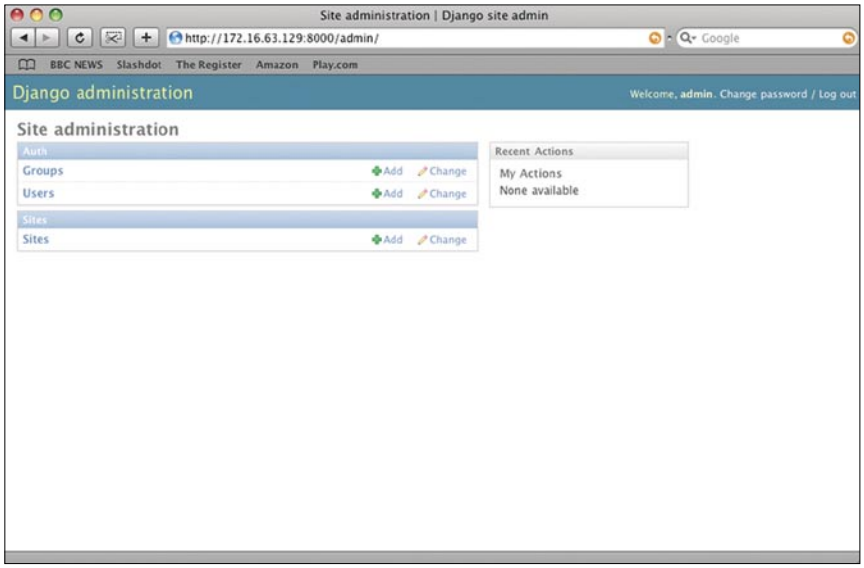


Figure 2. Admin default

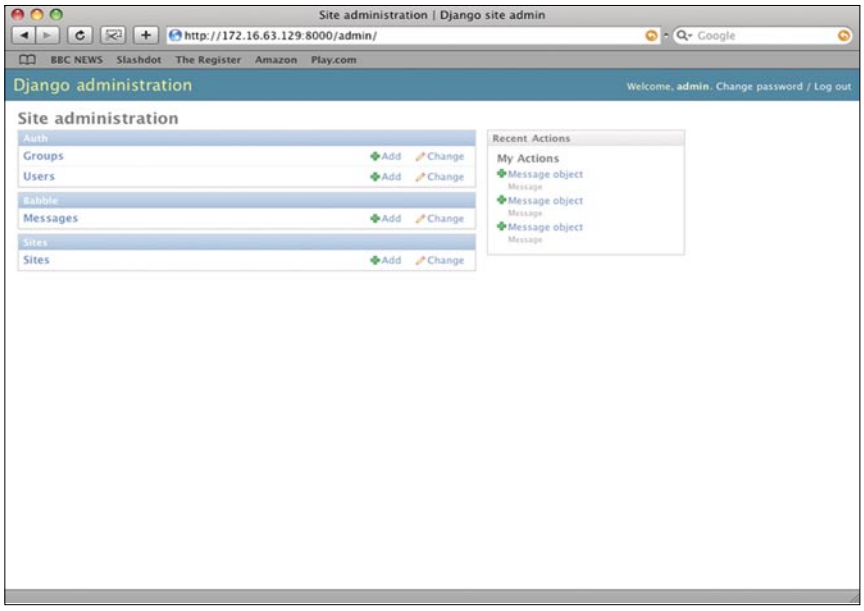


Figure 3. Admin with messages

body message, and a DateTimeField to hold the date and time that the message was created. This defaults to now, using the default keyword parameter to specify the now function from Python's datetime module.

The only other thing to do is to tell Django that it's OK to display the model in the admin interface. Open up admin.py in the same directory, and change it as follows:

```
from django.contrib import admin
from babble.models import Message
admin.site.register(Message)
```

Last coding step is to tell Django to load our Babble application, by adding it to the list of `INSTALLED_APPS` in the settings.py file. Amend that file as before, but `INSTALLED_APPS` should now look like this:

```
INSTALLED_APPS = (
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.sites',
    'django.contrib.admin',
    'babble',
)
```

Last thing to do is to ask Django to create your database tables. Rerun the python `manage.py` syncdb command, and you should see Django creating a table for your messages. Restart the development server, log in, and you should see that you now have an interface for adding and removing messages.

Go ahead and create some messages, and familiarise yourself with the way the Django admin application works.

Creating a public view

The final piece of our mini-application is to create a public view for people to see the messages that have been posted. Let's do that now. Open up the file babble/babble/views.py, and add the following code: see Listing 6.

Hopefully it's fairly simple to see what's going on here. We define a view function that takes a single argument, the request. We then use the ORM to fetch all the `Message` objects in the database, sorted in descending order

by the `created_on field` (that's what the dash before `created_on` signifies). This list is then stored in a dictionary under a key called `messages`, and passed to the template named `babble/messages.html`.

We now need to create the messages.html template for the view to render. Create a directory as follows:

```
mkdir -p ~/babble_env/babble/babble/templates/babble
```

Add a file called messages.html to that directory with the following content: see Listing 7.

Finally, we need to tell Django which URL should map to our view. We need to alter two files: the `urls.py` file in the babble application, which will map a URL to the view directly. Then we'll edit the `urls.py` file in babbleproj, to tell Django which URL to `mount` the application at. Let's look at Listing 8.

Most of this is boilerplate. The key code is in the penultimate line, where the `r'messages/$'` regular expression is mapped to the `view_messages` view function that we defined a moment ago in the `views.py` file.

That file controls only the URL configuration for the `Babble` application. We still have to tell the main project's URL file where to `mount` the application. Open up `~/babble_env/babbleproj/babbleproj/urls.py` and modify it so it looks like this: see Listing 9.

Hopefully this should now make a little more sense. You can see that before, when you uncommented, the admin lines, you 'mounted' the admin application at the `admin/` URL that you visited in the browser.

Now, as you add that last line, you should see how you are `mounting` the Babble application at `babble/`. URLs defined in Babble's urls.py file will appear based at the location specified in the project's `root` URL configuration file.

Start up the development server again, and visit *http://127.0.0.1:8000/babble/messages/*. You should see a plain text list of all the messages you created earlier in the admin interface, ordered in descending order of creation date.

Congratulations – you've created your first Django web site!

Where next?

Work through the Django tutorial (*http://docs.djangoproject.com/en/dev/intro/tutorial01/*), which will go into a lot more depth than we've had time to here. As well as complex models and referential integrity, it covers serving media such as images, javascript and CSS, more depth on the ORM programming interface, production deployment and much, much more.

There are also many third-party Django applications to explore and integrate into your own sites. *http://djangoplugables.com* is a great resource for keeping up-to-date on what's hosted on Google Code, and the weekly This Week In Django podcast (available from the iTunes store or directly from *http://thisweekindjango.com*) takes a look at new community developments each week, as well as keeping listeners abreast of changes in Django itself.

You can also join the django-users mailing list at *http://groups.google.com/group/django-users* or hop onto the `#django` IRC channel on irc.freenode.net. I look forward to seeing you there!

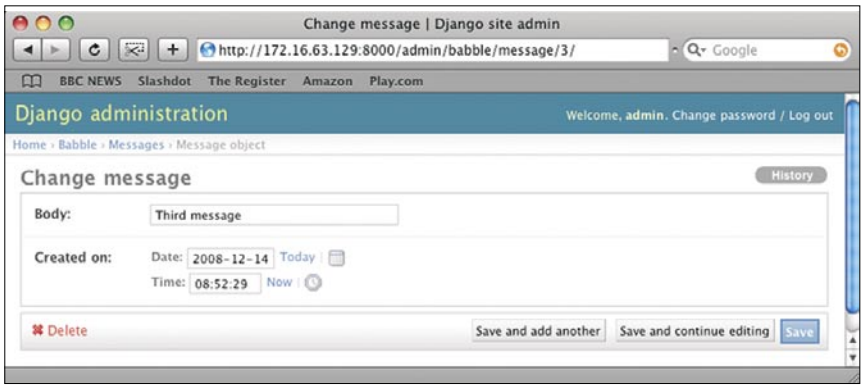


Figure 4. admin editing message

Figure 5. Success

About the Author

Dan Fairs is Director of Fez Consulting Ltd., a UK Internet services company. Graduating from the University of Bristol with a Master of Engineering degree in Computer Science in 2000, Dan has worked with many companies, ranging from small pharmaceutical software firms to retail and investment banks. Delivering solutions for the past three years with Zope and Plone, Dan has found Django to be an excellent delivery platform, well suited to the agile approach he uses for development. Dan is also a certified ScrumMaster, and can offer Scrum training alongside software consultancy and development services.

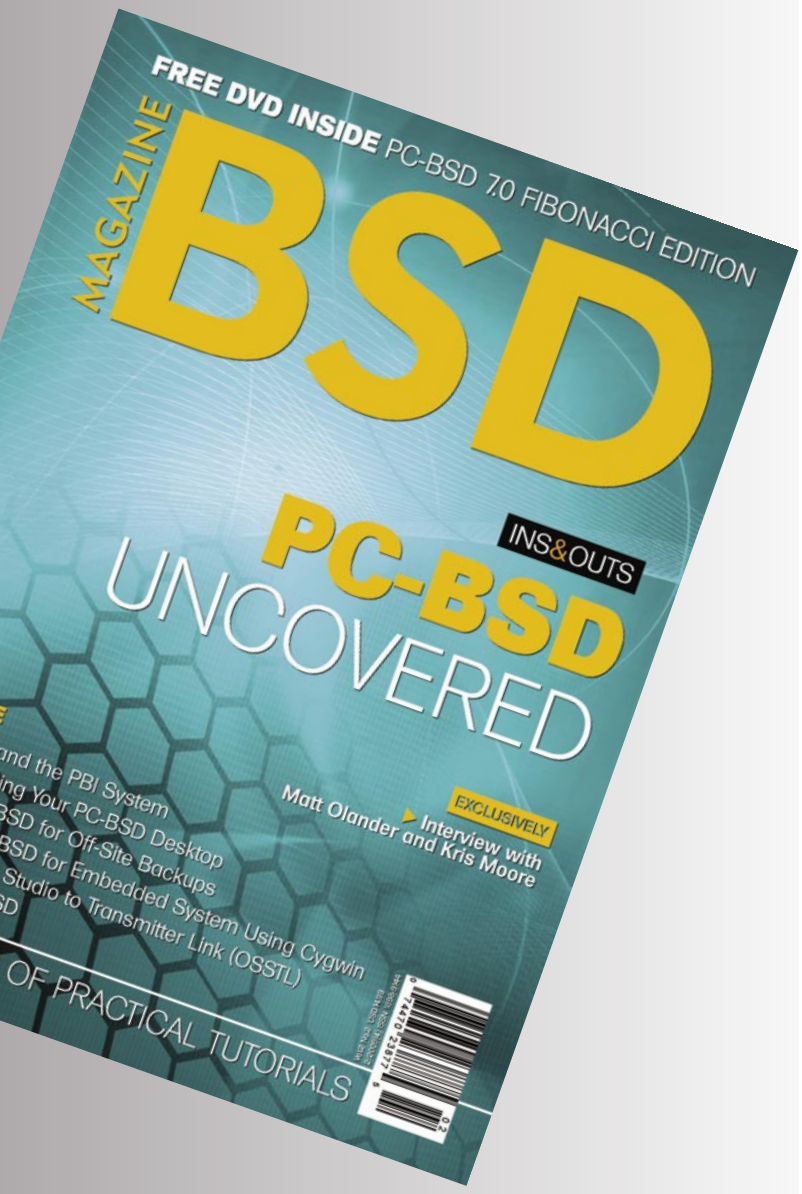
great subscription offer

SAVE $20!

Get your copy of BSD mag and save $20 of the shop prices

Why subscribe?

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



3 EASY WAYS to subscribe:

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

BSD magazine ORDER FORM

☐ **Yes**, I'd like to subscribe to BSD magazine from issue ☐ ☐ ☐ ☐ ☐ ☐
1 2 3 4 5 6

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 39.99€

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 Media LLC)

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

Open Source

Studio to Transmitter Link (OSSTL)

Jason Ellison

A local nonprofit radio station owns a studio that provides feeds for two AM radio stations and one FM radio station. The content provided for the two AM station's is mostly syndicated talk radio with a very little music content.

While the FM radio station's programming is a wide range of audio from talk radio to modern pop music. The studio sends the audio to two remote locations for transmission. The two AM station's mono audio were sent to a transmitter six miles away and the FM station's stereo audio was sent to a transmitter 15 miles away. This was accomplished using multiple microwave links in the 950MHz frequency range. These systems that are used to transmit the audio from the studio to the actual transmitter site are referred to as Studio to Transmitter Links, or STL's for short.

The studio was about to change locations and they were reconsidering the way they transported audio from the studio to the transmitter sites. The current system of using microwave STL links had some drawbacks. The microwave links required annual licensing fees for the 900MHz frequencies used by the system. The particular microwave based STL that the station used were older Marti STL-10's. Due to the age of the equipment in use, audio quality was slightly less than that of some of the competition. This is a major concern for radio stations because sound quality can have a huge impact on retaining listenership. The equipment also could not do some of the more modern tasks such as encoding the the sound into digital formats for instance, streaming to web listeners or archiving for future listeners.

Initial Discussion

I was approached about the possible use of open source software and commodity hardware to replace the current proprietary microwave based STL system. Knowing that moving audio from one location to another was completely in the realm of modern computers, I felt this would be an interesting project that on the surface seemed very doable. I had been playing with low power FM transmitters for a while and had previously considered the need for a low cost and flexible solution. Once you were in the realm of commodity hardware and open source software, although not trivial, adding functionality becomes a much easier task when compared to proprietary solutions.

Project requirements

The first step was to solidify the project requirements and get them down on paper. After a few initial discussions we were able to get the project requirements clearly defined. The requirements were as follows:

- Move CD quality (or better) audio across an Ipv4 network
- Maintain a constant system latency of less than 2 seconds (from audio input to audio output).
- Deliver the audio stream reliably.
- Provide digital versions of the audio streams for internet radio.
- Transmit the audio using no more than 1.5Mbps (T1)

Research

I scoured the web and newsgroups looking for projects that may accomplish the defined goals. I found a small amount of discussion and even fewer projects. All of the projects that I did find seemed to now be defunct. One such project is flow-stl hosted at sourceforge.net and previously used by Prometheus Radio. I managed to get into contact with Andy Gunn of Prometheus Radio. Unfortunately he informed me that the software was no longer in use and that they were currently using a proprietary software package. I also was unable to obtain a copy of the software they previously used. So it looked like I would not have the benefit of seeing how previous projects attempted this sort of thing.

For the next two months I spent my days and nights at my desk testing various software programs. Each software

package in various configurations. Listening to the audio for changes in latency... making sure that the end result sounded as good as the input. Verifying bandwidth utilization. The whole process very quickly became repetitive. I automated a few of the testing procedures, but it remained a very labor intensive process.

This system had to work over low bandwidth connections. To simulate the expected network conditions, I setup two Cisco 1720 routers with T1 WIC's in a back to back configuration. Thats fancy speak for saying I used a crossover cable to test the audio transfers across a T1 network. This allowed me to adjust network bandwidth and latency. These were configured at various times with QoS to simulate even lower bandwidth conditions, packet loss and network congestion (Figure 1).

Hardware

The first thing we choose were Audio cards. We needed something of professional quality. For the studio we choose the *M Audio Delta 44*. This card has four 1/4" TRS balanced analog inputs. For the transmitter side we went with the *M Audio Audiophile 192*, which offered two 1/4" TRS balanced analog outputs. These cards are very high quality, offering frequency responses from 20Hz to 20kHz @ 48kHz. These devices also have very little crosstalk, which means I should be able to use a stereo stream to send two mono audio streams to the AM transmitter site. More over these cards seem to be fairly well supported on open source operating systems either by the operating system itself or through a third

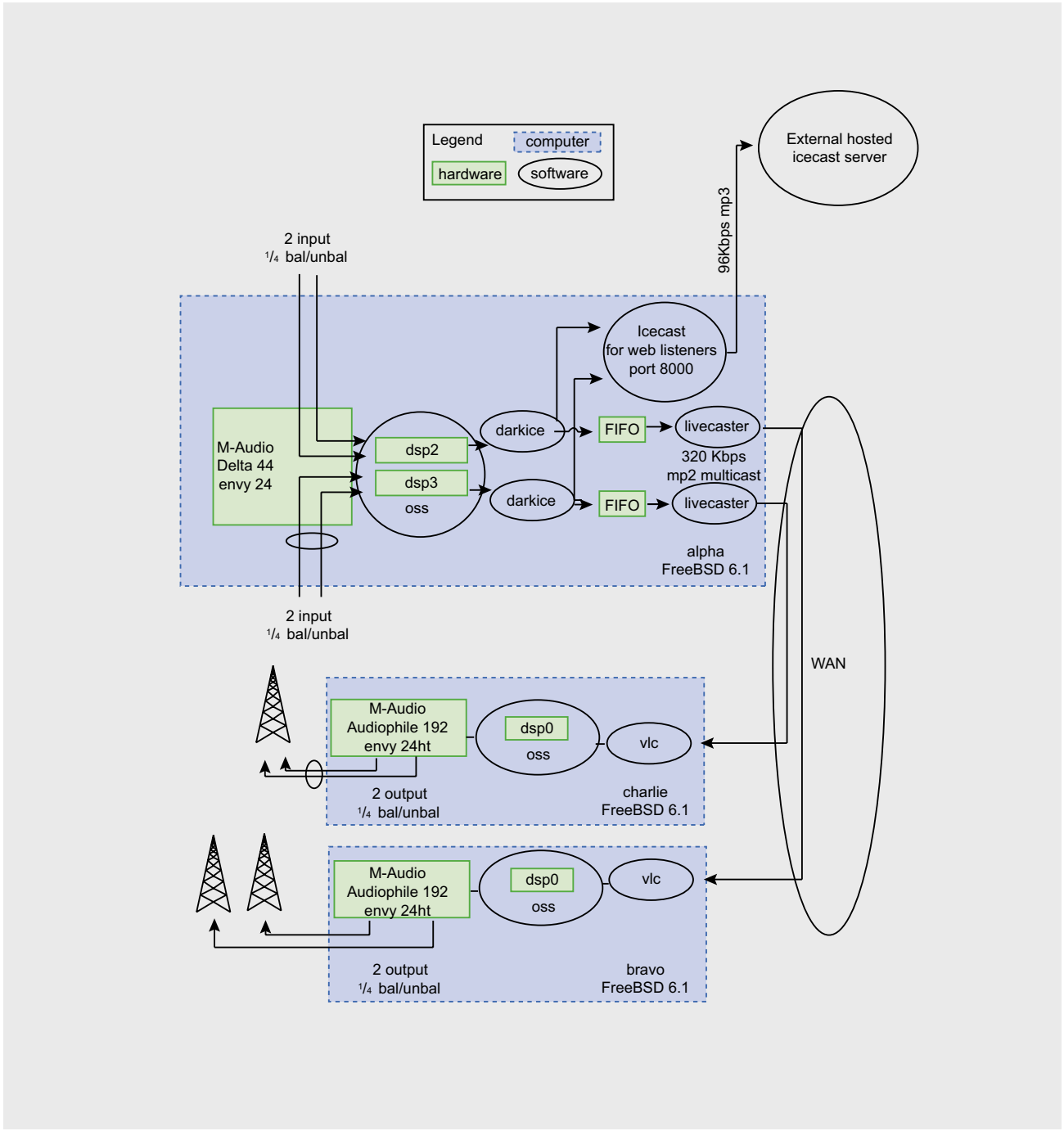


Figure 1. OSSTL Flowchart

party. At the time of the project FreeBSD stable was version 6.1 which did not support these cards. I was able to get support by using 4Front's proprietary OSS drivers.

(since beginning this project, FreeBSD appears to support the M-Audio soundcards via the open source Envy24 and Envy24HT sound drivers) Next were the computers. I was allocated three computers to build the system. The machines were whitebox OEM machines that I assembled to keep cost down. We used basic but reliable components. In the end, the specs I choose were a little overkill. But I wanted to be on the safe side.

- Studio side (hostname: alpha)
- CPU: AMD Athlon 64 Processor 3500+ (2.1GHz)
- RAM: 1GB of RAM.

In production use, top shows 498MB of memory usage and the cpu between 50% – 60% idle.

- Transmitter side (hostname: bravo and charlie)
- CPU: AMD Sempron Processor 2800+ (1.6GHz)
- RAM: 512MB of RAM.

In production use, top shows 146MB of memory usage and the cpu 99% idle.

Listing 1. Compile-Time Configuration Options

```
icecast-2.3.1

no custom configuration

lame-3.96.1

no custom configuration

twolame-0.3.7

no custom configuration

darkice-0.18

./configure --without-alsa --without-jack --without-faac -with-lame
--prefix=/usr/local --with-twolame-prefix=/usr/local

vlc-0.8.5

./configure --enable-oss --enable-twolame --disable-alsa --disable-ffmpeg \
  --disable-libmpeg2 --disable-wxwidgets --disable-skins2 \   --enable-
livedotcom -with-livedotcom-tree=/usr/local/src/live
```

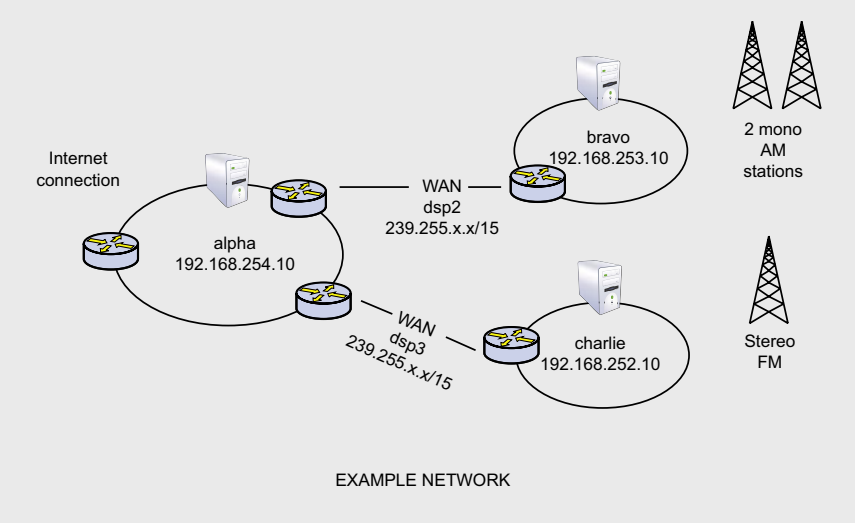


Figure 2. OSSTL Example Network

Software

For this project I spent countless hours researching many software applications. You could really get lost evaluating all of them. I kept to the projects that had been around for a while and had been well maintained. This was after all going to be used in a live radio station. I needed software that had been in the field for a while.

I settled on DarkIce for reading the analog data from the soundcard inputs. DarkIce is highly configurable and very stable. It allows for reading from oss interfaces which is exactly what I needed. For output, I do a few things...

To get the audio to the transmitter sites I use a bitrate of 320Kbps. This bitrate was chosen to avoid possible compression artifacts in the audio stream. This is because some of the programming that the station plays has already been compressed. The satellite feed for instance is already digitized at 256Kbps using an MP2 codec. satellite feeds are fairly common these days. During testing I found that when I played previously encoded files with a laptop through the system, I would get fairly severe artifacts unless I used a bitrate that was equal to or above the bitrate of what the original content was digitized at. The audio bound for the transmitter sites is encoded and written to a file which is actually a fifo.

Now that the audio is being fed into a local fifo I needed to get it out to the transmitter sites. This one took a while to figure out... I choose to go with liveCaster which uses the liveMedia library from *live555*. I found a liveCaster binary available from *live555.com*, It is free to use but does not come with the source. The LiveMedia library is the same library that VLC uses to read and write RTSP, RTP and SDP formats. liveCaster can be used to send MPEG audio data in RTP/UDP packets. It has the capability to send in either Unicast or Multicast. This application is rock solid. It does not transcode the audio data so it very fast also.

Listing 2a. Startup Scripts

/usr/local/etc/rc.d/darkice-a:

```
#!/bin/sh
# darkice and livecaster
# PROVIDE: darkice-a
# REQUIRE: DAEMON oss icecast2

. /etc/rc.subr
name="darkice-a"
stop_cmd="darkice_stop"
start_cmd="darkice_start"
pidfile=/var/run/$name

BPS=320
DSP=dsp2

darkice_start(){
  echo -e "\nstarting: $name"
  /usr/local/bin/darkice -v 5 \
  -c /usr/local/etc/dlc/$DSP/$BPS/darkice.cfg &
  echo $! > $pidfile
}
darkice_stop(){
  echo -e "\nstopping: $name"
  kill `cat $pidfile`
  rm -f $pidfile
}
darkice_restart(){
  darkice_stop;
  darkice_start;
}
rc_usage(){
  echo "Usage: $0 [start|stop]"
}
load_rc_config $name
run_rc_command "$1"
```

/usr/local/etc/rc.d/darkice-b:

```
#!/bin/sh
# darkice and livecaster
# PROVIDE: darkice-b
# REQUIRE: DAEMON oss icecast2

. /etc/rc.subr
name="darkice-b"
stop_cmd="darkice_stop"
start_cmd="darkice_start"
pidfile=/var/run/$name

BPS=320
DSP=dsp3

darkice_start(){
  echo -e "\nstarting: $name"
  /usr/local/bin/darkice -v 5 \
  -c /usr/local/etc/dlc/$DSP/$BPS/darkice.cfg &
  echo $! > $pidfile
}
darkice_stop(){
  echo -e "\nstopping: $name"
  kill `cat $pidfile`
  # 2 > /dev/null
  rm -f $pidfile
}
darkice_restart(){
  darkice_stop;
  darkice_start;
}
rc_usage(){
  echo "Usage: $0 [start|stop]"
}
load_rc_config $name
run_rc_command "$1"
```

/usr/local/etc/rc.d/livecaster-a:

```
#!/bin/sh
# livecaster
# PROVIDE: livecaster-a
# REQUIRE: DAEMON oss darkice-a

. /etc/rc.subr
name="livecaster-a"
stop_cmd="livecaster_stop"
start_cmd="livecaster_start"
pidfile=/var/run/$name

BPS=320
DSP=dsp2

livecaster_start(){
  #livecaster
  sleep 1
  echo "starting: $name"
  /usr/local/bin/lc \
  -d /usr/local/etc/dlc/$DSP/$BPS/ \
  -s /usr/local/etc/dlc/$DSP/$BPS/stream.fifo &
  echo $! > $pidfile
}
livecaster_stop(){
  echo "stopping: $name"
  kill `cat $pidfile` 2 > /dev/null
  rm -f $pidfile
}
livecaster_restart(){
  livecaster_stop
  sleep 1
  livecaster_start
}
rc_usage(){
```

Listing 2b. Startup Scripts & Various Run-Time Configuration Files

```
  echo "Usage: $0 [start|stop]"
}
load_rc_config $name
run_rc_command "$1"

/usr/local/etc/rc.d/livecaster-b:

#!/bin/sh
# livecaster
# PROVIDE: livecaster-b
# REQUIRE: DAEMON oss darkice-b

. /etc/rc.subr

name="livecaster-b"
stop_cmd="livecaster_stop"
start_cmd="livecaster_start"
pidfile=/var/run/$name

BPS=320
DSP=dsp3

livecaster_start(){
  sleep 1
  echo "starting: $name"
  /usr/local/bin/lc \
  -d /usr/local/etc/dlc/$DSP/$BPS/ \
  -s /usr/local/etc/dlc/$DSP/$BPS/stream.fifo &
  echo $! > $pidfile
}
livecaster_stop(){
  echo "stopping: $name"
  kill `cat $pidfile` 2 > /dev/null
  rm -f $pidfile
}
livecaster_restart(){
  livecaster_stop
  sleep 1
  livecaster_start
}
rc_usage(){
  echo "Usage: $0 [start|stop]"
}
load_rc_config $name
run_rc_command "$1"


bravo and charlie

/usr/local/etc/rc.d/020-vlc.sh:

#!/bin/sh
# vlc

echo -n "$0 : "
start(){
  echo "Start"
  ossmix vmix0-src Production ;
  ossmix envy24.rate 44100 ;
  ossmix envy24.ratelock on ;
  /usr/local/bin/vlc --daemon \
  --no-video \
  --audio --aout oss --aout-rate 44100 --dspdev /dev/
dsp \
  --no-oss-buggy --rt-priority --loop \
  --udp-caching 1000 \
  sdp:///usr/local/etc/livecaster/320bps
}
stop(){
  echo "Stop"
  killall vlc
  sleep 1
}
usage(){
  echo "missing parameter"
  echo "Usage: $0 [start|stop]"
}
case "$1" in
  start)
    start "$2" ;
    ;;
  stop)
    stop ;
    ;;
  *)
    usage ;
    ;;
esac



Configuration files:

/usr/local/etc/dlc/dsp3/320/darkice.cfg:

[general]
duration = 0 # duration of encoding, in seconds. 0
means forever
bufferSecs = 1 # size of internal slip buffer, in
seconds
reconnect = yes # reconnect to the server(s) if
disconnected
realtime = yes

[input]
device = /dev/dsp3 # OSS DSP soundcard device for the
audio input
sampleRate = 44100 # sample rate in Hz. try 11025,
22050 or 44100
bitsPerSample = 16 # bits per sample. Try 16
channel = 2 # channels. 1 = mono, 2 = stereo
```

Listing 2c. Run-Time Configuration Files

```
[icecast2-0]
bitrateMode = cbr
format = mp3
bitrate = 128
quality = 0.5
server = localhost
port = 8000
password = s3cr3t
mountPoint = lpfm-128.mp3
name = LPFM - Radio With A Vision
description = (C)2007 Simmon and Grundy
url = http://www.example.org/
genre = Other
public = yes

[icecast2-1]
bitrateMode = cbr
format = mp3
bitrate = 96
quality = 0.5
server = localhost
port = 8000
password = s3cr3t
mountPoint = lpfm-96.mp3
name = LPFM - Radio With A Vision
description = (C)2007 Simmon and Grundy
url = http://www.example.org/
genre = Other
public = yes

[icecast2-2]
bitrateMode = cbr
format = mp3
bitrate = 64
quality = 0.5
server = localhost
port = 8000
password = s3cr3t
mountPoint = lpfm-64.mp3
name = LPFM - Radio With A Vision
description = (C)2007 Simmon and Grundy
url = http://www.example.org/
genre = Other
public = yes

[icecast2-3]
bitrateMode = cbr
format = mp3
bitrate = 32
quality = 0.5
server = localhost
port = 8000
password = s3cr3t
mountPoint = lpfm-32.mp3
name = LPFM - Radio With A Vision
description = (C)2007 Simmon and Grundy
url = http://www.example.org/
genre = Other
public = yes

[file-0]
bitrateMode = cbr
format = mp2
bitrate = 320
quality = 0.8
fileName = /usr/local/etc/dlc/dsp3/320/stream.fifo
samplerate = 44100
highpass = 25000
lowpass = 10


/usr/local/etc/dlc/dsp2/320/darkice.cfg:

[general]
duration = 0 # duration of encoding, in seconds. 0
means forever
bufferSecs = 1 # size of internal slip buffer, in
seconds
reconnect = yes # reconnect to the server(s) if
disconnected
realtime = yes

[input]
device = /dev/dsp2 # OSS DSP soundcard device for the
audio input
sampleRate = 44100 # sample rate in Hz. try 11025,
22050 or 44100
bitsPerSample = 16 # bits per sample. Try 16
channel = 2 # channels. 1 = mono, 2 = stereo

[file-0]
bitrateMode = cbr
format = mp2
bitrate = 320
quality = 0.8
fileName = /usr/local/etc/dlc/dsp2/320/stream.fifo
samplerate = 44100
highpass = 25000
lowpass = 10

/usr/local/etc/icecast.xml:

<icecast>
<limits>
  <clients>10</clients>
  <sources>5</sources>
  <threadpool>5</threadpool>
  <queue-size>524288</queue-size>
  <client-timeout>30</client-timeout>
  <header-timeout>15</header-timeout>
  <source-timeout>10</source-timeout>
  <burst-on-connect>1</burst-on-connect>
  <burst-size>65535</burst-size>
</limits>
```

For the web listeners DarkIce encodes the audio into four different bitrates: 128Kbps, 96Kbps, 64Kbps and 32Kbps. These lower bitrates are all that are needed for delivery to a remote IceCast server. In fact the radio station is only using the 96Kbps and 32Kbps streams. The remote IceCast server is hosted on a well connected network so it can easily stream the data to a larger number of web listeners.

IceCast was implemented to stream music for internet radio listeners. The web listeners connect to an IceCast server hosted by a third party on a network with a bit more bandwidth. Originally the data was sent directly to that remote IceCast server from DarkIce. Unfortunately, we noticed during testing that a failure of the remote IceCast server can cause your local DarkIce server to crash. Adding a local IceCast server in the chain resolved these concerns.

At the actual transmitter sites I used VLC player to read the audio data from the network. VLC then outputs the sound via the M Audio Audiophile 192. And again, it uses the same liveMedia library to decode the network stream as we used to create it. This player just works. It is uses minimum resources and can be run from the command line.

A few other utilities I found invaluable were sox and festival. I used sox to generate test frequencies to check that the needed frequencies were traversing the entire system untouched by filters. Festival was used to generate human voices into wav files. I would have festival say, *1 second tone at 10KHz*... and then use sox to play a 10KHz sound for one second.

Listing 2d. IceCast Run-Time Configuration Files

```
<authentication>
  <source-password>s3cr3t</source-password>
  <relay-password>s3cr3t</relay-password>
  <admin-user>admin</admin-user>
  <admin-password>hackme</admin-password>
</authentication>
<hostname>alpha</hostname>
<listen-socket>
<port>8000</port>
</listen-socket>
<fileserve>1</fileserve>
<paths>
  <basedir>/usr/local/share/icecast</basedir>
  <logdir>/usr/local/var/log/icecast</logdir>
  <webroot>/usr/local/share/icecast/web</webroot>
  <adminroot>/usr/local/share/icecast/admin</adminroot>
  <alias source="/" dest="/status.xsl"/>
</paths>
<logging>
  <accesslog>access.log</accesslog>
  <errorlog>error.log</errorlog>
  <loglevel>3</loglevel> <!-- 4 Debug, 3 Info, 2 Warn, 1 Error -->
  <logsize>10000</logsize> <!-- Max size of a logfile -->
</logging>
<security>
  <chroot>0</chroot>
  <changeowner>
    <user>nobody</user>
    <group>nogroup</group>
  </changeowner>
</security>
</icecast>
```

Then I would play other frequencies for different periods. This was tremendously helpful in tweaking the setup (Figure 2).

Details

For installing FreeBSD 6.x OS, I followed the *installing FreeBSD* section of the FreeBSD Handbook (which is available online of course). One thing that I would have liked to have known from the beginning was the expected disk usage. In the end total disk usage was right at 2GB. The included all of the sources. The following is what disk usage looks like on alpha, which was used to build all the software.

```
alpha# df -h
Filesystem  Size   Used Avail Capacity
Mounted on
/dev/ad4s1a 989M    91M 819M  10% /
/dev/ad4s1e 989M   1.4M 909M   0% /tmp
/dev/ad4s1f  68G   1.6G 61G    3% /usr
/dev/ad4s1d 989M    72M 838M   8% /var
```

Building the software

Most of the software used in this project was compiled using original sources. This resulting system has proven to be extremely reliable. This says a lot about the quality of code. The fact that software is constantly evolving can not be denied. As the complexity of a project grows due to new features and enhancements: the chance for bugs to go unnoticed increases. Because of this it is important to noted exactly which versions of the relevant software packages I am using. Being that the software was downloaded and compiled from source, I also note which options the sources were compiled with (Listing 1).

Custom start up scripts

Custom startup scripts were created to control the various software components. I took advantage of the dependencies to control the start order of the services on the server at the studio. The desired order of start would be `oss`, `IceCast`, `DarkIce`, `liveCaster`. One `DarkIce` and one `liveCaster` process need to be started per soundcard (Listing 2).

The liveCaster GUI was used to generate the *SELF.txt* file. This file is used to configure the liveCaster

daemon. It was also used to generate the sdp file used to instruct VLC what audio to play from the multicast streams (Listing 3).

VLC's SDP files

VLC's knows which audio stream to play because of the information contained in the SDP file. The SDP file used by VLC matches the parameters found in the *SELF.TXT* file that configures liveCaster. Since we have two different streams being sent, we have two different SDP files. One of the SDP files describes the stereo FM audio stream. While the other SDP file is used descibes the two AM mono signals that are being streamed in the individual channels of the MP2 stereo signal (Listing 4).

Web interface

A simple web interface was created at all three locations that allows the operator to start and stop all daemons. It also allows you to view the current audio levels. This is very useful when trouble shooting a problem. I used PHP and bash/awk scripting to get the volume levels from the ossmix program output. With some Ajax I was able to display the audio levels using images. The result was similar to the lights you find on high end audio equipment.

Expansion

One of the things I wanted most for this project is add in multicast output directly into DarkIce. Even though liveCaster is available free, Ross Finlayson of live555 has not release the source code for the application. After posting my results to relevant mailing lists, I received a large amount of interest from people all over over the world. I believe that all portions of this setup should be open source. The availability of the source code would allow the community to make any needed future changes. Adding multicast output to DarkIce requires writing an additional *Sink* for DarkIce using the liveMedia library for encoding the data into RTP/UDP. This was something that I began work on but during testing liveCaster worked so well it became less important than some of the other issues the client had. After the project was proven stable for a month the system was put into production. Being that the system was in

Listing 3. liveCaster Run-Time Configuration Files

```
/usr/local/etc/dlc/dsp2/320/SELF.txt:

info dsp2-320bps MP2 audio
bwLimit 320
progId P:10.0.0.109:1172752654
nickname dsp2-320bps
outputMode audio
SDPdir
GroupEId 239.255.7.113 54010 {15 nokey}
tunnelState 0 {} 10100 10100 1

/usr/local/etc/dlc/dsp3/320/SELF.txt:

bwLimit 320
info dsp3-320bps MP2 audio
progId P:10.0.0.109:1172752749
SDPdir local-default
outputMode audio
nickname

 dsp3-320bps
tunnelState 0 {} 10100 10100 1
GroupEId 239.255.149.81 56910 {15 nokey}
```

Listing 4. SDP files used by VLC

```
Bravo:

/usr/local/etc/livecaster/320bps/dsp2-320bps.sdp:

v=0
o=- 1172752654 3381741506 IN IP4 10.0.0.109
s=dsp2-320bps
i=dsp2-320bps MP2 audio
b=AS:320
t=3381741506 3381743126
a=type:broadcast
a=tool:liveCaster
m=audio 54010 RTP/AVP 14
c=IN IP4 239.255.7.113/15
charlie:

/usr/local/etc/livecaster/320bps/dsp3-320bps.sdp:

v=0
o=- 1172752749 3381741714 IN IP4 10.0.0.109
s=dsp3-320bps
i=dsp3-320bps MP2 audio
b=AS:320
t=3381743230 3381744850
a=type:broadcast
a=tool:liveCaster
m=audio 56910 RTP/AVP 14
c=IN IP4 239.255.149.81/15
```

production, I wasn't real keen on making major changes.

There were some unknowns about the hardware requirements when I started this. I know now that this system could operate with less powerful hardware than what was used. Validating performance on various hardware platforms with various codecs and bitrates would be very valuable. I would especially like to validate small form factor devices. An environmentally hardened device with a watchdog timer would be ideal for the transmitter sites. With a watchdog timer on the device the system would reboot if it were to lockup... This would save a trip to the transmitter for the radio engineer.

Conclusion

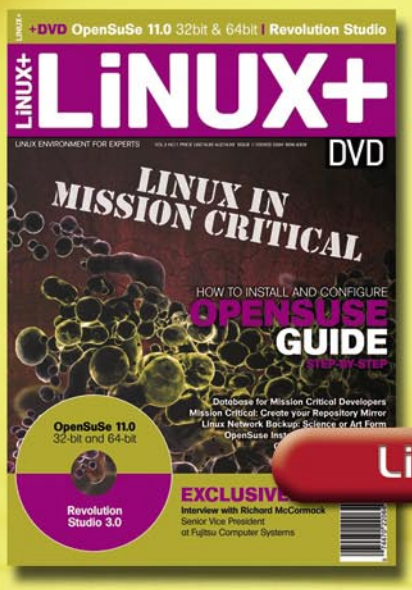
This was a very successful and exciting project. I met and exceeded all of the objectives laid out by the client. The quality of the radio stations audio broadcast has increased dramatically. After switching to this system, the radio station sounds as good as or better than any radio station on the dial. The reliability this system has increased. The ability to readily stream to web listeners on-line was an exciting feature for the station operators. This allowed them to reach a wider audience and offer the service in more convenient ways.

When I completed this project I emailed information to darkice, freebsd, and vlc mailing lists. The Community response I recieved was positive. I had people write from all over the world requesting more information. In the future I will probably start a website dedicated to OSSTL. This website will host all the documentation needed to recreate the system from scratch. It would also host a mailing list so that people could obtain help and share their experience.

References

- FreeBSD *http://www.freebsd.org/*
- DarkIce *http://darkice.tyrell.hu/*
- liveCaster and liveMedia streaming media library *http://www.live555.com/*
- VLC media player *http://www.videolan.org/vlc/*

Subscribe to BSD magazine and get 15% off on either:



Linux+

GOmotion

Flash & Flex

Hackin9

PC-BSD – Making Your Life Easier

Matt Olander

Accomplishing common tasks on PC-BSD may be executed effectively and efficiently by using built-in configuration tools and locating system settings that may increase overall usability and performance. Finding and using these tools correctly is often more difficult than accomplishing these tasks in the *normal* fashion. PC-BSD is a Unix-like operating system empowered by FreeBSD designed for the everyday desktop user. There are several configuration utilities written specifically for PC-BSD that allow the average user to execute tasks easily without any prior knowledge of PC-BSD, Unix, or FreeBSD. PC-BSD includes the KDE desktop environment as the default window manager, which brings several other useful utilities and functions as well.

Configuration Utilities

There are several custom utilities exclusive to PC-BSD that bring more power and control to the average user. Some of these utilities include; System Updater, Network Manager, WiFi Configuration Utility, Warden & Inmates, Add/Remove Programs, Display Setup Wizard, System Settings, and the CUPS Printing Manager.

System Updater

The System Updater Utility checks for System Updates and PBI Updates at startup. When multiple updates are available the user has the option of selecting which updates to install. You may also use the Configuration tab to select/deselect *Check for system updates*, it's recommended that you keep this option selected.

The same applies for the *Check for updates to installed PBIs* option, the utility compares the PBI version installed with the version available on the server and notifies you if an update is available. The updater essentially uninstalls the outdated PBI, downloads the new one, and installs it for you, and all without restarting!

You can turn off the System Updater by deselecting *Run the System Updater at Startup* and specify a custom temp directory to use instead og `/usr/local/tmp` if you need to.

Network Manager

The Network Management Utility displays all local network adapters and provides a graphical front-end for configuration. The *Advanced* network configuration tab provides quick access to changing the DNS, Hostname, Gateway, IPv6 Gateway, and PPPoE configuration.

WiFi Configurator

The Wireless Network Configuration Utility allows a user to set up individual profiles for each WiFi hotspot. The profile with the strongest signal will automagically become active when in range. The WiFi configurator will default to an encrypted network in the instance of having encrypted and unencrypted networks available.

The Warden & Inmates

The Warden is a graphical Jail Management Utility that brings operating-system level virtualization to the desktop user. Managing Jails has never been easier.

Each Jail is an independent `mini-system` that can be configured in just a couple of clicks. With the addition of The Warden's *Inmates*, things like an AMP Inmate (`apache/mysql/php`) makes deploying a jailed webserver easy and secure. This utility is a must have for web developers needing a friendly and secure way of developing and deploying web applications.

Display Setup Wizard

This wizard allows you to select your desired resolution, the video driver you wish to use, advanced options such as entering a specific monitor refresh rate as well as enabling dual-monitor support out of the box.

CUPS Printing Management

CUPS is a web-based printing management utility designed by Apple, Inc. for Unix.

A graphical front-end to CUPS is currently in the works and should be included in the next update to PC-BSD.

The following guide may be used when setting up a printer in CUPS until a front-end solution is added.

- Step One: Open up your favorite web browser and type *localhost:631* into the address bar.
- Step Two: Click the Administration tab and select Add Printer from the Printers section.
- Step Three: Enter a name, location, and description for your printer. You may type in whatever you like.
- Step Four: Select an applicable device and click continue. *HP Printer* is a common selection.
- Step Five: Type the printer's URI and click continue. For example; `socket://192.168.1.7:9100`
- Step Six: Select the Make/Manufacturer for your printer and click continue, or provide a PPD file and click *Add Printer*.
- Step Seven: Select a Model/Driver for your printer or provide a PPD file and click *Add Printer*.
  If you are prompted for a username and password, enter the username and password you use to login to PC-BSD.
- Step Eight: Click the 'Printers' tab and select *Print Test Page*.
  If your test page prints succesfully click *Set As Default*.
- Step Nine: Close CUPS and enjoy using your printer.

System Settings

The System Settings utility provides access to configure PC-BSD's appearance, user and accessibility options, network & connectivity, system administration such as display settings, sound, date & time, and a lot more. Think of it as your nexus to all system settings. Often a user's experience can improve drastically upon mastering the art of system setting modification.

Look & Feel

- Appearance – In Appearance you will find several sub sections to modify the style, color scheme, icons, fonts, and window borders. This section can drastically change the way your desktop looks, so modifying it will give your desktop that personal touch.
- Desktop – This section provides the option to enable desktop effects, configure multiple desktops, and set screensaver settings.
- Notifications – This section provides options for application notification settings.
- Window Behavior – The Window Behavior section has several tabs that allow modification to how windows are displayed. A noteworthy option is on the *Moving* tab, selecting *Centered* from the Placement drop-down menu opens all application windows in the center of your screen.

Personal

- About Me – Do not use this section to change your username or password. You may, however, change the home directory, desktop, and autostart paths in if needed.
- Accessibility – This section provides accessibility options for those who need them.
- Default Applications – Specify which applications are defaulted for your E-Mail client, text editor, Instant Messenger, Terminal emulator, and Web Browser.
- Regional & Language – You may change the language, currency, time & date, keyboard layout and other regional specific items.

Network & Connectivity

- Firewall – The firewall can be enabled or disabled from this section.
- Network Settings – Configure Proxy settings and connection preferences.
- Sharing – Specify the username and password to Windows shares on the network.

Computer Administration

- Add/Remove Software – Installing and Uninstalling applications has never been easier. This utility allows easy uninstallation of PBIs and System Components.
- Date & Time – Easily change the date, time, and timezone.
- Display – Resize and rotate your display. A useful section when connecting to devices like projectors.
- Font Installer – Install, manage, and preview fonts.
- Keyboard & Mouse – Change your mouse speed and theme, modify keyboard repeat settings, and add keyboard shortcuts.
- Online Update Manager – This will open the System Updater Utility.
- Password & User Account – Change your name, password, administrator password, or add accounts to your computer.
- Services Manager – This will display the services setup on PC-BSD. You can enable or disable them at will.kbai
- Sound – This section provides information on your sound card.
- System Manager – This will provide system specific information such as the version of PC-BSD you are currently running. This section also displays information of CPU Type, System Memory, and has the option to generate a diagnostic sheet for use by technicians for troubleshooting purposes.

Advanced System Settings

When default system settings are not quite enough. Advanced System Settings is located under the tab labeled *Advanced* next to the *General* tab found in System Settings. This section provides access to settings power-users may like to change, such as services, sessions, task scheduling, etc.

Advanced User Settings

- Audio CDS – View and modify various settings related to listening to audio CDs.
- Autostart – You may add, modify, or remove programs and scripts set to run at system startup.
- CDDB Retrieval – Retrieves track information on audio CDs inserted into your computer.
- Digital Camera – Add and manage digital cameras.
- File Associations – Application specific file type associations.
- Input Actions – View, add, or modify hotkeys and gestures.
- KDE Wallet – Manage stored passwords with one global password.
- Services Manager – Manage KDE On-Demand and Startup services.
- Session Manager – Configure the session manager and logout settings with ease. Specify whether you'd like to login with the previous session, a saved session, or an empty session.
- Solid – This section is a placeholder for the Power Management Backend, Network Management Backend, and Bluetooth Management Backend.

System

- Login Manager – Configure the login manager to look and feel the way you want it to.
- Task Scheduler – Display, add, modify and remove Personal and System scheduled tasks.

Becoming Empowered

The power of PC-BSD is only limited to one's knowledge of system utilities and functions, configuration options and the ability to apply this knowledge to customizing and using the desktop. With the right tools and knowledge, the average user can transform the factory default install to something straight out of science fiction with ease. For more help, tips, and tricks, join the active and informed PC-BSD community on *http://www.pcbsd.org*, the `#pcbsd` IRC channel on FreeNode. You may also follow PC-BSD on Twitter at: `http://www.twitter.com/pcbsd` for Micro-Updates on the going-ons of PC-BSD.

Interview with PC-BSD

To celebrate this issue of the magazine fully dedicated to PC-BSD, I had the opportunity to do a quick question and answer session with Kris Moore and Matt Olander.

We discussed the use of PC-BSD in virtualized environments, the system update process, the help iXsystems can provide with drivers, the link with the FreeBSD codebase, *The Warden*, and more.

Could you introduce yourself?

Kris Moore: I am the founder of the PC-BSD project. I live in eastern Tennessee with my wife and 3 children. I first got into FreeBSD in the mid nineties, while working at a small ISP. Several years later I decided that FreeBSD was great on the server, but it was time for an easy to use desktop as well, and PC-BSD was born.

Matt Olander: I am the Chief Technology Officer at iXsystems. Previous to working here, I worked at BSDi, the commercial entity responsible for BSD/OS. My background is in database and system administration where I became a huge fan and supporter of the FreeBSD operating system due to its performance, reliability, and stability.

How does the system update process work?

Kris Moore: The system updater will download updates, which may or may not contain updates to the base world/kernel. If it does, the update is applied during the next system startup, instead of during a running X session, since that may tend to get a bit messy.

Do you use PBI packages for the system too?

Kris Moore: We do release system updates via PBI as well, for all major point releases.

What happens if the update process gets interrupted?

Kris Moore: Depends upon which stage the update is in of course. If its during the download/extract phase, it'll just need to be re-downloaded/extracted to install. If the system gets rebooted during the actual upgrade process, it'll kick-off again at the next bootup.

Have you ever had the desire (or necessity) to modify parts of the FreeBSD system?

Kris Moore: Occasionally we have thrown little bits into the system outside of FreeBSD vanilla. Such as a patch to make Flash9 work properly, updated driver etc. But we try to stick as close to base as possible. Any patches we put in are ones which will be appearing in FreeBSD soon anyway, and we sometimes just include them a bit early, while waiting for them to MFC.

How is iXsystems helping with closed-source and/or missing drivers?

Matt Olander: We leverage whatever buying power we have with vendors as a server builder and integrator to persuade them to consider supporting FreeBSD and therefore PC-BSD. Being in the middle of San Jose, California, also known as the Silicon Valley, certainly has some advantages. Many large vendors are miles away from our headquarters. This makes getting together to discuss open source easier. For some vendors, initial discussions are the beginning of heading down a path of learning about open source in general. It's difficult to ask for driver support for an open source operating system when many of the management personnel have not heard about the open source movement to begin with.

If they are aware of open source, many times it is just a matter of getting the right person on the phone. Some of these organizations are large and it may be difficult to find the person that can

make the decision internally to support an open source operating system or provide the documentation so that somebody else can support their device. This seems to be getting easier as the word on open source spreads farther.

Do you have any plan to provide a version of Skype for BSD?

Kris Moore: We do offer a PBI which installs the Linux version of Skype, that seems to work pretty well. However, at the moment I haven't heard from the Skype folks if we can expect a BSD native solution in the near future. Users should send them e-mail asking about it, and maybe they'll think about releasing a client for us :)

Considering the type of user base of your project, what type of feedback do you receive?

Kris Moore: Normally we see a lot of more non-technical questions, just from users who want to do common tasks, like video/network setup, etc.

Is the DVD iso aLive? :)

Kris Moore: Not at the moment. We do want to release a live DVD, but we are waiting on some things to be fixed up beforehand, to improve performance and make it really usable :)

KDE or not KDE, that is the problem. Why did you choose KDE?

Kris Moore: We develop all of our GUI tools in the QT toolkit, so that makes KDE a perfect fit naturally. However, I also feel that KDE provides the best overall user experience for non-technical users, who wish to jump into the Open-Source arena with little or no experience.

How do you manage the risks involved in having GUIs running with high privileges to manage the system?

Kris Moore: Nothing special really. The X/KDE sessions run under user permissions, and the only time a user will need to enter a root password is when wanting to modify a system configuration, or install new software. We do include sudo, but it isn't used much by default.

After a default installation, what type of firewalling setup does the system provide?

Kris Moore: We use PF by default, but ipfw is available for users who prefer to work with it. By default almost everything is blocked, except a few netbios ports which allow us to locate samba shares.

Somewhere in the manual I read about a tool called PF Generator, but I wasn't able to find any detail... is it a secret project?

Kris Moore: No secret! That's actually probably referring to our Firewall GUI, which is a simple front-end to PF, allowing users to start/stop the firewall, and add exceptions for specified NICS very easily. Its not designed to be a end-all PF configuration tool, we offer PBIs such as fwbuilder for that kind of advanced usage.

How does PC-BSD work as guest or host with various virtualization solutions?

Kris Moore: We offer PC-BSD pre-installed in vmware for users to try, and I know it runs just fine in QEMU, after using it here for testing. It may work as a guest elsewhere, but I haven't done any testing personally.

As for a host, right now we offer a QEMU PBI, which includes the aQEMU GUI interface, and also supports kernel acceleration.

Do you have any advice for people that want to use it in a virtualized environment?

Kris Moore: The biggest thing would be to make sure you disable all/any graphical effects in KDE, since a virtualized environment will most likely be using vesa mode, and may feel more sluggesh with them enabled. 256-512MB of RAM or more is always a plus as well.

What is The Warden?

Kris Moore: The Warden is a new tool, specific to PC-BSD. It offers a GUI front-end to the FreeBSD *jails* system, which allows you to create/manage/delete virtual systems on your desktop. It comes in very handy for running services, such as apache, sql, php or others, which you may not want running on your system without the security that a jail provides.

The Warden also supports backing up and restoring jails, which makes it effortless to move a jail from one host to another. This is very useful when you migrate to a more powerful machine, or simply want to *clone* jails to different systems.

How does PC BSD differ from Desktop BSD?

Kris Moore: PC-BSD is similar to DesktopBSD in the sense that we both offer a GUI installer, and a pre-built desktop environment, however we differ in areas of package management, with us offering the PBI system, and DesktopBSD providing a front-end to FreeBSD ports. Our PBI system is designed take all the guesswork out of package management, and make it easy for users coming from a Windows or Mac environment to immediately feel right at home on PC-BSD.

Does imitation make an OS stronger?

Kris Moore: Only when you attempt to improve. Imitation itself is nice, but often what makes a OS better is taking an idea, and trying to improve upon it to make it easier and more accessible for the public.

Should PC-BSD attempt to replicate the Windows user experience better?

Kris Moore: In the sense, that somebody who isn't a hard-core computer user, feels comfortable enough to sit down and use the system, yes. That doesn't mean that we have to replicate the Windows user experience completely, but it does mean that usability is king. With PC-BSD we strive to make the system as easy and usable as possible, so that we can attract a much broader user-base, not just technical open-source enthusiasts, but the other 99% of the population as well, who just want a system that works.

Is innovation and new metaphors for interacting requireed to establish a strong user base, and differentiate the product, ala MAC OSX?

Kris Moore: Not necessarily. While new innovations may be novel at first, long-term usability will be the key to getting and keeping a user base. We don't need to re-invent the wheel for every computing metaphor to be successful, but instead need to offer a system which works as well, or is easier to use than the competition to make it popular. Then we can also tout the other benefits of BSD, such as better speed & security, but these by themselves may not be worth a lot to end-users if the system isn't friendly enough to use out of box.

Thank you for your time!

Green Eggs & BSD...

Mikel King

BSD is here BSD is there, BSD is everywhere. Well not exactly, but there certainly is a proliferation of BSD throughout the Internet. The problem is and quite honestly has been quantifying the impact of BSD on the corporate LAN. Often times I read a messages from a lowly junior admin, on the mailing lists, who wants to use a BSD for some project. Only he's not sure how to get management to buy in. Worse management wants some hard core numbers to justify the installation and because it so difficult gleaning truly useful statistics from the internet these projects never see the light of day.

Sure there are projects like BSDStats and of course solutions like Netcraft to capture some statistical data however this is not exactly the sort of metric a CIO is looking for when he's asked, *Could you, would you run BSD outside the box, would you, could you run BSD with a fox?* Invariably they end up sticking with a familiar old smelly shoe of a solution based on some huge corporate entity. Sure it has holes and doesn't quite fit anymore but it's what you know. One that is clearly out of touch with the stability many CIO's want on their front lines.

Of course should a CIO answer, *Hey, you know running BSD is not bad, and I will run BSD where I can. I do like BSD Sam-I-am!* Ultimately the tasks BSD is often relegated to are web serving, or DNS and in some rare cases you may find it operating as the corporations' mail server. It is the rare company that will roll out an entire server infrastructure on BSD. Honestly there are too few like Yahoo!

Rarer still is to find BSD as the desktop OS deployed throughout an organization. The proliferation of Windows, Linuxes and of course Mac OS X on the desktop it is often not even considered as a desktop OS. Which is truly disappointing to say the very least. Though it is worth noting that Mac OS X is and has been based on a BSD since it's inception. Originally OpenBSD and currently FreeBSD. While she may not be the prettiest Bella at the party, one thing is for sure if only given the chance man can she dance!

Thus the deeper question is why is it that many sites choose not to report their BSD usages? Is it that they are afraid for security reasons? Do they fear being labeled a maverick? Or is it perhaps that they do not even know there is a need to report their usage? It's not like there's a DHD (*Dial Home Device* for you none Stargate fans) built into the various BSD installers which captures and reports dmesg statistics automatically.

While there are several volunteer solutions, they are an opt-in kind of solution. Perhaps we should consider changing this to an opt-out kind of system? Install the BSDStats reporting system by default to facilitate so fact gathering. I know many will find this offensive to even contemplate however were this simply yet another ncurses pop up (as in the case of the FreeBSD installer. A question like *Would you like to participate in BSD statistical reporting* yes/no.

I think many would hit yes. Especially if it is explained that the information gathered was for usage statistics. If this information were properly collated then it would help far more than the advocacy types promote BSD in general. It would help everyone in the BSD community understand where, how and which the BSDs are being used. The developers would be able to see what kinds of hardware configurations are predominately being used. The BSD Certification Group would be better able to tailor exams content based on actual usage ratios. The various BSD foundations would be able to utilize such metrics in fundraising efforts as well as negotiating with hardware vendors for better support.

All of this brings us back to the crux of the problem which is how does this help advocate the deployment of more BSD. How can we help our brother and sister admins spread the BSD? We as IT Directors, Managers, and CIO's have to step out from behind the curtain. We have to come out of the UNIX closet and stop hiding behind the LINUX moniker. We have to stop settling status quo.

We know that BSD has some of the best operating system performances not to mention the best stability hands down. We all know that we can repurpose an old PC into a dhcp, DNS, firewall, even intranet web server. All of this and more can be handily accomplished with any BSD. Isn't it better to take that old non ROHS compliant PC and put it into service as something useful rather than relegating it to the landfill?

Don't throw your old hardware away, running BSD is green, I say. So could you, would you run BSD on a boat, would you, could run BSD with a goat?

So what do you say? *Of course you may!!!*

Let's face it overall reporting your usages accurately will do nothing but help BSD grow. Then you too can proudly exclaim, that you *will run BSD on a train, but maybe not in the rain.*

In the upcoming issue of BSD magazine...

The complete guide to FreeBSD!

Don't miss your chance to explore BSD world!

Coming up in June 2009

GREENSERVERS

Introducing the iX-Earth Series

iXsystems leverages advanced technology and system design expertise to minimize power consumption in all of our servers, making us a leader in power saving technology. Helping companies minimize their carbon footprint is a primary server design objective that results in a significant reduction in the Total Cost of Ownership (TCO) of our systems, through savings on energy and cooling costs.

Given this set of initiatives, iXsystems is proud to introduce the iX-Earth Series line of environmentally friendly servers. The iX-Earth server line features power-efficient 2.5" hard disk drives, low-power Intel® CPU's, Low-Voltage Memory, as well as the world's first 1U "silver level" power supply with 85%+ power efficiency. The 2.5" SAS and/or SATA hard drives featured in the iX-Earth server configurations can reduce power consumption by 40% over standard 3.5" hard drives, and Solid State Drives (SSD) equate to a power savings of up to 95% over standard 3.5" spinning drives.

Improved power efficiency does not mean compromised performance!

The iX-Earth series is optimized for high performance applications. The iX-1282 1U configuration within the iX-Earth series features up to 8 cores, up to 64GB of low voltage RAM, and 8 hot-swap 2.5" SAS/SATA. The iX-2216 2U server provides up to 8 cores, up to 128GB of low voltage RAM, and 16 hot swap 2.5" SAS/SATA hard drive bays in an efficient 2U form factor designed for the most frequently encountered applications. The chassis is also equipped with a redundant 900W high-efficiency (85%+) redundant power supply for outstanding power savings, and fault-tolerance.

Both the iX-1282/iX-1282R and the iX-2216 are compatible with Intel® Solid State Drives (SSD) for increased performance and even lower power consumption. Unlike traditional hard disk drives, Intel® Solid State Drives have no moving parts, resulting in a quiet, cool storage solution that also offers significantly higher performance than traditional server drives.

The iX-Earth series is an excellent choice for HPCs, server farms, and other datacenters where space, cost, energy-efficiency, and density are high priorities. By reducing your server(s) power consumption, you will decrease your total cost of ownership (TCO) –without sacrificing performance – all while decreasing environmental impact.

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 Green Server class contact iXsystems at (408)943-4100 or visit our website at http://www.ixsystems.com/green 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.

iXsystems
Enterprise Servers for Open Source

© http://www.sxc.hu/br0

IX-1282

- 1U Form Factor with 8 Hot-swap SAS/SATA 2.5" Drive Bays
- Dual Intel® 64-bit Xeon® Quad-Core or Dual-Core, with 1600/1333/1066 MHz FSB
- Intel® 5400 (Seaburg) Chipset - 5400/5300/5200/5100 Series Processors
- Up to 64GB DDR2 800/667/533 SDRAM Fully Buffered DIMM (FB-DIMM) 1.8V or 1.5V supported
- Left Slot (Full-height / Full-length): 2 (x8) PCI-Express (Gen 2.0)
- Right Slot (Low-Profile w/ 5.5" length): 1 (x8) PCI-Express (Gen 2.0)
- Intel® 82575EB Dual-port Gigabit Ethernet Controller w/ IOAT
- ATI ES1000 Graphics with 32MB video memory
- Remote Management Card, IPMI 2.0 (optional)
- Slim DVD
- 560W 85%+ High-efficiency Power Supply or
- (Optional) 650W Redundant 85%+ High-efficiency Power Supply (IX-1282R)

IX-2216

- 2U Form Factor with 16 Hot-Swap SAS/SATA 2.5" Drive Bays
- Dual Intel® 64-bit Xeon® Quad-Core or Dual-Core, with 1600/1333/1066 MHz FSB
- Intel® 5400 (Seaburg) Chipset - 5400/5300/5200/5100 Series Processors
- Up to 128GB DDR2 800/667/533 SDRAM Fully Buffered DIMM (FB-DIMM) 1.8V or 1.5V supported
- 3 (x8) PCI-Express (2 w/ Gen 2.0), 1 (x4) PCI-Express (Using x8 slot),2x 64-bit 133MHz PCI-X
- Intel® 82575EB Dual-port GigabitEthernet Controller w/ IOAT
- ATI ES1000 Graphics with 32 MB video memory
- Remote Management Card, IPMI 2.0 (optional)
- 5.25" Drive Bay
- Slim DVD
- 900W 85%+ High-Efficiency Redundant (1+1) Power Supply