MAGAZINE

BSD

FOR NOVICE AND ADVANCED USERS

EMBEDDED BSD

EXCLUSIVELY

▶ MAHESHABSD

INSIDE

VOL.3 NO. 5
ISSUE 5/2010(11)
1ISSN 898-9144

(PC-BSD Version 8.0 Hubble Edition Screenshot)

Professional Enterprise and Desktop Support for Your Peace of Mind

From optimizing your small office set-up to guidance on large-scale deployments, the iXsystems team can ensure you get the most from your PC-BSD and FreeBSD systems

iXsystems understands the need for both casual and professional users to have peace of mind with their information technology operations. Our professional services staff aims to provide just that. Whether you are running a single desktop or have an HPC, datacenter, or server farm, the experienced technicians at iXsystems can assist you with all aspects of the FreeBSD and PC-BSD operating systems.

Industry Leading Technical Expertise - The iXsystems Professional Services team is comprised of long-time FreeBSD developers, administrators, and project committers. From development to consultation to emergency problem solving, our team will put their many years of experience to work for you. With support staff stationed in North America, Europe and Asia, you can rest assured your operations are in good hands round the clock and around the world.

Large Rollouts - The Professional Services Team provides installation support for large networks. Let our technicians put their expertise to work for you to determine your operational needs and provide the critical support your system administrators need for rollouts and system migrations.

Escalation Management - When the iXsystems Professional Services Team encounters a confirmed bug, we can escalate the bug to the FreeBSD engineering team. We can also work with The FreeBSD Project to create and submit patches to the FreeBSD community for possible inclusion in the latest release.

Custom Development and Consultation Services - iXsystems employs and partners with some of the most brilliant minds in the FreeBSD Community to offer custom development and advanced level FreeBSD consulting solutions. Our Account Management Service Professionals will work with you to coordinate and develop software solutions specific to your business operations. iXsystems offers kernel tuning and system optimization, device driver creation, kernel, userland, and embedded systems development, and a host of other services that allow your company to fully utilize the FreeBSD and PC-BSD platforms.

For more information contact iXsystems at **+1-800-820-BSDi** or visit our website at **http://www.iXsystems.com/bsdsupport** and fill out the inquiry form. We will pair you up with an Account Management Service Professional that can assess your needs and create a custom FreeBSD support plan for your organization!

1 2 3 4

Installation and Configuration

PC-BSD Desktop Support means you will have all the assistance you need getting your system up, running, and configured for optimal performance.

Troubleshooting Assistance

In the event something does not work properly for you, our expert technicians will walk you through the troubleshooting process to determine the cause of your problem and provide you with a solution.

Knowledge Database

With your Desktop Support package, you will gain access to our Knowledge Database. The Knowledge Database contains problems and solutions associated with previous support inquiries, saving valuable time when issues arise.

PBI Creation and Application Installation

Our Professional Services Team can create custom PBIs (push button installers) for your PC-BSD system. These self-contained applications eliminate the problem of shared dependencies and provide the user with point-and-click program installation and removal.

Accelerated Support Solutions

From desktop support to Professional Enterprise Service Level packages, the iXsystems Professional Services Team can support all of your FreeBSD and PC-BSD related needs.

Desktop support packages include 8x5 support with same day/next day prioritized responses. Professional Enterprise Service Level packages are available in both 8x5 and 24x7 models. Pricing is determined by hourly blocks or on a per unit basis.

- ✔ **Increased Uptime**
- ✔ **Improved Productivity**
- ✔ **Better Security**

iXsystems™

Dear Readers!

Dear Readers!

I am happy to introduce you our May issue in new format!

We thought few words about what should you know before reading article and few words about what will you learn after reading the article will be helpful for you.
I hope you will like it and we look forward to your feedback.
I want to thank all authors for contributing to this issue, you did really great job!

We want to remind you about answering a short questionnaire concerning our magazine. We also look forward to your questions on BSD topic.
This will certainly help us to improve our magazine and make it more interesting than ever before!

Thank you and enjoy your reading!

Olga Kartseva
Editor in Chief

MAGAZINE BSD

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

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

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

Art Director:
Agnieszka Marchocka

DTP:
Ireneusz Pogroszewski

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

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

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

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

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

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

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

The editors use automatic DTP system AUPOS

Mathematical formulas created by Design Science MathType™.

what's new

get started

how-to's

let's talk

A Live CD Project

From The Lake Mansarovar

Lake Manasarovar in the Himalayan Mountains is one of the highest fresh-water lakes in the world. One who drinks its water will go to Swarga, which is the Lord Shiva's abode.

MaheshaBSD is the name for a Live CD project. Why Mahesha? What does it mean? Mahesha is one of the 1008 names of Lord Shiva – Supreme God of the universe who stands above all gods. This name was chosen because Shiva's weapon is the same as the FreeBSD's one – the trident. There is yet another important correlation – supremacy of the BSD code, which (as many IT professionals believe) stands supreme above all operating systems. The connection of Lord Shiva and BSD is therefore logical.

MaheshaBSD is a homemade Live CD FreeBSD 8.0 (i386) – it is free, but contrary to other Live CD's in the BSD world, it can play Adobe Flash video.

Some licenses prohibit redistribution of software, therefore you must download Adobe Flash (for Linux) separately and save it (gzipped) into your `/root` or `/home/guest` directory in the environment of this Live CD. After your download is finished, run a little script *flash* that will do the trick for you – it will gunzip and install Adobe Flash after you download the plugin into the above directories.

MaheshaBSD can play mantras, as it is fully *audio equipped*, and users can use it for a number of purposes, too – to boot anywhere off the CD, watch youtube videos, be anonymous, share files with other computers, or play chess.

Looking Into The MaheshaBSD's (X) Window

MaheshaBSD has, too, educational information for FreeBSD beginners – they can look into scripts and learn how to work with this Live CD and with FreeBSD, watch films with MPlayer or watch youtube video instructions about installation of FreeBSD, be anonymous (TOR), recover their lost partitions (TestDisk), scan NTFS partitions for viruses (Clamav Antivirus), or undelete files (PhotoRec). To copy data to a NTFS partition, mount it with the following command:

```
ntfs-3g /dev/ad0s1 /mnt
```

Typing *startx* from the console after you log in (pass: root) starts X Window (IceWM with Hindu and BSD icons on the desktop). The default video driver is *vesa*, but you may also use the nvidia driver by running *startxnv*, or ati video driver (startxati).

MaheshaBSD is Qemu and VMware friendly. To start X Window, run *startxcirrus* (Qemu) or *startxvmware* (VMware).

On one of my notebooks the vesa driver did not appear to work. In such a case (and similar ones), run the command:

```
Xorg -configure
```

Then copy the newly generated `/root/xorg.conf.new` into your `/etc/X11` directory (rename `xorg.conf.new` to `xorg.conf`) and add the following line into your `/etc/X11/xorg.conf` (into the `ServerLayout` Section – needed only for mouse to work):

```
Option "AllowEmptyInput" "off"
```

MaheshaBSD is modular and intended to be customized on the fly. When it boots, you will log in to the MaheshaBSD's minimal MFS (root) environment (memory file system), which you may always expand by typing the `opencd` command. You may, too, go back with the `goback` script – return to MFS, and then open (mount)

this Live CD (USB/DVD) again, but with a different command – `opendvd`, for example, which will mount the usrdvd.uzip file you may probably prepare later.

Uzip is compression similar to cloop or squash in Linux. In FreeBSD, you can use mkuzip to compress files the similar way like in Linux. After the user mounts the (mkuzipped) image file (virtual file that may contain anything), it is then uncompressed on the fly and can be used as any directory mounted in Unix with only exception that it is mostly read/only.

How To Prepare Your Own Uzip File?

First, you must make an image from a directory (`/usr` in our case). In FreeBSD (and Unix), makefs will do the trick. The command is as follows:

```
makefs -t ffs /mnt3/usr.img /usr
```

The above command will copy the /usr directory into the file `usr.img`.

Then you need to compress it with mkuzip:

```
mkuzip -o /mnt3/mahesha/usr.uzip /mnt3/usr.img
```

`/mnt3/mahesha/usr.uzip` – is the place where (target destination) the uzip file will be created.

`/mnt3/usr.img` – is the image file – (`uncompressed`) source for the subsequent uzip compression that will be made of the same thing in `/mnt3/mahesha/usr.uzip`.

To mount such a compressed file in FreeBSD, you must first load the module `geom_uzip` into the kernel: kldload `geom_uzip`

then use mdconfig: `mdconfig -a -t vnode -f /mnt3/mahesha/usr.uzip`

then mount the compressed file: `mount -o ro /dev/md0.uzip /usr`

You do not need to worry about knowing which `/dev/md*` device to mount – as soon as you run the mdconfig command, the system will inform you which md* device was associated with the `usr.uzip` file.

FreeBSD's mdconfig is the command used (among other things) for mounting image files. It can mount ISO images, images with FAT32/NTFS file systems, etc.

In MaheshaBSD, you can expand the CD with `openmincd`, which is intended for minimal (min) memory resources. The MaheshaBSD's minimal requirements are at least 139 MB of RAM. No hard disk is required.

Anonymous surfing is established via TOR and polipo (proxy server). Users just need to click on the icon of the Dillo browser (in X), but before they must execute the TOR and polipo software. They only need to type `anon` (in the root account) and then start Dillo. If users are in the guest account, they must start polipo as root.

Sharing Mahesha

The Lord Shiva's Live CD has, too, VNC Server – tightvnc, with the assistance of which you will log in to desktops of other computers. To connect to the desktop of MaheshaBSD, type `vncserver` either in the root or guest account (on the computer that will act as a server). FTP Server does not miss here either. A very secure FTP server (vsftpd) is packaged into the distribution. You will start it by running the following command (as `root`): `/usr/local/libexec/vsftpd`

Its configuration file is customized, thus you can both upload and download files if you have MaheshaBSD running on two computers. This may prove very helpful while doing recovery of notebooks (with not working CD-ROM drives), etc.

MaheshaBSD supports text to sound conversion (espeak), so you may listen to it by typing `speaktips`, for example. If the sound driver does not work, it is the matter of the drivers loaded into the kernel – they must be either unloaded (with kldunload), or changed.

MaheshaBSD was tested on several computers and it may happen that something will not work. To control sound volume in the text console, users should use aumix. Another sound mixer, too, has its icon (gmixer) on the IceWM desktop.

If you download a static version of Skype for Linux (Skype is again something that cannot be redistributed), you will most probably be able to call your friends, as Linux emulation is activated in this thing.

To put MaheshaBSD on a USB stick, just type *tips* and learn what to do. The system is downloadable from the following URL: *http://www.freebsd.nfo.sk/maheshaeng.htm*

The updated documentation of the distro is available only at the above website.

I want to thank the *http://www.rootbsd.net* team for allowing me to distribute this thing.

JURAJ SIPOS

Juraj lives in Slovakia and works in a library in an educational institute (school of psychology). Some time in the past he was fortunate enough to travel around the world and spend a bit of time in India and Australia. Juraj's hobbies are computers, mostly Unix and also spirituality. He has also translated several books from English, for example - Zen Flesh, Zen Bones by Paul Reps. He started with FreeBSD in 1997. He wrote the Xmodmap Howto „http://tldp.org/HOWTO/Intkeyb/" In addition to computers, he is very interested in Hinduism but not really the guru side of things, but more-so freedom and self actualization. His website has more information: http://www.freebsd.nfo.sk/

OpenBSD as a

Primary Domain Controller

Once a Windows-based network grows beyond around a dozen computers, setting up a Primary Domain Controller to simplify and centralize the management of users, computers and network resources becomes a must.

What you will learn…

- How to create a central repository for information about domain users
- How to turn OpenBSD server into Primary Domain Controller and file server using Samba

What you should know…

- Bind, Samba, OpenLDAP
- Working knowledge on OpenBSD
- Experience on setting fully functional Domain Name Service

But does the Domain Controller necessarily have to be a Windows machine, thus meaning the end of our project of a completly OpenBSD-based server network (*http://www.kernel-panic.it/openbsd.html#world*)?

Of course not! Once again, OpenBSD (*http://www.openbsd.org/*) comes to our rescue and, with the help of a few additional pieces of software, it will turn into a full-blown, secure and reliable Domain Controller. In particular, the pieces of software we will use are the following:

- OpenLDAP (*http://www.openldap.org/*) – an *open source implementation of the Lightweight Directory Access Protocol (LDAP)*;
- Samba (*http://us3.samba.org/samba/*) – an *Open Source/Free Software suite* that provides *secure, stable and fast file and print services for all clients using the SMB/CIFS protocol, such as all versions of DOS and Windows, OS/2, Linux and many others*;
- IDX-smbldap-tools (*https://gna.org/projects/smbldap-tools/*) – a *set of perl scripts designed to manage user and group accounts stored in an LDAP directory*;
- Bind (*Berkeley Internet Name Domain https://www.isc.org/software/bind*) – an *open-source software that implements the Domain Name System (DNS) protocols for the Internet*;
- Clam AntiVirus (*http://www.clamav.net/*) – a *open source (GPL) anti-virus toolkit for UNIX*;
- Samba-vscan (*http://www.openantivirus.org/projects.php*) – a *proof-of-concept module for Samba, which uses the VFS (virtual file system) features of Samba 2.2.x/3.0 to provide an on-access Samba anti-virus*;
- CUPS (*Common UNIX Printing System http://www.cups.org/*) – a *standards-based, open source printing system*.

We have already discussed Bind configuration in a previous document (*http://www.kernel-panic.it/openbsd/dns/*) entirely dedicated to OpenBSD and DNS, so we won't come back to this topic now. Therefore, throughout this document, I will assume that you have already set up a fully functional Domain Name Server and that it correctly resolves the domain names of the client machines that will connect to the Domain Controller. Please note that this is a fundamental prerequisite for successfully building the Primary Domain Controller, since `nmbd(8)` will rely on DNS to resolve unregistered NetBIOS names.

Also a working knowledge of OpenBSD is assumed, since we won't delve into system management topics such as base configuration or packages/ports (*http://www.openbsd.org/faq/faq15.html*) installation.

OpenLDAP

OpenLDAP is an open source implementation of the Lightweight Directory Access Protocol. It will allow us to create a central repository for information about domain users, groups and computers, and make this information available to Samba (and any other LDAP-aware services) for authentication, authorization and management purposes.

The LDAP protocol

The *Lightweight Directory Access Protocol* (LDAP) is a networking protocol for accessing X.500-based directory services. A directory is a specialized database optimized for reading, browsing and searching and supports sophisticated filtering capabilities ([*OLDAP*] *http://www.openldap.org/doc/admin23/intro.html#What%20is%20a%20directory%20service*).

Similarly to the Unix file system or the *Domain Name System* (*http://www.kernel-panic.it/openbsd/dns/dns2.html*), the structure of this database is a hierarchical inverted tree, with the root at the top; for example: see Figure 1.

As in the above picture, the topmost levels of the LDAP tree are often arranged based upon domain names, thus allowing for directory services to be located using the Domain Name System.

Each node in the LDAP tree is called an entry and is uniquely identified by its Distinguished Name (DN), which is made up of the name of the entry itself (called the Relative Distinguished Name, RDN, usually derived from some attribute in the entry), comma-concatenated to the names of its parent entries. For instance, the DN of the entry highlighted in the following Figure 2.

Is made up of the sequence `uid=Danix`, `ou=Users`, `dc=kernel-panic` and `dc=it`, and is therefore written as `uid=Danix`, `ou=Users`, `dc=kernel-panic`, `dc=it` (see [RFC4514] *http://www.ietf.org/rfc/rfc4514.txt* for a full description of the DN format).

An entry consists of a set of attributes; each attribute has a name (or type) and one or more values. The name is typically a mnemonic string, like `dc` for `Domain Component` or `cn` for `Common Name`, and determines the syntax of the corresponding value. ObjectClasses define the attribute structure of an LDAP entry, i.e. which attributes *must* and which *may* be present in a specific LDAP entry. Both ObjectClasses and Attributes are defined within Schemas.

Though LDAP is a binary protocol, entries can be represented in a human-readable format by using the LDIF format; for example: see Listing 1.

LDAP queries can be represented by means of URLs, which allow you to specify the scope of the search and the search query, and to select which attibutes to return. The syntax of an LDAP URL is:

```
ldap://[host[:port]]/[DN[?[attributes][?[scope][?[filter][?
                    extensions]]]]]
```

Most of the URL components are optional:

- `host` is the name or address of the LDAP server to query;
- `port` is the network port the LDAP server is listening on (default is TCP port 389);
- `DN` is the *Distinguished Name* to use as the base object of the LDAP search (default is the root DN);
- `attributes` specifies which attributes should be returned from the entries (default is all attributes);
- `scope` is the scope of the search to perform. Available scopes are *base* (default) for a base object search, *one* for a one-level search, or *sub* for a subtree search;
- `filter` is the search filter to apply to entries within the specified scope during the search (default is `(objectClass=*)`);
- `extensions` are extensions to the LDAP URL format (default is no extensions).

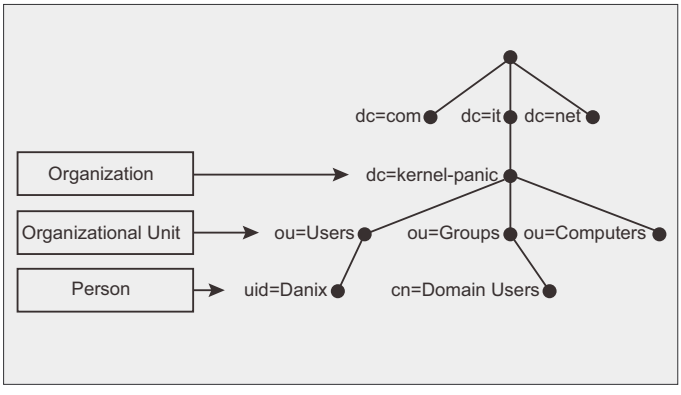

Figure 1. *The structure of database*

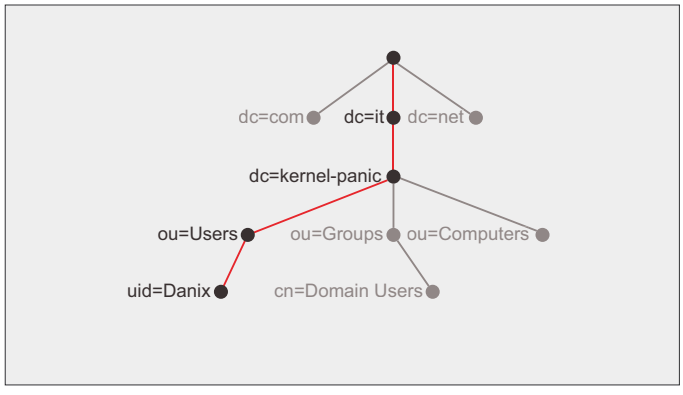

Figure 2. *Highlighted DN of the entry*

For example, the following URL:

```
ldap://ldap.kernel-panic.it/uid=Danix,ou=Users,dc=kernel-
                    panic,dc=it
```

refers to all attributes in a specific user entry, and an URL like:

```
ldap:///dc=kernel-panic,dc=it?sn?sub?(givenName=Daniele)
```

refers to the `sn` (surname) attribute of all entries with a `givenName` of *Daniele*. For further details, please refer to [RFC4516] *http://www.ietf.org/rfc/rfc4516.txt*.

Installation and configuration

Enough with the theory for now, and on to practice! OpenLDAP is available through OpenBSD's packages and ports system (*note*: unfortunately, the bdb flavor, providing support for the bdb and hdb backends, is marked as broken since OpenBSD 4.3 *http://www.openbsd.org/faq/faq15.html*); the following is the list of packages to be installed:

- `cyrus-sasl-x.x.x.tgz`
- `openldap-client-x.x.x.tgz`
- `openldap-server-x.x.x.tgz`

Listing 1. *LDAP represented in human readable format*

```
dn: uid=danix,ou=Users,dc=kernel-panic,dc=it
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
objectClass: posixAccount
objectClass: shadowAccount
objectClass: sambaSamAccount
cn: Daniele Mazzocchio
sn: Mazzocchio
givenName: Daniele
uid: Danix
uidNumber: 2000
gidNumber: 513
homeDirectory: /home/danix
loginShell: /bin/ksh
gecos: Daniele Mazzocchio
structuralObjectClass: inetOrgPerson
[ ... ]
```

And the installation is over! OpenLDAP configuration files are stored in `/etc/openldap`. Client-side configuration is contained in the `ldap.conf(5)` (*http://www.openldap.org/software/man.cgi?query=ldap.conf&format=html*) file; below is a sample configuration file: see Listing 2.

The `slapd.conf(5)` file provides configuration information for the Standalone LDAP Daemon, `slapd(8C)` (*http://www.openldap.org/software/man.cgi?query=slapd&format=html*): see Listing 3.

We can use the `slaptest(8C)` (*http://www.openldap.org/software/man.cgi?query=slaptest&format=html*) command to check the validity of our `slapd.conf(5)` (*http://www.openldap.org/software/man.cgi?query=slapd.conf&format=html*) file:

```
# install -d -o _openldap /var/run/openldap
# slaptest -u
config file testing succeeded
#
```

The `slapd.conf(5)` file, containing the `rootpw` password, should have restrictive permissions:

```
# chgrp _openldap /etc/openldap/slapd.conf
# chmod 640 /etc/openldap/slapd.conf
```

Ok, now everything should be ready for starting `slapd(8C)`. The first time you may want to invoke it with the `-d` option to turn on debugging and keep the daemon in the foreground, to immediately notice any error:

```
# /usr/local/libexec/slapd -4 -d 256 -u _openldap -g
                    _openldap
```

Listing 2. *Sample configuration file*

```
/etc/openldap/ldap.conf
# URI of the LDAP server to which the LDAP library
                    should connect
URI             ldap://ldap.kernel-panic.it
# The default base DN to use when performing LDAP
                    operations
BASE            dc=kernel-panic,dc=it

# Size limit to use when performing searches
SIZELIMIT       12
# Time limit to use when performing searches
TIMELIMIT       15
# Never derefernce aliases
DEREF           never
```

Listing 3. *Configuration information for the Standalone LDAP Daemon*

```
/etc/openldap/slapd.conf
# Include the necessary schema files. core.schema is required by default, the
# other ones are needed for sambaSamAccount. The samba.schema file can be found
# here and must be copied in /etc/openldap/schema/.
include         /etc/openldap/schema/core.schema
include         /etc/openldap/schema/cosine.schema
include         /etc/openldap/schema/inetorgperson.schema
include         /etc/openldap/schema/nis.schema
include         /etc/openldap/schema/samba.schema
# Absolute path to the PID file
pidfile         /var/run/openldap/slapd.pid
# Absolute path to the file that will hold slapd's command line options
argsfile        /var/run/openldap/slapd.args
# Type of database backend
database        ldbm
# DN suffix of queries that will be passed to this backend database
suffix          "dc=kernel-panic,dc=it"
# Database directory
directory       /var/openldap-data
# The Distinguished Name of the administrator of this database
rootdn          "cn=Manager,dc=kernel-panic,dc=it"
# Password (or password hash) for the rootdn. Clear-text passwords are allowed
# but strongly discouraged; the password hash can be generated using the
# slappasswd(8C) command; e.g.:
# # slappasswd
# New password: <password>
# Re-enter new password: <password>
# {SSHA}d1bjQZEA43NFKNL7g48XjaNv/W6DG0fY
rootpw          {SSHA}d1bjQZEA43NFKNL7g48XjaNv/W6DG0fY
# Maintain indices on the most useful attributes to speed up searches made on
# the sambaSamAccount, posixAccount and posixGroup objectClasses
index   objectClass             eq
index   cn                      pres,sub,eq
index   sn                      pres,sub,eq
index   uid                     pres,sub,eq
index   displayName             pres,sub,eq
index   uidNumber               eq
index   gidNumber               eq
index   memberUid               eq
index   sambaSID                eq
index   sambaPrimaryGroupSID    eq
index   sambaDomainName         eq
index   default                 sub
# Access control configuration.
The rootdn can always read and write everything                                 by * none
access to attrs=userpassword,sambaLMPassword,sambaNTPassword                access to *
    by anonymous auth                                                           by self write
    by self write                                                               by * read
```

```
[ ... ]
slapd starting
```

You can check that everything is running correctly by issuing the `ldapsearch(1)` (*http://www.openldap.org/software/man.cgi?query=ldapsearch&format=html*) command:

If everything is working fine, we can configure the system to start `slapd(8C)` on boot, by adding the following line (containing the command line arguments) to `/etc/rc.conf.local(8)` (*http://www.openbsd.org/cgi-bin/man.cgi?query=rc.conf.local&sektion=8*): see Listing 5.

Listing 4. *Instalation and configuration*

```
# ldapsearch -x -b '' -s base '(objectclass=*)'
                    namingContexts
# extended LDIF
#
# LDAPv3
# base <> with scope baseObject
# filter: (objectclass=*)
# requesting: namingContexts
#

#
dn:
namingContexts: dc=kernel-panic,dc=it

# search result
search: 2
result: 0 Success

# numResponses: 2
# numEntries: 1
#
```

Listing 5. *Configure the system by adding the command line arguments*

```
/etc/rc.conf.local
slapd_flags="-4 -u _openldap -g _openldap"
and the following commands to /etc/rc.local(8):
/etc/rc.local
if [ "$slapd_flags" != "NO" -a -x /usr/local/libexec/
                    slapd ]; then
    echo -n ' slapd'
    install -d -o _openldap /var/run/openldap
    /usr/local/libexec/slapd $slapd_flags
fi
```

LDAP over TLS/SSL

OpenLDAP comes with built-in support for the TLS/SSL protocols to provide integrity and confidentiality to LDAP connections by means of public-key cryptography. Enabling TLS/SSL will prevent traffic from traveling in the clear over the network, thus protecting users' passwords from eavesdroppers.

Setting up the PKI

TLS relies on public key certificates for authentication and therefore requires that you first set up a basic *Public Key Infrastructure* (PKI) for managing digital certificates. As a preliminary step, we will create the directories where certificates will be stored:

Listing 6. *The creation of the root CA certificate*

```
# openssl req -days 3650 -nodes -new -x509 -keyout
                    /etc/ssl/private/ca.key \
> -out /etc/openldap/ssl/ca.crt
[ ... ]
Country Name (2 letter code) []: IT
State or Province Name (full name) []: Italy
Locality Name (eg, city) []: Milan
Organization Name (eg, company) []: Kernel Panic Inc.
Organizational Unit Name (eg, section) []: LDAP CA
Common Name (eg, fully qualified host name) []:
                    ca.lan.kernel-panic.it
Email Address []: <enter>
#
```

Listing 7. *The creation of the private key and Certificate Signing Request*

```
# openssl req -days 3650 -nodes -new -keyout /etc/
                    openldap/ssl/private/server.key \
> -out /etc/openldap/ssl/private/server.csr
[ ... ]
Country Name (2 letter code) []: IT
State or Province Name (full name) []: Italy
Locality Name (eg, city) []: Milan
Organization Name (eg, company) []: KP Inc.
Organizational Unit Name (eg, section) []: LDAP Server
Common Name (eg, fully qualified host name) []:
                    ldap.kernel-panic.it
Email Address []: <enter>
Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []: <enter>
An optional company name []: <enter>
#
```

```
# install -m 700 -d /etc/openldap/ssl/private
```

The first step in setting up the PKI is the creation of the root CA certificate (/etc/openldap/ssl/ca.crt) and private key (/etc/ssl/private/ca.key) using openssl(1): see Listing 6.

Listing 8. *Generated Signed Certificate*

```
# openssl x509 -req -days 3650 -in /etc/openldap/ssl/
                    private/server.csr \
> -out /etc/openldap/ssl/server.crt -CA /etc/openldap/
                    ssl/ca.crt \
> -CAkey /etc/ssl/private/ca.key -CAcreateserial
Signature ok
subject=/C=IT/ST=Italy/L=Milan/O=Kernel Panic Inc./
                    OU=LDAP Server/CN=ldap.kernel-
                    panic.it
Getting CA Private Key
#
```

Listing 9. *Generated Client Certificate*

```
# openssl req -days 3650 -nodes -new -keyout /etc/
                    openldap/ssl/private/client.key \
> -out /etc/openldap/ssl/private/client.csr
[ ... ]
Country Name (2 letter code) []: IT
State or Province Name (full name) []: Italy
Locality Name (eg, city) []: Milan
Organization Name (eg, company) []: KP Inc.
Organizational Unit Name (eg, section) []: LDAP Client
Common Name (eg, fully qualified host name) []:
                    ldap.kernel-panic.it
Email Address []: <enter>

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []: <enter>
An optional company name []: <enter>
# openssl x509 -req -days 3650 -in /etc/openldap/ssl/
                    private/client.csr \
> -out /etc/openldap/ssl/client.crt -CA /etc/openldap/
                    ssl/ca.crt \
> -CAkey /etc/ssl/private/ca.key
Signature ok
subject=/C=IT/ST=Italy/L=Milan/O=Kernel Panic Inc./
                    OU=LDAP Client/CN=ldap.kernel-
                    panic.it
Getting CA Private Key
#
```

The next step is the creation of the private key (/etc/openldap/ssl/private/server.key) and *Certificate Signing Request* (/etc/openldap/ssl/private/server.csr) for the server: see Listing 7.

Finally, the CA will generate the signed certificate out of the certificate request: see Listing 8.

You can generate the client certificate by repeating the last two steps: see Listing 9. As a finishing touch, we need to assign restrictive permissions to the private keys, in order to prevent unauthorized access:

```
# chown -R _openldap:_openldap /etc/openldap/ssl/private
# chmod 600 /etc/openldap/ssl/private/*
```

OpenLDAP configuration

Configuring the slapd(8C) daemon for TLS operation simply requires that you add a few lines to slapd.conf(5), right after the rootpw parameter, containing the cipher suites to accept and the paths to the certificates: see Listing 10.

In the client configuration file, ldap.conf(5), you have to change the URI scheme to ldaps and specify the path to the CA certificate and the acceptable cipher suites: see Listing 11. As a final step, add the -h ldaps:/// option to the slapd(8C) command line arguments to make the daemon listen only for LDAP over TLS on TCP port 636:

```
/etc/rc.conf.local
# Only listen for LDAP over TLS (port 636)
slapd_flags="-4 -u _openldap -g _openldap -h ldaps:///"
```

and restart slapd(8C).

Listing 10. *Configuring the slaped (8C) daemon TLS operation*

```
/etc/openldap/slapd.conf
# TLS configuration
TLSCipherSuite          HIGH:MEDIUM:+SSLv3
TLSCACertificateFile    /etc/openldap/ssl/ca.crt
TLSCertificateFile      /etc/openldap/ssl/server.crt
TLSCertificateKeyFile   /etc/openldap/ssl/private/
                    server.key
```

Listing 11. *Changing the URI scheme to ldaps and specification of the path to CA certificate and cipher suits*

```
/etc/openldap/ldap.conf
[ ... ]
URI             ldaps://ldap.kernel-panic.it
# TLS configuration
TLS_CACERT      /etc/openldap/ssl/ca.crt
TLS_CIPHER_SUITE HIGH:MEDIUM:+SSLv3
```

Listing 12. *Sample Configuration File*

/etc/samba/smb.conf

```
##############################################################################
# Parameters in the [global] section apply to the server as a whole, or are    #
# defaults for sections that do not specifically define certain items          #
##############################################################################
[global]
# Domain name to use
    workgroup = KERNEL-PANIC
# String that will appear in browse lists next to the machine name
    server string = Samba Server
# Set the Samba server to user-level security (more details on security modes
# can be found here)
    security = user
# List of hosts permitted to access Samba services
    hosts allow = 172.16.0. 127.
# Negotiate encrypted passwords with the clients
    encrypt passwords = yes

# Use a separate log file for each machine that connects
    log file = /var/log/samba/smbd.%m
# Maximum size, in KB, of the log files
    max log size = 1024

# Select the backend(s) to retrieve and store passwords with. The LDAP URL is
# optional and defaults to 'ldap://localhost' (set the URI scheme to 'ldaps' if
# you're using LDAP over TLS/SSL)
    passdb backend = ldapsam:ldap://ldap.kernel-panic.it
# Avoid substituting %-macros in the passdb fields
    passdb expand explicit = no
# File containing the mapping of Samba users to local Unix users
    username map = /etc/samba/smbusers

# This socket option should give better performance
    socket options = TCP_NODELAY

# Allow nmbd(8) to try to become the local master browser
    local master = yes
# Tell Samba to be the Domain Master Browser for its workgroup
    domain master = yes
# A domain controller must have the 'os level' set at or above a value of 32
    os level = 33
# Make nmbd(8) force a local browser election on startup, also giving it a
# slightly higher chance of winning the election
    preferred master = yes
# A domain controller must provide the network logon service
    domain logons = yes
# Uncomment the following parameter to disable roaming profiles
```

Listing 12. *Sample Configuration File*

```
#    logon path =
# Name of an (optional) logon script (you can make it user-specific with '%U').
# The script must be in DOS format
    logon script = netlogon.bat

# Make nmbd(8) act as a WINS server
    wins support = yes
# Try to resolve NetBIOS names via DNS lookups
    dns proxy = yes

# LDAP options
    ldap suffix = dc=kernel-panic,dc=it
    ldap machine suffix = ou=Computers
    ldap user suffix = ou=Users
    ldap group suffix = ou=Groups
    ldap idmap suffix = ou=Idmap
    ldap admin dn = cn=Manager,dc=kernel-panic,dc=it
    ldap ssl = no
    ldap passwd sync = Yes

# Range of user and group ids allocated for mapping UNIX users to NT user SIDs
    idmap uid = 2000-4000
    idmap gid = 2000-4000

# Scripts to run when managing users with remote RPC (NT) tools
    add user script = /usr/local/sbin/smbldap-useradd -a -g 512 -m %u
                      add group script = /usr/local/sbin/smbldap-groupadd %g
                      add machine script = /usr/local/sbin/smbldap-useradd -w -g 515 %u
                      delete user script = /usr/local/sbin/smbldap-userdel -r %u
                      delete user from group script = /usr/local/sbin/smbldap-groupmod -x %u %g
                      delete group script = /usr/local/sbin/smbldap-groupdel -r %g


############################################################################
# Users' home directories. If no path is specified, the path is set to the     #
# (Unix) user's home directory (tipically '/home/<username>')                  #
############################################################################
[homes]
    comment = Home Directories
    browseable = no
    writable = yes


############################################################################
# The netlogon service allows you to specify the path to the logon scripts     #
############################################################################
[netlogon]
    comment = Share for logon scripts
```

Listing 12. *Sample Configuration File*

```
    path = /var/netlogon
    read only = yes
    write list = @"Domain Admins"
    browseable = no


############################################################################
# Shares definitions. The name of a section corresponds to the name of the     #
# shared resource. The following are just some examples, feel free to modify   #
# them according to your needs.                                                #
############################################################################

# A temporary directory for people to share files
[tmp]
    comment = Temporary file space
    path = /tmp
    read only = no
    public = yes

# A publicly accessible directory, but read only, except for people in the
# "staff" group
[public]
    comment = Public Stuff
    path = /home/samba
    public = yes
    writable = yes
    write list = @staff

# Define a share accessible only to a selected group of users. This directory
# should be writable by both users and should have the sticky bit set on it to
# prevent abuse
[myshare]
    comment = Mary's and Fred's stuff
    path = /usr/somewhere/shared
    valid users = mary fred
    public = no
    writable = yes
    create mask = 0660
    directory mask = 1770

# A service pointing to a different directory for each user that connects.
# %U gets replaced with the user name (in lower case) that is connecting
[private]
    comment = User data
    path = /var/data/%U
    valid users = %U
    public = no
    writable = yes
```

A bit of Samba

Samba is an Open Source software suite that, since 1992, *has provided secure, stable and fast file and print services for all clients using the SMB/CIFS protocol, such as all versions of DOS and Windows, OS/2, Linux and many others*. It will allow us to turn our OpenBSD server into a Primary Domain Controller and file server, able to interoperate with Windows-based client machines.

Installation and configuration

We can install most of the required software from the pre-compiled packages:

- libiconv-x.x.x.tgz
- popt-x.x.tgz
- gettext-x.x.tgz
- pcre-x.x.tgz
- glib2-x.x.x.tgz
- desktop-file-utils-x.x.tgz
- xdg-utils-x.x.x.tgz
- jpeg-x.tgz
- png-x.x.x.tgz
- tiff-x.x.x.tgz
- gdbm-x.x.x.tgz
- libdaemon-x.x.tgz
- lzo-x.x.tgz
- libgpg-error-x.x.tgz
- libgcrypt-x.x.x.tgz
- libtasn1-x.x.tgz
- gnutls-x.x.x.tgz
- dbus-x.x.x.tgz
- avahi-x.x.x.tgz
- cups-x.x.x.tgz
- libutf8-x.x.tgz

but we will compile Samba from the ports, because the antivirus module requires the Samba source code to successfully compile (of course feel free to install the pre-compiled package, `samba-x.x.x-cups-ldap.tgz`, if you don't need antivirus support).

```
# cd /usr/ports/net/samba
# env FLAVOR="cups ldap" make install
[ ... ]
```

Most of Samba configuration takes place in the `/etc/samba/smb.conf(5)` (*http://samba.org/samba/docs/man/manpages-3/smb.conf.5.html*) file. It is an INI-formatted file, made up of multiple sections, each beginning with the name of a shared resource (except for the `[global]` section) and containing a variable number of parameters, in the form `name = value`. Each parameter has a default value which will be retained if the parameter is omitted.

There are three special sections:

- `[global]` – defines global parameters and default values for the other sections;
- `[homes]` – allows on-the-fly creation of home directories for users connecting to the server;
- `[printers]` – allows users to connect to any printer specified in the local host's `/etc/printcap(5)` file.

Lines beginning with a semicolon (;) or hash (#) character are treated as comments; parameters may span across multiple lines using a back-slash (\). Listing 12 is a sample configuration file.

Now we need to create the file containing the mapping of Samba users to local Unix users, `/etc/samba/smbusers`.

Listing 13. *Testing the configuration*

```
# testparm
Load smb config files from /etc/samba/smb.conf
Processing section "[homes]"
Processing section "[tmp]"
Processing section "[public]"
Processing section "[myshare]"
Processing section "[private]"
Loaded services file OK.
Server role: ROLE_DOMAIN_PDC
Press enter to see a dump of your service definitions
[ ... ]
```

Listing 14. *Ading variables*

```
/etc/rc.conf.local
smbd_flags="-D"
nmbd_flags="-D"
and the appropriate startup commands to /etc/
                    rc.local(8):
/etc/rc.local
if [ "$smbd_flags" != "NO" -a -x /usr/local/libexec/
                    smbd ]; then
    echo -n ' smbd'
    /usr/local/libexec/smbd $smbd_flags
fi

if [ "$nmbd_flags" != "NO" -a -x /usr/local/libexec/
                    nmbd ]; then
    echo -n ' nmbd'
    /usr/local/libexec/nmbd $nmbd_flags
fi
```

In particular, we need to map the Domain Administrator user to root, in order to grant it the privileges it will need to manage the domain.

```
/etc/samba/smbusers
root = administrator
```

We can test our configuration by running the `testparm(1)` (*http://samba.org/samba/docs/man/manpages-3/testparm.1.html*) command: see Listing 13.

The last step is telling Samba the password to use to bind to the LDAP server (i.e. the (unencrypted) value of the rootpw parameter in `slapd.conf(5)`). Samba will store that password in `/etc/samba/secrets.tdb`:

```
# smbpasswd -w <password>
Setting stored password for "cn=Manager,dc=kernel-
                    panic,dc=it" in secrets.tdb
```

Now we can configure the system to start Samba on boot by adding a couple of variables to the `/etc/rc.conf.local(8)` file: see Listing 14.

Finally, we are ready to start Samba, though it will be pretty useless until the LDAP database has been populated; so that's what we're going to do in the next chapter (*http://www.kernel-panic.it/openbsd/pdc/pdc4.html*).

```
# mkdr /var/log/samba
# /usr/local/libexec/smbd -D
# /usr/local/libexec/nmbd -D
```

The IDX-smbldap-tools

Smbldap-tools (*https://gna.org/projects/smbldap-tools/*) is *a set of perl scripts designed to manage user and group accounts stored in an LDAP directory*. These scripts will make our lives much easier by providing a set of simple commands for carrying out the most common user

Listing 15. *Seting global parameters*

```
/etc/smbldap-tools/smbldap.conf
# SID and domain name
SID="S-1-5-21-2855447605-3248580512-2157288933"
sambaDomain="KERNEL-PANIC"

# LDAP servers and ports (if you're using LDAP over
                    TLS/SSL, set the URI schemes
# to 'ldaps' and the ports to '636')
slaveLDAP="ldap://ldap.kernel-panic.it"
slavePort="389"
masterLDAP="ldap://ldap.kernel-panic.it"
masterPort="389"

# TLS configuration (set ldapTLS to '1' to enable TLS)
ldapTLS="0"
verify="none"
cafile="/etc/openldap/ssl/ca.crt"
clientcert="/etc/openldap/ssl/client.crt"
clientkey="/etc/openldap/ssl/private/client.key"

# LDAP configuration
suffix="dc=kernel-panic,dc=it"
usersdn="ou=Users,${suffix}"
computersdn="ou=Computers,${suffix}"
groupsdn="ou=Groups,${suffix}"
idmapdn="ou=Idmap,${suffix}"
sambaUnixIdPooldn="sambaDomainName=KERNEL-
                    PANIC,${suffix}"
scope="sub"
hash_encrypt="SSHA"
crypt_salt_format="%s"

# Unix accounts configuration
userLoginShell="/bin/ksh"
userHome="/home/%U"
userHomeDirectoryMode="700"
userGecos="System User"
defaultUserGid="513"
defaultComputerGid="515"
skeletonDir="/etc/skel"
defaultMaxPasswordAge="45"

# Samba configuration
userSmbHome=""
userProfile=""
userHomeDrive="H:"
userScript="logon.bat"
mailDomain="kernel-panic.it"

# smbldap-tools configuration
with_smbpasswd="0"
smbpasswd="/usr/local/bin/smbpasswd"
with_slappasswd="0"
slappasswd="/usr/local/sbin/slappasswd"
```

administration tasks, thus saving us from dealing with the internals of LDAP and making managing Samba users almost as easy as managing normal system users.

Please note that, though Samba account information will be stored in LDAP, `smbd(8)` will still obtain the user's UNIX account information via the standard C library calls, such as `getpwnam()` (see documentation *http://www.samba.org/samba/docs/man/Samba-HOWTO-Collection/passdb.html#id357234*), which don't natively support LDAP. This means we'll also need to configure the `ypldap(8)` daemon, which will act as an interface between LDAP and OpenBSD's authentication routines.

Configuration

The smbldap-tools require the installation of quite a few perl modules:

- p5-Jcode-x.x.tgz
- p5-Unicode-String-x.x.tgz
- p5-Unicode-Map8-x.x.tgz
- p5-Unicode-Map-x.x.tgz
- p5-Unicode-MapUTF8-x.x.tgz
- p5-Convert-ASN1-x.x.tgz
- p5-Digest-SHA1-x.x.tgz
- p5-Digest-HMAC-x.x.tgz
- p5-GSSAPI-x.x.tgz
- p5-Authen-SASL-x.x.tgz
- p5-Net-SSLeay-x.x.tgz
- p5-IO-Socket-SSL-x.x.tgz
- p5-XML-Parser-x.x.tgz
- p5-XML-SAX-Writer-x.x.tgz
- p5-XML-SAX-x.x.tgz
- p5-XML-NamespaceSupport-x.x
- p5-Text-Iconv-x.x
- p5-XML-Filter-BufferText-x.x
- p5-URI-x.x.tgz
- p5-ldap-x.x.tgz
- p5-Crypt-SmbHash-x.x.tgz
- smbldap-tools-x.x.x.tgz

The `/etc/smbldap-tools/smbldap_bind.conf` file contains the parameters to connect to the LDAP server; they should match the `rootdn` and `rootpw` parameters in `/etc/openldap/slapd.conf`. Make sure this file has restrictive permissions (`600`) to protect the passwords from unauthorized access.

```
/etc/smbldap-tools/smbldap_bind.conf
slaveDN="cn=Manager,dc=kernel-panic,dc=it"
slavePw="password"
masterDN="cn=Manager,dc=kernel-panic,dc=it"
masterPw="password"
```

Before editing the next configuration file, we need to retrieve the SID for the domain:

```
# net getlocalsid
SID for domain SAMBA is: S-1-5-21-2855447605-3248580512-
                    2157288933
```

Listing 16. *Creating the structure of the LDAP tree*

```
# /usr/local/sbin/smbldap-populate
Populating LDAP directory for domain KERNEL-PANIC
                    (S-1-5-21-2855447605-3248580512-
                    2157288933)
(using builtin directory structure)

adding new entry: dc=kernel-panic,dc=it
adding new entry: ou=Users,dc=kernel-panic,dc=it
adding new entry: ou=Groups,dc=kernel-panic,dc=it
adding new entry: ou=Computers,dc=kernel-panic,dc=it
adding new entry: ou=Idmap,dc=kernel-panic,dc=it
adding new entry: uid=root,ou=Users,dc=kernel-
                    panic,dc=it
adding new entry: uid=nobody,ou=Users,dc=kernel-
                    panic,dc=it
adding new entry: cn=Domain Admins,ou=Groups,dc=kernel
                    -panic,dc=it
adding new entry: cn=Domain Users,ou=Groups,dc=kernel-
                    panic,dc=it
adding new entry: cn=Domain Guests,ou=Groups,dc=kernel
                    -panic,dc=it
adding new entry: cn=Domain Computers,ou=Groups,dc=ker
                    nel-panic,dc=it
adding new entry: cn=Administrators,ou=Groups,dc=kerne
                    l-panic,dc=it
adding new entry: cn=Account Operators,ou=Groups,dc=ke
                    rnel-panic,dc=it
adding new entry: cn=Print Operators,ou=Groups,dc=kern
                    el-panic,dc=it
adding new entry: cn=Backup Operators,ou=Groups,dc=ker
                    nel-panic,dc=it
adding new entry: cn=Replicators,ou=Groups,dc=kernel-
                    panic,dc=it
adding new entry: sambaDomainName=KERNEL-
                    PANIC,dc=kernel-panic,dc=it

Please provide a password for the domain root:
Changing UNIX and samba passwords for root
New password: <admin_passwd>
Retype new password: <admin_passwd>
#
```

Listing 17. *Getting a LDIF dump*

```
# ldapsearch -x -LL -b 'ou=Users,dc=kernel-
                    panic,dc=it' -s sub
version: 1

dn: ou=Users,dc=kernel-panic,dc=it
objectClass: top
objectClass: organizationalUnit
ou: Users

[ ... ]
```

Listing 18. *Initializing the YP server as a master*

```
# ypinit -m
Server Type: MASTER Domain: kernel-panic.it
Creating an YP server will require that you answer a
                    few questions.
Questions will all be asked at the beginning of the
                    procedure.

Do you want this procedure to quit on non-fatal
                    errors? [y/n: n] <Enter>

Ok, please remember to go back and redo manually
                    whatever fails.
If you don't, something might not work.

At this point, we have to construct a list of this
                    domain's YP servers.
smb.kernel-panic.it is already known as master server.
Please continue to add any slave servers, one per line.
                    When you are
done with the list, type a <control D>.
        master server   :  smb.kernel-panic.it
        next host to add:  ^D
The current list of NIS servers looks like this:

smb.kernel-panic.it

Is this correct?  [y/n: y] <Enter>
Building /var/yp/kernel-panic.it/ypservers...
smb.kernel-panic.it has been setup as an YP master
                    server.
Edit /var/yp/kernel-panic.it/Makefile to suit your
                    needs.
After that, run 'make' in /var/yp.
#
```

The `/etc/smbldap-tools/smbldap.conf` file allows you to set global parameters that will be readable by everybody see Listing 15.

Populating the LDAP database

Now we can create the structure of the LDAP tree by inserting the base entries in the database; the `smbldap-populate` script will take care of everything for us: see Listing 16.

The last step of the above command doesn't actually change the UNIX password for the root account: it only sets the password for the domain administrator (in LDAP).

You can test that the database now contains the base entries by running the `ldapsearch(1)` (*http://www.openldap.org/software/man.cgi?query=ldapsearch&format=html*) command; you can get an LDIF dump of the users defined in the LDAP database by running: see Listing 17.

In addition to the default groups created by `smbldap-populate`, you may also want to define some additional groups, e.g.:

```
# smbldap-groupadd -g 1500 Accounting
[ ... ]
```

Now we need to create the appropriate user for every computer that will need to connect to Samba (the `$` sign at the end of each name is mandatory):

```
# smbldap-useradd -w -u 3000 computer1$
# smbldap-useradd -w -u 3001 computer2$
[ ... ]
```

Finally, we can create the actual Samba users; each user will have a home directory that will be automatically connected as drive `H:` at logon:

```
# smbldap-useradd -a -u 2000 -g 512 -G 513 -N Daniele -S
                    Mazzocchio \
>   -c "Daniele Mazzocchio" danix
# smbpasswd -a danix
New SMB password: password
Retype new SMB password: password
#
```

Now we can (re)start the Samba processes:

```
# pkill .mbd
# /usr/local/libexec/smbd -D
# /usr/local/libexec/nmbd -D
```

Don't forget to assign the correct permissions and ownerships to the Samba shares.

Configuring ypldap(8)

YP(8) (*http://www.openbsd.org/cgi-bin/man.cgi?query=yp&sektion=8*) is a directory service originally developed by Sun Microsystems which, long before LDAP, allowed network management of password, group and hosts file entries. Starting with release 4.5 (*http://www.openbsd.org/45.html*), OpenBSD provides an additional YP daemon, ypldap(8), which uses LDAP as a backend in place of the traditional db(3) (*http://www.openbsd.org/cgi-bin/man.cgi?query=db&sektion=3*) database.

Since YP is the only directory service that can be accessed directly using standard C-library functions like getpwent(3), getgrent(3), gethostbyname(3) and so on [FAQ10] (*http://www.openbsd.org/faq/faq10.html#Dir*), it will act as an interface between the system's authentication routines (used by smbd(8) *http://samba.org/samba/docs/man/manpages-3/smbd.8.html*) and the LDAP directory.

As a first step in configuring the YP subsystem, we will set the YP domain of the host, which is an arbitrary string identifying the hosts that share (part of) their system configuration data through YP (and has nothing to do with Samba or DNS domains); the YP domain for a host is set with domainname(1) and can be made permanent across reboots by putting it into the file /etc/defaultdomain(5):

```
# domainname kernel-panic.it
# echo "kernel-panic.it" > /etc/defaultdomain
```

Before initializing the YP server, you may want to edit /var/yp/Makefile.yp in order to create only the necessary YP maps, by modifying the all target:

```
/var/yp/Makefile.yp
all: passwd group netid
```

Now we are ready to initialize the YP server as a master by issuing the ypinit(8) command: see Listing 18.

The default configuration file for ypldap(8) (*http://samba.org/samba/docs/man/manpages-3/smbd.8.html*) is /etc/ypldap.conf(5), which is made up of three sections: macros, global configuration settings and the declaration of one or more directories. In Listing 19 is a sample configuration file.

Since it contains sensitive information, the ypldap.conf(5) file should have restrictive permissions (600); the -n flag of ypldap(8) allows you to check the configuration file for validity:

```
# chmod 600 /etc/ypldap.conf
# ypldap -n
configuration OK
```

Listing 19. *Sample Configuration file*

```
/etc/ypldap.conf
# Macros
# Optional macros go here...

# Global settings
domain       "kernel-panic.it"
interval     3600
# Specify the maps that ypldap should provide
provide map  "passwd.byname"
provide map  "passwd.byuid"
provide map  "group.byname"
provide map  "group.bygid"

# Directory declaration
directory "ldap.kernel-panic.it" {
    binddn    "cn=Manager,dc=kernel-panic,dc=it"
    bindcred  "password"
    basedn    "ou=Users,dc=kernel-panic,dc=it"

    # passwd maps configuration
    passwd filter "(objectClass=posixAccount)"

    attribute name maps to "uid"
    fixed attribute passwd "*"
    attribute uid maps to "uidNumber"
    attribute gid maps to "gidNumber"
    attribute gecos maps to "gecos"
    attribute home maps to "homeDirectory"
    # LDAP users are not interactive system users
    fixed attribute shell "/sbin/nologin"
    fixed attribute change "0"
    fixed attribute expire "0"
    fixed attribute class "default"

    # group maps configuration
    group filter "(objectClass=posixGroup)"

    attribute groupname maps to "cn"
    fixed attribute grouppasswd "*"
    attribute groupgid maps to "gidNumber"
    list groupmembers maps to "memberUid"
}
```

To actually tell the system to include user and group accounts from the YP domain, we need to add the default YP markers to the passwd(5) and group(5) files:

```
# echo "+:*::::::::" >> /etc/master.passwd
# pwd_mkdb -p /etc/master.passwd
# echo "+:*::" >> /etc/group
```

Well, now we're ready to start all the required daemons! YP uses RPC(3) (*http://www.openbsd.org/cgi-bin/man.cgi?query=rpc&sektion=3*) to communicate with clients, so it requires that the portmap(8) daemon be enabled. Also the ypbind(8) daemon is required for the server to use its own maps.

```
# portmap
# ypldap
```

Listing 20. *Starting automatically the daemons*

```
/etc/rc
#       if [ -d /var/yp/'domainname' ]; then
#               # YP server capabilities needed...
#               echo -n ' ypserv';
                  ypserv ${ypserv_flags}
#               #echo -n ' ypxfrd';
                  ypxfrd
#       fi

#       if [ -d /var/yp/binding ]; then
#               # YP client capabilities needed...
#               echo -n ' ypbind';
                  ypbind
#       fi
and add the following commands to /etc/rc.local(8),
                  right after the startup of
                  slapd(8C):
/etc/rc.local
if [ -d /var/yp/$(domainname) ]; then
    echo -n ' ypldap'
    ypldap ${ypldap_flags}
    # Wait 5 seconds to fully initialize ypldap before
                  starting ypbind
    sleep 5
fi

if [ -d /var/yp/binding ]; then
    echo -n ' ypbind'
    ypbind
fi
```

```
# ypbind
Enabling yp client subsystem.
To disable: kill ypbind and remove /var/yp/binding
#
```

You can test that the system is correctly retrieving user information from the YP directory by using the getent(1) command:

```
# getent passwd
[ ... ]
danix:*:2000:512:Daniele Mazzocchio:/home/danix:/sbin/
                  nologin
#
```

To automatically start the daemons on boot, add the following lines to the /etc/rc.conf.local(8) file:

```
/etc/rc.conf.local
portmap=YES
ypldap_flags=""
```

comment out the following lines in /etc/rc(8) (which would start ypserv(8) instead of ypldap(8)): see Listing 20.

Well, now we have a fully functional *Primary Domain Controller*: then we can start joining computers (*http://technet2.microsoft.com/windowsserver/en/library/34f9c7c0-50c4-4adf-9106-db9c7e671d091033.mspx?mfr=true*) to our fresh new domain and perform all the necessary tests! The next chapters will discuss a couple of additional features you may find very useful: antivirus support and printer shares.

Keeping viruses away with Samba-vscan

So we have a fully functional file server and primary domain controller now. However, you may want to add some nice additional features to it, such as antivirus support to detect and quarantine viruses in real time.

Samba-vscan (*http://www.openantivirus.org/projects.php#samba-vscan*) is a proof-of-concept module for Samba, which uses the VFS (virtual file system) features of Samba 2.2.x/3.0 to provide an on-access Samba antivirus. Samba-vscan currently supports several antivirus softwares, including ClamAV (*http://www.clamav.net/*), which we will use as the backend antivirus engine.

We already discussed ClamAV (*http://www.kernel-panic.it/openbsd/mail/mail6.html#mail-6.2*) installation and configuration in a previous document (*http://www.kernel-panic.it/openbsd/mail/*), so we won't dwell upon it now and I assume you already have a clamd daemon up and running on the file server itself or on another machine.

Compiling Samba-vscan requires the prior installation of the following packages:

- autoconf-2.61p3.tgz
- libmagic-x.x.tgz
- gmake-x.x.tgz
- bzip2-x.x.x.tgz

As a preliminary step, we will also need to `make proto` the Samba port; therefore, go to the `/usr/ports/obj/samba/w-samba-x.x.x-cups-ldap/samba-x.x.x/source/` directory and edit the `autogen.sh` file, by replacing the first lines after the initial comments with:

```
/usr/ports/obj/samba/w-samba-x.x.x-cups-ldap/samba-x.x.x/
                    source/autogen.sh
TESTAUTOHEADER="autoheader-2.61"
TESTAUTOCONF="autoconf-2.61"
```

Then, still from within that directory, run:

```
# ./autogen.sh
[ ... ]
# ./configure
[ ... ]
# make proto
[ ... ]
```

Now we are ready to download (*http://www.openantivirus.org/projects.php*), extract and compile Samba-vscan: see Listing 21.

The configuration file for Samba-vscan (with ClamAV support) is named `/etc/samba/vscan-clamav.conf`: see Listing 22.

The last step is updating Samba configuration to include antivirus support by adding the following lines in each section corresponding to a share you want to protect against viruses, or in the `[global]` section if you want to protect all of your shares.

```
/etc/samba/smb.conf
    vfs object = vscan-clamav
    vscan-clamav: config-file = /etc/samba/vscan-clamav.conf
```

and reload Samba configuration:

```
# pkill -HUP smbd
```

Sharing printers with CUPS

The Common UNIX Printing System (CUPS *http://www.cups.org/*) is a software providing a portable printing layer for UNIX-based operating systems. It will allow us to turn the system into a printer server and share printers with Samba; though this is not a particularly difficult task, please be sure to closely follow this procedure to successfully export the `printer(s)` to Samba through the `cupsaddsmb(8)` (*http://www.cups.org/documentation.php/man-cupsaddsmb.html*) command.

Listing 21. *Download, extract and compile Samba-vscan*

```
# tar -zxvf samba-vscan-x.x.x.tar.gz
[ ... ]
# cd samba-vscan-x.x.x/
# env LDFLAGS=-L/usr/local/lib/ CPPFLAGS=-I/usr/local/
                    include/ ./configure \
>  --with-samba-source=/usr/ports/net/samba/w-samba-
                    x.x.x-cups-ldap/samba-x.x.x/
                    source/
[ ... ]
# gmake clamav
[ ... ]
# cp vscan-clamav.so /usr/local/lib/samba/vfs/
# cp clamav/vscan-clamav.conf /etc/samba/
```

Listing 22. *Configuration file for Samba-vscan*

```
/etc/samba/vscan-clamav.conf
[samba-vscan]
max file size = 10485760
verbose file logging = no

scan on open = yes
scan on close = yes

deny access on error = no
deny access on minor error = no

send warning message = yes
infected file action = nothing
quarantine directory  = /var/clamav/quarantine/
quarantine prefix = vir-

max lru files entries = 100
lru file entry lifetime = 5
exclude file types =
scan archives = yes

clamd socket name = /var/clamav/clamd.sock
libclamav max files in archive = 1000
libclamav max archived file size = 10485760
libclamav max recursion level = 5
```

You should already have installed CUPS as a dependency when adding the Samba package. CUPS configuration goes beyond the scope of this document, so please refer to the documentation (*http://www.cups.org/documentation.php*) for a detailed description of its features and options. The following configuration will refer to my own printer (a Dell 1600n Laser printer), so make sure to correctly configure your own printer(s) before proceeding to Samba configuration. The printers are defined in the `/etc/cups/printers.conf(5)` configuration file:

Listing 23. *Exporting printers to Samba*

```
/etc/samba/smb.conf
[global]
    [ ... ]

    load printers = yes
    printing = cups
    printcap name = cups
    show add printer wizard = Yes
    use client driver = No

[dp1600n]
    comment = Dell Laser MFP 1600n
# Users must have write access to the spool directory
    valid users = root @DomainUsers
    path = /var/spool/samba/printing
    printer = dp1600n
    public = no
    writable = no
    printable = yes

[print$]
    comment = Printer Drivers
    path = /etc/samba/drivers
    browseable = no
    guest ok = no
    read only = yes
    write list = root
```

Listing 24. *Finding the PostScript drivers and PPD file(s)*

```
# ls -l /etc/samba/drivers/W32X86/3/
total 2884
-rwxr--r--  1 root  wheel   25729 Feb 28 01:55
                    dp1600n.ppd
-rwxr--r--  1 root  wheel  129024 Feb 28 01:49
                    ps5ui.dll
-rwxr--r--  1 root  wheel   26038 Feb 28 01:55
                    pscript.hlp
-rwxr--r--  1 root  wheel  792644 Feb 28 01:55
                    pscript.ntf
-rwxr--r--  1 root  wheel  455168 Feb 28 01:49
                    pscript5.dll
#
```

```
/etc/cups/printers.conf
<DefaultPrinter dp1600n>
    Info          Dell Laser Printer 1600n
    Location      Room 123
    DeviceURI     ipp://prn1.lan.kernel-panic.it/
    State         Idle
    StateMessage  Printer is idle
    Accepting     Yes
</Printer>
```

Getting the driver files

Now we have to retrieve the correct driver files. First, we need the Universal PostScript printer drivers for Windows from the Adobe website. You can download them here (*http://www.adobe.com/support/downloads/product.jsp?product=44&platform=Windows*): select the installer for your language and install the drivers on a Windows machine. At the end of the installation, you should find the following files in the `C:\WINDOWS\system32\spool\drivers\w32x86\3` folder:

- PS5UI.DLL
- PSCRIPT.HLP
- PSCRIPT.NTF
- PSCRIPT5.DLL

Now create the `/usr/local/share/cups/drivers` directory on the file server:

```
# mkdir /usr/local/share/cups/drivers/
```

and copy the above files into it (*warning*: on the file server, driver file names must be lowercase!).

Next, we need to download (*http://www.cups.org/windows/software.php?6.0*) the Windows CUPS drivers and extract and copy them to the drivers directory:

```
# tar -zxvf cups-windows-6.0-source.tar.gz
[ ... ]
# cd cups-windows-6.0/i386
# cp cups6.ini cupsui6.dll cupsps6.dll /usr/local/share/
                    cups/drivers/
```

The last file you need to retrieve is the PPD file appropriate to your printer. Fortunately, if you can't find the file on

the printer driver CD, Easy Software Products (*http://www.easysw.com/*) provides a huge collection of PPD files which includes support for the most common printers. Download (*http://ftp.easysw.com/pub/printpro/4.5.12/printpro-4.5.12-linux-intel.tar.gz*) the Linux file (portable format), extract it, look for the PPD file appopriate to your printer and copy it to `/etc/cups/ppd/`; for example:

```
# tar -zxvf printpro-4.5.12-linux-intel.tar.gz
[ ... ]
# tar -zxvf printpro-dell.ss
[ ... ]
# gunzip -o /etc/cups/ppd/dp1600n.ppd usr/share/cups/
                    model/en/dp1600n.ppd.gz
```

Please note that the PPD file has exactly the same name (`dp1600n`) as the printer defined in `/etc/cups/printers.conf(5)` (plus the `.ppd` extension). If the two names differ, you may encounter problems when running the `cupsaddsmb(8)` command later.

Exporting printers to Samba

Now we can proceed to update Samba configuration by adding a few options to the `[global]` section and by defining a couple of additional sections: see Listing 23.

The spool directory must be writeable by the users authorized to print and have the sticky-bit set; for example:

```
# chgrp 513 /var/spool/samba/printing
# chmod 1770 /var/spool/samba/printing
```

Now we can start the `cupsd(8)` daemon and reload Samba configuration:

```
# /usr/local/sbin/cupsd
# pkill -HUP smbd
```

Well, so we're finally ready to issue the `cupsaddsmb(8)` command, which will actually export printers to samba:

```
# mkdir /etc/samba/drivers
# cupsaddsmb -H localhost -U root -v -a
[ ... ]
Printer Driver dp1600n successfully installed.
[ ... ]
Succesfully set dp1600n to driver dp1600n.
#
```

If everything went fine, now you should find the PostScript drivers and the PPD file(s) in the fresh new `/etc/samba/drivers/W32X86/3` directory: see Listing 24.

The last step is configuring the system to run `cupsd(8)` on boot, by adding the following lines to the `/etc/rc.local` file, before the start of Samba:

```
/etc/rc.local
if [ -x /usr/local/sbin/cupsd ]; then
    echo -n ' cupsd'
    /usr/local/sbin/cupsd
fi
```

Bibliography

- *Lightweight Directory Access Protocol* (LDAP): The Protocol – *http://www.rfc-editor.org/rfc/rfc4511.txt*
- OpenLDAP Software 2.3 Administrator's Guide – *http://www.openldap.org/doc/admin23/*
- The Official Samba-3 HOWTO and Reference Guide – *http://us4.samba.org/samba/docs/man/Samba-HOWTO-Collection/*
- Samba-3 by Example – *http://us4.samba.org/samba/docs/man/Samba-Guide/*
- Using Samba, Jay Ts, Robert Eckstein & David Collier-Brown, O'Reilly, 2003 – *http://us4.samba.org/samba/docs/using_samba/toc.html*
- OpenBSD as a File Server – *http://www.onlamp.com/pub/a/bsd/2000/11/14/OpenBSD.html*
- Security with LDAP – *http://www.skills-1st.co.uk/papers/security-with-ldap-jan-2002/security-with-ldap.html*

References

- [OLDAP] – *http://www.openldap.org/doc/admin23/intro.html#What%20is%20a%20directory%20service* – Introduction to OpenLDAP Directory Services
- [RFC4514] – *http://www.rfc-editor.org/rfc/rfc4514.txt* – RFC 4514 – Lightweight Directory Access Protocol (LDAP): String Representation of Distinguished Names
- [RFC4516] – *http://www.rfc-editor.org/rfc/rfc4516.txt* – RFC 4516 – Lightweight Directory Access Protocol (LDAP): Uniform Resource Locator
- How do I configure the BDB backend? *http://www.openldap.org/faq/data/cache/893.html*
- The Official Samba-3 HOWTO and Reference Guide – Password Backends *http://us4.samba.org/samba/docs/man/Samba-HOWTO-Collection/passdb.html#id327194*
- [FAQ10] – *http://www.openbsd.org/faq/faq10.html#Dir* – FAQ 10 – System Management – Directory services

Appendix

Special thanks to Michael Cooter for motivating me to write this document and for his useful suggestions and comments.

DANIELE MAZZOCCHIO

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

FreeBSD MySQL

Clustering How-to

The PHP, MySQL and Apache stack is a very popular implementation on standalone BSD servers but in demanding high availability [HA] environments the twin spectres of redundancy and fail-over rear their heads. In these scenarios, it is essential to eliminate the single point of failure which is the enemy of 100% uptime.

What you will learn…

- The difference between replication / clustering and how to configure a basic MySQL cluster from scratch.

What you should know…

- How to install FreeBSD from scratch including basic IP configuration

Over the years, many advances have been made in the areas of hardware redundancy, with hot plug / hot swap power supplies, motherboards etc. which builds redundancy into individual servers but not into the system per se. Whilst these features go a long way to provide stability, there is the possibility that data corruption can arise where data is mirrored across drives. If a hard disk controller doesn't have the intelligence to identify an error condition (e.g. an application generated corruption or even a failure in the hardware itself) the corrupted data will be byte copied at low level across to the redundant server, thereby ruining the clean copy. While no system is 100% foolproof, there are further measures we can take to improve reliability. Where a database application is required, the two most common scenarios to reduce risk are clustering and replication.

Clustering versus Replication

Both solutions have advantages and disadvantages. Replication requires a master server and N slaves, and updates are performed sequentially (synchronous communication). If multiple slaves are present, this gives good redundancy under normal circumstances as copies of the database are distributed across multiple servers. However, if a particularly complex SQL query is processed, this could cause the slave[s] to fall out of sync with the master for a period of time which may not be desirable. If the master were to fail after a transaction (but before the database was committed to the slave), data consistency would be lost if a non-transaction-safe database was in use. Replication is useful with very large datasets, as clustering can become impractical with extremely large databases due to the sheer quantity and specification of hardware required.

By design, a MySQL cluster does not have any single point of failure, and all databases are updated simultaneously (asynchronous communication). As the database is stored and accessed in RAM, transactions are

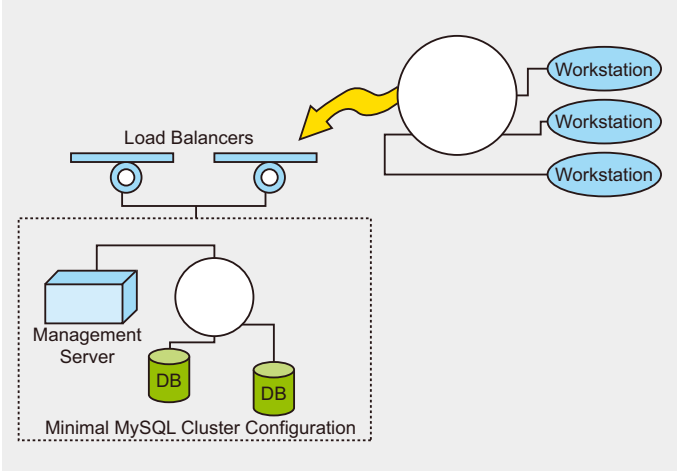

Figure 1. *Minimal Configuration*

saved to disk when a node is shut-down and conversely loaded into RAM when the node is started. In the event of a catastrophic failure anywhere within the cluster, the other nodes are not affected. MySQL clustering also offers benefits such as high speed connectivity using SCI (Scalable Coherent Interface), the ability to start and stop nodes in real time without affecting applications, scalability and a response time in milliseconds. However, clustering poses some limitations from the the design perspective (e.g. no support for fulltext indexes) and all machines in the cluster must be the same architecture. Ideally, an enterprise grade system will utilise both clustering and replication to leverage both technologies.

Load Balancing and security

As each node appears as an individual database instance, in a HA scenario a load balancer (or ideally redundant load balancers to remove the single point of failure) would be required. This may be achieved at the TCP/IP or at the application layer. As each node is visible on the network as a separate entity, consideration should be made about firewall configuration – or best case – running the cluster on a totally separate network. Depending on the scenario, various possible techniques may be used such as FreeBSD CARP, DNS Round Robin or MySQL Proxy etc. See Figure 1.

Listing 1. *Management server config.ini*

```
[NDBD DEFAULT]
NoOfReplicas=2
DataDir=/usr/local/mysql-cluster
[MYSQLD DEFAULT]
[NDB_MGMD DEFAULT]
[TCP DEFAULT]
# Management Node
[NDB_MGMD]
HostName=192.168.0.30   # The IP of the Management
                server
# Storage Nodes
[NDBD]
HostName=192.168.0.28   # The IP of the first node
[NDBD]
HostName=192.168.0.29   # The IP of the second node
# SQL Nodes
[MYSQLD]
HostName=192.168.0.28
[MYSQLD]
HostName=192.168.0.29
```

Requirements

A minimum of 3 servers are required for a cluster. While MySQL will support domain names within the cluster, for this example I have used a static IP address for simplicity. A patched minimal FreeBSD 8.0 installation (x3) is required as well as the patched mysql-cluster port available from Alex Dupre (see table On the 'Net). The following cluster was tested under VirtualBox with the nodes having an IP address of 192.168.0.28 and 192.168.0.29 respectively and a management node IP of 192.168.0.30. As this port is classed as experimental, use in a production environment is left to the discretion of the sys admin as *your mileage may vary* (YMMV), but please note that MySQL does not support clustering in a VM environment – my setup was purely for convenience of testing.

Installation and configuration

Step 1 – Install the port

Download and copy the `mysql_cluster` port and as root, extract compile and install:

```
su
tar -xvzf mysql-cluster.tar.gz
cd mysql-cluster
```

Listing 2. *NDB cluster running successfully*

```
-- NDB Cluster -- Management Client --
ndb_mgm> SHOW
Connected to Management Server at: localhost:1186
Cluster Configuration
---------------------
[ndbd(NDB)]   2 node(s)
id=2   @192.168.0.28  (mysql-5.1.37 ndb-7.0.8,
                    Nodegroup: 0, Master)
id=3   @192.168.0.29  (mysql-5.1.37 ndb-7.0.8,
                    Nodegroup: 0)

[ndb_mgmd(MGM)]   1 node(s)
id=1   @192.168.0.30  (mysql-5.1.37 ndb-7.0.8)
[mysqld(API)]   2 node(s)
id=4   @192.168.0.28  (mysql-5.1.37 ndb-7.0.8)
id=5   @192.168.0.29  (mysql-5.1.37 ndb-7.0.8)

ndb_mgm>
```

```
make install clean
```

Edit the `/usr/local/etc/rc.d/mysql-server` file and replace

```
%%RC_SUBR%% with /etc/rc.subr
```

```
vi /usr/local/etc/rc.d/mysql-server

Edit line 22 to read:
. /etc/rc.subr
```

Create the data directory for the cluster:

```
mkdir /usr/local/mysql-cluster
```

Repeat Step 1 for each of the other 2 servers

Step 2 – Configure the Management server

Login to your chosen management server [in this example 192.168.0.30] and su to root:

```
su
vi /usr/local/etc/config.ini
```

Listing 3. *Creating a test database using the NDBCLUSTER engine*

```
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 2
Server version: 5.1.37-ndb-7.0.8a FreeBSD port: mysql-
                    cluster-7.0.8a
Type 'help;' or '\h' for help. Type '\c' to clear the
                    current input statement.
mysql> use test;
Database changed
mysql> CREATE TABLE cluster_test (i INT)
                    ENGINE=NDBCLUSTER;
Query OK, 0 rows affected (1.66 sec)
mysql> INSERT INTO cluster_test () VALUES (1);
Query OK, 1 row affected (0.04 sec)
mysql> SELECT * FROM cluster_test;
+------+
| i    |
+------+
|    1 |
+------+
1 row in set (0.00 sec)
mysql>
```

Add the following lines: see Listing 1.

Run the `ndb` management server:

```
/usr/local/libexec/ndb_mgmd -f /usr/local/etc/config.ini
```

You should see a message similar to the following:

```
2010-04-25 13:16:34 [MgmtSrvr] INFO     -- NDB Cluster
                    Management Server. mysql-5.1.37 ndb-
                    7.0.8a
2010-04-25 13:16:34 [MgmtSrvr] INFO     -- Reading cluster
                    configuration from '/usr/local/etc/
                    config.ini'
```

On each of the cluster nodes, su to root and edit `/etc/my.cnf`

```
su
```

Listing 4. *Error proving cluster has consistency*

```
Welcome to the MySQL monitor.  Commands end with ; or
                    \g.
Your MySQL connection id is 2
Server version: 5.1.37-ndb-7.0.8a FreeBSD port: mysql-
                    cluster-7.0.8a
Type 'help;' or '\h' for help. Type '\c' to clear the
                    current input statement.
mysql> USE test;
Reading table information for completion of table and
                    column names
You can turn off this feature to get a quicker startup
                    with -A
Database changed
mysql> CREATE TABLE cluster_test (i INT)
                    ENGINE=NDBCLUSTER;
ERROR 1050 (42S01): Table 'cluster_test' already
                    exists
mysql> INSERT INTO cluster_test () VALUES (1);
Query OK, 1 row affected (0.07 sec)
mysql> SELECT * FROM cluster_test;
+------+
| i    |
+------+
|    1 |
|    1 |
+------+
2 rows in set (0.00 sec)
mysql> exit
```

```
vi /usr/local/etc/my.cnf
```

Add the following:

```
[mysqld]
ndbcluster
ndb-connectstring=192.168.0.30 # the IP of the Management
                    server
[mysql_cluster]
ndb-connectstring=192.168.0.30 # the IP of the Management
                    server
```

Listing 5. *Management cluster displaying second node offline*

```
ndb_mgm> SHOW
Cluster Configuration
---------------------
[ndbd(NDB)]   2 node(s)
id=2   @192.168.0.28  (mysql-5.1.37 ndb-7.0.8,
                    Nodegroup: 0, Master)
id=3 (not connected, accepting connect from 192.168.0.29)

[ndb_mgmd(MGM)]   1 node(s)
id=1   @192.168.0.30  (mysql-5.1.37 ndb-7.0.8)

[mysqld(API)]   2 node(s)
id=4   @192.168.0.28  (mysql-5.1.37 ndb-7.0.8)
id=5   @192.168.0.29  (mysql-5.1.37 ndb-7.0.8)

ndb_mgm>
```

Listing 6. *Query returns correct results only with 1 node running*

```
mysql> USE test;
Reading table information for completion of table and
                    column names
You can turn off this feature to get a quicker startup
                    with -A
Database changed
mysql> SELECT * FROM cluster_test;
+------+
| i    |
+------+
|    1 |
|    1 |
+------+
2 rows in set (0.00 sec)

mysql>
```

Listing 7. *How to benchmark different MySQL engines. You will need to tune MySQL to get the best for your individual setup*

```
/usr/local/bin/mysqlslap --auto-generate-sql --concurrency=100 --number-of-queries=500 -engine=NDBCLUSTER

/usr/local/bin/mysqlslap --auto-generate-sql --concurrency=100 --number-of-queries=500 -engine=INNODB

/usr/local/bin/mysqlslap --auto-generate-sql --concurrency=100 --number-of-queries=500 -engine=MYISAM
```

Initialise the connection on each node:

```
/usr/local/libexec/ndbd --initial

[Node 1 - 192.168.0.28]
2010-04-25 10:05:57 [ndbd] INFO -- Configuration fetched
                    from '192.168.0.30:1186', generation: 1

/usr/local/etc/rc.d/mysql-server onestart
Starting mysql.
```

Repeat for the second node:

```
/usr/local/libexec/ndbd --initial
[Node 2 - 192.168.0.29]
2010-04-25 10:23:50 [ndbd] INFO  -- Configuration fetched
                    from '192.168.0.30:1186', generation:
                    1
/usr/local/etc/rc.d/mysql-server onestart
Starting mysql.
```

Note – The – initial switch should only be used again if the configuration changes on the management console.

Step 3 – Test and populate the database

Switch to the management console and still as root, enter the management console :

```
su
ndb_mgm
```

At the prompt, type `SHOW`, you should be greeted with see Listing 2.

If there are no entries under NDB there the modes have not started, check config.ini and `my.cnf`. If there are no entries under API, MySQL is not started on the clusters or there is a configuration issue.

Go back to the first node, and as root type the following to test the cluster: `mysql` see Listing 3.

Repeat for the second cluster node and you should get: see Listing 4. Let's see how resilient the cluster is, as root …

```
 ps -x | grep ndbd
831  ??  Is     0:00.00 /usr/local/libexec/ndbd --initial
832  ??  I      0:03.48 /usr/local/libexec/ndbd --initial
949   0  RL+    0:00.00 grep ndbd
 kill -9 831 832
```

On the management server type `SHOW` at the management console to prove we have lost a node: see Listing 5.

At the node that is currently still up, run a query: see Listing 6.

Thankfully, it looks like our data is still there. To restart the node we have shut down, type:

```
/usr/local/libexec/ndbd
```

To benchmark the different database engines: see Listing 7.

Further steps

There is a lot to improve upon with the current configuration, not least in the areas of security and the addition of load balancing to the architecture. Consideration also needs to be taken about the internal database structures, as this will also impact performance and redundancy. Finally, my

On the 'Net

- *http://dev.mysql.com/doc/refman/5.1/en/mysql-cluster.html* – Mysql website
- *http://www.alexdupre.com/mysql-cluster.tar.gz* – FreeBSD Clustering port
- *http://www.freebsd.org* – FreeBSD website

ROB SOMERVILLE

Rob Somerville has a keen passion for all things BSD / Open Source and has been working with technology since the early Eighties. His biggest claim to fame was designing an on-line search engine for a database company when 2400 Baud modems were cutting-edge. Married with 1 daughter, he shares the house with many computers, 2 cats, a dog and an extensive collection of O'Reilly books.

Shameless stock web 2.0-like art bug to get your attention.

We used the color red to grab your attention, it's an old trick.

Now, let us introduce ourselves; we're Superb Internet, and we want your business. We sell dedicated servers, hosting, rack space, etc. Oh yeah, and we have an awesome network, it's up 100% of the time.

30-Day Money-Back Guarantee SLA

Award-Winning Customer Service

100% Uptime Guarantee

Yup, awesome network map. As awesome as our real Tier-1 network, with three amazing datacenters.

Man, that istock girl is HOT! We used her photo to make you actually want to talk to us.

WWW.SUPERB.NET/NO_BS

Go here and check us out...you might like it

Superb INTERNET

BSD File sharing

Part 3. FTP

Last time I wrote on SAMBA on different BSD's. This time I am going to dedicate the article of the series to FTP.

What you will learn…

- how to set up a ftp server on OpenBSD, FreeBSD and NetBSD
- few basic facts about how ftp works
- how to adjust firewall settings when running ftp server

What you should know…

- how to use „man" command
- how to do very basic linux/unix admin routines (such as useradd, chown, chmod)

Some people do not know that the FTP protocol is the true BSD heritage, as it originated in the 1970's at Berkeley University, so it is the right thing to dedicate it some space in the BSDMag anyway.

FTP – a bit of theory

Well, FTP (*file transfer protocol*) is not really a sharing protocol in the same sense as NFS that would let you add a partition, but as it says it is a transfer protocol. So you can use it to offer your files to others, but they should first download them and then work with them. Even if clever programs such as gnome-vfs can make you think you actually *work there, on the server*. Many people would probably argue about the security of FTP and offer SFTP instead, and we intend to give space to that debate in the next contribution (about sftp and ssh in general).

Depending on what we actually want from the server there are basically two ways how to share files on a FTP server. If you want to share all files with wide public, usually only download, and you want to offer an access to a repository, pool or something similar, you can take advantage of an anonymous ftp server. While if you want to offer hosting and let each person access his/her dedicated space with a web site files, then you have to let users access your server with login, and preferably, lock them chrooted into their homes.

Active or Passive – that is the question. Another choice you have to make before starting an ftp server is whether to enable communication in the active mode (old unix default, nowadays mainly Microsoft) or passive mode (more secure, unix default). If I simplify the difference, in active mode, the server sends data on port 20 and commands on port 21, in passive mode, the server sends commands also on port 21, however opens a range of ports, usually on ports 1020 and higher. The disadvantage of active mode is that clients behind nat may have problems with file transfer and many web browsers do not support browsing FTP in active mode. The possible issue with passive mode is how to configure a firewall, that is usually done by opening a range of ports for the FTP server. A packet filter setting for example can be configured as follows, concerning FTP in passive mode:

```
fxp0="internet"
tcp_services="{ 21 1023:1060 50000: 65535 }"
```

and then consequently in the further section you will find:

```
pass in quick on $internet proto TCP from any to any port
                    $tcp_services
```

as you can see, port 21 is running and then two ranges of ports, here some people would definitely not be happy about the number of ports opened to the attacker.

Starting FTP server

There are well known ftp-server applications such as pureftpd, wu-ftpd, proftpd or vfstpd, however you do not have to install anything on FreeBSD, OpenBSD or NetBSDs since the default installs come with a native ftpd installed. The server can be started as an independent service or as a part of inetd. Advantages and disadvantages on both sides. Inetd has probably bigger overheads with big load, while when little traffic, it can save resources. Also with inetd you do not have to restart the server after each change in configuration. Personally I like to start it standalone as it is easier for me to have better control over its run. All the following examples worked for me but I warn you against simple copy and paste without knowing what you are doing at all. Read your system's documentation carefully, it is quite possible that your release, flavour or fork of BSD uses a different option, like e.g. -a instead of -A.

Starting via inet

On all BSD's add this line to the `/etc/inetd.conf`

```
ftp stream tcp  nowait root /usr/libexec/ftpd ftpd -options
```

The difference is in the options. On most BSD's the options are:

- `-A` – anonymous access allowed
- `-l` – logs access, if repeated logs more details such as transfers, etc.
- `-D` – runs as a daemon (standalone)
- `-S` – logs all anonymous access

Starting standalone

Freebsd

```
edit /etc/rc.conf:
ftpd_enable="YES"
then you can specify the options by
ftpd_flags="-your_options"
```

Netbsd

```
edit /etc/rc.conf:
ftpd="YES"
then you can specify the options by
ftpd_flags="-your_options"
```

OpenBSD

```
edit /etc/rc.conf:
ftpd_flags="-D"
```

then you can add more options to the `-D` flag.

Configuring FTP server

Configuring ftpd is no wizardry if you know where to look, here are the basics:

- `/etc/ftpwelcome` – contains a message displayed to the client before login
- `/etc/motd` – contains a message displayed to the client after login
- `/etc/ftpchroot` – lists users who have permission to login into their chrooted directories
- `/etc/ftpusers` – lists users who are restricted and may not login (to list user *root* and other privileged user here is highly recommended)
- `/etc/ftpd.conf` – contains various configuration options and fine tuning

An anonymous FTP server

Even if it is not in the headline, we run the FTP server chrooted too. It is chrooted in the directory we choose. That will eventually be the directory where all anonymous users can have access. We can restrict their privileges so that they only can read or also can write or have absolutely no access to various folders just simply by system rules using chmod.

First pick cleverly the storage, for example, `/home/ftp` because `/home` is the biggest slice with a lot of space on my PC. Remember, ftp will run chrooted so no links outside the chroot will work.

```
# groupadd -g 1000 ftp
# useradd -u 500 -g ftp -c 'anonymous FTP user' -s /sbin/
                    nologin -d /home/ftp -m ftp
```

(with some bsd's the propper nologin is `/usr/sbin/nologin`)

The above lines will add a ftp group and a ftp user with chroot specified by `-d` flag. Change it at will. If not specified, most systems will default to `/var/ftp`, which is a logical choice for servers with large `/var` slice ready for an ftp or http server. Next we have to equip the chroot with important directories and change permissions.

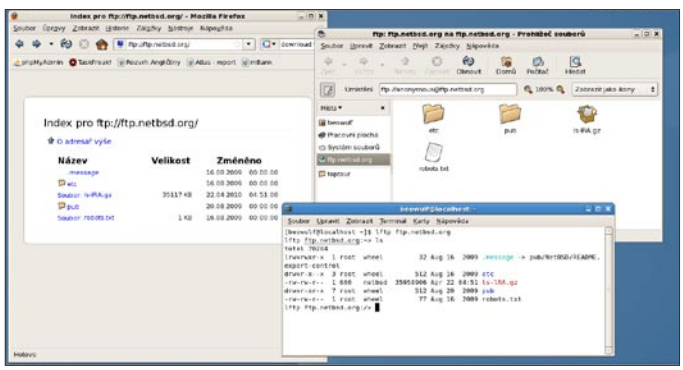

Figure 1. *Three ftp clients*

```
# cd /home/ftp
# mkdir pub
# mkdir upload
# mkdir etc
# mkdir bin
```

Now we will make sure all ftp clients will work well:

```
# cp /bin/ls bin/ls
```

And adjust permissions so that people can upload files to the proper directory:

```
# chmod 755 etc pub bin
# chmod 777 upload
```

Harden security:

```
# chown root:ftp /home/ftp etc pub bin
```

You can do much more tuning with chmod, chown and umask, but basicaly we are ready. Put some files into pub directory and people can see them and download. They can also send you files to upload directory. You can fire up the server in your BSD's flavour way, and enjoy its services, but don't forget, the proper flags to use with ftpd serving anonymous users are: -AS

Chrooted user FTP server

As you can guess, it is going to be similar. This time we will assume that you want to let users access files in their web-site directories. For example in OpenBSD the place to store web sites is `/var/www/htdocs/` and my website is called openunix. Therefore I do:

```
# groupadd -g 1000 ftp
# useradd -u 1001 -g ftp -c 'openunix website FTP user' -s /sbin/nologin \
-d /var/www/htdocs/openunix -m openunix
```

(again, depending on your flavour, check `/etc/passwd` for the proper `../nologin` location)

On the 'Net

- *http://www.tongatapu.net.to/nix/OpenBSD/ftpServer.htm*
- *http://www.bsdguides.org/guides/freebsd/networking/anonymous_ftp*
- *http://www.openbsd.org/faq/faq10.html*
- *http://www.freebsddiary.org/ftp-anonymous.php*
- http://linux-bsd-sharing.blogspot.com/2008/10/howto-setup-and-anonymous-ftp-server-on.html
- *http://slacksite.com/other/ftp.html*
- *http://www.troubleshootingnetworks.com/ftpinfo.html*

```
# passwd openunix#x0d;
New UNIX password:
Retype new UNIX password:
```

At this stage the user exists, but cannot login, so now we enable ftp access for the user:

```
# echo 'openunix'>>/etc/ftpchroot
```

Start the FTP server with `-l` flag. Of course, you can add more options. Now the users' personal space is ready and she/he can upload her/his web site files into the directory and run the web. (Needless to say it needs appropriate configuration in the `htttpd.conf`). And how to access the dedicated space on the ftp server?

Client side of FTP

The range of ftp clients is neverending. Almost all web browsers make quite good ftp clients for simple reading and download. File browsers such as nautilus or konqueror can serve you with more complex ftp transfers up and down with ease. If you fancy command line, ftp from the system is there for you, if you work a lot, then probably lftp from the ports/pkgsrc. Please do not be unhappy if I did not list your favourite filezilla or whatever here, the number of handy ftp clients is large. To give an example how different experience you get with the clients above named I include a screenshot with Firefox (left), Nautilus (right) and command line lftp in the foreground. Possible caveats on the client side can occur if the demanded server presents itself with a different IP address then it uses internaly. That is, when it is behind a nat, for example. The client can even authorize, then it goes into the passive mode and stops. The solution is to setup the client to ignore the difference between the servers external and offered IP. Configuration depends on the client you use, but many modern clients do that out of the box.

Summary

FTP is an old and well-tested way of sharing files and just works. You can run it in active or passive mode. Active is less usable for FTP clients, passive may open issues with firewalls. Anonymous access is useful for public providing access to widely-shared files, chrooted users access is better if you need to keep access to storages private. FTP has issues with security, namely with logins.

PETR TOPIARZ

Petr Topiarz is a co-owner and manager of a small language school in Prague, involved also in an EU-financed online teaching project. He started with Linux and BSD back in 2004 and since 2005 he has done the upkeep of three BSD-served networks and has becoome a great BSD fan. He runs a modest portal called openunix.eu dealing with BSD and similar issues.

HAKIN9

PRACTICAL PROTECTION

IT SECURITY MAGAZINE

WWW.HAKIN9.ORG/EN

Exploring HAMMER

One of DragonFly's features is a new file system, called HAMMER. HAMMER has, to quote from the man page, instant crash recovery, large file systems spanning multiple volumes, data integrity checking, fine-grained history retention, mirroring capability, and pseudo file systems HAMMER is available by default on DragonFly BSD.

What you will learn…

- How to locate and retrieve historical data on a HAMMER file system either in the form of individual files or as a file system snapshot.

What you should know…

- General familiarity with Unix command line navigation, understanding of how to mount volumes

While there's many new capabilities suggested by all these different features, for this article we'll just concentrate on history retention. Why history retention? If you've been using computers for any length of time, you'll recognize the feeling in the pit of your stomach that comes from staring at a screen where you just typed `rm -rf some_important_directory` and realized that was your only copy. Even worse, when you've modified a file several times and lost track of what the original looked like, or what you've done during that process.

Hardier souls reading this article are no doubt smirking and saying to themselves, *Ha! That is why I have my entire home directory stored in a version control system*. Well, good for you. The real advantage of file history retention comes when it is pervasive and continuous, not limited to a given directory that you happened to set up previously with a separate software package. Having that history built into the file system, so that it is automatic, makes a big difference.

HAMMER offers that universal coverage. As data is synced to disk, the file system keeps track of changes and maintains them on a schedule determined by the user. Because this happens at the system level and not the file level, the entire tracked file system can be recreated from history, rather than retrieving individual files. While you can retrieve previous versions of files by appending the hexidecimal transaction ID to the filename, HAMMER offers infinite snapshots for each pseudo file system on a HAMMER volume.

How to use HAMMER transaction history

Here's the scenario: you have a file that used to contain useful data. Maybe you scrambled it by overwriting it, or maybe you've made so many modifications that you've lost track of your known good configuration. Either way, you need to go back in time to see the old version of the file.

Every time a system writes data to the disk on a HAMMER filesystem, metadata is also written out that describes where and when the data is located. This metadata is used whenever an old version of the file is accessed. Once data hits the drive, it's saved. This realistically grants about 30-60 seconds of time between saves of metadata, since that's when the physical drive writes data from memory to disk.

We'll use a test case of a single file named *test.txt*, with 3 versions, each containing a single ine. The original version when first saved to disk:

```
This is the first version of the file.
```

The text after I reopened the file and modified it:

Listing 1.

```
# undo -a test.txt
test.txt: ITERATE ENTIRE HISTORY
>>> test.txt 0001 0x0000000caf2c2390 25-Apr-2010 01:23:29
This is the first version of the file.
>>> test.txt 0002 0x0000000caf2c2a30 25-Apr-2010 01:29:14
This is the second version of the file.
>>> test.txt 0003 0x0000000caf2c2c10 25-Apr-2010 01:30:02
lk;ajsf;alwkjeflkasdjfalks;djfasldkfj
```

```
This is the second version of the file.
```

... And the third, final incarnation of the file:

```
lk;ajsf;alwkjeflkasdjfalks;djfasldkfj
```

Saving all this version data is great, but it brings a problem of abundance: how do you find the data you actually want to have? The *undo* utility will list all the saved data around a file, the 64-bit transaction ID associated with it, and the actual timestamp that this data was saved.

```
# undo -i test.txt
test.txt: ITERATE ENTIRE HISTORY
        0x0000000caf2c2390 25-Apr-2010 01:23:29
        0x0000000caf2c2a30 25-Apr-2010 01:29:14
        0x0000000caf2c2c10 25-Apr-2010 01:30:02
```

Here it shows the creation of the file, modification about 6 minutes later, and then the last change within a minute later. If you just know the most recent version is the scrambled one (known as the OHCRAPOHCRAP manuver), the undo utility will default to the most recent historical version, which in this case was the second saved version on disk. You can retrieve it with undo:

```
# undo test.txt
>>> test.txt 0000 0x0000000caf2c2a30 25-Apr-2010 01:29:14
This is the second version of the file.
```

All the versions of the file can be dumped to stdout, to a separate file, or in diff format. Here's a raw dump of all the versions of this test file: see Listing 1.

If you haven't guessed yet, on that last version of the file, I randomly smashed the keyboard to simulate bad data. I use the same technique to write magazine articles.

Specifying the transaction id to the undo command will retrieve the associated copy of the file. This command will retrieve the very first version of the file saved to disk and place it in a file called *originaltest.txt*.

```
# undo -t 0x0000000caf2c2390 -o originaltest.txt test.txt
```

You can even treat it like a rough version control system, and get a unified diff of the file between any historical version and what's current. In this example, it's a diff reaching back to the original copy: see Listing 2.

History on a grand scale

Pulling out old versions of a file has been done before; there's various undelete utilities out there for a variety of file systems that will pull out whatever can be found on the disk. Having this capability at the file system level, however, adds a new level of capability to this. HAMMER, by default, takes a snapshot of a file system every 24 hours and saves 60 days worth of those snapshots.

Here's a more exact definition of what's being described: a HAMMER volume can have multiple file

Listing 2.

```
> undo -d -t 0x0000000caf2c2390 test.txt
diff -N -r -u test.txt@@0x0000000caf2c2390 test.txt (to 01-Jan-1970 00:00:00)
--- test.txt@@0x0000000caf2c2390        2010-04-25 01:23:29 +0000
+++ test.txt    2010-04-25 01:30:02 +0000
@@ -1 +1,2 @@
-This is the first version of the file.
+lk;ajsf;alwkjeflkasdjfalks;djfasldkfj
+
```

Listing 3.

```
# ls /var/hammer/usr
obj                     snap-20100316-0306      snap-20100404-0326
snap-20100224-0309      snap-20100317-0308      snap-20100405-0337
snap-20100225-0310      snap-20100318-0307      snap-20100406-0316
snap-20100226-0308      snap-20100319-0311      snap-20100408-0310
snap-20100228-0308      snap-20100320-0307      snap-20100409-0310
snap-20100301-0308      snap-20100321-0311      snap-20100412-0313
snap-20100303-0309      snap-20100322-0307      snap-20100413-0324
snap-20100304-0308      snap-20100323-0308      snap-20100414-0312
snap-20100305-0308      snap-20100324-0309      snap-20100415-0306
snap-20100306-0308      snap-20100325-0310      snap-20100416-0311
snap-20100307-0311      snap-20100326-0309      snap-20100417-0308
snap-20100308-0305      snap-20100327-0309      snap-20100418-0311
snap-20100309-0310      snap-20100328-0309      snap-20100419-0309
snap-20100310-0308      snap-20100329-0312      snap-20100420-0310
snap-20100311-0309      snap-20100330-0309      snap-20100421-0311
snap-20100312-0308      snap-20100331-0310      snap-20100422-0310
snap-20100313-0308      snap-20100401-0313      snap-20100423-0316
snap-20100314-0326      snap-20100402-0311      snap-20100424-0309
snap-20100315-0310      snap-20100403-0316
```

systems on it. Technically, the file system is HAMMER, so each mounted section is what's called a *pseudo file system*. Each one of these file systems can have different schemes for snapshots and data retention. For example, with enough disk space, `/usr` can have monthly snapshots retained for 6 months, while `/home` could be snapshotted nightly and have those snapshots kept for 2 weeks. Both of these pseudo file systems would be on the same disk.

All this history is saved in `/var/hammer`, under the name of the mounted pseudo file system. For example, here's all the available snapshots of `/usr` on a HAMMER system: see Listing 3.

Entering any of these directories will show a read-only version of the filesystem as it existed at that point in time. Files can be copied out of the directory normally, or the `mount_null` command can be used to make the entire directory available.

```
# mount_null /var/hammer/usr/snap-20100401-0313 /home/
                    aprilfools_usr
```

Snapshots can even be created on demand with the snapshot argument to the `hammer(8)` command:

```
# hammer snapshot /usr /my_snaps/usr_snaps
```

These snapshots are sparse, meaning that the space taken is only the changes to the data between versions, not the complete data from that time. The busier the disk between versions, the bigger the snapshots can be. This is why HAMMER is only recommended for disks over 50G, though it is possible to manage smaller drives using reduced amounts of file retention and very careful cleanup policies.

Final notes

HAMMER has many different features to explore. The one explored here, and of most use to a panicked sysadmin, is historical data retention. The ability to undelete files is not new to file systems in general, but when it happens automatically across an entire disk, new flexibility emerges that isn't available in any other solution. HAMMER's automatic ability to retrieve old data will metaphorically save your life, sooner or later.

A last note for the curious: even though the HAMMER name is specified in all-caps, it's not an acronym.

JUSTIN SHERRILL

Justin Sherrill has been posting daily DragonFly news to the DragonFly Digest at http://www.shiningsilence.com/dbsdlog/ since 2004. He works in a mine and lives in the northeast United States.

BSDCan 2010

The Technical BSD Conference

High value. Low cost. Something for everyone.

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 2010 will be held on 13-14 May 2010 at University of Ottawa, and will be preceded by two days of Tutorials on 11-12 May 2010.

There will be related events (of a social nature, for the most part) on the day before and after the conference.

http://bsdcan.org/

BSDCan 2010

Embedded OpenBSD

Unix-like operating systems aren't picky at all. Despite the extreme physical conditions, they can take root on those old computers where most (proprietary) operating systems risk extinction and help them, after years of faithful service, to start new lives as firewalls, routers, proxies...

What you will learn…

- Different way of how to install the operating system

What you should know…

- Working knowledge on openBSD
- Some experience with embedded computers and flash memory cards

But sometimes this is not enough: servers must be reliable and old computers are (guess what?) *...old*, and this increases their risk of disease. That's why embedded systems are a great option: they are (relatively) inexpensive, silent, small, reliable... What else could you need?

Ok, you have to learn to cohabit with very basic hardware, but the right OS, with the right configuration, will wallow in it!

The use of these computers ranges from firewalls (*http://www.kernel-panic.it/openbsd/carp/index.html*) to access points, to VPN servers (*http://www.kernel-panic.it/openbsd/vpn/index.html*) and so on; what characterizes them is their minimal hardware configuration (especially the small amount of disk space) which may make the installation procedure a bit unusual and custom. However, post-installation configuration is absolutely normal; that's why, throughout this document, we will only focus on the main methods to enclose our favourite OS in those few inches of integrated circuits.

The basic tools we will use are:

- OpenBSD (*http://www.openbsd.org/*) – the *secure by default* (*http://www.openbsd.org/security.html#default*) operating system, particularly well suited for *ultra-light* installations and security-critical applications;
- an embedded computer – to be precise a net4521 (*http://www.soekris.com/net4521.htm*) board (in the picture), manufactured by Soekris Engineering, Inc (*http://www.soekris.com/*). WRAP (*http://www.pcengines.ch/wrap.htm*) and ALIX (*http://www.pcengines.ch/alix.htm*) boards, by PC Engines GmbH (*http://www.pcengines.ch/*), are a great option too;
- a 64MB Compact Flash memory card – used as mass memory; some embedded computers also support 2.5" disks, but all examples can be easily extended to them.

Figure 1. *64MB Compact Flash Memory Card*

A solid knowledge of OpenBSD is assumed, since we will have to go through building a custom kernel (and we won't dwell on this topic too long) and finding out all the configuration, startup and executable files strictly necessary to build a minimal, yet fully functional, system.

Installation modes

There are many ways to install the operating system, each with its own peculiarities and, therefore, best suited for different situations and needs:

- using an installation script (like BowlFish *http://www.kernel-panic.it/software/bowlfish/*) is very easy and will let you install, in a few minutes, a deeply customized OpenBSD system. However, if you're reading these lines, you probably prefer having full control over the installation process; therefore, we won't examine this installation procedure here;
- writing directly to disk will let you fully customize the system, using minimal disk space (the system largely fits into a 32 MB compact flash card). However, it requires a good knowledge of the operating system, which must be built file by file;
- diskless installation, mounting the entire filesystem through NFS, makes you save the money of mass memory and allows you to simplify and centralize maintenance; on the other hand, it requires a more complex network configuration and the setup of additional servers (PXE, NFS...);
- network installation requires a non-trivial configuration too (PXE server) and is much more difficult to customize, being, after all, a standard installation. Therefore, this is probably the best option if you have enough disk space (256MB CF or 2.5" disk);

In any case, if you use a Compact Flash card as mass memory, keep in mind that it has a limited number of write cycles and therefore must be mounted read-only. Logging or swapping to it would quickly render it unusable. The most common configuration is to mount the whole filesystem read-only, except for the `/tmp`, `/root` and `/var` directories, which are mapped to memory. Anyway, this doesn't mean you won't be able to make changes to the filesystem, but only that every time you will need to edit a file on the disk you will have to first mount it read-write:

```
# mount -o rw,noatime /dev/wd0a /
```

and then rememeber to mount it back read-only when you're done.

```
# mount -o ro /dev/wd0a /
```

Installing directly to disk

Before delving into the inners of the installation, we need to retrieve the disk geometry values, which, as we will see, will come in handy more than once. To get this information, insert the Compact Flash card into its socket, attach to the device's serial console with a null modem cable, connect with `cu(1)` (*http://www.openbsd.org/cgi-bin/man.cgi?query=cu&sektion=1*) and power the system up. You should get something like: see Listing 1.

The numbers 490, 8 and 32 are, respectively, the number of cylinders, heads (i.e. tracks per cylinder) and sectors (per track) of the disk.

Ok, now let the fun begin! We will create a bootable filesystem on the flash card and copy the files we need from the OS. To make fewer write operations on the memory card, the best thing is to create a disk-image file of the size of the CF card (see `vnd(4)` *http://www.openbsd.org/cgi-bin/man.cgi?query=vnd&sektion=4* for details) and eventually copy it to the device. To create a 64MB virtual disk image, type:

```
# dd if=/dev/zero of=net4521.img bs=512 count=125440
125440+0 records in
125440+0 records out
64225280 bytes transferred in 1.399 secs (45875823 bytes/
                    sec)
# vnconfig -c svnd0 net4521.img
```

The `bs` parameter sets the block (sector) size (usually 512 bytes), and `count` the number of sectors, obtained by multiplying the disk geometry values (32 * 8 * 490). *Note*: if you want to write directly to the disk, without bothering with the virtual disk image, simply replace `svnd0` with

Listing 1. *Instaling directly to disc*

```
# cu -s 19200 -l cua00

comBIOS ver. 1.26a  20040819  Copyright (C) 2000-2004
                    Soekris Engineering.

net45xx

0064 Mbyte Memory                       CPU 80486 133 Mhz

Pri Mas  SanDisk SDCFB-64       LBA 490-8-32  62 Mbyte
[...]
```

the appropriate disk drive (e.g. `sd0`) in the subsequent examples.

After creating the virtual disk, we need to `disklabel(8)` (*http://www.openbsd.org/cgi-bin/man.cgi?query=disklabel&sektion=8*) it, build the filesystem and make it bootable; but to fully understand these steps, we must first discuss how OpenBSD boots on the i386 architecture. So let's take a look, in parallel, at the boot process and how it reflects upon our installation procedure (for more information, please refer to [FAQ14 *http://www.openbsd.org/faq/faq14.html#Boot386*]).

Master Boot Record

The Master Boot Record is the first physical sector (512 bytes) on the disk; it is loaded by the BIOS after the POST and it contains the primary partition table (*Master Partition Table*) and a small program (Master Boot Code) to load the Partition Boot Record (see First and second stage boot loaders).

OpenBSD provides a *MBR template file* (`/usr/mdec/mbr`) which we can install with `fdisk(8)` (*http://www.openbsd.org/cgi-bin/man.cgi?query=fdisk&sektion=8*):

```
# fdisk -c 490 -h 8 -s 32 -iyf /usr/mdec/mbr svnd0
Writing MBR at offset 0.
#
```

We need to specify the disk geometry (we have seen before how to retrieve these data) because we're not installing directly to the disk now, but to a virtual disk image. *Note*: if the OpenBSD release you're installing from is not the same as the release you're installing, you can extract the `mbr` file from the `baseXX.tgz` file set.

First and second stage boot loaders

Next, the OpenBSD boot process goes through two stages:

- in the first stage, the MBR loads the PBR (*Partition Boot Record* or first-stage boot loader), which is the first physical sector (512 bytes) on the OpenBSD primary partition. It contains a small program, `biosboot(8)` (*http://www.openbsd.org/cgi-bin/man.cgi?query=biosboot&sektion=8*), which has the task of loading the second-stage boot loader (`/boot`);
- in the second stage, `/boot`, the second-stage boot loader, has the task of accessing the OpenBSD file system through the machine's BIOS, and locating and loading the actual kernel.

Before installing the boot loaders, we need to create the `disklabel(5)` (*http://www.openbsd.org/cgi-bin/man.cgi?query=disklabel&sektion=5*), which contains detailed information about disk geometry and partitions and acts as an interface between the disk and the disk drivers contained within the kernel. The `disklabel(8)` (*http://www.openbsd.org/cgi-bin/man.cgi?query=disklabel&sektion=8*) utility allows you to write the label on the disk (once again, disk geometry information will come in handy): see Listing 2.

We have created only a single `/` partition: swapping on the compact flash is obviously strongly discouraged! Now we can build the filesystem: see Listing 3.

Mount it and install the two boot loaders with the `installboot(8)` (*http://www.openbsd.org/cgi-bin/man.cgi?query=installboot&sektion=8*) command:

Listing 2. *Writing a lable on the disc*

```
# disklabel -E svnd0
Label editor (enter '?' for help at any prompt)
> e
Changing device parameters for /dev/rsvnd0c:
disk type: [vnd] ESDI
label name: [fictitious] net4521
sectors/track: [100] 32
tracks/cylinder: [1] 8
sectors/cylinder: [100] 256
number of cylinders: [1254] 490
total sectors: [125440] <enter>
rpm: [3600] <enter>
interleave: [1] <enter>
> a a
offset: [0] 63
size: [125377] <enter>
FS type: [4.2BSD] <enter>
> q
Write new label?: [y] y
#
```

Listing 3. *Building the file system*

```
# newfs -S 512 /dev/rsvnd0a
newfs: reduced number of fragments per cylinder group
                    from 7832 to 7792 to enlarge last
                    cylinder group
/dev/rsvnd0a: 61.2MB in 125376 sectors of 512 bytes
5 cylinder groups of 15.22MB, 974 blocks, 2048 inodes
                    each
super-block backups (for fsck -b #) at:
 32, 31200, 62368, 93536, 124704,
#
```

Listing 4. *Building the custom kernel. Configuration file*

```
/usr/src/sys/arch/i386/conf/NET4521
# OpenBSD config file for Soekris net4521 embedded system

machine         i386                    # architecture,
                    used by config; REQUIRED
option          I486_CPU

# Operation Related Options
option          DUMMY_NOPS              # speed hack;
                    recommended

# Debugging Options
option          DDB

# Filesystem Options
option          FFS
option          MFS
option          NFSCLIENT
option          FDESC
option          FIFO

# Miscellaneous Options
option          PCIVERBOSE
option          CRYPTO
option          TIMER_FREQ=1189161
option          PCCOMCONSOLE
option          CONSPEED=19200

# Networking Options
option          INET
option          INET6
option          TCP_SACK
option          TCP_FACK
option          TCP_SIGNATURE
option          IPSEC
option          KEY
option          ALTQ
option          ALTQ_NOPCC

maxusers        5                       # estimated
                    number of users
config          bsd root on wd0a

mainbus0        at root

cpu0            at mainbus?
bios0           at mainbus0
pcibios0        at bios0 flags 0x0000   # use 0x30 for
                    a total verbose

isa0            at mainbus0
pci*            at mainbus0

# power management and other environmental stuff
elansc*         at pci?                 # AMD Elan
                    SC520 System Controller
gpio*           at elansc?

# CardBus bus support
cardbus*        at cardslot?
pcmcia*         at cardslot?
cbb*            at pci?
cardslot*       at cbb?

npx0            at isa? port 0xf0 irq 13        # math
                    coprocessor
isadma0         at isa?

pccom0          at isa? port 0x3f8 irq 4        #
                    standard PC serial ports
pccom1          at isa? port 0x2f8 irq 3

# IDE
wdc0            at isa? port 0x1f0 irq 14 flags 0x00
                    # WD100x compatible hard disk
                    controller driver
wd*             at wdc? flags 0x0000    # WD100x
                    compatible hard disk driver

# Networking devices
sis*            at pci?                 # SiS 900/7016
                    ethernet Fast Ethernet driver
nsphyter*       at mii? phy ?           # NS and
                    compatible PHYs

# Wireless network cards
wi*             at pcmcia?              # PRISM 2-3
                    wireless network driver

# Pseudo-devices
pseudo-device   mtrr            1       # driver for
                    CPU memory range attributes
pseudo-device   nvram           1       # driver for
                    reading PC NVRAM contents
```

Listing 4. *Building the custom kernel. Configuration file*

```
pseudo-device   bio             1       # ioctl tunnel pseudo-device
pseudo-device   hotplug         1       # devices hot plugging

pseudo-device   ksyms           1       # kernel symbol table device
pseudo-device   systrace        1       # enforce and generate policies for system calls

pseudo-device   pf                      # Packet filter
pseudo-device   pflog                  # Packet filter logging interface
pseudo-device   pfsync                  # Packet filter state table logging interface
pseudo-device   loop            2       # Loopback
pseudo-device   bpfilter        16      # Berkeley Packet Filter
pseudo-device   tun             2       # Network tunnel pseudo-device
pseudo-device   enc             1       # IPSEC encapsulating Interface
pseudo-device   bridge          2       # Ethernet bridge interface
pseudo-device   vlan            32      # IEEE 802.1Q encapsulation/decapsulation pseudo-device
pseudo-device   gre             4       # GRE encapsulating network device
pseudo-device   pty             32      # Pseudo-terminals
pseudo-device   gif             4       # Generic tunnel interface
```

Listing 5. *Building the custom kernel. Installation*

```
# cd /usr/src/sys/arch/i386/conf
# config NET4521
Don't forget to run "make depend"
# cd ../compile/NET4521
# make clean && make depend && make
[...]
# cp bsd /mnt/net4521/
```

Listing 6. *Populating the file system*

```
# mkdir /mnt/net4521/tmp{late,}
# ln -s /tmp/{var,root} /mnt/net4521/
# mkdir -p /mnt/net4521/tmplate/var/cron/{tabs,atjobs}
# chmod 555 /mnt/net4521/tmplate/var/cron
# chmod 1770 /mnt/net4521/tmplate/var/cron/atjobs
# chmod 1730 /mnt/net4521/tmplate/var/cron/cron
# mkdir -p /mnt/net4521/tmplate/root
# chmod 700 /mnt/net4521/tmplate/root
```

Listing 7. *Generating users configuration files*

```
# echo "root:$(encrypt -b 8 mypasswd):0:0:daemon:0:0:Charlie " \
> "&,,,:/root:/bin/ksh" >> /mnt/net4521/etc/master.passwd
# echo "wheel:*:0:root" >> /mnt/net4521/etc/group
# pwd_mkdb -d /mnt/net4521/etc /mnt/net4521/etc/master.passwd
```

Listing 8. *Creating log files*

```
# mkdir -p /mnt/net4521/tmplate/var/{log,run/dev}
# touch /mnt/net4521/tmplate/var/log/{authlog,daemon,mess
                    ages,secure}
# touch /mnt/net4521/tmplate/var/run/utmp
# chmod 640 /mnt/net4521/tmplate/var/log/{authlog,daemon}
# chmod 600 /mnt/net4521/tmplate/var/log/secure
# chmod 664 /mnt/net4521/tmplate/var/run/utmp
```

```
# mount /dev/svnd0a /mnt/net4521
# cp /usr/mdec/boot /mnt/net4521/
# /usr/mdec/installboot /mnt/net4521/boot /usr/mdec/
                    biosboot svnd0
#
```

We can now set up some boot parameters in the /etc/boot.conf(5) (*http://www.openbsd.org/cgi-bin/man.cgi?query=boot.conf&sektion=5*) configuration file. We will use it to set up the serial console, which has a default baud rate of 19200 (or 38400 for WRAP and ALIX boards):

```
/mnt/net4521/etc/boot.conf
set tty com0
stty com0 19200
```

Building a custom kernel

Now that the disk is ready, we only have to populate it. Let's start with the kernel, for which we have two options: if the CF card is not too small, the easy and smooth (and recommended) solution is copying the standard bsd kernel to it:

```
# cp /bsd /mnt/net4521/
```

Or else, if you want the kernel to be smaller and faster at boot time, you can build a custom kernel with only the bare minimum features. The following is a sample configuration file suitable for the latter case: see Listing 4.

So let's build the kernel and install it: see Listing 5.

Populating the filesystem

Next we will create the necessary configuration files in /etc (well, for the moment /mnt/net4521/etc/) We will only see the main ones here: a comprehensive list would depend too much on the use of the device.

- /etc/fstab(5) (*http://www.openbsd.org/cgi-bin/man.cgi?query=fstab&sektion=5*) this file contains information about the filesystems.

```
/mnt/net4521/etc/fstab
/dev/wd0a       /       ffs     ro                      1 1
swap            /tmp    mfs     rw,nosuid,-P=/tmplate,-
                    s=16384   0 0
```

As stated before, we map the /tmp filesystem to memory. /var and /root, which must be read-write, will be symbolic links to /tmp/var. We will also create a /tmplate directory containing the directory tree which mount_mfs(8) (*http://www.openbsd.org/cgi-bin/man.cgi?query=mount_mfs&sektion=8*) will use to populate /tmp after its creation (we will put pseudo-devices and files required by syslogd(8) (*http://www.openbsd.org/cgi-bin/man.cgi?query=syslogd&sektion=8*) into this directory later); see Listing 6.

- network configuration files – /etc/hosts(5) (*http://www.openbsd.org/cgi-bin/man.cgi?query=hosts&sektion=5* host name database), /etc/hostname.if(5) (*http://www.openbsd.org/cgi-bin/man.cgi?query=hostname.if&sektion=5* interface-specific configuration files), /etc/myname(5) (*http://www.openbsd.org/cgi-bin/man.cgi?query=myname&sektion=5* default hostname), /etc/mygate(5) (*http://www.openbsd.org/cgi-bin/man.cgi?query=mygate&sektion=5* default gateway), /etc/resolv.conf(5) (*http://www.openbsd.org/cgi-bin/man.cgi?query=resolv.conf&sektion=5* resolver configuration file);
- users configuration files – /etc/group(5) (*http://www.openbsd.org/cgi-bin/man.cgi?query=group&sektion=5* group permissions file) and /etc/master.passwd(5) (*http://www.openbsd.org/cgi-bin/man.cgi?query=master.passwd&sektion=5* password file); the other files (/etc/passwd(5) *http://www.openbsd.org/cgi-bin/man.cgi?query=passwd&sektion=5*, /etc/pwd.db, /etc/spwd.db) will be generated by the pwd_mkdb(8) (*http://www.openbsd.org/cgi-bin/man.cgi?query=pwd_mkdb&sektion=8*) command: see Listing 7.

Feel free to add all the system and administrative users and groups you will need. If you want to use sudo(8) (*http://www.openbsd.org/cgi-bin/man.cgi?query=sudo&sektion=8*), which is usually a good idea, you need to create the sudoers(5) (*http://www.openbsd.org/cgi-bin/man.cgi?query=sudoers&sektion=5*) file (using the visudo -f /mnt/net4521/etc/sudoers *http://www.openbsd.org/cgi-bin/man.cgi?query=visudo&sektion=8* command);

- pf(4) (*http://www.openbsd.org/cgi-bin/man.cgi?query=pf&sektion=4*) configuration files /etc/pf.conf(5) (*http://www.openbsd.org/cgi-bin/man.cgi?query=pf.conf&sektion=5* configuration and rules) e /etc/pf.os(5) (*http://www.openbsd.org/cgi-bin/man.cgi?query=pf.os&sektion=5* OS fingerprints);
- ssh(1) (*http://www.openbsd.org/cgi-bin/man.cgi?query=ssh&sektion=1*) configuration files /etc/ssh/ssh_config (*http://www.openbsd.org/cgi-bin/man.cgi?query=ssh_config&sektion=5* SSH client configuration file), /etc/ssh/sshd_config (*http://www.openbsd.org/cgi-bin/man.cgi?query=sshd_config&sektion=5* SSH daemon configuration file),

/etc/moduli(5) (*http://www.openbsd.org/cgi-bin/man.cgi?query=moduli&sektion=5* system Diffie-Hellman moduli file). We can also generate the host private keys right now:

```
# ssh-keygen -t rsa -f /mnt/net4521/etc/ssh/ssh_host_rsa_
                    key -N ""
# ssh-keygen -t rsa1 -f /mnt/net4521/etc/ssh/ssh_host_key
                    -N ""
# ssh-keygen -t dsa -f /mnt/net4521/etc/ssh/ssh_host_dsa_
                    key -N ""
```

- /etc/syslog.conf(5) (*http://www.openbsd.org/cgi-bin/man.cgi?query=syslog.conf&sektion=5*)

Containing syslogd(8) (*http://www.openbsd.org/cgi-bin/man.cgi?query=syslogd&sektion=8*) configuration. All log files have to be created (a touch(1) (*http://www.openbsd.org/cgi-bin/man.cgi?query=touch&sektion=1*) will suffice), otherwise syslogd(8) (*http://www.openbsd.org/cgi-bin/man.cgi?query=syslogd&sektion=8*) will complain on boot: see Listing 8.

You may schedule newsyslog(8) (*http://www.openbsd.org/cgi-bin/man.cgi?query=newsyslog&sektion=8*) to periodically archive log files:

```
# echo "0 * * * *  /usr/bin/newsyslog" > \
> /mnt/net4521/tmplate/var/cron/tabs/root
# chmod 600 /mnt/net4521/tmplate/var/cron/tabs/root
```

Anyway, since /var will reside on volatile memory, it is recommended to forward log messages to a remote log host;

- /etc/ttys(5) (*http://www.openbsd.org/cgi-bin/man.cgi?query=ttys&sektion=5*) the terminal initialization file, modified according to our configuration: see Listing 9.
- /etc/sysctl.conf(5) (*http://www.openbsd.org/cgi-bin/man.cgi?query=sysctl.conf&sektion=5*) containing sysctl(8) (*http://www.openbsd.org/cgi-bin/man.cgi?query=sysctl&sektion=8*) variables to set at system startup; e.g.:

```
/mnt/net4521/etc/sysctl.conf
net.inet.ip.forwarding=1
[...]
```

Next, we need to copy the startup scripts (rc(8), rc.local(8), rc.securelevel(8), rc.conf(8), rc.conf.local(8), rc.shutdown(8), netstart(8) (*http://www.openbsd.org/cgi-bin/man.cgi?query=netstart&sektion=8*)) and create the device files: see Listing 10.

Note: rc(8) (*http://www.openbsd.org/cgi-bin/man.cgi?query=rc&sektion=8*) clears the /tmp directory on boot, thus removing the contents of the /var and /root directories; therefore, I would recommend that you delete the following lines from /mnt/net4521/etc/rc (*http://www.openbsd.org/cgi-bin/man.cgi?query=rc&sektion=8*):

```
/mnt/net4521/etc/rc
(cd /tmp && rm -rf [a-km-pr-zA-Z]* &&
    find . ! -name . ! -name lost+found ! -name
                quota.user \
        ! -name quota.group -execdir rm -rf -- {} \;
                -type d -prune)
```

Listing 9. *The terminal initialization file, modified according to configuration*

```
/mnt/net4521/etc/ttys
console "/usr/libexec/getty Pc"         vt220   off secure
ttyC0   "/usr/libexec/getty Pc"         vt220   off secure
ttyC1   "/usr/libexec/getty Pc"         vt220   off secure
ttyC2   "/usr/libexec/getty Pc"         vt220   off secure
ttyC3   "/usr/libexec/getty Pc"         vt220   off secure
tty00   "/usr/libexec/getty std.19200"  vt100   on  secure
tty01   "/usr/libexec/getty std.9600"   unknown off
tty02   "/usr/libexec/getty std.9600"   unknown off
tty03   "/usr/libexec/getty std.9600"   unknown off
ttyp0   none                            network
ttyp1   none                            network
ttyp2   none                            network
[...]
```

Listing 10. *Copying the start up scripts and creating the device files*

```
# mkdir /mnt/net4521/dev/
# cp /dev/MAKEDEV /mnt/net4521/dev/
# cd /mnt/net4521/dev/
# ./MAKEDEV tun0 tun1 tun2 tun3 bpf0 bpf1 bpf2 bpf3 bpf4 bpf5
                    bpf6 bpf7 bpf8   \
> bpf9 fd1 fd1B fd1C fd1D fd1E fd1F fd1G fd1H fd0 fd0B fd0C
                    fd0D fd0E fd0F     \
> fd0G fd0H random crypto pf pctr systrace sd0 sd1 sd2 sd3
                    sd4 wd0 wd1 wd2 wd3 \
> fd tty00 tty01 tty02 tty03 ttyc0 ttyc1 ttyc2 ttyc3 apm std
#
```

/dev/log, used by syslogd(8) (*http://www.openbsd.org/cgi-bin/man.cgi?query=syslogd&sektion=8*), must be writable: therefore, we turn it into a symlink to /var/run/dev/log. The same applies to pseudo terminals, which must be able to change owner and permissions: see Listing 11.

Finally, we can install binaries and libraries. The simplest way is copying them from the system currently in use, or you may extract them from the installation file set (baseXX.tgz). To save some time, you can create a file with the list of the binaries to copy (a good starting point is flashsmall.txt from flashdist *http://www.nmedia.net/~chris/soekris/*): see Listing 12.

If you wish to further decrease the binaries disk space, you can take a look at crunchgen(8) (*http://www.openbsd.org/cgi-bin/man.cgi?query=crunchgen&sektion=8*), which builds them all in a single binary file which modifies its behaviour according to argv[0], or remove the debugging symbols from the shared libraries using the strip(1) (*http://www.openbsd.org/cgi-bin/man.cgi?query=strip&sektion=1*) command:

```
# strip -S /mnt/net4521/usr/lib/lib*
```

Now we only have to transfer the virtual filesystem we have created to the memory card:

```
# umount /mnt/net4521
# vnconfig -u svnd0
# dd if=net4521.img of=/dev/sd0c bs=512
125440+0 records in
125440+0 records out
64225280 bytes transferred in 383.307 secs (167556 bytes/
                    sec)
#
```

Plug the compact flash into the device, power it up and ...uncork the champagne!

Diskless installation

Creating an embedded system with no mass memory offers several benefits:

- no need to use compact flash cards or 2.5" disks anymore, thus saving a little money;
- by using NFS, you will probably have larger disks available;
- you can centrally manage disks;
- you can share filesystems (usually /usr, which rarely changes) among multiple hosts, thus making maintenance and upgrading easier and faster;

But there are also some drawbacks:

- a new server becomes necessary, to provide all services needed to boot the devices;
- on security-critical systems, like firewalls, using NFS is often a poor option;
- boot server configuration may not be trivial.

So let's get to the configuration! We need to set up a boot server, on which most of the installation will take place; all we need from the embedded device is its MAC address. To get it, you just have to attach to the console and power it up: see Listing 13.

We will now take a look at how to compile a diskless kernel, and then step through the system boot process to understand which network services we will need to set up on the boot server.

Building a custom kernel

Everything we have seen before about kernel configuration and compiling still applies; just make sure you specify, in the configuration file, that the system must look for the root and swap filesystems on NFS:

Listing 11. *Pseudo terminals which must be able to change owner and permissions*

```
# ln -s /var/run/dev/log /mnt/net4521/dev/log
# cd /mnt/net4521/tmplate/var/run/dev/
# /dev/MAKEDEV pty
# for dev in [tp]typ?; do
>     ln -s /var/run/dev/$dev /mnt/net4521/dev/$dev
> done
#
```

Listing 12. *Creating a file with the list of binaries to copy*

```
# tar -I bin_list.txt -cf - | tar -C /mnt/net4521/ -xpf -
tar: Removing leading / from absolute path names in the
                    archive
# while read file; do
>     ldd $file 2>/dev/null | egrep 'rlib|rtld' | awk
                    '{ print $7 }'
> done < bin_list.txt | sort -u | xargs tar -cvf - |
                    tar -C /mnt/net4521/ -xpf -
tar: Removing leading / from absolute path names in the
                    archive
[...]
#
```

```
/usr/src/sys/arch/i386/conf/NET4521
[...]
config          bsd root on nfs swap on nfs
[...]
```

rarpd(8)

On boot, the device first tries to configure its network settings. Since it only knows its MAC address, it generates a RARP request to get an IP address. Therefore, we must enable the `rarpd(8)` (*http://www.openbsd.org/cgi-bin/man.cgi?query=rarpd&sektion=8*) daemon in the boot server's `/etc/rc.conf.local(8)` file:

```
/etc/rc.conf.local
rarpd_flags="-a"
```

If you don't want the daemon to listen on all the interfaces, just replace the `-a` parameter with the name of the interface to listen on. To honour RARP requests, the daemon uses two files:

- `/etc/ethers(5)` (*http://www.openbsd.org/cgi-bin/man.cgi?query=ethers&sektion=5*) which maps ethernet addresses to host names:

```
/etc/ethers
00:00:24:c3:c1:b0       net4521.kernel-panic.it
```

- `/etc/hosts(5)` (*http://www.openbsd.org/cgi-bin/man.cgi?query=hosts&sektion=5*), which maps IP addresses to host names:

Listing 13. *Attaching a console and powering it up*

```
# cu -s 19200 -l cua00

comBIOS ver. 1.26a  20040819  Copyright (C) 2000-2004
                    Soekris Engineering.

net45xx

0064 Mbyte Memory                          CPU 80486 133
                    Mhz


Slot   Vend Dev  ClassRev Cmd  Stat CL LT HT  Base1
                    Base2   Int
-------------------------------------------------------
                    ------------
0:00:0 1022 3000 06000000 0006 2280 00 00 00 00000000
                    00000000
0:17:0 104C AC51 06070000 0107 0210 10 3F 82 A0000000
                    020000A0 10
0:17:1 104C AC51 06070000 0107 0210 10 3F 82 A0001000
                    020000A0 10
0:18:0 100B 0020 02000000 0107 0290 00 3F 00 0000E101
                    A0002000 11
0:19:0 100B 0020 02000000 0107 0290 00 3F 00 0000E201
                    A0003000 05

 1 Seconds to automatic boot.   Press Ctrl-P for
                    entering Monitor.

NSC DP83815/DP83816 Fast Ethernet UNDI, v1.03
Copyright (C) 2002, 2003 National Semiconductor
                    Corporation
All rights reserved.

Pre-boot eXecution Environment  PXE-2.0 (build 082)
Copyright (C) 1997-2000  Intel Corporation


CLIENT MAC ADDR: 00 00 24 C3 C1 B0
[...]
```

Listing *14. Enabling a daemon in the boot service*

```
/etc/rc.conf.local
dhcpd_flags=""
and configure it:
/etc/dhcpd.conf
[...]
# Diskless devices group
group {
        filename "pxeboot";             # Boot file
        #next-server  pxe-server;       # PXE server
                    (if different from the DHCP server)

        host net4521 { hardware ethernet 00:00:c8:c1:
                    24:57; }
}
[...]
```

```
/etc/hosts
172.16.0.10             net4521.kernel-panic.it
```

If the requesting host does not exist in both files, the daemon won't be able to send a reply.

dhcpd(8)

Now that it has got its own IP address, the embedded device will look for the boot file. To get the file name, it will send a DHCP request, to which our server will be glad to reply. Therefore, we need to enable the `dhcpd(8)` (*http://www.openbsd.org/cgi-bin/man.cgi?query=dhcpd&sektion=8*) daemon in the boot server's `/etc/rc.conf.local(8)` (*http://www.openbsd.org/cgi-bin/man.cgi?query=rc.conf.local&sektion=8*) file: see Listing 14.

tftpd(8)

Ok, now that it knows the name of the boot file, the diskless device will attempt to download it, via `tftp(1)` (*http://www.openbsd.org/cgi-bin/man.cgi?query=tftp&sektion=1*), from the server in the *next-server* parameter or from the DHCP server itself. To enable `tftpd(8)` (*http://www.openbsd.org/cgi-bin/man.cgi?query=tftpd&sektion=8*) on our boot server, we need to uncomment the following line in `/etc/inetd.conf(8)` (*http://www.openbsd.org/cgi-bin/man.cgi?query=inetd&sektion=8*):

```
/etc/inetd.conf
tftp            dgram   udp     wait    root    /usr/
                    libexec/tftpd       tftpd -s /tftpboot
```

create the `/tftpboot` directory and populate it with the appropriate files: `pxeboot(8)` (*http://www.openbsd.org/cgi-bin/man.cgi?query=pxeboot&sektion=8* the second-stage PXE boot loader), bsd (the custom kernel) and `/tftpboot/etc/boot.conf(8)` (*http://www.openbsd.org/cgi-bin/man.cgi?query=boot&sektion=8*), which contains the boot parameters:

```
/tftpboot/etc/boot.conf
set tty com0
stty com0 19200
```

bootparamd(8)

Now the system will boot, until it needs to mount the NFS filesystems. To find them out, it will broadcast a BOOTPARAMS request, waiting for some `rpc.bootparamd(8)` daemon to tell it the parameters of the NFS filesystems to mount. Therefore, we need to start the `bootparamd(8)` daemon on our server. Once again, we have to edit a couple of variables in `/etc/rc.conf.local(8)`:

```
/etc/rc.conf.local
bootparamd_flags=""
portmap="YES"
```

As you can see, to make `bootparamd(8)` work, we need to start the `portmap(8)` daemon too, which converts RPC program numbers into DARPA protocol port numbers. `bootparamd(8)` has its own configuration file, `/etc/bootparams(5)`, which must contain an entry for each client, specifying the pathnames for its root and (optionally) swap areas (fields are delimited with blank or tab, and entries may span across multiple lines using a back-slash):

```
/etc/bootparams
net4521 root=boot-srv:/exports/net4521/root/ \
        swap=boot-srv:/exports/net4521/swap
```

nfs

The last step to complete the boot process is to mount the NFS filesystems. Therefore, we must set up the NFS server; let's edit the `/etc/rc.conf.local(8)` file once again to set a couple of variables:

```
/etc/rc.conf.local
nfs_server="YES"
nfsd_flags="-tun 4"
```

and set up the filesystems to mount:

- `/exports/net4521/root`, the directory that will contain the whole filesystem of the embedded device (except `/usr`: in fact, if the systems have the same architecture, the server can save a lot of disk space exporting its own `/usr` directory); we have seen before how to populate the filesystem;
- `/exports/net4521/swap`, the file that will contain the system's swap area; you can build it by running:

```
# dd if=/dev/zero of=/exports/net4521/swap  bs=1m
                    count=128
```

which creates a 128MB swap file.

On the NFS server, the `/etc/exports(5)` file lists the exported filesystems and sets the hosts and export options for each one:

```
/etc/exports
/usr -ro 172.16.0.10
/export/net4521 -maproot=root -alldirs 172.16.0.10
```

Listing *15. Normal boot and installation proces*

```
# cu -l cua00 -s 19200

comBIOS ver. 1.26a  20040819  Copyright (C) 2000-2004 Soekris Engineering.

net45xx

0064 Mbyte Memory                          CPU 80486 133 Mhz


Slot   Vend Dev  ClassRev Cmd  Stat CL LT HT  Base1    Base2   Int
------------------------------------------------------------------
0:00:0 1022 3000 06000000 0006 2280 00 00 00 00000000 00000000
0:17:0 104C AC51 06070000 0107 0210 10 3F 82 A0000000 020000A0 10
0:17:1 104C AC51 06070000 0107 0210 10 3F 82 A0001000 020000A0 10
0:18:0 100B 0020 02000000 0107 0290 00 3F 00 0000E101 A0002000 11
0:19:0 100B 0020 02000000 0107 0290 00 3F 00 0000E201 A0003000 05

 5 Seconds to automatic boot.   Press Ctrl-P for entering Monitor.

comBIOS Monitor.   Press ? for help.

> boot F0

NSC DP83815/DP83816 Fast Ethernet UNDI, v1.03
Copyright (C) 2002, 2003 National Semiconductor Corporation
All rights reserved.

Pre-boot eXecution Environment  PXE-2.0 (build 082)
Copyright (C) 1997-2000  Intel Corporation


CLIENT MAC ADDR: 00 00 24 C3 C1 B0
CLIENT IP: 172.16.0.10  MASK: 255.255.255.0  DHCP IP: 172.16.0.4
GATEWAY IP: 172.16.0.4
probing: pc0 com0 com1 pxe![2.1] mem[639K 63M a20=on]
disk:
net: mac 00:00:24:c3:c1:b0, ip 172.16.0.10, server 172.16.0.4
>> OpenBSD/i386 PXEBOOT 1.02
switching console to com0
>> OpenBSD/i386 PXEBOOT 1.02
com0: changing speed to 19200 baud in 5 seconds, change your terminal to match!

com0: 19200 baud
booting tftp:bsd.rd: 4302596+825452 [52+147936+134838]=0x5291b0
entry point at 0x100120

Copyright (c) 1982, 1986, 1989, 1991, 1993
```

Listing *15. Normal boot and installation proces*

```
        The Regents of the University of California.  All rights reserved.
Copyright (c) 1995-2005 OpenBSD. All rights reserved.  http://www.OpenBSD.org

OpenBSD 4.1 (RAMDISK_CD) #573: Sun May 20 00:27:05 MST 2007
    deraadt@i386.openbsd.org:/usr/src/sys/arch/i386/compile/RAMDISK_CD
cpu0: AMD Am486DX4 W/B or Am5x86 W/B 150 ("AuthenticAMD" 486-class)
cpu0: FPU
real mem  = 66691072 (65128K)
avail mem = 54427648 (53152K)
using 839 buffers containing 3436544 bytes (3356K) of memory
mainbus0 (root)
bios0 at mainbus0: AT/286+(00) BIOS, date 20/40/19, BIOS32 rev. 0 @ 0xf7840
pcibios0 at bios0: rev 2.0 @ 0xf0000/0x10000
pcibios0: pcibios_get_intr_routing - function not supported
pcibios0: PCI IRQ Routing information unavailable.
pcibios0: PCI bus #2 is the last bus
bios0: ROM list: 0xc8000/0x9000
cpu0 at mainbus0
pci0 at mainbus0 bus 0: configuration mode 1 (no bios)
pchb0 at pci0 dev 0 function 0 "AMD ElanSC520 PCI" rev 0x00
cbb0 at pci0 dev 17 function 0 "Texas Instruments PCI1420 CardBus" rev 0x00: irq 10
cbb1 at pci0 dev 17 function 1 "Texas Instruments PCI1420 CardBus" rev 0x00: irq 10
sis0 at pci0 dev 18 function 0 "NS DP83815 10/100" rev 0x00: DP83816A, irq 11, address 00:00:24:c3:c1:b0
nsphyter0 at sis0 phy 0: DP83815 10/100 PHY, rev. 1
sis1 at pci0 dev 19 function 0 "NS DP83815 10/100" rev 0x00: DP83816A, irq 5, address 00:00:24:c3:c1:b1
nsphyter1 at sis1 phy 0: DP83815 10/100 PHY, rev. 1
cardslot0 at cbb0 slot 0 flags 0
cardbus0 at cardslot0: bus 1 device 0 cacheline 0x10, lattimer 0x3f
pcmcia0 at cardslot0
cardslot1 at cbb1 slot 1 flags 0
cardbus1 at cardslot1: bus 2 device 0 cacheline 0x10, lattimer 0x3f
pcmcia1 at cardslot1
isa0 at mainbus0
isadma0 at isa0
pckbc0 at isa0 port 0x60/5
pckbd0 at pckbc0 (kbd slot)
pckbc0: using irq 1 for kbd slot
wskbd0 at pckbd0 (mux 1 ignored for console): console keyboard
wdc0 at isa0 port 0x1f0/8 irq 14
wd0 at wdc0 channel 0 drive 0: <SanDisk SDCFB-64>
wd0: 1-sector PIO, LBA, 61MB, 125440 sectors
wd0(wdc0:0:0): using BIOS timings
npx0 at isa0 port 0xf0/16: using exception 16
pccom0 at isa0 port 0x3f8/8 irq 4: ns16550a, 16 byte fifo
pccom0: console
pccom1 at isa0 port 0x2f8/8 irq 3: ns16550a, 16 byte fifo
biomask f7c5 netmask ffe5 ttymask ffe7
```

Listing *15. Normal boot and installation proces*

```
rd0: fixed, 3800 blocks
wi0 at pcmcia0 function 0 "NETGEAR MA401RA Wireless PC, Card, ISL37300P" port 0xa000/64
wi0: PRISM2.5 ISL3873, Firmware 1.0.7 (primary), 1.3.6 (station), address 00:09:5b:3b:89:58
root on rd0a
rootdev=0x1100 rrootdev=0x2f00 rawdev=0x2f02
erase ^?, werase ^W, kill ^U, intr ^C, status ^T
(I)nstall, (U)pgrade or (S)hell? i
[...]
```

The client filesystem table, `/etc/fstab(5)` (which, to be precise, resides on the server, in `/exports/net4521/root/etc/fstab`), will look like:

```
/etc/fstab
boot-srv:/exports/net4521/root  /      nfs     rw      0 0
boot-srv:/usr                   /usr   nfs     rw      0 0
```

Bibliography

- FAQ 4 – *http://www.openbsd.org/faq/faq4.html* OpenBSD Installation Guide
- FAQ 5 – *http://www.openbsd.org/faq/faq5.html* Building the System from Source
- FAQ 6 – Networking *http://www.openbsd.org/faq/faq6.html*
- Soekris on OpenBSD Running Diskless *http://www.256.com/gray/docs/soekris_openbsd_diskless/*
- Soekris Router Project *http://www.glozer.net/soekris/soekris.html*
- Homemade Embedded BSD Systems *http://www.onlamp.com/pub/a/bsd/2004/03/11/Big_Scary_Daemons.html*
- Diskless, Low-Form-Factor OpenBSD Systems *http://www.onlamp.com/pub/a/bsd/2004/04/29/Big_Scary_Daemons.html*

References

- Soekris Engineering, Inc. *http://www.soekris.com/*, Soekris Engineering's website
- PC Engines GmbH *http://www.pcengines.ch/*, PC Engines – your flexible partner in Embedded PC Design
- BowlFish *http://www.kernel-panic.it/software/bowlfish/*, an OpenBSD installer for embedded systems
- Flashdist *http://www.nmedia.net/~chris/soekris/*, an embedded OpenBSD installer
- [FAQ14] *http://www.openbsd.org/faq/faq14.html#Boot386*, How does OpenBSD/i386 boot?

Now we only have to power up the device and, if some champagne remained from the previous chapter, now is time to go get it and finish it.

Network installation

The network installation has many steps in common with the diskless installation: once again, we will have to set up the `rarpd(8)`, `dhcpd(8)` and `tftpd(8)` (*http://www.kernel-panic.it/openbsd/embedded/embedded3.html#3.3*) servers. This time, however, the kernel to boot is `bsd.rd` instead of `bsd`. It's a RAM disk kernel which, after boot, provides a RAM-based filesystem containing various interesting utilities for system maintenance and installation. Therefore, the boot configuration file will contain an additional line:

```
/tftpboot/etc/boot.conf
set tty com0
stty com0 19200
boot bsd.rd
```

To boot from network you must press [*ctrl-P*] at system startup to enter into the BIOS menu and then type *boot F0*. An absolutely normal boot and installation processes will follow: see Listing 15.

DANIELE MAZZOCCHIO

Latest release: http://www.kernel-panic.it/openbsd/embedded/

11th Libre Software Meeting

Free Admission

Bordeaux - Pessac - Talence

July 6th - 11th, 2010

http://rmll.info

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

MANIFESTATION RESPONSABLE

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

Making Sense of

Data Management on Intelligent Devices

The demand for embedded devices is growing rapidly, and there is a clear need for development of advanced software to deliver new features on limited hardware. Data management is a critical component in these new software systems.

What you will learn…

- You will also learn that data management for embedded systems and devices is a major concern and needs to be addressed during the development.

What you should know…

- You should know that embedded systems and devices that rely on flat files and non-relational databases will have major problems in a long term.

Embedded databases are used by portable media players to store information about music and video, GPS devices to store map data, and monitoring systems to log information. These and other leading-edge industries have learned the importance of managing data reliably with a relational embedded data management system.

Developers face unique challenges when designing and implementing software for custom embedded hardware. Embedded processor architectures, such as ARM, PowerPC, Atom™, each have unique characteristics. Footprint and performance are especially important, and access to source code for all software components is required for customization and portability.

What is an embedded database?

- An embedded database is a software library used by application developers to store data.
- The library adds database features to the application such as transaction logging, scalable index algorithms, and isolated concurrency.
- Unlike enterprise databases, an embedded database is distributed with the application and is not installed separately by the end-users.
- Embedded databases are especially well-suited for special-purpose devices and embedded systems with limited resources and a dedicated user interface.

To meet these requirements, embedded developers have often relied on custom solutions, using flat file formats to store data. However, increasing hardware capabilities make it possible to store more information on embedded devices than ever before. Fast read and write operations, protection from data loss and corruption, and multi-user access have become important requirements for embedded systems. Flat files are not able to fully address these issues.

Design Considerations for Embedded Data Management

- *Critical performance demands*: Embedded devices operate under strict time constraints. Whether to satisfy impatient users or to keep up with a constant stream of incoming sensor data, performance is always important.
- *Fail-safe reliability*: Embedded systems are subject to failure from unexpected power loss and other crash scenarios. If such a situation occurs during a write operation, data may be lost or even corrupted. Redundancy is necessary to ensure reliability.
- *Sharing data between concurrent tasks*: Modern embedded systems are connected and intelligent, performing several tasks at once and often sharing data between those tasks. Locking primitives, such as mutexes, are cumbersome to use directly in complex scenarios.

- *Portability*: The exact format of data in memory is determined by the processor architecture and the compiler. But platform-specific details, such as byte order, alignment, and structure padding, should not affect the format of data stored on persistent media, such as flash.

Custom Flat File Solutions

For stand-alone applications that store little data, flat files are a straightforward method to save information. Data can be written in a human-readable text file format, or stored in a custom binary format. In either case, the application developer is responsible for serializing and deserializing data in a format that is only meaningful to the application.

Unless the application developer is willing to invest significant time in building the data model, custom formats do not scale to large data sets that exceed the size of memory, offer no protection against data loss or corruption, and are difficult to share with other applications.

Consider a simple example using flat files. Suppose that the full data set is read into memory and is periodically saved by writing all data to the file system. Listing 1 shows a trivial function to save the data set, calling a helper function to serialize the data. Listing 2 shows a replacement for this function that provides some protection against unexpected power loss. By alternating between two different files, it ensures that at least one good copy of the data will survive.

However, this solution has some limitations. While it uses a counter to identify the most recently saved file, it is difficult to determine whether the most recent file is complete and accurate. It also requires that the entire data set be written each time a change is saved. Because a full flush operation is required between each save operation, this significantly limits the frequency of updates.

Embedded Relational Database

While each individual problem, in isolation, has a straightforward solution, it is difficult to address one requirement without compromising on the others. Just as saving data safely can limit throughput and the size of the data set, sharing access to the database complicates safe storage and also degrades performance. An embedded relational database management system (embedded RDBMS) provides a complete solution that carefully balances these requirements.

Embedded databases are used in a variety of applications, each with different requirements. To accommodate this, many options are available to control the behavior of the database:

- High performance read and write
- Main-memory and disk-based tables
- Single-user, multi-threaded, and client/server access models
- SQL queries and direct table cursors
- Integrated C/C++ APIs and ODBC connectivity

Listing 1. *Unsafe save function*

```
int save_data(data_t* data_set)
{
  static char* file = "file.db";
  int result;
  FILE* fp;

  fp = fopen(file, "wb");
  result = write_data(data_set, fp);
  fclose(fp);

  return result;
}
```

Listing 2. *Safe commit function*

```
// Alternate between file1 and file2
// each time the phone book is saved.
char file1[] = "file1.db";
char file2[] = "file2.db";

// Maintain a counter to identify the
// most recently saved file.
int counter = 0;

int commit_data(data_t* data_set)
{
  static char* file = file1;
  int result;

  FILE* fp = fopen(file, "wb");
  counter++;
  fwrite(&counter, sizeof(int), 1, fp);
  result = write_data(data_set, fp);
  // Flushing has a very high performance cost,
  // but must be completed before the next commit.
  fflush(fp);
  fclose(fp);
  file = (file == file1) ? file2 : file1;
  return result;
}
```

Relational Model

In a running application, data is organized in data structures, such as classes, that reference each other directly, sometimes in a hierarchy, but usually in a complex network. However, direct references are difficult to maintain when data is stored persistently, especially if it is shared with other tasks that approach the data in a different way. Even small changes to the application can easily break backward compatibility. Porting to a new processor or operating system, or even changing the compiler, can raise unexpected problems.

Instead, relational databases organize data in tables, where related tables share common fields. In this way, relationships are maintained naturally and can always be used in both directions. Data is easily accessed through SQL queries and standard interfaces such as ODBC. And because there is a clear boundary between the representation of data in a working application and the representation used when that data is stored, changing the application or supporting another platform is a straightforward process.

Consistent, Scalable Performance

Embedded devices need consistent, scalable performance across all operations, whether reading or writing to the database. Indexes are used to efficiently search the database and traverse the relationships between tables. B+ tree indexes are optimized to minimize disk I/O, and offer consistent performance regardless of the size of the table, even with limited random-access memory. For tables that can fit entirely in main memory, T-tree indexes ensure that processor instructions are minimized.

Shared Access with Multi-user Connections

An embedded database can be shared between several concurrent tasks, and can present each task with what seems to be exclusive access to the data for a short time. By automatically locking individual rows as they are read and modified, the database enables tasks to safely work in different parts of the database in tandem, only pausing or moving on to other work when they would interfere with each other.

Whether an application needs no shared access to a database, access from several threads, or from several processes, embedded databases can accommodate each scenario, and the same application code can be used in all cases.

Database Recovery

When a sudden power failure or crash occurs while writing to a file, data corruption and inconsistency can result. To prevent corruption, embedded databases first write each change to a separate log file before modifying the database file. Using the log, incomplete changes can be rolled back to restore the database to a known good state.

If the database software uses write-ahead logging, also known as undo/redo logging, changes can be written to the database file either before or after a transaction is committed. This significantly reduces write operations without compromising data integrity. In this way, high-throughput tasks that frequently update the database can coexist with tasks that modify a large portion of the database at once.

Conclusion

Flat file formats are not robust enough to handle all of the problems that embedded developers will face as storage media continues to grow in size. A relational embedded database is a powerful and important tool in any embedded developer's arsenal. And while many off-the-shelf solutions are available, it is important to select a product that can fully meet your application's needs. Some databases provide only basic functionality, with limited support for concurrency and mediocre performance in serious applications. Others are bloated with features that are unnecessary on embedded systems, requiring complicated installation procedures and consuming more system resources than the application itself. Starting with a solution that is designed to meet the requirements of embedded systems and devices has a significant impact on the performance, maintainability, and extensibility of the application.

ITTIA DB SQL is a pure relational database library that provides embedded applications with a single solution to the most important challenges of data storage. ITTIA DB SQL is fully functional, supporting write-ahead logging, B+ and T-tree indexes, complex multi-user shared access, and more. A variety of platforms are supported, and source code is available for porting to new platforms, customizing the feature set to minimize the already low footprint, or just for the assurance of having total control. With a high-performance relational embedded database like ITTIA DB SQL, application developers can focus on the business logic that makes each product unique.

RYAN PHILLIPS

Ryan Phillips is the Lead Engineer for ITTIA DB SQL. Ryan has worked closely on projects both for back-end enterprise databases and for embedded systems, and now finds ways to combine the lessons learned from the diverse history of these two fields.

In the next issue:

-BSD FILE SHARING Part 4
-Questions from Readers
- and Other !

Next issue is coming in June!

BSD in the Industry

After several years of slavery with windows based programs, many programs related with Industry or Engineering are opening the doors to the new trends of UNIX like OS. This is a natural evolution because as the Economy crisis strikes on whole World, the IT infrastructures are also under pressure to decrease at maximum the overall cost.

Despite of this serious crisis, still many Companies continue using proprietary software in their infrastructures. Although the technicians decrease their internal costs using Free Software improving not only the costs but also their performance, the policies and agreements of their Companies do not allow to the total implantation of this software. However, even when the close mind Corporations are starting to be in evolution, the presence of Free Software in large Companies is anecdotic.

But what happens when we speak about the implementation of IT infrastructures in Enterprises? In the world of Steel Industry and also in the Energy Industry, the trend of Engineering Companies is to provide a *turn-key* Project, that is, not only the classic implementation of Production Processes but also of all those Processes that support it such as software. Basically these kinds of Projects integrate the software as support services to the Production: automation control of machines trough PLCs, visualization of Production parameters values trough HMI, support for Quality, Safety and Environment, support the Maintenance and an ERP package.

There are several software Providers that offer these programs from some time ago, but ironically these programs are written or his platform is supported only under Windows environments. For about five years and at the request of the Customers, some of them also offer support for GNU / Linux. But as this Market is so enormously promising to the Free Software, why do not use BSD? Most of IT managers knows the benefits of BSD over GNU / Linux and is not my intention in this paper to discuss or describe these advantages but if they are opening this Market, why the BSD users did not? Final target of this paper is to show in which areas BSD users and developers, integrators are failing.

In this case, the advantage of GNU / Linux is that they're offering their OS and Programs, beyond the classic IT environment. His legacy UNIX allows them to offer a reliable platform on its traditional use but ongoing Partnerships with Leading Companies in other Sectors such as the ERP. This approach allows them to enter an unfamiliar environment for them, through the knowledge of these Companies are experts in their field supported by an Open Source software platform that allows both to benefit. These Companies change their Services decreasing or entirely eliminating licensing costs without reducing their benefits come through the implementation costs and GNU / Linux is benefiting from the cost of these implementations and the experience acquired in the Sector during the same. In addition, this synergy allows the End Customer to enjoy a tangible improvement in flexibility and reliability. If the results are good ?Why would it be this paper? As far as everybody can contribute and wants to use the last geek features released with the distros, as fast the lack of stability and serious security issues becomes in the implementations. This is the point that could allow to BSD flavours to be the right choice.

In next papers I will describe all of the necessary Support Services to the Production in the world of Steel and Energy Industry, starting with my personal selection and highlighting where and how BSD can and must be evolved. Also my intention is to guide and help as much as possible to BSD related Companies to be introduced in these fields.

JOSEBA MENDEZ

NOW FEATURING THE INTEL® XEON® PROCESSOR 5600 SERIES

Orion II iX-N4236

Powerful *4U* Orion II Storage Series

- ✔ Outstanding Performance
- ✔ Excellent Cooling Efficiency
- ✔ Up to 24 Processing Cores with Hyper-Threading
- ✔ Up to 72TB in 4U, Unparalleled Storage Density
- ✔ Up to 432TB in 20U of Rack Space Utilizing Optional Orion II JBOD Expansion Units

To order today call: **1-800-820-BSDi**

Notable features include:

- Dual Intel® 64-Bit Socket 1366 Six-Core, Quad-Core, or Dual-Core, Intel® Xeon® Processor 5600/5500 Series
- 4U Storage Server Chassis with up to 72 TB storage capacity
- 36 x 3.5 Hot-Swap SAS/SATA HDDs (24 front side + 12 rear side)
- 1400 W (1+1) Redundant High Efficiency Power Supply (Gold level 93%+ power efficiency)
- Dual Intel® 5520 chipsets with QuickPath Interconnect (QPI) up to 6.4 GT/s
- Up to 192GB DDR3 1333/1066/800 MHz ECC Registered DIMM/24 GB Unbuffered DIMM
- 2 (x16) PCI-E 2.0, 4 (x8) PCI-E 2.0 (1 in x16 slot), 1 (x4) PCI-E (in x8 slot)
- Intel® 82576 Dual-port Gigabit Ethernet Controller

iXsystems Introduces the Orion II 4U Storage Solution

The iX-N4236 boasts energy efficient technology and maximum, high density storage capacity, creating a 4U powerhouse with superior cooling.

The Orion II has **thirty-six hot-swappable SAS/SATA drive bays**, providing 50% more storage density than its predecessor. By delivering high-end storage density within a single machine, iXsystems cuts operating costs and reduces energy requirements.

Storage sizes for the iX-N4236 are customizable, with 250GB, 500GB, 750GB, 1TB, and 2TB hard drives available. For environments requiring maximum storage capacity and efficiency, 2TB Enterprise-class drives are available from Western Digital®, Seagate®, and Hitachi. These drives feature technologies to prevent vibration damage and increase power savings, making them an excellent choice for storage-heavy deployment schemes.

The Intel® Xeon® Processor 5600 Series (Six/Quad-Core) and Intel® Xeon® Processor 5500 Series (Quad/Dual-Core) have a light energy footprint, while creating a perfect environment for intense virtualization, video streaming, and management of storage-hungry applications. Energy efficient DDR3 RAM complements the other power saving components while still providing 18 slots and up to 192GB of memory overall.

100% cooling redundancy, efficient airflow, and intelligent chassis design ensure that even under the heaviest of workloads, the Orion II remains at an optimal temperature, while still drawing less power than other servers in its class. With a 1400 W Gold Level (93%+ efficient) power supply, the entire system works together to efficiently manage power draw and heat loss.

For more information or to request a quote, visit:
http://www.iXsystems.com/Orion2

intel Xeon inside™
Powerful. Intelligent.