BSD MAGAZINE

FOR NOVICE AND ADVANCED USERS

ZFS BOOT ENVIRONMENTS

INSIDE

VOL.7 NO.8
ISSUE 08/2013(49)
1898-9144

855-GREP-4-IX
www.iXsystems.com
Enterprise Servers and Storage for Open Source

✓ Rock-Solid Performance

✓ Professional In-House Support

E5-2600

High Performance,
High Density Servers for
Data Center, Virtualization, & HPC

MODEL: iXR-22X4IB

http://www.iXsystems.com/e5

768GB
of RAM in 1U

KEY FEATURES

iXR-22X4IB	iXR-1204+10G
• Dual Intel® Xeon® Processors E5-2600 Family per node	• Dual Intel® Xeon® Processors E5-2600 Family
• Intel® C600 series chipset	• Intel® C600 series chipset
• Four server nodes in 2U of rack space	• Intel® X540 Dual-Port 10 Gigabit Ethernet Controllers
• Up to 256GB main memory per server node	• Up to 16 Cores and 32 process threads
• One Mellanox® ConnectX QDR 40Gbp/s Infiniband w/QSFP Connector per node	• Up to 768GB main memory
• 12 SAS/SATA drive bays, 3 per node	• Four SAS/SATA drive bays
• Hardware RAID via LSI2108 controller	• Onboard SATA RAID 0, 1, 5, and 10
• Shared 1620W redundant high-efficiency Platinum level (91%+) power supplies	• 700W high-efficiency redundant power supply with FC and PMBus (80%+ Gold Certified)

Call iXsystems toll free or visit our website today! **1-855-GREP-4-IX | www.iXsystems.com**

High-Density iXsystems Servers powered by the Intel® Xeon® Processor E5-2600 Family and Intel® C600 series chipset can pack up to 768GB of RAM into 1U of rack space or up to 8 processors - with up to 128 threads - in 2U.

On-board 10 Gigabit Ethernet and Infiniband for Greater Throughput in less Rack Space.

Servers from iXsystems based on the Intel® Xeon® Processor E5-2600 Family feature high-throughput connections on the motherboard, saving critical expansion space. The Intel® C600 Series chipset supports up to 384GB of RAM per processor, allowing performance in a single server to reach new heights. This ensures that you're not paying for more than you need to achieve the performance you want.

The iXR-1204 +10G features dual onboard 10GigE + dual onboard 1GigE network controllers, up to 768GB of RAM and dual Intel® Xeon® Processors E5-2600 Family, freeing up critical expansion card space for application-specific hardware. The uncompromised performance and flexibility of the iXR-1204 +10G makes it suitable for clustering, high-traffic webservers, virtualization, and cloud computing applications - anywhere you need the most resources available.

For even greater performance density, the iXR-22X4IB squeezes four server nodes into two units of rack space, each with dual Intel® Xeon® Processors E5-2600 Family, up to 256GB of RAM, and an on-board Mellanox® ConnectX QDR 40Gbp/s Infiniband w/QSFP Connector. The iXR-22X4IB is perfect for high-powered computing, virtualization, or business intelligence applications that require the computing power of the Intel® Xeon® Processor E5-2600 Family and the high throughput of Infiniband.

HIGH Throughput & INCREDIBLE Performance Density

IXR-1204+10G: **10GbE On-Board**

4 Server Nodes in 2U

IXR-22X4IB

Call iXsystems toll free or visit our website today! **1-855-GREP-4-IX | www.iXsystems.com**

Dear Readers,

This issue of BSD Magazine is dedicated to boot environments, including the article on ZFS Boot Environments by Kris Moore. Moreover, you can read articles about email gateway, service spawner, web programming, PKGNG and Apache THRIFT.

We start with the Let's Talk section, where Rob Somerville shares his thoughts on "opt in" legislation for access to adult material.

Next, in the Developer's Corner, we take a look at the article about Apache THRIFT by Chirag Maheshwari.

Then, in the What's New section, we take a look at an article about ZFS Boot Environments by Kris Moore.

Alexandro Silva describes how to build an email gateway with FreeBSD to prevent malware and undesirable messages.

Next, Daniel Dettlaff talks a bit about The Service Spawner that does user-side installation, software configuration, and maintains your software automatically.

This month's Admin section continues with the menu navigation system and the usage of Javascript.

Finally, Joe Maloney talks about PKGNG: The future of packages on FreeBSD and PC-BSD.

We hope you enjoy this issue and find many interesting articles inside!

Kamil Sobieraj
Editor of BSD Magazine
& BSD Team

MAGAZINE BSD

Editor in Chief:
Ewa Dudzic
ewa.dudzic@software.com.pl

Supportive Editor:
Kamil Sobieraj
kamil.sobieraj@software.com.pl

Contributing:
Kris Moore, Michał Hewelt, Alexandro Silva, Rob Somerville, Joe Maloney, Chirag Maheshwari

Top Betatesters & Proofreaders:
Annie Zhang, Luca Ferrari, Eric Geissinger, Winston Weinert, Russ Harvey, Imad Soltani, Olaoluwa Omokanwaye, Edward Werzyn, Eric De La Cruz Lugo, Robert Cabrera, Raul Mesa Ros, Denise Ebery, Ben Milman

Special Thanks:
Annie Zhang
Eric Geissinger

Art Director:
Ireneusz Pogroszewski

DTP:
Ireneusz Pogroszewski
ireneusz.pogroszewski@software.com.pl

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

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

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

Advertising Sales:
Kamil Sobieraj
kamil.sobieraj@software.com.pl

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

Software Media 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.

Quis Custodiet Ipsos Custodes

(Who will guard the guards themselves)?

With the UK government in alliance with Internet Service Providers determined to introduce "opt in" legislation for access to adult material, has the Net come of age or are we entering a new period of censorship?

There will be thousands of words, both printed and typed, on the blogo-sphere arguing the pros and cons of filtering pornographic material via "Opt In" at the UK ISP's level. I will take a very British view on this, in so much that what two consenting adults do in the privacy of their own home – provided it is legal and no harm becomes of others – is of no concern to others. At the end of the day, as adults we have a conscience to guide us, and indeed the foundation of Western criminal law is based on *mens rea* – the act is not culpable unless the mind is guilty. So I will not add to the noise by debating the rights or wrongs of a piece of forthcoming legislation which seems based on a populist knee jerk reaction – however well intentioned it may be. I suspect history will prove it to be ineffectual, a political "get out of jail free" card that absolves those in positions of responsibility by allowing them to say with mealy mouth, "at least we tried".

The scenario goes much deeper than just protecting children. The definition of a civilised society is how well we look after the poor, the sick, and those that cannot look after themselves. No right thinking adult could argue with that premise. But what happens when the law of unintended consequences takes hold? The proponents of this legislation would argue that it is not censorship – but they cannot argue against the fact that hooks are being put in place to potentially control those that choose to access adult material – whether sexual or otherwise. In other words, the camel's nose of censorship is now well inside the tent or to use a very English phrase, this is just the thin edge of the wedge.

The potential for abuse is already evident from the adult filters on mobile networks in the UK. Gun, alternative news political, community and technology sites and those with "inappropriate" content are all lumped together with the more carnal purveyors of adult entertainment. To add insult to injury, the filters are either on or off. You can either receive a bland diet of what our moral guardians believe is "safe" or the whole gamut, including the most distasteful the Internet has to offer. This is not a hypothetical list – these sites have already been blocked in the UK by adult filtering. The difficulty is that the appeal process is unmerciful, and it is doubtful that in law any web-master having his or her site incorrectly included in the list would be able to reclaim loss of income or reputation via the courts. And let's be honest, the technology isn't there to truly determine the really nasty stuff. So the potential to monetise freedom has now arrived at a computer near you. Those behind the curtain yet again hold the power but escape accountability.

Once the majority of people realise this, they will vote with their feet and choose to have the filter removed. Then what? Your local friendly ISP (Or worse still some quasi-government agency) will have a subscribers list that marketeers, blackmailers, the media, law

enforcement agencies and other interested parties would give their right arm for. Human nature being what it is, this information will get out. Be it through commercial forces (Sell us your database, we'll give you a pound per pervert) or "public interest" (Political opponent X enjoys watching "Adult" movies) the vista for abuse and control is limitless. As for black hat hackers, you get the point.

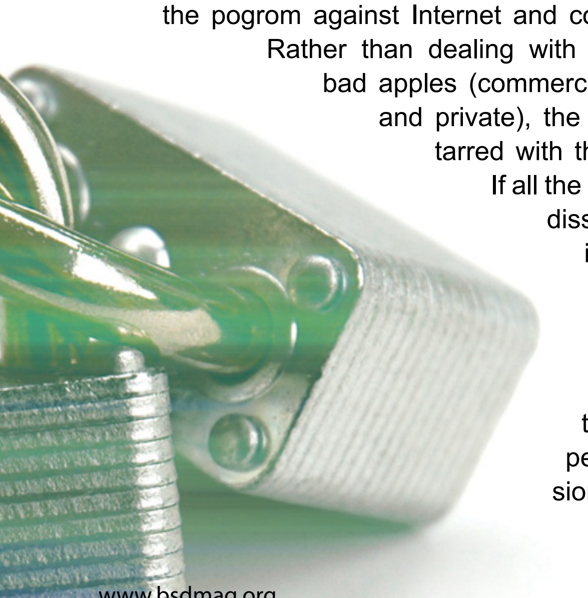
The problem, of course, is that the Internet needs to police itself rather than abrogating responsibility to lawyers, politicians and others who wouldn't recognise an encrypted tunnel or a VPN if it fell on them with the router still attached via steel cable rather than CAT 6. As a community, we need to expose those that damage the reputation of the Internet, but unfortunately the political will just isn't there to deal with the real offenders. A case in point – I recently came across a live phishing email on a Friday evening pointing to a major UK bank. The bank wasn't interested, and if I had contacted the police, I would have been given short shrift. The ISP responsible wasn't that bothered either. We live in a 24/7 culture, and we need a 24/7 response to the threats that materialise and disappear just as quickly. Of course, vexatious complaints would be a problem, but the community has a way of identifying and isolating those that are troublemakers. Please don't feed the trolls. While a global Internet police force would be a bad thing, we do need some sort of mechanism free of political and commercial bias that has teeth. And it needs to work both ways – If Mr. Dictator decides his populace cannot view content from another country we do not have a World Wide Web, we have a filtered LAN. No comfortable deals made in smoke-filled rooms.

We are sliding towards a crisis of medieval proportions where those with vested interests can control not only what the public can or cannot read and view but also when and how. DRM and the threadbare Intellectual Property rights argument are only the beginning of the pogrom against Internet and content freedom. Rather than dealing with the minority of bad apples (commercial, government and private), the whole barrel is tarred with the same brush. If all the energy that was dissipated in dealing with torrents and illegal file sharing was directed towards corruption that costs people their pensions, livelihood – and in developing countries, their lives – the world would be a better place. But hey, that would affect our bottom line. Go for the easy solution rather than dealing with the true perpetrators.

In the age of technology, the Internet remains the last bastion of true democracy. What is done in the dark needs to be exposed, and that cuts both ways. The whole panoply of human character is available on the web, as it is in any gathering of people. It is too late to argue if the community has lost the technological or PR battle; the legislators are gearing up and the commercial and legal realities will mean any compromised organisations will crumple. Google has already been in front of the select committee for tax avoidance in the UK, and strategically where Google goes, a significant percentage will follow. To the community's chagrin, Google has capitulated freedom while the East probes Western servers with impunity. True freedom and democracy are scary responsibilities, but as usual, a pragmatic political compromise leaves an unpleasant taste in the mouth. Idealism has a short shelf life where hard currency is involved.

Tragically, the community is faced with a paradox that embraces all innovators, visionaries and growing societies – how do we police ourselves without delegating responsibility? More challenging still, how can we prevent our idealism from being corrupted by those that would seek to capitalise on human weakness and vulnerability to their advantage? In the year 2013 with scandals affecting the financial markets, traditional institutions and indeed nations, the Internet community is not alone in asking these questions.

Ultimately, the whole Internet is stronger than the sum of its parts. Those in the dark have been running scared, as has been proven when the kill switch has been activated on previous occasions. If we can maintain the moral high ground, police ourselves with integrity, hopefully we can shrug off the pseudo-ethical straitjacket those that do not understand technology seek to impose.



ROB SOMERVILLE

Rob Somerville has been passionate about technology since his early teens. A keen advocate of open systems since the mid eighties, he has worked in many corporate sectors including finance, automotive, airlines, government and media in a variety of roles from technical support, system administrator, developer, systems integrator and IT manager. He has moved on from CP/M and nixie tubes but keeps a soldering iron handy just in case.

Apache THRIFT: A Much Needed Tutorial

This is the first article on Apache Thrift, as there is neither any official documentation nor any tutorial available for the fabulous tool Apache Thrift. This article tries to bridge that gap and introduce you to Apache Thrift and how, when and why you should use it.

What you will learn…

- What is Apache Thrift
- Why we should use Apache Thrift
- How to use Apache Thrift
- Comparison with similar tools
- How it can be scaled

What you should know…

- basic shell commands
- basic knowledge of programming

This article is about Apache Thrift and how it can be used. It also tries to address the challenge of scalability by discussing how Thrift can be used to meet enterprise expectations.

What is Apache Thrift

From the Thrift Website (*http://thrift.apache.org/*):

Thrift is a software framework for scalable cross-language services development. It combines a software stack with a code generation engine to build services that works efficiently and seamlessly between C++, Java, Python, PHP, Ruby, Erlang, Perl, Haskell, C#, Cocoa, JavaScript, Node.js, Smalltalk, and OCaml.

Thrift is an *Interface Definition Language* (IDL) which is used to define and create services between numerous languages as a *Remote Procedure Call* (RPC). Its lightweight framework and support for cross-language communication makes it more robust and efficient than other RPC frameworks like SOA (REST/SOAP) for most of the operations. Through a simple and straightforward IDL, it allows you to create services that are usable by numerous languages. Using code generation, Thrift creates a set of files that can be used to create clients and/or servers. In addition to interoperability, Thrift can be very efficient because of a serialization mechanism which can save both space and time.

In other words, Apache Thrift lets you create a service to send/receive data between two or more softwares that are written in completely different languages/platforms.

History

Thrift was originally developed by the folks at Facebook. It's also one of the "core parts of their infrastructure". After a while, they decided to make it Open Source and finally contributed it to Apache Software Foundation (ASF) in April 2007 in order to increase usage and development. Thrift was then released under the Apache 2.0 license.

The choice of programming language at Facebook is based on what language is best suited for the task at hand. While pragmatic, this flexibility resulted in difficulties when these applications needed to call one another. After some analysis, Facebook engineers did not find anything currently existing which could meet their needs of interoperability, transport efficiency, and simplicity amongst others. Out of this need, they

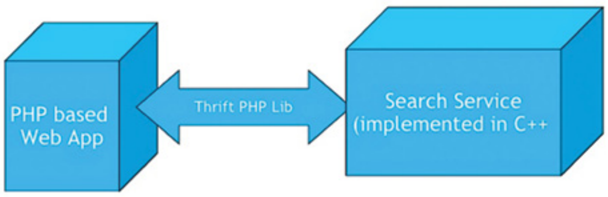


Figure 1. *Facebook Search Service Architecture*

developed efficient protocols and a service infrastructure that became Thrift. Facebook now uses Thrift for their back-end services, the reason for which it was initially designed.

Who uses Thrift?

- Facebook Thrift is one of the core components of Facebook infrastructure. Its search services are implemented in C++, but its Web Application is based on PhP. Thus, to bridge the gap between them, Facebook uses Thrift (Figure 1).
- Evernote Thrift is extensively used in a variety of the Evernote public API.
- Scribe is also built on top of Thrift.
- HBase leverages Thrift for cross-language API.
- The whole list can be found here: *http://wiki.apache.org/thrift/PoweredBy*

How to Download and Install

The stable release of Apache Thrift can be downloaded from: *http://thrift.apache.org/download/*.

Listing 1. *Installing Thrift on Debian/Ubuntu*

```
##Install the pre-requisites
sudo apt-get install libboost-dev libboost-test-
                    dev libboost-program-options-dev
                    libevent-dev automake libtool flex
                    bison pkg-config g++ libssl-dev
sudo apt-get install php5-dev php5-cli
cd /usr/tmp/
fetch "http://www.trieuvan.com/apache/thrift/0.9.0/
                    thrift-0.9.0.tar.gz"
tar xvzf thrift-0.9.0.tar.gz
cd thrift-0.9.0
./bootstrap.sh
./configure
sudo make
sudo make install
```

Listing 2. *Installing Thrift on FreeBSD*

```
##Install the pre-requisites and then,
##on freeBSD, Thrift can be installed from FreeBSD
                    ports

cd /usr/ports/devel/thrift
./bootstrap.sh
./configure
make
make install
```

[GEEKED AT BIRTH]

University of Advancing Technology
UAT
Learn. Experience. Innovate.

You can talk the talk.
Can you walk the walk?

[IT'S IN YOUR DNA]

LEARN:
Advancing Computer Science
Artificial Life Programming
Digital Media
Digital Video
Enterprise Software Development
Game Art and Animation
Game Design
Game Programming
Human-Computer Interaction
Network Engineering
Network Security
Open Source Technologies
Robotics and Embedded Systems
Serious Game and Simulation
Strategic Technology Development
Technology Forensics
Technology Product Design
Technology Studies
Virtual Modeling and Design
Web and Social Media Technologies

www.uat.edu > 877.UAT.GEEK

Install on Ubuntu

Commands are given in Listing 1.

Install on FreeBSD

Commands are given in Listing 2.

Install on Windows

- First install some pre-requisites
  - MinGW
  - GNU Build Tools
  - g++ 4.0+
  - bison 2.3-1
  - boost 1.33.1-4
  - boost-devel 1.33.1-4
  - flex 2.5.33-1
  - pkgconfig
  - libtool
- Download thrift.exe
- Copy it to C:\Thrift\thrift.exe
- Now, add "C:\Thrift\" to your PATH environment variable
- Now the thrift compiler can directly be used from CMD prompt.

Install Thrift Editor for Eclipse

- Open Eclipse
- Eclipse –> help –> Install new Software

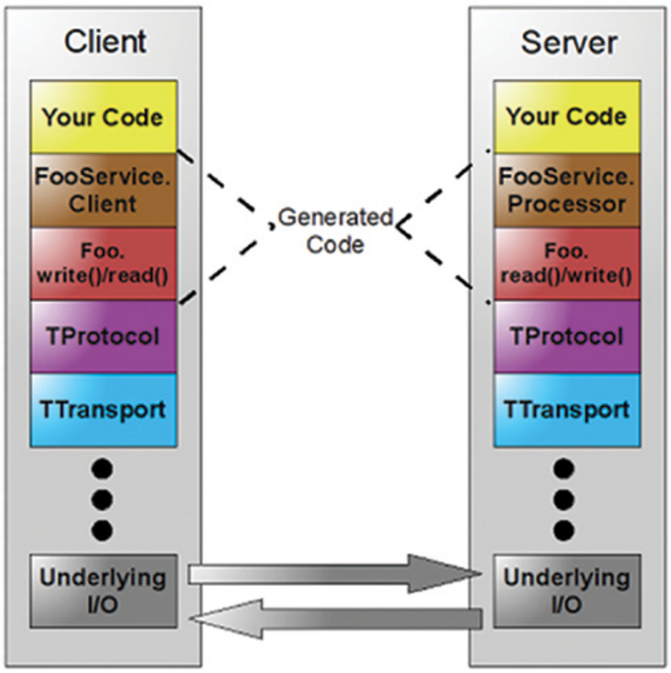


Figure 2. *Apache Thrift client-server architecture*

- Add the URL: *http://thrift4eclipse.sourceforge.net/updatesite/*
- Check the only available package
- Install the package

Architecture

Thrift includes a complete stack for creating clients and servers. Figure 2 depicts the Thrift Stack.

The top portion of the stack is generated code from your Thrift definition file. Thrift services result in generated client and processor code. These are represented by the brown boxes in the figure. The data structures that are sent (other than built-in types) also result in generated code. These result in the red boxes. The protocol and transport are part of the Thrift runtime library. Therefore with Thrift, you can define a service and have the freedom to change the protocol and transport without regenerating your code. Thrift also includes a server infrastructure to tie the protocols and transports together. There are blocking, non-blocking, single and multithreaded servers available. The "Underlying I/O" portion of the stack differs based on the language in question. For Java and Python network I/O, the built-in libraries are leveraged by the Thrift library, while the C++ implementation uses its own custom implementation.

Thrift allows you to choose independently between your protocol, transport and server. With Thrift being originally developed in C++, Thrift has the greatest variation among these in the C++ implementation.

Transport Layer

The transport layer provides simple abstraction for reading/writing to/from the network. The transport layer basically describes "how" data is transmitted. This layer decouples the underlying transport from the rest of the system, exposing only the following interface:

- open
- close
- read
- write
- flush

In addition to the above interface, Thrift also uses ServerTransport interface on the server side to accept or create transport objects. The interface includes:

- open
- listen
- accept
- close

There are a number of transports supported by Thrift:

- TSocket: Uses blocking socket I/O for transport.
- TFramedTransport: Sends data in frames, where each frame is preceded by a length. This transport is required by a non-blocking server.
- TFileTransport: This transport writes to a file. This transport is not included with the Java implementation.
- TMemoryTransport: Uses memory for I/O. The Java implementation uses a simple ByteArrayOutputStream internally.
- TZlibTransport: Performs compression using zlib. It should be used in conjunction with another transport. Not available in the Java implementation.

Protocol Layer

The protocol abstraction defines a mechanism to map in-memory data structures to a wire-format. It specifies how datatypes use the underlying Transport to encode/decode themselves. Separates Data Structure from Transport representation. Thus the protocol implementation governs the encoding scheme and is responsible for (de)serialization. Thrift protocols are stream oriented by nature thus there is no need for any explicit framing. In other words, Protocols describe "WHAT" is actually transmitted. Thrift supports both text and binary protocols. The binary protocols almost always outperforms text protocols, but sometimes text protocols may prove to be useful in cases of debugging. The Protocols available for the majority of the Thrift-supported languages are:

TBinaryProtocol
A straightforward binary format encoding numeric values as binary, rather than converting to text. They are not optimized for space efficiency.

TCompactProtocol
Very efficient and dense encoding of data. This protocol writes numeric tags for each piece of data. The recipient is expected to properly match these tags with the data (If the data is not present, no tag/data pair is present). For Integers, this protocol performs compression using Variable-Length Quantity (VLQ) encoding from the MIDI file format.

TDenseProtocol
It's similar to TCompactProtocol but strips off the meta information from what is transmitted and adds it back at the receiver side. It is still experimental and not yet implemented in Java.

TJSONProtocol
Uses JSON for data encoding.

TSimpleJSONProtocol
A write-only protocol using JSON. Suitable for parsing by scripting languages.

TDebugProtocol
Sends data in the form of human-readable text format. It can be well used in debugging applications involving Thrift.

Processor Layer

A processor encapsulates the ability to read data from input streams and write to output streams. The input and output streams are represented by protocol objects. The processor interface is extremely simple. Service-specific processor implementations are generated by the Thrift compiler.

Thus, the generated code makes the Process Layer of the architecture stack. The processor essentially reads data from the wire (using the input protocol), delegates processing to the handler (implemented by the user), and writes the response over the wire (using the output protocol).

Server Layer

A server pulls together all of the various functionalities described above to complete the Thrift server. First, it creates a transport and specifies input/output protocols for

Listing 3. *file: hello.thrift*

```
namespace java helloworld
service HelloService {
   string sayHello()
}
```

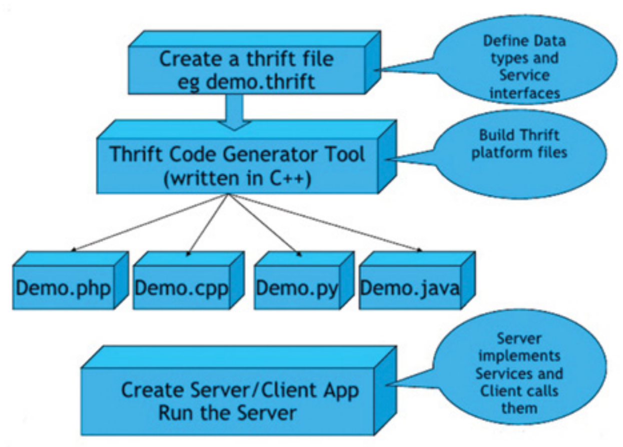

Create a thrift file
eg demo.thrift
Define Data types and Service interfaces
Thrift Code Generator Tool
(written in C++)
Build Thrift platform files
Demo.php
Demo.cpp
Demo.py
Demo.java
Create Server/Client App
Run the Server
Server implements Services and Client calls them

Figure 3. *Thrift Application Development flow-chart*

Listing 4. *Generating Processors in Thrift*

```
##Generating the processor for Server Side
thrift --gen java hello.thrift
##Generating the processor for Client Side
thrift --gen php hello.thrift
```

Listing 5. *file: HelloServiceImpl.java implementing the hello world service*

```
package helloworld;
import org.apache.thrift.TException;

public class HelloServiceImpl implements HelloService.Iface {

   public String sayHello() throws TException {
      return "Hello World!!";
   }
}
```

Listing 6. *file: HelloServer.java which is code for Java server.*

```
package helloworld;

import org.apache.thrift.server.TServer;
import org.apache.thrift.server.TThreadPoolServer;
import org.apache.thrift.transport.TServerSocket;
import org.apache.thrift.transport.TTransportException;

public class HelloServer implements Runnable {

   private static final int PORT =   9090;

   public void run() {
      try {
         TServerSocket serverTransport = new
                  TServerSocket(PORT);
         HelloService.Processor processor = new
                  HelloService.Processor(new
                  HelloServiceImpl());
         TServer server = new TThreadPoolServer(new
                  TThreadPoolServer.Args(serverTransport).
                  processor(processor));
         System.out.println("Starting server on port: "+PORT);
         server.serve();

      } catch(TTransportException e) {
         System.out.println("Message: "+e.getMessage());
         System.out.println("StackTrace: ");
         e.printStackTrace();
      }
   }

   public static void main(String[] args) {
      new Thread(new HelloServer()).run();
   }
}
```

Listing 7. *file: client.php, Client side code.*

```
<?php
   // defining the port and server to listen
   define("PORT", '9090');
   define("SERVER", 'localhost');

   //Global variable where the php library files are stored
   $GLOBALS['THRIFT_ROOT'] = 'thrift';

   //including the library files
   require_once $GLOBALS['THRIFT_ROOT'].'/Thrift.php';
   require_once $GLOBALS['THRIFT_ROOT'].'/protocol/
                TBinaryProtocol.php';
   require_once $GLOBALS['THRIFT_ROOT'].'/transport/
                TSocket.php';
   require_once $GLOBALS['THRIFT_ROOT'].'/transport/
                TBufferedTransport.php';

   //loading the auto-generated package
   require_once $GLOBALS['THRIFT_ROOT'].'/packages/
                hello/HelloService.php';
?>
<?php
   try {
      //create a thrift connection
      $socket = new TSocket(SERVER, PORT);
      $transport = new TBufferedTransport($socket);
      $protocol = new TBinaryProtocol($transport);

      //create a new hello service client
      $client = new HelloServiceClient($protocol);

      //open the connection
      $transport->open();

      $result = $client->sayHello();
      echo "Result: ".$result;

      $transport->close();
   } catch(TException $tx) {
      echo "Thrift Exception: ".$tx->getMessage()."\r\n";
   }
?>
```

the transport. Then, it creates a processor based on the I/O protocols. It finally waits for incoming connections. When a connection is made, it hands them off to the processor to handle the processing of the request.

Thrift provides a number of servers:

TSimpleServer
A single-threaded server using standard blocking I/O socket. Mainly used for testing purposes.

TThreadPoolServer
A multi-threaded server with N worker threads using standard blocking I/O. It generally creates *N=5* minimum threads in the pool if not specified otherwise.

TNonBlockingServer
A multi-threaded server using Non-Blocking IO (Java implementation uses NIO channels). TFramedTransport must be used with this server.

THttpServer
HTTP server (for JS clients) optionally with REST like URLs.

TForkingServer
Forks a process for each request to server.

TProcessPoolServer
Available in Python. Pre-forks workers to avoid Global Interpreter Lock.

Listing 8. *Non-Blocking server in Java*

```
//Works with Asynchronous client too

public class NonblockingServer {

private static final int PORT = 7911;
    private void start() {
        try {
            TNonblockingServerTransport serverTransport
                    = new TNonblockingServerSocket(PORT);
            HelloService.Processor processor =
                    new HelloService.Processor(new
                    HelloServiceImpl());

            TServer server = new TNonblockingServer(new
                    TNonblockingServer.
                    Args(serverTransport).
                     processor(processor));
            System.out.println("Starting server on port
                    '' + PORT.toString() + '' ...");
            server.serve();
        } catch (TTransportException e) {
            e.printStackTrace();
        }
    }

    public static void main(String[] args) {
        NonblockingServer srv = new NonblockingServer();
        srv.start();
    }
}
```

Listing 9. *Non-Blocking Client in Java*

```
//Note the usage of TFramedTransport for Non-Blocking
                    Server which would frame the data
//sent over the wire

public class NonblockingClient {
    private void invoke() {
        TTransport transport;
        try {
            transport = new TFramedTransport(new
                    TSocket("localhost", 7911));
          TProtocol protocol = new TBinaryProtocol(transport);

            HelloService.Client client = new
                    HelloService.Client(protocol);
            transport.open();

            System.out.println(client.sayHello());

            transport.close();
        } catch (TTransportException e) {
            e.printStackTrace();
        } catch (TException e) {
            e.printStackTrace();
        }
    }

    public static void main(String[] args) {
        NonblockingClient c = new NonblockingClient();
        c.invoke();
    }
}
```

Listing 10. *Asynchronous Client written in Java*

```
//Uses the Non-Blocking Server given in Listing 8.
//Note that for each operation of the service, a callback will be defined. Also, a new client will
//have to be used with every different operation else an exception will be thrown

public class AsyncClient {

    private void invoke() {
        try {
            HelloService.AsyncClient client = new HelloService.
                    AsyncClient(new TBinaryProtocol.Factory(), new TAsyncClientManager(),
                                new TNonblockingSocket("localhost", 7911));

            client.sayHello(new sayHelloMethodCallback());
        } catch (TTransportException e) {
            e.printStackTrace();
        } catch (TException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    public static void main(String[] args) {
        AsyncClient c = new AsyncClient();
        c.invoke();
    }

    class sayHelloMethodCallback
            implements AsyncMethodCallback<HelloService.AsyncClient.sayHello_call> {

        public void onComplete( HelloService.AsyncClient.sayHello_call sayHello_call) {
            try {
                string result = sayHello_call.getResult();
                System.out.println("Result from server: " + result);
            } catch (TException e) {
                e.printStackTrace();
            }
        }

        public void onError(Exception e) {
            System.out.println("Error : ");
            e.printStackTrace();
        }

    }
```

Thrift allows only one service per server. This is certainly a limitation and will be addressed later in this article under multiplexing.

How to use?

Figure 3 describes how the flow of a Thrift Application is made and Figure 4 shows the anatomy of a Thrift service.

First we'll make a small "Hello World" application which aims to make RPC's between Java (Server) and PHP (Client).

We'll start by making a Thrift IDL file named "hello.thrift" as shown in Listing 3. The full tutorial for making a Thrift IDL file can be found here: *http://diwakergupta.github.io/thrift-missing-guide/thrift.pdf*.

Then we'll generate the files for processors layer using the Thrift compiler for both Client and Server using the commands given in Listing 4.

Make a new project in Eclipse with type, "Dynamic Web Project". Put the "thrift file" in the `<project-name>/Java/Resources/src/` directory. Copy the lib files (`libthrift-<version>.jar, build/lib/*`) to `<project-name>/WebContent/WEB-INF/lib/` folder. Now, we have to implement the services mentioned in the thrift file. Write a new class named "HelloServiceImpl" implementing the "HelloService.Iface" interface under the package "helloworld". The code for the same is given in Listing 5.

Now, we have to write the server which actually caters to the client's requests in Java. We will call this "HelloServer" implementing the "Runnable" interface. This class will also come under the package "helloworld". The code for the same is given in Listing 6. As you can see, here we are.

Now finally, we have to make the Client, which is written in PHP. We already generated the client side processor from the Thrift IDL. So first, we include all the Thrift run-time libraries by creating a new directory named "thrift" and copy all the php library files available in the directory `/path/to/thrift-version-folder/lib/php/src/` to the newly created directory. Also, create a new sub-directory named "packages" in "thrift" directory and copy the auto-generated PHP package here. Create a new file <client-file>.php adjacent to the "thrift" directory. The contents of the PHP file are given in Listing 7.

We now know how to make a basic Server-Client Application in Thrift which uses different language/platform for both the server and client. We will now try and explore the different available server and client combinations available to us for improving performance. The previous example we gave was of *Blocking* Server and Client. Now we'll make another server and client which is *Non-Blocking* in nature. The codes for Non-Blocking server and client are given in Listing 8 and Listing 9 respectively.

Last but not least, we will try and implement an *Asynchronous* Server and Client. We can write asynchronous clients to call a Thrift service. A callback needs to be registered which will get invoked at successful completion of the request. Blocking mode server didn't work (method invocation returned with an empty response) with the asynchronous client (Maybe it's because we are using TNonblockingSocket at the client side. See construction of HelloService.AsyncClient in Listing 10. So this may be the proper behaviour). Non-blocking mode server given in Listing 8 seems to work without an issue. So you can use the non blocking server from earlier with the client shown in Listing 10 to implement Asynchronous mode.

So this is how we can use different combinations of transports, protocols, and servers to suit our needs without changing much of the code.

It is now worth summarizing the pros and cons of the Thrift approach:

Benefits

- Cross-Language serialization with lower overhead than alternatives such as SOAP due to use of binary format.
- A lean and clean library. Neither any framework to code nor any XML configuration files.
- The language bindings are very natural. For example, Java uses `ArrayList<int>` and similarly C++ uses `std::vector<int>`.
- The application level wire format and the serialization level wire format are clearly separated. They can be modified independently.
- Soft versioning of the protocol. Thrift does not require a centralized and explicit mechanism like major-version/minor-version. Loosely coupled teams can freely evolve RPC calls.
- No build dependencies or non-standard software. No mix of incompatible software licenses.
- Changing the current interface is very easy. The old Client or Server remain compatible with the new Server or Client respectively, silently ignoring any deprecated or newly added fields if necessary.
- Thrift supports a wide variety of languages and environments.
- Service inheritance, subservices implement all functions of their base services and can have additional functions.

Limitations

- Only one service per server. This can be addressed using multiplexing discussed in a short while but still adds to the complexity.
- There can be no cyclic structs. Structs can only contain structs that have been declared before it. A struct also cannot contain itself.
- Important OOP concepts like inheritance and polymorphism are not supported.
- Lacks full documentation.
- Null cannot be returned by a server. Instead a wrapper struct or value is expected.
- No out-of-the-box authentication service available between server and client.
- No Bi-Directional messaging is available.

Multiplexing of services

Despite being a powerful and efficient cross-language communication tool, Thrift's services are challenged by high administrative and maintenance overheads. The fact remains that *every Thrift server is only capable of exposing only one service at a time*. In order to host multiple functions, Thrift provides organizations with the following options:

- Write a monolithic, unwieldy implementation and host it as a single service.
- Host multiple small services across a series of ports.

If we follow the first option, writing monolithic services elevates the cost of development and maintenance as time passes since the complexity increases with the addition of any new service. Even the second option can prove deadly in the long run as the number of ports consumed to host the multiple services keeps on increasing. Ports are a limited resource and need to be judiciously used. Clients will have to maintain too many connections for each service they want to use. Also, as many ports are opened, security needs to be properly scrutinized, which introduces more overhead.

Thus, we introduce multiplexing by extending the Thrift framework to create and host multiple services on each server. The baseline approach is to assign a symbolic name to each service which is referred to as "service context". This will help us host multiple services on each server where each service can be recognized by its respective service context. The basic architecture can be seen in Figure 4.

Let me explain the different components involved:

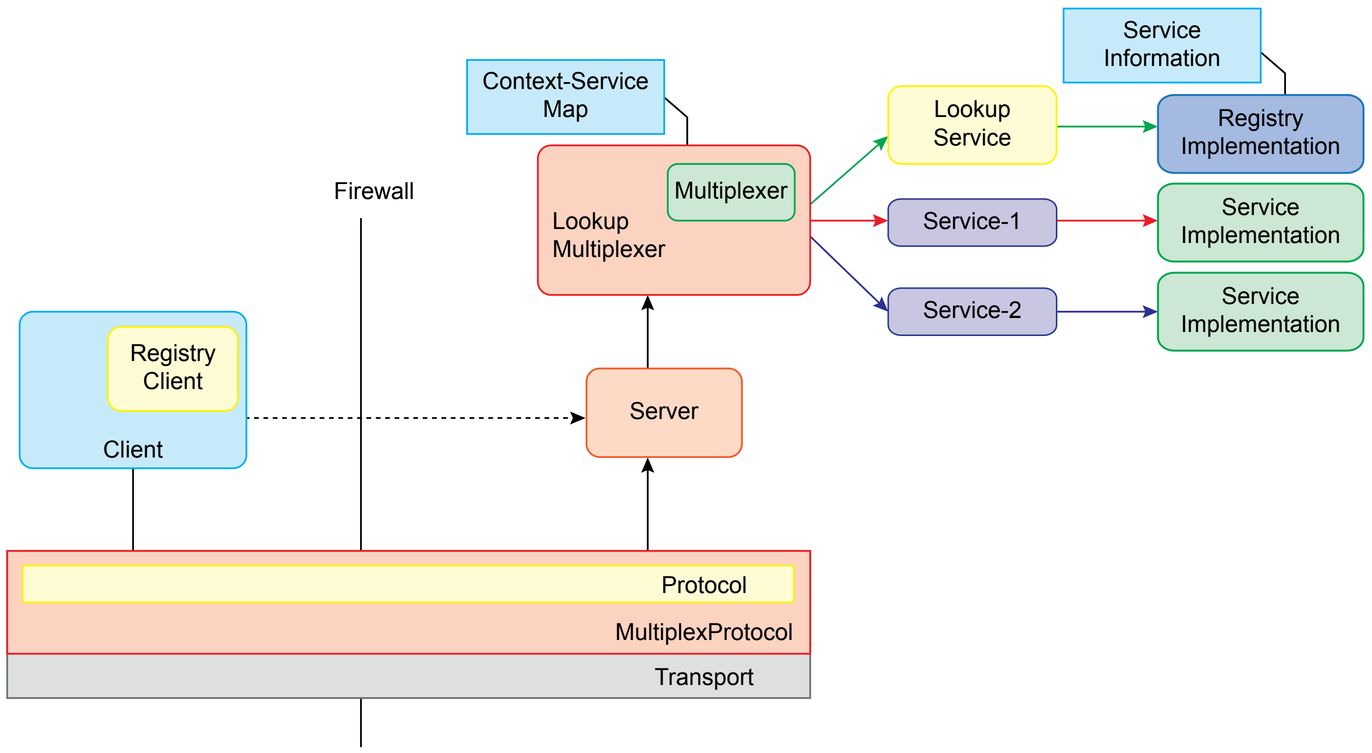


Figure 4. *Thrift Multiplexing architecture with registry lookup*

Multiplexer
This acts as the server side request breaker and identifying the service that the client has requested for, based on the service context provided by the client. This component maintains a mapping between the service context and the available services in that context.

MultiplexProtocol
It is a wrapper class around the underlying protocol that is capable of embedding service context to the message on the client side and fetching it on the server side.

RegistryLookup
In order to reduce the overhead associated with managing the service context manually, we have created a registry system that is responsible for managing meta information about the service context and their services.

ServiceInformation
This class captures/represents the information regarding services on a particular server. This object should be capable of transmitting across the network and hence be used by the client to get service information. Service information consists of service context, service name and description.

MultiplexServer
It is a new abstract server (wrapper to the old Server) which is capable of hosting any server implementation on any transport and protocol using TMultiplexLookup. It basically provides an additional degree of freedom when it comes to hosting a new service on different transport and protocols with no additional coding effort.

Table 1. *Protocol efficiency study.*

Technology	Message Size (The smaller the better)	% larger than TCompact Protocol
Thrift – TCompactProtocol	260	--NA--
Thrift – TBinaryProtocol	445	71.15%
Protocol Buffers	255	-1.9%
Remote Method Invocation	880	238.46%
REST – JSON	536	106.15%
REST – XML	809	211.15%

Comparison With Similar Technologies

As there was no data available on the internet about how different protocols perform against each other, I tried to compare some of the available technologies for RPC between different languages, namely, Apache Thrift (TCompactProtocol, TBinaryProtocol), Google Protocol Buffers, Java Remote Method Invocation, REST using JSON and XML. These tests were conducted on a simple machine available at any home. To compare the protocols, I send out the same amount of data from the different technologies. I used WireShark to capture the sizes that were transferred between Client and Server. You can also perform the same exercise to see the performances for yourself. The results are tabulated and shown in Table 1.

Thrift has a clear advantage in the size of its payload particularly compared to RMI and XML-based REST. Protocol Buffers from Google are effectively the same given that the Protocol Buffers number excludes messaging overhead.

To compare the runtime performances of the different technologies, I used the same computer to run both client and server whose specifications are as follows:

- Operating System: Ubuntu Linux 12.04
- System CPU: Intel Core i5-430M processor
- Memory: 4GB
- Java Version: Java(TM) SE Runtime Environment (build 1.7.0_04-b20)

I made around 100,000 queries from the Client monitoring the CPU usage and the total time taken. The results are tabulated and shown in Table 2.

Table 2. *Runtime Performance comparison*

Technology	CPU usage (SERVER)	CPU Usage (CLIENT)	Average Wall Time (mm:ss)
Thrift – TCompactProtocol	22%	30%	01:12
Thrift – TBinaryProtocol	20%	33%	01:35
Protocol Buffers	41%	28%	01:20
Remote Method Invocation	46%	16%	02:18
REST – JSON	73%	20%	05:42
REST – XML	81%	12%	06:38

The tests yielded some interesting observations. In terms of wall time, Thrift clearly outperformed REST and RMI. In fact, TCompactProtocol took less than 20% of the time it took REST-XML to transmit the same data. The clear dominance of the binary protocols should not be too surprising, as binary data transmission is well-known to have higher performance than text-based protocols. RMI in fact significantly out-performed JSON-based REST in wall time, despite its significantly larger payload size.

References

1. Apache Thrift official: *http://thrift.apache.org/*
2. Wikipedia Page: *http://en.wikipedia.org/wiki/Apache_Thrift*
3. Thrift missing guide: *http://diwakergupta.github.io/thrift-missing-guide/thrift.pdf*
4. Slide on Thrift: *http://www.slideshare.net/talentica/building-scalable-and-language-independent-java-services-using-apache-thrift-6286467*
5. slide on facebook architecture: *http://www.slideshare.net/dleyanlin/facebook-architecture-13994448*
6. Silde on thrift: *http://www.slideshare.net/dvirsky/introduction-to-thrift*
7. java & web develop tips: *http://www.javawebtips.com/117109/*
8. Apache Thrift: *http://jnb.ociweb.com/jnb/jnbJun2009.html*
9. Thrift Home: *http://incubator.apache.org/thrift*
10. Thrift Wiki: *http://wiki.apache.org/thrift*
11. Thrift Installation: *http://wiki.apache.org/thrift/ThriftInstallation*
12. Thrift Installation (Win32): *http://wiki.apache.org/thrift/ThriftInstallationWin32*
13. Wikipedia Page on Variable-Length Quantity(VLQ): *http://en.wikipedia.org/wiki/Variable_length_unsigned_integer*

The CPU percentages yielded some interesting numbers. While the Thrift and Protocol Buffers servers had the highest server CPU percentages, the REST clients had the highest CPU percentages of the clients. For whatever reason, Thrift and REST disproportionately place their CPU loads on their clients and servers. Protocol Buffers balanced its load most evenly between client and server, but then again this was a simple quick hand-rolled server that I wrote for this article. While I did not have time to analyze the cause of the CPU load, the Thrift and Protocol Buffers examples needed to do manual conversion of objects between what is transmitted and what is used. The RMI and REST implementations required no such object conversion. This extra bit of work may account for the additional CPU utilization on the Thrift and Protocol Buffers servers.

Given the poor performance of REST, there may certainly be higher performing servlet containers than Jetty that could be used as part of this test. Jetty was merely chosen because of its relative ease in implementation and ease in bundling the sample code used in this article for download. Doing some quick searches, I found one performance comparison that showed Apache Tomcat to be faster than Jetty and another that showed them at parity. Neither study showed anywhere near a performance difference to make up for the wall time performance of the binary protocols.

All of these technologies are roughly equivalent in the amount of coding complexity required to make them work. This excludes Protocol Buffers of course as it contains no services infrastructure. It should also be noted that Thrift generates all the code you need for a client or server for each language it supports. Java was the server of choice in this article, but other languages could be used if they are better suited – one of the main reasons Thrift was developed in the first place. That being said, I found many of the implementations incomplete. As mentioned previously, the Python implementation, for instance, only had the TBinaryProtocol implemented.

Conclusion

This article introduced you to the tool Apache Thrift and the different ways you can use it and shows how Apache Thrift can fulfill any needs with the flexibility it provides in choosing the different layers of the architecture separately. It outmatches any other similar technology available, even Google protocol buffers, by providing more language support and flexibility. A sneak peek was given on how multiplexity can be used to tackle one of the limitations Apache Thrift has; a full implementation of it can be a whole article in itself and may be discussed in another article.

CHIRAG MAHESHWARI

Chirag is a pre-final year undergraduate (B.Tech) student at the Indian Institute of Technology Guwahati, studying Mathematics and Computing. He is currently working as a Software Developer Intern at S&P Capital IQ, Hyderabad. He's a technology enthusiast and likes to stay updated with all the latest in the field. He can be contacted online at http://digital-madness.in/.

NET OPEN SERVICES IS AN APPLICATION HOSTING COMPANY FOCUSED ON OPEN SOURCE APPLICATIONS MANAGEMENT IN HIGH AVAILABILITY ENVIRONMENT.

NET OPEN SERVICES IS PROUD TO PROVIDE A HIGH QUALITY SERVICE TO OUR CUSTOMERS SINCE 10 YEARS.

OUR EXPERTISE INCLUDES:

- **CLOUD COMPUTING, PUBLIC, PRIVATE AND HYBRID CLOUD MANAGEMENT (OPENSTACK, CLOUDSTACK, RED HAT ENTERPRISE VIRTUALIZATION)**
- **REMOTE MONITORING AND MANAGEMENT 24/7**
- **NETWORKING AND SECURITY (OPEN BSD, IP TABLE, CHECKPOINT, CISCO,...)**
- **OS AND APPLICATION MANAGEMENT (FREE BSD, OPEN BSD, SOLARIS, UNIX, LINUX, AIX, MS WINDOWS)**
- **DATABASE MANAGEMENT (ORACLE, MYSQL, CASSANDRA, NOSQL, MS SQL, SYBASE...)**
- **MANAGED HOSTING IN CARRIER CLASS DATA CENTERS**
- **DISASTER RECOVERY**

Net Open SERVICES

WE PROVIDE SERVICES IN EVERY STEP OF THE PROJECT LIFE, DESIGN, DEPLOYMENT, MANAGEMENT AND EVOLUTIONS. **NETOPENSERVICES** TEAM INCLUDES EXPERIENCED LEADERS AND ENGINEERS IN THE INTERNET SERVER INDUSTRY.

OUR TEAM HAS **15 YEARS OF EXPERIENCE** IN DEVELOPING INTERNET INFRASTRUCTURE-GRADE SOLUTIONS AND PROVISIONING INTERNET DATACENTERS AND GLOBAL SERVICE NETWORKS TOGETHER.

WE OFFER EXCEPTIONAL HARDWARE SUPPORT AS SOFTWARE SUPPORT ON UNIX/LINUX AND OPEN SOURCE APPLICATION. **NETOPENSERVICES** DELIVERS THESE CUSTOM-BUILT LINUX AND UNIX SERVERS, AS WELL AS PRECONFIGURED SERVERS AND SCALABLE STORAGE SOLUTIONS, TO OUR CUSTOMERS. WE ALSO OFFER CUSTOM DEVELOPMENT AND ADVANCED-LEVEL UNIX/LINUX CONSULTING SOLUTIONS.

WWW.NETOPENSERVICES.COM • CONTACT@NETOPENSERVICES.COM

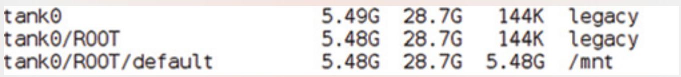
A Closer Look at the Changes in PC-BSD/TrueOS 9.2. Part 1

ZFS Boot Environments

With PC-BSD 9.2 just around the corner, this is a good time to take an advanced look at some of the new functionality coming with it. While there are a number of new features in the works, today we will look specifically at the implementation of ZFS Boot Environments. In coming issues, we will also look at PC-BSD's home directory encryption via PEFS and its PKGNG support. Let us get started by first looking at ZFS Boot Environments and the `beadm` command.

ZFS Boot Environments

While ZFS boot environments have been around in Solaris for a while, they are still relatively new to the FreeBSD ecosystem and many users may be unfamiliar with them. So what is a Boot environment (BE)? Simply put, it is a fancy way to leverage the power of ZFS snapshots to create instant system backups, which can be easily booted from in case of disaster. This is accomplished on PC-BSD and TrueOS by using a specific ZFS dataset layout, the `beadm` utility, and the GRUB boot-loader. In order to make Boot Environments contain the proper data for backup / restore, we have to start with a slightly different layout than a traditional ZFS layout (Figure 1).

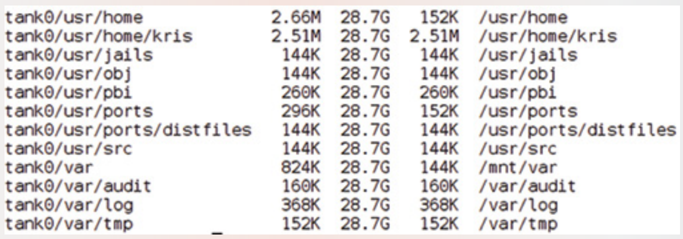


```
tank0                      5.49G  28.7G   144K  legacy
tank0/ROOT                 5.48G  28.7G   144K  legacy
tank0/ROOT/default         5.48G  28.7G  5.48G  /mnt
```

Figure1. *ZFS Layout of the "default" boot environment*

The first change is that we are creating a special `<tank>/ROOT/default` dataset. This dataset is the primary source for taking and cloning ZFS snapshots, which means anything outside of this dataset will *not* be included in a Boot Environment. It will be mounted as '/' on your system. However, for a Boot Environment to work, we will want to include `/usr`, `/usr/local`, and such within our snapshot, while still allowing child datasets such as `/usr/home` and `/usr/jails` to be created. To accomplish this, we create a `/usr` ZFS dataset and set the `canmount=off` flag. The same is done for the `/var` dataset, allowing us to include it in the snapshots, while creating `/var/log`, `/var/tmp`, and `/var/audit` to persist between Boot Environments (Figure 2).

These layout choices ensure that when we take a snapshot of the system, we end up with all the system files and packages necessary to boot back up to a working desktop or server environment. Including `/usr/local` makes Boot Envi-



```
tank0/usr/home              2.66M  28.7G   152K  /usr/home
tank0/usr/home/kris         2.51M  28.7G  2.51M  /usr/home/kris
tank0/usr/jails              144K  28.7G   144K  /usr/jails
tank0/usr/obj                144K  28.7G   144K  /usr/obj
tank0/usr/pbi                260K  28.7G   260K  /usr/pbi
tank0/usr/ports              296K  28.7G   152K  /usr/ports
tank0/usr/ports/distfiles    144K  28.7G   144K  /usr/ports/distfiles
tank0/usr/src                144K  28.7G   144K  /usr/src
tank0/var                    824K  28.7G   144K  /mnt/var
tank0/var/audit              160K  28.7G   160K  /var/audit
tank0/var/log                368K  28.7G   368K  /var/log
tank0/var/tmp                152K  28.7G   152K  /var/tmp
```

Figure 2. *Additional ZFS file-systems excluded from a Boot Environment*

ronments a good choice for backups before performing package updates, because `/usr/local` updates are often just as critical to a user's system as the kernel and world are. With the default layout ready for Boot Environments, we can now take a look at the `beadm` command to manage your backups. The `beadm` command, available in ports under `sysutils/beadm`, provides an easy-to-use framework to create, destroy, and otherwise manage your various Boot Environments. The subcommands we are most interested in will be *list*, *create*, and *destroy*. On a freshly installed system, you will start with a single Boot Environment named "default" (Figure 3).

The "NR" flags indicate that this BE is currently active *Now*, and will still be active on next *Reboot*. Creating a new BE will take a snapshot of the current BE at the current moment, so it would be wise to remember to do this *before* starting anything potentially dangerous. To add a new Boot Environment, simply run the `beadm create <nickname>` command: Figure 4.

As you can see, we have created a new BE nicknamed "newbe". A quick re-run of the list command will show the newly created environment, along with the time and current size cost (Figure 5).

During the creation of the new BE, you probably saw the notice about generating the grub.cfg file. This brings us to the point of putting the "Boot" into Boot Environment. Similar to Solaris, PC-BSD and TrueOS 9.2 include the GRUB boot loader out of box and have it integrated into the `beadm` command in order to provide boot-time functionality. If we now reboot our system, we will be presented with a new Boot Environment menu: Figure 6.

As you can see, our `newbe Boot Environment` has been added to the menu, along with the creation date. By default, a timer exists which will boot the first environment after 5 seconds. However, if you press any key and interrupt this timer, you can arrow up and down to select the environment you wish to boot. Once you have selected the BE you wish to boot, GRUB will load the kernel + modules

```
[root@pcbsd-8613] ~# beadm list
BE      Active Mountpoint  Space Created
default NR     /            5.5G 2013-07-19 16:27
```

Figure 3. *Performing the initial list of Boot Environments*

```
[root@pcbsd-8613] ~# beadm create newbe
Generating grub.cfg ...
Found theme: /boot/grub/themes/pcbsd/theme.txt
done
Created successfully
```

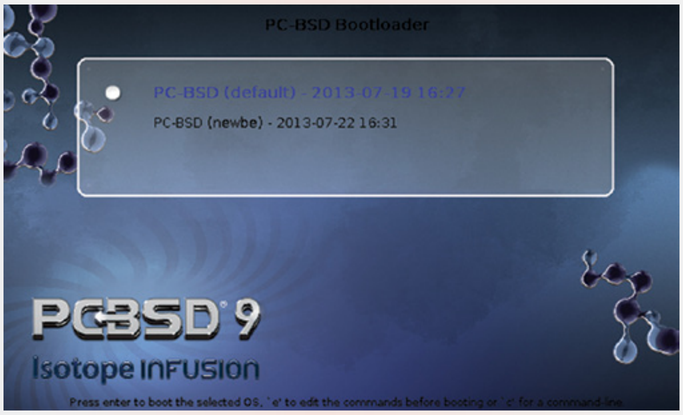
Figure 4. *Creating a new Boot Environment*

```
[root@pcbsd-8613] ~# beadm list
BE      Active Mountpoint  Space Created
default NR     /            5.5G 2013-07-19 16:27
newbe   -      -          672.0K 2013-07-22 16:31
```

Figure 5. *Getting a listing of available Boot-Environments*

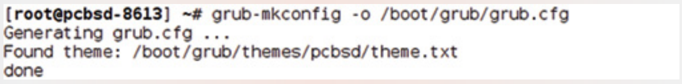
from this BE and boot the system. In addition, the GRUB menu will be updated with your current BEs any time you perform a `beadm create` or `beadm destroy` command.

Since GRUB is performing the initial boot of the system, there are a few new commands and configuration options available to the end user. First among these is the ability to regenerate the `/boot/grub/grub.cfg` file manually. This file is re-created every time you change BEs, but there may be cases when you want to manually re-create it, such as after placing a new module or setting into `/boot/loader.conf`. To start the re-creation, we will use the `grub-mkconfig` command: Figure 7. After running this command, any changes in the original `/boot/grub/grub.cfg` will be lost and replaced with an updated configuration file using details about your current BEs. However, there may be cases where you want



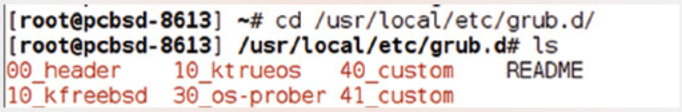
```
PC-BSD Bootloader

PC-BSD (default) - 2013-07-19 16:27
PC-BSD (newbe) - 2013-07-22 16:31

Press enter to boot the selected OS, `e' to edit the commands before booting or `c' for a command-line
```

Figure 6. *Bootloader*



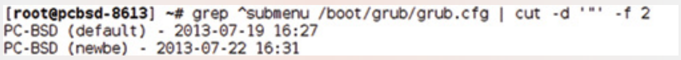
```
[root@pcbsd-8613] ~# grub-mkconfig -o /boot/grub/grub.cfg
Generating grub.cfg ...
Found theme: /boot/grub/themes/pcbsd/theme.txt
done
```

Figure 7. *Rebuilding grub.cfg*



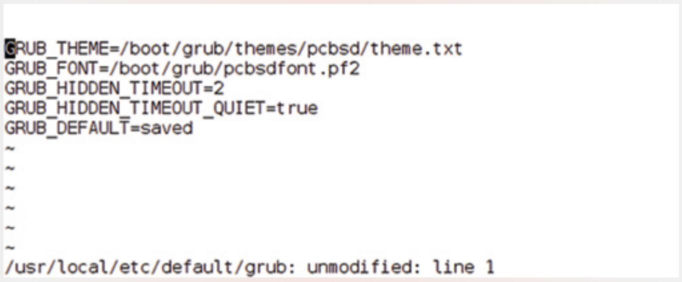
```
[root@pcbsd-8613] ~# cd /usr/local/etc/grub.d/
[root@pcbsd-8613] /usr/local/etc/grub.d# ls
00_header     10_ktrueos    40_custom    README
10_kfreebsd   30_os-prober  41_custom
```

Figure 8. *The GRUB configuration script directory*



```
[root@pcbsd-8613] ~# grep ^submenu /boot/grub/grub.cfg | cut -d '"' -f 2
PC-BSD (default) - 2013-07-19 16:27
PC-BSD (newbe) - 2013-07-22 16:31
```

Figure 9. *Adjusting the /usr/local/etc/default/grub file*



```
GRUB_THEME=/boot/grub/themes/pcbsd/theme.txt
GRUB_FONT=/boot/grub/pcbsdfont.pf2
GRUB_HIDDEN_TIMEOUT=2
GRUB_HIDDEN_TIMEOUT_QUIET=true
GRUB_DEFAULT=saved
~
~
~
~
~
~
/usr/local/etc/default/grub: unmodified: line 1
```

Figure 10. *Getting a list of BEs for booting*

to adjust your boot menu with other options, such as a new OS as an additional boot option or changing the default BE to boot after the countdown. Let's take a look at how to accomplish these tasks (Figure 8). The first directory of importance is `/usr/local/etc/grub.d`. Within this directory is a collection of files starting with a numerical prefix. These files are shellscripts, run by the `grub-mkconfig` command which echo out various parts of the resulting `grub.cfg` file. These scripts will normally be replaced when updating the GRUB package, however, it is possible to add your own scripts to this directory. These new scripts will then be executed along with the others and their output added to the end of the grub.cfg file created by the `grub-mkconfig` command. If you have a fairly static group of options, it is also possible to create a `/boot/grub/custom.cfg` file, which will be included at the tail end of the grub configuration. Setting the default GRUB Boot Environment or OS entry is also easy. To get started, we will need to edit the file `/usr/local/etc/default/grub` and add the `GRUB_DEFAULT=saved` line (Figure 9).

Anytime we adjust the grub defaults file, we will need to re-create the grub configuration file using the command `grub-mkconfig -o /boot/grub/grub.cfg`. After `grub-mkconfig` finishes, we can then set the default boot menu option with the `grub-set-default` command. This command will take a numerical argument for the particular menu entry to boot, starting with 0. So the first entry will be 0, the second will be 1, and so on. Using the `grep` command, you can easily see which Boot Environments are available for booting. (NOTE: after setting a default Boot Environment with the `grub-set-default` command, if you remove a BE you may change the order of the environments and you should check your grub.cfg file to make sure you are still booting the desired BE) (see Figure 10).

In this article, we've taken a look at how PC-BSD and TrueOS use `ZFS`, `beadm` and `GRUB` to manage backups and set the booting of different Boot Environments. In the coming months, we expect to add new GUI utilities to assist desktop users with the management of these technologies, and users who want to stay current with these changes are encouraged to join us on the PC-BSD Testing & Development mailing lists (*http://lists.pcbsd.org/mailman/listinfo*).

KEN MOORE

Ken Moore co-created EasyPBI with Jesse Smith in 2011 and took over full development of it for the PC-BSD project in 2012. He lives in Tennessee with his wife and two sons and is always looking for ways to make computers simpler, but no less powerful, for the average user. He is currently employed by iXsystems to work on the PC-BSD Project as both a developer and as the manager for the PC-BSD PBI repository. He can be reached at: ken@pcbsd.org.

What do all these have in common?

They all use Nipper Studio

to audit their firewalls, switches & routers

Nipper Studio is an award winning configuration auditing tool which analyses vulnerabilities and security weaknesses. You can use our point and click interface or automate using scripts. Reports show:

1) Severity of the Threat & Ease of Resolution

2) Configuration Change Tracking & Analysis

3) Potential Solutions including Command Line Fixes to resolve the Issue

Nipper Studio doesn't produce any network traffic, doesn't need to interact directly with devices and can be used in secure environments.

SME pricing from £650 scaling to enterprise level

evaluate for free at www.titania.com

2012 Computing Security Awards
WINNER
Enterprise Security Solution of the Year

2012 Computing Security Awards
WINNER
Network Security Solution of the Year

2012 Computing Security Awards
Runner-up
SME Security Solution of the Year

titania nipper STUDIO

www.titania.com
T: +44 (0) 1905 888785

An email Gateway

with FreeBSD to Prevent Malware and Undesirable Messages

Controlling inbound and outbound mail messages is a big challenge for sysadmins. Malware can spread quickly, infecting dozens of mailboxes and relentless spammers send thousands of messages with unsolicited advertising and phishing scams.

What you will learn…

- Basic email gateway setup

What you should know…

- FreeBSD shell command line
- Knowledge of the SMTP, POP and IMAP protocols

This article will show you how to prevent viruses, spam and malware by configuring a FreeBSD email gateway system using MailScanner, SpamAssassin, ClamAV and Postfix. The email gateway is responsible for analyzing, filtering, and cleaning or removing malicious files and messages (Figure 1). Figure 2 shows how the mail is scanned by the MailScanner. It is delivered to the incoming folder where it is analyzed by SpamAssassin and ClamAV and then delivered to the quarantine or queue folders. The cleaned messages are then delivered to the internal mail server.

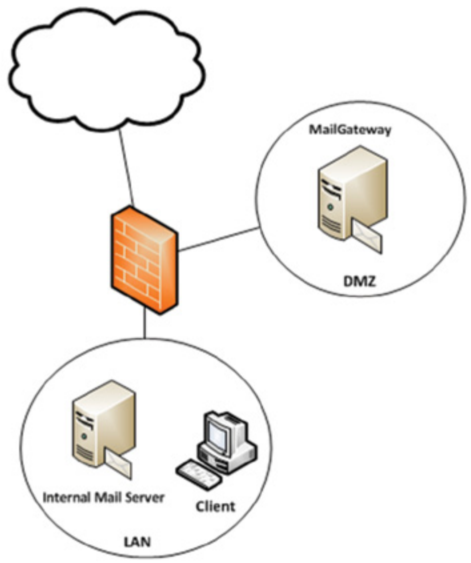


Figure 1. *Basic mail gateway topology*

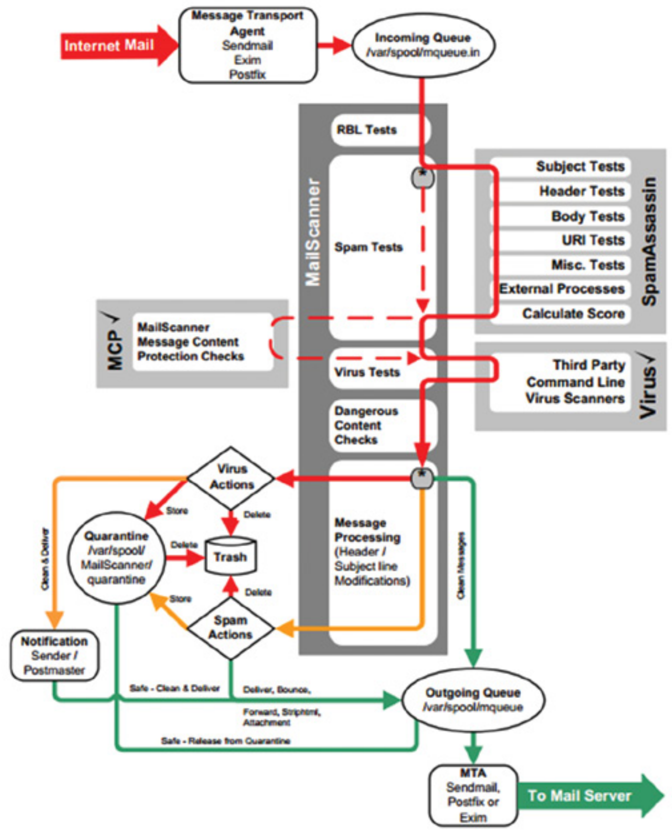


Figure 2. *MailScanner process flow*

Installing

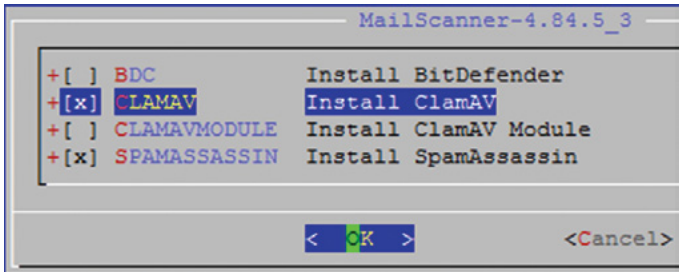
Before you install the applications, it is necessary to update the FreeBSD ports tree to ensure you use the patched versions.

```
#cd /usr/ports
#portsnap fetch update
```

Installing Postfix MTA.

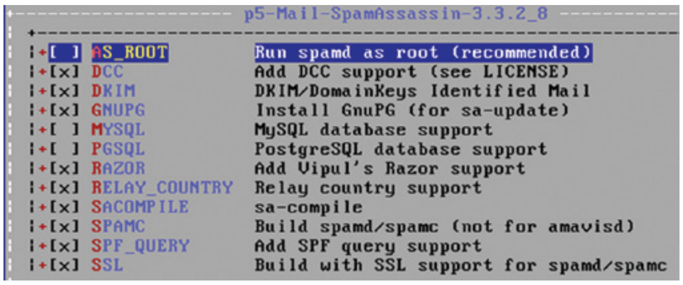
```
#cd /usr/ports/mail/postfix
#make install clean
#cd /usr/local/etc/postfix
#postalias aliases
```

Installing MailScanner.



```
MailScanner-4.84.5_3

+[ ] BDC              Install BitDefender
+[x] CLAMAV           Install ClamAV
+[ ] CLAMAVMODULE     Install ClamAV Module
+[x] SPAMASSASSIN     Install SpamAssassin

        <  OK  >                <Cancel>
```

Figure 3. *Installing MailScanner and dependencies*



```
p5-Mail-SpamAssassin-3.3.2_8

+[ ] AS_ROOT          Run spamd as root (recommended)
+[x] DCC              Add DCC support (see LICENSE)
+[x] DKIM             DKIM/DomainKeys Identified Mail
+[x] GNUPG            Install GnuPG (for sa-update)
+[ ] MYSQL            MySQL database support
+[ ] PGSQL            PostgreSQL database support
+[x] RAZOR            Add Vipul's Razor support
+[x] RELAY_COUNTRY    Relay country support
+[x] SACOMPILE        sa-compile
+[x] SPAMC            Build spamd/spamc (not for amavisd)
+[x] SPF_QUERY        Add SPF query support
+[x] SSL              Build with SSL support for spamd/spamc
```

Figure 4. *Disabling spamd as root*

```
#cd /usr/ports/mail/mailscanner
#make install clean
```

In the SpamAssassin options, uncheck the AS_ROOT for security reasons (Figure 4).

Configuring Postfix MTA

Configure Postfix by editing the `/usr/local/etc/postfix/main.cf` file (Listing 1), with some basic parameters described below:

- myhostname is the internet hostname of this mail system, use the FQDN;
- mydestination is the list of domains that are delivered via the mail delivery transport;
- mynetworks_style is the method used to generate the default value for the mynetworks parameter. The host value is when Postfix should "trust" only the local machine.

Create the transport_maps (Listing 2) to set the next hop; in our case, that is the internal mail server and header_checks (Listing 3) for content inspection of primary non-MIME message headers.

```
#cd /usr/local/etc/postfix
#vi transport
#portmap transport
#cd /usr/local/etc/postfix
#vi header_checks
```

The Sendmail is the default MTA in FreeBSD, so it is necessary to disable it and enable Postfix to start at boot (Listing 4).

Listing 1. *Basic settings in main.cf*

```
myhostname = mailgw.acme.local
mydomain = localhost
inet_interfaces = all
mydestination = $myhostname, localhost.$mydomain,
                    localhost
mynetworks_style = host
mynetworks = 192.168.0.0/24, 127.0.0.0/8
relay_domains = acme.local
transport_maps = hash:/usr/local/etc/postfix/transport
header_checks = regexp:/usr/local/etc/postfix/header_
                    checks
```

Listing 2. *The /usr/local/etc/postfix/transport file content*

```
acme.local smtp:[192.168.0.2]
```

Listing 3. *The /usr/local/etc/postfix/header_checks file content*

```
/^Received:/ HOLD
```

Listing 4. *Disabling Sendmail and enabling Postfix in the /etc/rc.conf file*

```
sendmail_enable="NONE"
sendmail_msp_queue_enable="NO"
sendmail_outbound_enable="NO"
sendmail_submit_enable="NO"
postfix_enable="YES"
```

Listing 5. *Configuring Mailscanner*

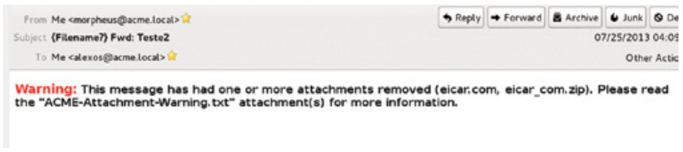
```
%org-name% = ACME
%web-site% = www.acme.local
Run As User = postfix
Run As Group = postfix
Incoming Queue Dir = /var/spool/postfix/hold
Outgoing Queue Dir = /var/spool/postfix/incoming
MTA = postfix
Virus Scanners = clamav
Still Deliver Silent Viruses = yes
SpamAssassin User State Dir = /var/spool/MailScanner/
                    spamassassin
}
```

Listing 6. *Enabling Mailscanner, ClamAV and SpamAssassin in the /etc/rc.conf file*

```
mailscanner_enable="YES"
spamass_milter_enable="YES"
spamd_enable="YES"
clamav_clamd_enable="YES"
clamav_milter_enable="YES"
clamav_freshclam_enable="YES"
```

Listing 7. *DNS example*

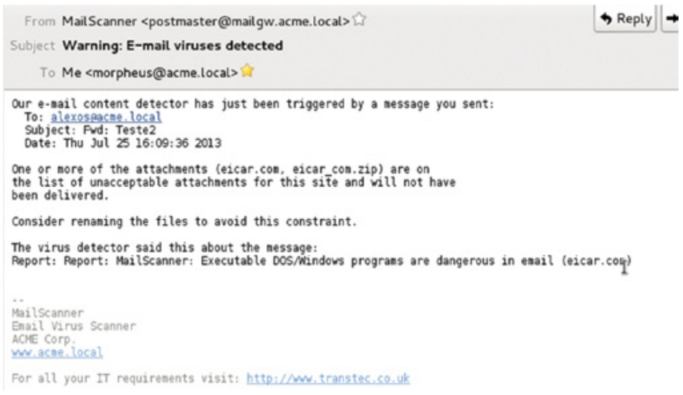
```
acme.local. IN MX 10 mailgw.acme.local.
acme.local. IN A 192.168.0.2
mail IN A 192.168.0.2
mailgw IN A 192.168.0.1
}
```



From Me <morpheus@acme.local>
Subject (Filename?) Fwd: Teste2
To Me <alexos@acme.local>
07/25/2013 04:09
Other Actio

Warning: This message has had one or more attachments removed (eicar.com, eicar_com.zip). Please read the "ACME-Attachment-Warning.txt" attachment(s) for more information.

Figure 5. *Message cleaned*



From MailScanner <postmaster@mailgw.acme.local>
Subject Warning: E-mail viruses detected
To Me <morpheus@acme.local>

```
Our e-mail content detector has just been triggered by a message you sent:
  To: alexos@acme.local
  Subject: Fwd: Teste2
  Date: Thu Jul 25 16:09:36 2013

One or more of the attachments (eicar.com, eicar_com.zip) are on
the list of unacceptable attachments for this site and will not have
been delivered.

Consider renaming the files to avoid this constraint.

The virus detector said this about the message:
Report: Report: MailScanner: Executable DOS/Windows programs are dangerous in email (eicar.com)


--
MailScanner
Email Virus Scanner
ACME Corp.
www.acme.local

For all your IT requirements visit: http://www.transtec.co.uk
```

Figure 6. *Alert to the sender*

On The Web

- Postfix official Web Site: *http://www.postfix.org/*
- MailScanner Web Site: *http://www.mailscanner.info/*
- SpamAssassin Web Site: *https://spamassassin.apache.org/*
- MailScanner User Guide: *http://www.mailscanner.info/files/MailScanner-Guide.pdf*

Configuring MailScanner

Configure MailScanner by editing the `/usr/local/etc/MailScanner/MailScanner.conf` file, to manage the entire mail security system. Listing 5 shows the basic parameters necessary to configure it. The parameter Still Deliver Silent Viruses should not be used in a production environment, because the system will send thousands of MailScanner messages to the users. Improve MailScanner performance by increasing the Max Children parameter. Keep in mind each process consumes 20 MB. Create directory and configure necessary permissions.

```
#mkdir -p /var/spool/Mailscanner/incoming && mkdir /var/
spool/Mailscanner/quarantine && mkdir /var/spool/
                    Mailscanner/spamassassin
#chown -R postfix:postfix /var/spool/Mailscanner
#chmod -R 775 /var/spool/Mailscanner
```

Edit the `/etc/rc.conf` file to enable MailScanner, ClamAV and SpamAssassin to start at boot (Listing 6).

Testing

To test your system, you need to configure the SMTP email client to use the email gateway. In my tests, I configured an MX record pointing to the email gateway (Listing 7) set in my client. I used the EICAR file to test the malware detection and system response. When sending an infected file, the system cleans the message, forwards it to the recipient (Figure 5) and alerts the sender (Figure 6).

Conclusion

The MailScanner is a very powerful security tool and one should test several configurations to best understand how it can help one's mail service. MailScanner helps in the effective management of your mail security system that scales well in larger environments.

ALEXANDRO SILVA AKA ALEXOS

Alexandro Silva aka Alexos lives in Salvador, Bahia, Brasil. He is an Information Security Consultant at iBliss Segurança & Inteligência. He has been using FreeBSD since the 4.11 release and can be reached on line at http://alexos.org.

VERISIGN®

vBSDCon · SAVE THE DATE!

DULLES, VA · OCTOBER 25-27, 2013

Please join us October 25-27, 2013 at the Hyatt in Dulles, Virginia for the first biennial vBSDCon event. This exciting weekend will bring together members of the BSD community for a series of roundtable discussions, educational sessions, best practice conversations, and exclusive networking opportunities. See below for details on this industry weekend not to be missed:

AGENDA

- **Friday, October 25:** Evening Reception
- **Saturday, October 26:** General Session, Birds of a Feather Sessions
- **Sunday, October 27:** General Session, Breakout Sessions

WHO SHOULD ATTEND

- Developers • Engineers • Administrators • Innovators

TOPICS

- PkgNG w/ Baptiste Daroussin
- A comprehensive look at bsdinstall with Devin Teske
- Netflix Demo/Presentation with Scott Long
- netmap with Luigi Rizzo
- Migration from GCC to LLVM/Clang with David Chisnall

REGISTRATION INFORMATION WILL BE SENT TO YOU IN MAY!

Questions? Please contact: eventsteam@verisign.com

VerisignInc.com

The Service Spawner

If you have ever dreamed of software that does user-side installation, software configuration, and maintains your software automatically without the need for a vast amount of UNIX systems knowledge, you should probably take a look at the universal Service Spawner.

What you will learn…

- How software is maintained in the Open Source world.
- How to maintain your applications easily on your servers.

What you should know…

- TheSS is based on software built by Sofin. You should know basic Sofin principles (BSD Mag from June 2013).
- If you want to make your own service igniters, you should have (at least) basic knowledge of shell scripting and JSON scripting.

Last month, I wrote about Sofin – Software Installer. The story continues and this time, it is about software built on top of architecture used by Sofin – The Service Spawner.

It is common in small and medium-sized companies where there's "monit", or a similar software combined with system RC configurations, used to launch all user services. In the simple case, this is probably enough, but the fun starts when you have dozens of users, each with different server configuration demands. After a while, you end up with tons of manually written configurations and scripts, all on the root side, so you then need to perform each change manually for each user demand. I could play dumb and do some sort of a hack with sudo, but it would work only for users who know *monit* configuration syntax. How many of your users do that? So I assumed that the whole user service configuration should be located on the user-side, not the system side. It should be easy and straightforward for any user to spawn a new service. So I raged, and it happened.

The *SS* is a system agnostic mechanism to maintain server software, written in C / C++ using the Qt4 framework (by Trolltech, currently Nokia). It might be considered controversial that I used a framework designed for GUI applications to write server software, but I consider Qt4 to be one of the best cross-platform, high level API's available for C++. Besides, TheSS requires only the QtCore part of the Qt framework to build (Sofin already provides definition with only the core part of Qt without any GUI). That is enough for the introduction. Let's talk about some specifics.

Design Assumptions

I started work on TheSS with a simple idea: "I want to spawn any kind of server service, from database servers, to web app with ease". The goal was to create an ultimate solution which will provide:

- service installation
- service configuration
- service starting/ restarting / autostarting
- service dependencies support (with explicit order of execution)
- service scheduling (built in CRON-compliant asynchronous scheduler)
- service TCP port checking, random port generation for services (including support for static ports)
- automatic service monitoring (including watches on: TCP/UDP ports, PID files, process status and custom availability checks defined in shell script)
- flexibility (with JSON "software igniters")
- easy domain maintenance for services (including built-in domain resolving)
- support for expectations and problem notifications (low level notification centre)

- ability to upgrade itself without shutting down running services (live upgrades)
- easy to use management panel
- live debugging (log level trigger while running)
- work both for regular users (user services) and superuser (root services)
- distribution model in mind
- simplicity in mind (KISS)
- … and more

There were many more design assumptions on which TheSS was based. I will mention only those most important, to help you understand how it works under the hood.

Usually, on standard systems, there are a couple of locations where the service configuration might be placed. They are `/etc/mysoftware`, or `~/.mysoftware`. But sometimes, you will find them also in `/usr/local/etc` (software from FreeBSD ports) or even more fancy places – depending on the build configuration, operating system and so on. TheSS is software that always keeps *everything* – from service configuration, environment, pids, sockets, logs and data – in one place, which is `~/SoftwareData/Mysoft` for user and `/SystemUsers/SoftwareData/Mysoft for root`. The structure of "SoftwareData", for example service called "Mysoft" is:

- …/SoftwareData/Mysoft/.autostart – service autostart trigger (TheSS will autostart the service if such a file exists)
- …/SoftwareData/Mysoft/.domain – service domain (might be set in "igniter")
- …/SoftwareData/Mysoft/.ports/0 – service default port file (might be pool of ports)
- …/SoftwareData/Mysoft/service.conf – service configuration
- …/SoftwareData/Mysoft/service.pid – service pid
- …/SoftwareData/Mysoft/service.sock – service UNIX socket file
- …/SoftwareData/Mysoft/service.env – service environment settings
- …/SoftwareData/Mysoft/service.log – service log
- …/SoftwareData/Mysoft/database/ – service database directory (only if your service is some kind of database)
- …/SoftwareData/Mysoft/app/ – service application data directory (i.e., web application root)

a d v e r i s e m e n t

m:tier

www.mtier.org
contact@mtier.org

- BSD development and consultancy
- Zabbix Monitoring
- Bacula enterprise backup
- BSD Thin Client
- Corporate BSD Desktop
- Solution management with Puppet
- and more ...

It is turning the current FHS models inside out, but there are more upsides than downsides for this solution. It gives you unification and flexibility of configuration for each service, which is very important to implement automatic service management well.

Service Igniters

To be flexible enough, all services spawned by TheSS have a form of JSON "igniter". The igniter is just a JSON file (with comments support) with defined options for services. By default, TheSS will look for igniters in:

For regular user:

- /Common/Igniters/Services
- ~/Igniters/Services

For root:

- /Common/Igniters/Services
- /SystemUsers/Igniters/Services

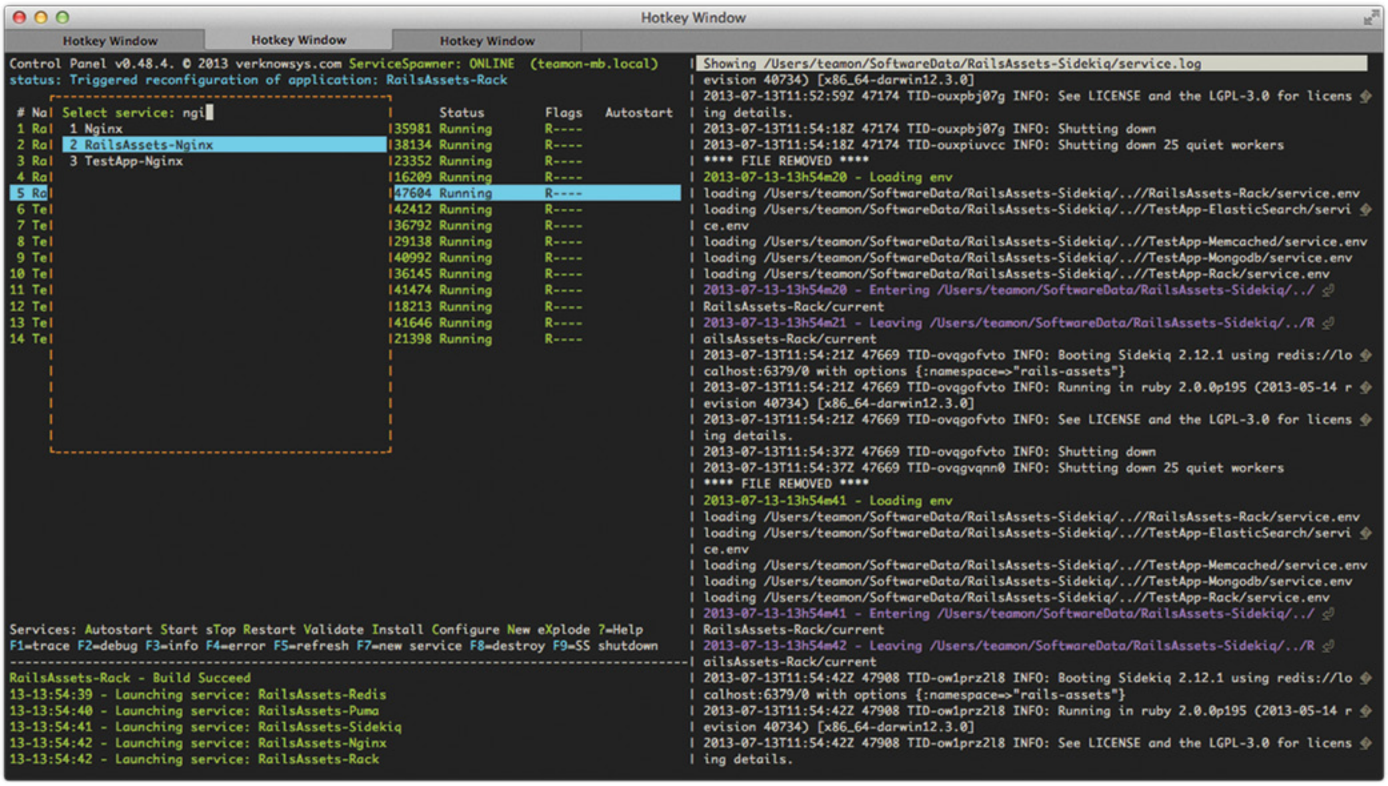
Each igniter may implement its own "service hooks":

- configure – creates service.conf if it doesn't exist
- reconfigure – recreates service.conf, and restarts service
- start – starts service
- afterStart – is called after service start
- stop – stops service
- afterStop – is called after service stop
- restart – performs stop and start of service
- validate – used by service monitoring mechanism, service won't start if this validation fails
- reload – by default sends SIGHUP to service
- install – installs service if it's not installed

Each hook consists of "commands" and "expectations" where commands are just plain Shell script, and expectations are the way of being notified about hook failures. Similar to Sofin, TheSS has Default.json "super igniter" which is the core of all defined service igniters. The basic idea was an ability to write igniters by hand (simple cases) but also to give support for any kind of utility to generate those automatically. One of those utilities written in Ruby, by Tymon (teamon) Tobolski, is on the way. It is called Hussar and is not released yet.

A major feature of igniters are built in constants. Igniter constants are automatically replaced with proper values before invoking every hook. Currently there are:

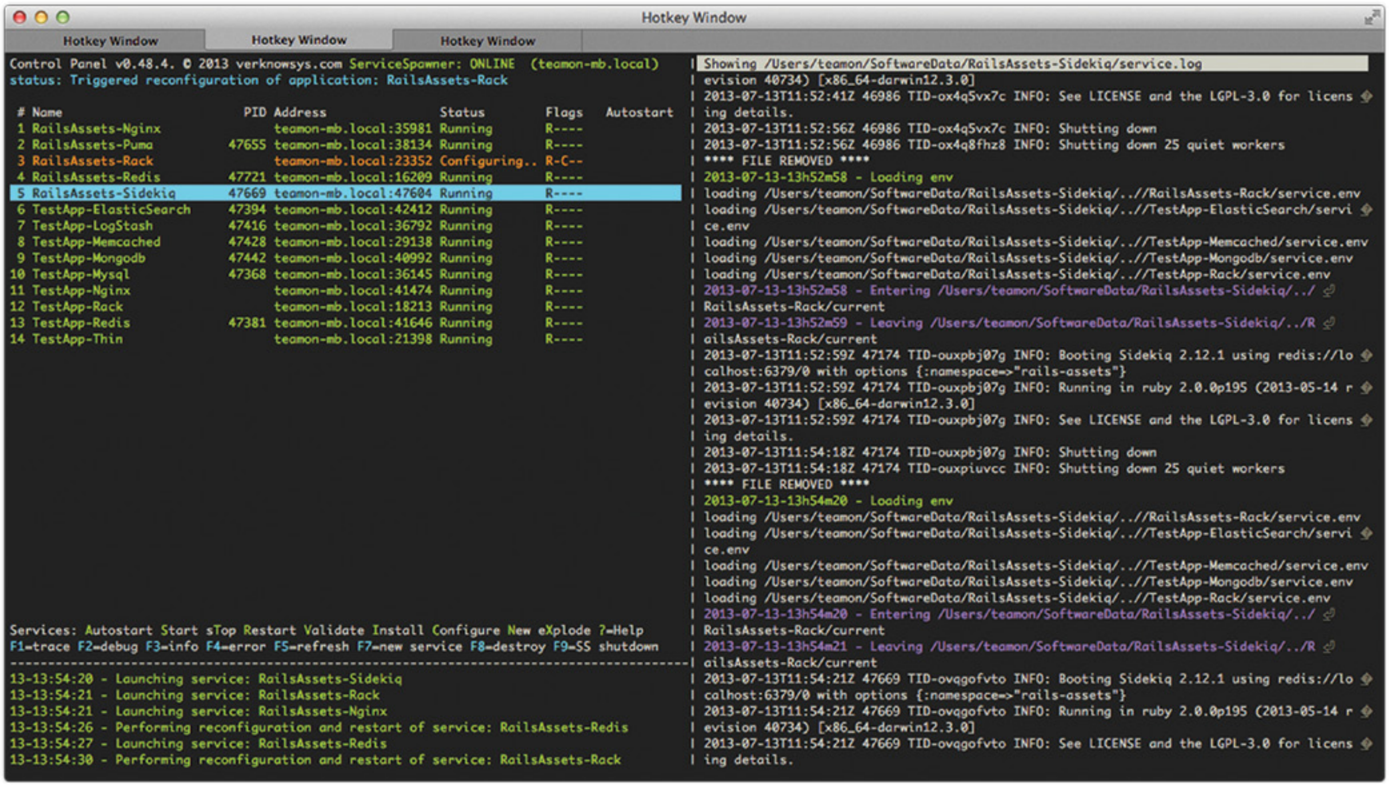
- SERVICE_PREFIX – service prefix directory (f.e. `~/SoftwareData/AppName/`)



```
Control Panel v0.48.4. © 2013 verknowsys.com ServiceSpawner: ONLINE  (teamon-mb.local)
status: Triggered reconfiguration of application: RailsAssets-Rack
     ---------------------------------------
 # Na| Select service: ngi               |        Status      Flags    Autostart
 1 Ra|  1 Nginx                          |35981 Running     R----
 2 Ra|  2 RailsAssets-Nginx              |38134 Running     R----
 3 Ra|  3 TestApp-Nginx                  |23352 Running     R----
 4 Ra|                                   |16209 Running     R----
 5 Ra|                                   |47604 Running     R----
 6 Te|                                   |42412 Running     R----
 7 Te|                                   |36792 Running     R----
 8 Te|                                   |29138 Running     R----
 9 Te|                                   |40992 Running     R----
10 Te|                                   |36145 Running     R----
11 Te|                                   |41474 Running     R----
12 Te|                                   |18213 Running     R----
13 Te|                                   |41646 Running     R----
14 Te|                                   |21398 Running     R----
     |                                   |
     ---------------------------------------

Services: Autostart Start sTop Restart Validate Install Configure New eXplode ?=Help
F1=trace F2=debug F3=info F4=error F5=refresh F7=new service F8=destroy F9=SS shutdown
--------------------------------------------------------------------------------------
RailsAssets-Rack - Build Succeed
13-13:54:39 - Launching service: RailsAssets-Redis
13-13:54:40 - Launching service: RailsAssets-Puma
13-13:54:41 - Launching service: RailsAssets-Sidekiq
13-13:54:42 - Launching service: RailsAssets-Nginx
13-13:54:42 - Launching service: RailsAssets-Rack
```

```
| Showing /Users/teamon/SoftwareData/RailsAssets-Sidekiq/service.log
| evision 40734) [x86_64-darwin12.3.0]
| 2013-07-13T11:52:59Z 47174 TID-ouxpbj07g INFO: See LICENSE and the LGPL-3.0 for licens
| ing details.
| 2013-07-13T11:54:18Z 47174 TID-ouxpbj07g INFO: Shutting down
| 2013-07-13T11:54:18Z 47174 TID-ouxpiuvcc INFO: Shutting down 25 quiet workers
| **** FILE REMOVED ****
| 2013-07-13-13h54m20 - Loading env
| loading /Users/teamon/SoftwareData/RailsAssets-Sidekiq/..//RailsAssets-Rack/service.env
| loading /Users/teamon/SoftwareData/RailsAssets-Sidekiq/..//TestApp-ElasticSearch/servi
| ce.env
| loading /Users/teamon/SoftwareData/RailsAssets-Sidekiq/..//TestApp-Memcached/service.env
| loading /Users/teamon/SoftwareData/RailsAssets-Sidekiq/..//TestApp-Mongodb/service.env
| loading /Users/teamon/SoftwareData/RailsAssets-Sidekiq/..//TestApp-Rack/service.env
| 2013-07-13-13h54m20 - Entering /Users/teamon/SoftwareData/RailsAssets-Sidekiq/../
| RailsAssets-Rack/current
| 2013-07-13-13h54m21 - Leaving /Users/teamon/SoftwareData/RailsAssets-Sidekiq/../R
| ailsAssets-Rack/current
| 2013-07-13T11:54:21Z 47669 TID-ovqgofvto INFO: Booting Sidekiq 2.12.1 using redis://lo
| calhost:6379/0 with options {:namespace=>"rails-assets"}
| 2013-07-13T11:54:21Z 47669 TID-ovqgofvto INFO: Running in ruby 2.0.0p195 (2013-05-14 r
| evision 40734) [x86_64-darwin12.3.0]
| 2013-07-13T11:54:21Z 47669 TID-ovqgofvto INFO: See LICENSE and the LGPL-3.0 for licens
| ing details.
| 2013-07-13T11:54:37Z 47669 TID-ovqgofvto INFO: Shutting down
| 2013-07-13T11:54:37Z 47669 TID-ovqgvqnn0 INFO: Shutting down 25 quiet workers
| **** FILE REMOVED ****
| 2013-07-13-13h54m41 - Loading env
| loading /Users/teamon/SoftwareData/RailsAssets-Sidekiq/..//RailsAssets-Rack/service.env
| loading /Users/teamon/SoftwareData/RailsAssets-Sidekiq/..//TestApp-ElasticSearch/servi
| ce.env
| loading /Users/teamon/SoftwareData/RailsAssets-Sidekiq/..//TestApp-Memcached/service.env
| loading /Users/teamon/SoftwareData/RailsAssets-Sidekiq/..//TestApp-Mongodb/service.env
| loading /Users/teamon/SoftwareData/RailsAssets-Sidekiq/..//TestApp-Rack/service.env
| 2013-07-13-13h54m41 - Entering /Users/teamon/SoftwareData/RailsAssets-Sidekiq/../
| RailsAssets-Rack/current
| 2013-07-13-13h54m42 - Leaving /Users/teamon/SoftwareData/RailsAssets-Sidekiq/../R
| ailsAssets-Rack/current
| 2013-07-13T11:54:42Z 47908 TID-ow1prz2l8 INFO: Booting Sidekiq 2.12.1 using redis://lo
| calhost:6379/0 with options {:namespace=>"rails-assets"}
| 2013-07-13T11:54:42Z 47908 TID-ow1prz2l8 INFO: Running in ruby 2.0.0p195 (2013-05-14 r
| evision 40734) [x86_64-darwin12.3.0]
| 2013-07-13T11:54:42Z 47908 TID-ow1prz2l8 INFO: See LICENSE and the LGPL-3.0 for licens
| ing details.
```

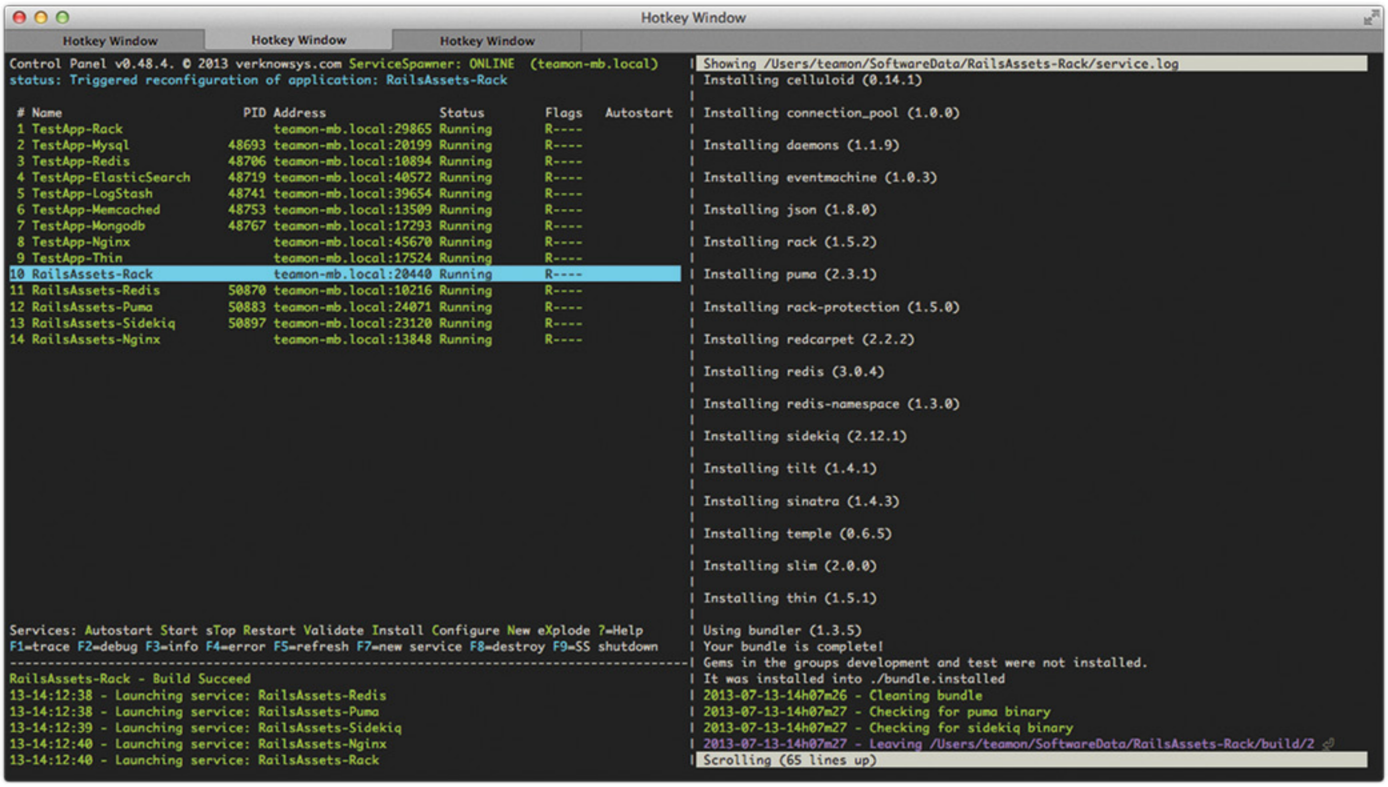
Figure 1. *Panel – initializing new service*



```
Control Panel v0.48.4. © 2013 verknowsys.com ServiceSpawner: ONLINE  (teamon-mb.local)
status: Triggered reconfiguration of application: RailsAssets-Rack

 # Name                     PID Address              Status        Flags   Autostart
 1 RailsAssets-Nginx            teamon-mb.local:35981 Running       R----
 2 RailsAssets-Puma       47655 teamon-mb.local:38134 Running       R----
 3 RailsAssets-Rack             teamon-mb.local:23352 Configuring.. R-C--
 4 RailsAssets-Redis      47721 teamon-mb.local:16209 Running       R----
 5 RailsAssets-Sidekiq    47669 teamon-mb.local:47604 Running       R----
 6 TestApp-ElasticSearch  47394 teamon-mb.local:42412 Running       R----
 7 TestApp-LogStash       47416 teamon-mb.local:36792 Running       R----
 8 TestApp-Memcached      47428 teamon-mb.local:29138 Running       R----
 9 TestApp-Mongodb        47442 teamon-mb.local:40992 Running       R----
10 TestApp-Mysql          47368 teamon-mb.local:36145 Running       R----
11 TestApp-Nginx                teamon-mb.local:41474 Running       R----
12 TestApp-Rack                 teamon-mb.local:18213 Running       R----
13 TestApp-Redis          47381 teamon-mb.local:41646 Running       R----
14 TestApp-Thin                 teamon-mb.local:21398 Running       R----

Services: Autostart Start sTop Restart Validate Install Configure New eXplode ?=Help
F1=trace F2=debug F3=info F4=error F5=refresh F7=new service F8=destroy F9=SS shutdown
-----------------------------------------------------------------------------------------
13-13:54:20 - Launching service: RailsAssets-Sidekiq
13-13:54:21 - Launching service: RailsAssets-Rack
13-13:54:21 - Launching service: RailsAssets-Nginx
13-13:54:26 - Performing reconfiguration and restart of service: RailsAssets-Redis
13-13:54:27 - Launching service: RailsAssets-Redis
13-13:54:30 - Performing reconfiguration and restart of service: RailsAssets-Rack
```

```
| Showing /Users/teamon/SoftwareData/RailsAssets-Sidekiq/service.log
| evision 40734) [x86_64-darwin12.3.0]
| 2013-07-13T11:52:41Z 46986 TID-ox4q5vx7c INFO: See LICENSE and the LGPL-3.0 for licens
| ing details.
| 2013-07-13T11:52:56Z 46986 TID-ox4q5vx7c INFO: Shutting down
| 2013-07-13T11:52:56Z 46986 TID-ox4q8fhz8 INFO: Shutting down 25 quiet workers
| **** FILE REMOVED ****
| 2013-07-13-13h52m58 - Loading env
| loading /Users/teamon/SoftwareData/RailsAssets-Sidekiq/..//RailsAssets-Rack/service.env
| loading /Users/teamon/SoftwareData/RailsAssets-Sidekiq/..//TestApp-ElasticSearch/servi
| ce.env
| loading /Users/teamon/SoftwareData/RailsAssets-Sidekiq/..//TestApp-Memcached/service.env
| loading /Users/teamon/SoftwareData/RailsAssets-Sidekiq/..//TestApp-Mongodb/service.env
| loading /Users/teamon/SoftwareData/RailsAssets-Sidekiq/..//TestApp-Rack/service.env
| 2013-07-13-13h52m58 - Entering /Users/teamon/SoftwareData/RailsAssets-Sidekiq/../
| RailsAssets-Rack/current
| 2013-07-13-13h52m59 - Leaving /Users/teamon/SoftwareData/RailsAssets-Sidekiq/../R
| ailsAssets-Rack/current
| 2013-07-13T11:52:59Z 47174 TID-ouxpbj07g INFO: Booting Sidekiq 2.12.1 using redis://lo
| calhost:6379/0 with options {:namespace=>"rails-assets"}
| 2013-07-13T11:52:59Z 47174 TID-ouxpbj07g INFO: Running in ruby 2.0.0p195 (2013-05-14 r
| evision 40734) [x86_64-darwin12.3.0]
| 2013-07-13T11:52:59Z 47174 TID-ouxpbj07g INFO: See LICENSE and the LGPL-3.0 for licens
| ing details.
| 2013-07-13T11:54:18Z 47174 TID-ouxpbj07g INFO: Shutting down
| 2013-07-13T11:54:18Z 47174 TID-ouxpiuvcc INFO: Shutting down 25 quiet workers
| **** FILE REMOVED ****
| 2013-07-13-13h54m20 - Loading env
| loading /Users/teamon/SoftwareData/RailsAssets-Sidekiq/..//RailsAssets-Rack/service.env
| loading /Users/teamon/SoftwareData/RailsAssets-Sidekiq/..//TestApp-ElasticSearch/servi
| ce.env
| loading /Users/teamon/SoftwareData/RailsAssets-Sidekiq/..//TestApp-Memcached/service.env
| loading /Users/teamon/SoftwareData/RailsAssets-Sidekiq/..//TestApp-Mongodb/service.env
| loading /Users/teamon/SoftwareData/RailsAssets-Sidekiq/..//TestApp-Rack/service.env
| 2013-07-13-13h54m20 - Entering /Users/teamon/SoftwareData/RailsAssets-Sidekiq/../
| RailsAssets-Rack/current
| 2013-07-13-13h54m21 - Leaving /Users/teamon/SoftwareData/RailsAssets-Sidekiq/../R
| ailsAssets-Rack/current
| 2013-07-13T11:54:21Z 47669 TID-ovqgofvto INFO: Booting Sidekiq 2.12.1 using redis://lo
| calhost:6379/0 with options {:namespace=>"rails-assets"}
| 2013-07-13T11:54:21Z 47669 TID-ovqgofvto INFO: Running in ruby 2.0.0p195 (2013-05-14 r
| evision 40734) [x86_64-darwin12.3.0]
| 2013-07-13T11:54:21Z 47669 TID-ovqgofvto INFO: See LICENSE and the LGPL-3.0 for licens
| ing details.
```

Figure 2. *Panel – launching service with dependencies*



```
Control Panel v0.48.4. © 2013 verknowsys.com ServiceSpawner: ONLINE  (teamon-mb.local)
status: Triggered reconfiguration of application: RailsAssets-Rack

 # Name                     PID Address              Status        Flags   Autostart
 1 TestApp-Rack                 teamon-mb.local:29865 Running       R----
 2 TestApp-Mysql          48693 teamon-mb.local:20199 Running       R----
 3 TestApp-Redis          48706 teamon-mb.local:10894 Running       R----
 4 TestApp-ElasticSearch  48719 teamon-mb.local:40572 Running       R----
 5 TestApp-LogStash       48741 teamon-mb.local:39654 Running       R----
 6 TestApp-Memcached      48753 teamon-mb.local:13509 Running       R----
 7 TestApp-Mongodb        48767 teamon-mb.local:17293 Running       R----
 8 TestApp-Nginx                teamon-mb.local:45670 Running       R----
 9 TestApp-Thin                 teamon-mb.local:17524 Running       R----
10 RailsAssets-Rack             teamon-mb.local:20440 Running       R----
11 RailsAssets-Redis      50870 teamon-mb.local:10216 Running       R----
12 RailsAssets-Puma       50883 teamon-mb.local:24071 Running       R----
13 RailsAssets-Sidekiq    50897 teamon-mb.local:23120 Running       R----
14 RailsAssets-Nginx            teamon-mb.local:13848 Running       R----

Services: Autostart Start sTop Restart Validate Install Configure New eXplode ?=Help
F1=trace F2=debug F3=info F4=error F5=refresh F7=new service F8=destroy F9=SS shutdown
-----------------------------------------------------------------------------------------
RailsAssets-Rack - Build Succeed
13-14:12:38 - Launching service: RailsAssets-Redis
13-14:12:38 - Launching service: RailsAssets-Puma
13-14:12:39 - Launching service: RailsAssets-Sidekiq
13-14:12:40 - Launching service: RailsAssets-Nginx
13-14:12:40 - Launching service: RailsAssets-Rack
```

```
| Showing /Users/teamon/SoftwareData/RailsAssets-Rack/service.log
| Installing celluloid (0.14.1)
|
| Installing connection_pool (1.0.0)
|
| Installing daemons (1.1.9)
|
| Installing eventmachine (1.0.3)
|
| Installing json (1.8.0)
|
| Installing rack (1.5.2)
|
| Installing puma (2.3.1)
|
| Installing rack-protection (1.5.0)
|
| Installing redcarpet (2.2.2)
|
| Installing redis (3.0.4)
|
| Installing redis-namespace (1.3.0)
|
| Installing sidekiq (2.12.1)
|
| Installing tilt (1.4.1)
|
| Installing sinatra (1.4.3)
|
| Installing temple (0.6.5)
|
| Installing slim (2.0.0)
|
| Installing thin (1.5.1)
|
| Using bundler (1.3.5)
| Your bundle is complete!
| Gems in the groups development and test were not installed.
| It was installed into ./bundle.installed
| 2013-07-13-14h07m26 - Cleaning bundle
| 2013-07-13-14h07m27 - Checking for puma binary
| 2013-07-13-14h07m27 - Checking for sidekiq binary
| 2013-07-13-14h07m27 - Leaving /Users/teamon/SoftwareData/RailsAssets-Rack/build/2
| Scrolling (65 lines up)
```

Figure 3. *Panel – each service information is under a hotkey: "K" (configuration), "L" (log) and "E" (environment)*

- PARENT_SERVICE_PREFIX – prefix of service parent (filled only if service has parent)
- SERVICE_DOMAIN – service domain name
- SERVICE_ADDRESS – service address (resolved from SERVICE_DOMAIN)
- SERVICE_ROOT – service application root (f.e. `~/Apps/AppName`)
- SERVICE_VERSION – service version (taken from Sofin's: `~/Apps/AppName/appname.version`)

Please note that igniter constants are values set on runtime. Igniter constants work for all hook commands (including the built in cron scheduler) in all igniters.

The last feature worth mentioning is igniter live updates. When you change an igniter, which is used to spawn your service, it is automatically reloaded. You do not need to restart or reload anything manually to update your scheduler entry. Just edit the igniter, save it, and you are done.

TheSS in Action

First of all, TheSS service must be running as a given user. Usually, the only thing required is to run two commands: `sofin get thess` and `svdss`. Before the first run, the default igniters must be installed. To do so, run `ignitersinstall` as root (required only once). It will install default igniters to `/Common` and `/SystemUsers/Igniters`. After that, TheSS is ready to do its job. If you need your own igniters, just put them into `~/Igniters`. The panel comes up next. To run panel, just run panel. On the first run, you will see just lots of empty space which will be used later for services list, notifications and service logs. To see the panel help, press "?". To initialize a new service from igniter, hit "N" and type the name. If your igniters were installed properly, you should see a scrollable list of igniter names there (Figure 1).

I will just pick one of my testing igniters with dependencies, based on a Ruby on Rails app. Move the cursor with up/down arrows to pick the igniter and press Enter to confirm. You will see the initialized service on the list. To launch it, press "S" (Figure 2 nad Figure 3).

If your igniter is defined properly, you will see your service and its dependencies up and running in a matter of seconds.

Understanding TheSS

There are several TheSS requirements, all of which must be met, to launch your service properly:

- TheSS requires Sofin to be installed on the system. Different software build methods are not supported, and there is no plan to change that.

Glossary

- FHS – *http://en.wikipedia.org/wiki/Filesystem_Hierarchy_Standard*
- KISS – An acronym of "Keep It Simple Stupid".
- RC – default mechanism of launching services in open source systems (rc.d on Linuxes)

On the Web

- TheSS project page: *https://github.com/VerKnowSys/TheSS*
- Sofin project page: *http://verknowsys.github.io/sofin/*
- My website: *http://dmilith.verknowsys.com/WebLog/Core.html*

- Currently, default TheSS shell (used by all igniters) is: `/Software/Zsh/exports/zsh`. If you want TheSS to use a different shell, you need to change it in globals.h and rebuild the whole software package.
- Service hooks are spawned synchronously, hence service hook commands cannot block execution flow. If they do so, you will end up with a blocked execution in a middle of some stage. This will look like hang, but it is not.
- TheSS assumes that every igniter is "perfect". If it is not, then your service probably will not work. Please take a look at the example igniters bundled with TheSS before defining your own.

Future

One of the most important upcoming features of TheSS is an implementation of the services distribution model. The design provides a mechanism of distributed communication between cluster nodes, using TheSS to automatically move software between machines, scale services without downtime, and to provide automatically configured domain servers, load balancers and more.

Summary

TheSS is actively developed but may be considered stable. We have been using it on production servers since version 0.22.x (current version at the time of writing this article is 0.48.5). Watch for new features!

DANIEL (DMILITH) DETTLAFF

Sysadmin of several medium-sized companies in Poland. Enterprise and cloud hater. Currently working at Monterail.com, LLC. Working on building self-healing, self-manageable, distributed, AI-driven systems. He makes new ideas real while systems are automatically doing his work. Constantly learning how to become a good software architect. Musician, lyricist, writer in his free time.

Acknowledgments

Thanks to Michał (wick3d) Hewelt for help with my English grammar and Tymon (teamon) Tobolski for panel screenshots.

Great Specials

On FreeBSD® & PC-BSD® Merchandise

Give us a call & ask about our
SOFTWARE BUNDLES
1.925.240.6652

$39.95
FreeBSD 9.1 Jewel Case CD Set
or FreeBSD 9.1 DVD

$29.95
PC-BSD 9.1 DVD

Save a BUNDLE!

$49.95
The PC-BSD 9.0 Users Handbook
PC-BSD 9.1 DVD

$99.95
The FreeBSD CD or DVD Bundle

Inside each CD/DVD Bundle, you'll find:
FreeBSD Handbook, 3rd Edition
Users Guide FreeBSD Handbook, 3rd Edition, Admin Guide
FreeBSD 9.1 CD or DVD set
FreeBSD Toolkit DVD

Stylish Dress Attire
Look Your Professional Best

Comfy Apparel
Stay Warm in Zip Ups & Pullovers

T-Shirts
Lots of Styles to Choose From

FreeBSD 9.1 Jewel Case CD/DVD $39.95

CD Set Contains:

Disc 1	Installation Boot LiveCD (i386)
Disc 2	Essential Packages Xorg (i386)
Disc 3	Essential Packages, GNOME2 (i386)
Disc 4	Essential Packages (i386)

FreeBSD 9.0 CD	$39.95
FreeBSD 9.0 DVD	$39.95

FreeBSD Subscriptions

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

FreeBSD Subscription, start with CD 9.1	$29.95
FreeBSD Subscription, start with DVD 9.1	$29.95
FreeBSD Subscription, start with CD 9.0	$29.95
FreeBSD Subscription, start with DVD 9.0	$29.95

PC-BSD 9.1 DVD (Isotope Edition)

PC-BSD 9.1 DVD	$29.95
PC-BSD Subscription	$19.95

The FreeBSD Handbook

The FreeBSD Handbook, Volume 1 (User Guide)	$39.95
The FreeBSD Handbook, Volume 2 (Admin Guide)	$39.95

The FreeBSD Handbook Specials

The FreeBSD Handbook, Volume 2 (Both Volumes)	$59.95
The FreeBSD Handbook, Both Volumes & FreeBSD 9.1	$79.95

PC-BSD 9.0 Users Handbook	$24.95
BSD Magazine	$11.99
The FreeBSD Toolkit DVD	$39.95
FreeBSD Mousepad	$10.00
FreeBSD & PCBSD Caps	$20.00
BSD Daemon Horns	$2.00

Bundle Specials!
Save $$$

Just Plain Fun
Mousepads & Novelty Horns

BSD Magazine
Available Monthly

FreeBSD Mall

For even MORE items
visit our website today!
www.FreeBSDMall.com

FreeBSD Programming Primer – Part 7

In the seventh part of our series on programming, we will continue with the menu navigation system and using Javascript.

What you will learn…

- How to configure a development environment and write HTML, CSS, PHP and SQL code

What you should know…

- BSD and general PC administration skills

So far, we have built navigation section buttons that represent the three content types that we have defined in content.inc: pages, news and FAQ's. When the button is pressed, a Javascript popup alerts the user as to what button was clicked via the onclick event (Figure 1). We now need to add additional functionality – when the page is loaded, by default the page's links should be displayed, the menu option (or filter) needs to be displayed to the user, and when the button is clicked, the menu content needs to be rebuilt (Figure 2). Later we will build a more sophisticated menu using the Jquery library.

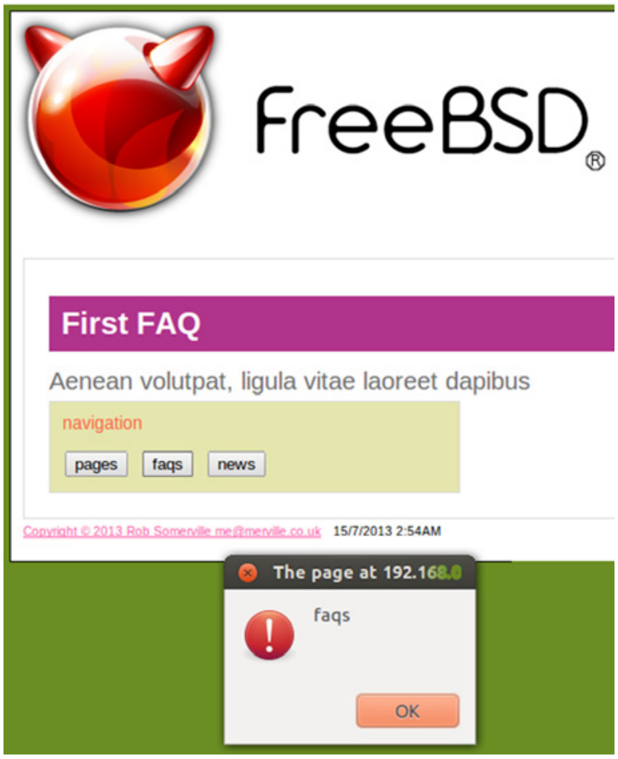

FreeBSD
First FAQ
Aenean volutpat, ligula vitae laoreet dapibus
navigation
pages
faqs
news
Copyright © 2013 Rob Somerville me@merville.co.uk 15/7/2013 2:54AM
The page at 192.168.0
faqs
OK

Figure 1. *FAQ with Javascript onclick buttons*

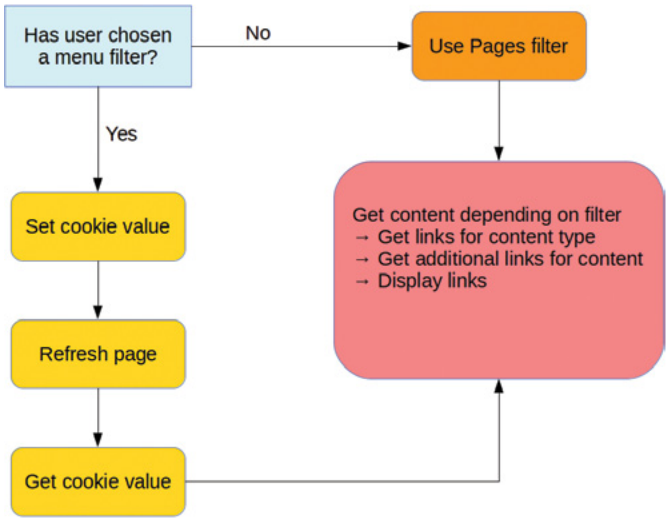
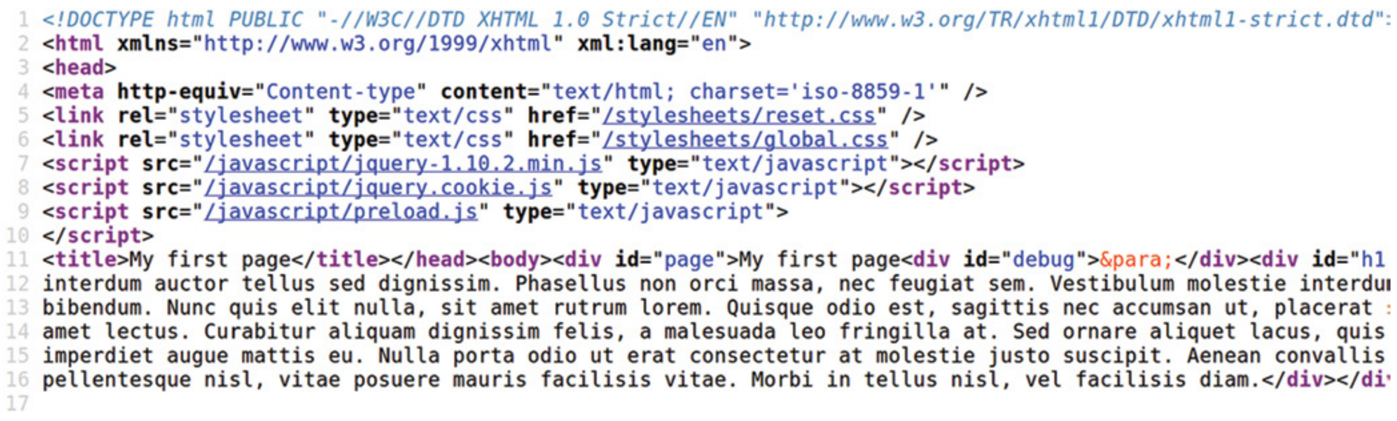

Has user chosen a menu filter?
No
Use Pages filter
Yes
Set cookie value
Get content depending on filter
→ Get links for content type
→ Get additional links for content
→ Display links
Refresh page
Get cookie value

Figure 2. *Logic for the navigation menu*



```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd":
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
<head>
<meta http-equiv="Content-type" content="text/html; charset='iso-8859-1'" />
<link rel="stylesheet" type="text/css" href="/stylesheets/reset.css" />
<link rel="stylesheet" type="text/css" href="/stylesheets/global.css" />
<script src="/javascript/jquery-1.10.2.min.js" type="text/javascript"></script>
<script src="/javascript/jquery.cookie.js" type="text/javascript"></script>
<script src="/javascript/preload.js" type="text/javascript">
</script>
<title>My first page</title></head><body><div id="page">My first page<div id="debug">&para;</div><div id="h1
interdum auctor tellus sed dignissim. Phasellus non orci massa, nec feugiat sem. Vestibulum molestie interdu
bibendum. Nunc quis elit nulla, sit amet rutrum lorem. Quisque odio est, sagittis nec accumsan ut, placerat
amet lectus. Curabitur aliquam dignissim felis, a malesuada leo fringilla at. Sed ornare aliquet lacus, quis
imperdiet augue mattis eu. Nulla porta odio ut erat consectetur at molestie justo suscipit. Aenean convallis
pellentesque nisl, vitae posuere mauris facilisis vitae. Morbi in tellus nisl, vel facilisis diam.</div></di
```

Figure 3. *Page source showing Javascript Jquery libraries loaded*

Listing 1. *postload.js*

```
// Set navigation menu cookie

function setnavitem(item){

   $.cookie("navmenuitem", item);

}
```

Listing 2. *menu.inc*

```
<?php

function menu($type) {

    require INCLUDES . 'content.inc';

    if ($type == 'navigation') {

        // Build select statement for each content type
                   in turn
        // Omit the UNION keyword on the last item

        $offset = 1;
        $categories = count($content_tables);
        $sql = '';
        $option = '';

        // Get the value of the cookie if set

        if(isset($_COOKIE['navmenuitem'])){

            $menuvalue = $_COOKIE['navmenuitem'];

        }else{

            $menuvalue = 'pages';

        }

        foreach ($content_tables as $contenttype) {

            // Build the option for the content type

            $option .= '<button onclick="setnavitem(\''.
                   $contenttype.'\');
document.location.reload(true);">'.$offset.'.
'.ucfirst($contenttype).'</button> &nbsp;';
            $offset ++;

        }

        $menu = '';

        $menu .= '<div class ="menu-' . $type . '">';
        $menu .= '<h2>' . ucfirst($type) . ' (' .
                   $menuvalue.')</h2>';
        $menu .= '<p>&nbsp;</p>';
        $menu .= $option;
        $menu .= '</div>';

        return $menu;
    }
}
```

Step 1 – Handling the user interaction

Ensure you have downloaded the Jquery libraries as detailed in the previous article. If you view the page source for any page, it should be similar to (Figure 3). Modify postload.js and menu.inc as follows (Listing 1 – 2).

If you now navigate to *http://yoursiteip/faq/1*, you should now see a page similar to (Figure 4). If you click on the buttons, instead of a Javascript popup you should see the navigation menu title changing to reflect the new selection. Using Firebug and the Cookie console, you will see the content of the cookie changing when a new menu item is selected. Deleting the cookie and refreshing the

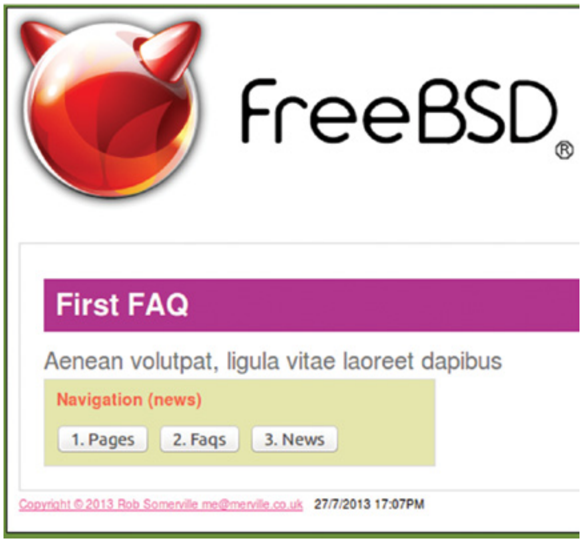
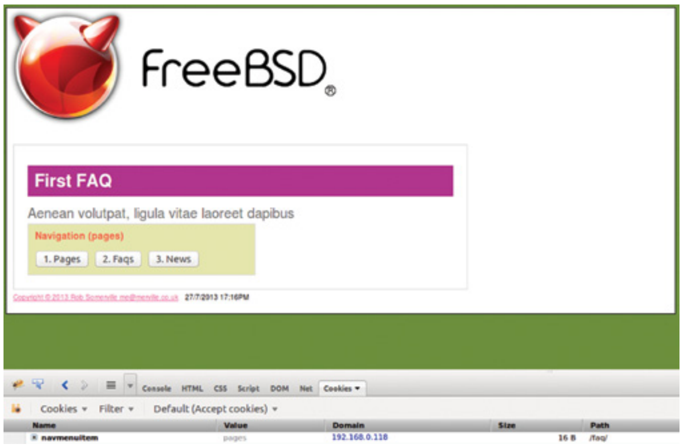

FreeBSD®
First FAQ
Aenean volutpat, ligula vitae laoreet dapibus
Navigation (news)
1. Pages
2. Faqs
3. News
Copyright © 2013 Rob Somerville me@merville.co.uk
27/7/2013 17:07PM

Figure 4. *FAQ page with Javascript and cookie control*



Name	Value	Domain	Size	Path
navmenuitem	pages	192.168.0.118	16 B	/faq/

Figure 5. *Cookie set in Firebug*

Listing 3. *add to preload.js*

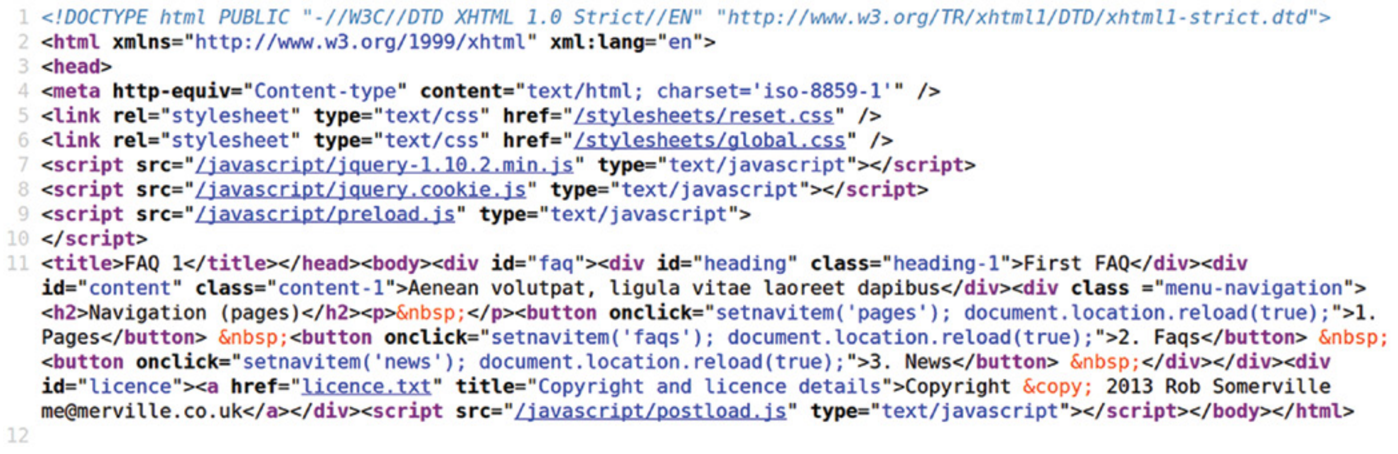
```
// Init routines

function preinit(){

   document.body.style.display = 'none';

}

function postinit(){

   $(document.body).fadeIn(500);

}
```

Listing 4. *Add to core.inc*

```
Add just after echo BODY;

echo "<script>preinit();</script>";
```

Listing 5. *Add to core.inc*

```
Add just before ob_end_flush()

echo "<script>postinit();</script>";
```



```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
<head>
<meta http-equiv="Content-type" content="text/html; charset='iso-8859-1'" />
<link rel="stylesheet" type="text/css" href="/stylesheets/reset.css" />
<link rel="stylesheet" type="text/css" href="/stylesheets/global.css" />
<script src="/javascript/jquery-1.10.2.min.js" type="text/javascript"></script>
<script src="/javascript/jquery.cookie.js" type="text/javascript"></script>
<script src="/javascript/preload.js" type="text/javascript">
</script>
<title>FAQ 1</title></head><body><div id="faq"><div id="heading" class="heading-1">First FAQ</div><div
id="content" class="content-1">Aenean volutpat, ligula vitae laoreet dapibus</div><div class ="menu-navigation">
<h2>Navigation (pages)</h2><p>&nbsp;</p><button onclick="setnavitem('pages'); document.location.reload(true);">1.
Pages</button> &nbsp;<button onclick="setnavitem('faqs'); document.location.reload(true);">2. Faqs</button> &nbsp;
<button onclick="setnavitem('news'); document.location.reload(true);">3. News</button> &nbsp;</div></div><div
id="licence"><a href="licence.txt" title="Copyright and licence details">Copyright &copy; 2013 Rob Somerville
me@merville.co.uk</a></div><script src="/javascript/postload.js" type="text/javascript"></script></body></html>
```

Figure 6. *Page source showing button options*

Listing 6. *Add to mysql.inc*

```
Returns an array of rows or NULL on no result

function mysql_fetchrows($sql) {

   // Returns an array of rows or NULL if no result

   $db = new mysqli(DBSERVER, DBUSER, DBPASSWORD,
                   CMSDB);

   if ($db->connect_errno > 0) {
       die('Unable to connect to database [' .
                   $db->connect_error . ']');
   }

   if (!$result = $db->query($sql)) {
       if (DEBUG) {
           die('There was an error running the query ['
                   . $db->error . ']');
       } else {
           die('');
       }
   }

   while ($row = $result->fetch_row()) {
       $r[] = $row;
   }

   // Free the result

   $result->free();

   // Close the connection

   $db->close();

   if (isset($r)) {
       return $r;
   } else {
       return NULL;
   }
}
```

Listing 7. *Add to core.inc*

```
function arraytolinks($mysqlfetchrows){

   require INCLUDES . 'content.inc';

   // Convert a MySQL result set into a set of links

   // Requires ID (page id), title and contenttype

   $links = '<div class="menulinks">';
   $links .= '<ul>';

   if($mysqlfetchrows){

   foreach ($mysqlfetchrows as $key => $value) {

       // Convert the content type to the relevant
                   table name.
       // See content.inc

       $path = array_search($value[2], $content_
                   tables);

       $links .= '<li><a href="/'.$path.'/'.$value[0].'"
                   title="'.$value[1].'">'.
       $value[1].'</a></li>';

   }

   $links .= '</ul>';
   $links .= '</div>';

   }else{

       $links .= "<li>Sorry - no content available</
                   li></ul></div>";

   }

   return $links;
}
```

Listing 8. *Full listing of menu.inc*

```
<?php

function menu($type) {

   require INCLUDES . 'content.inc';

   if ($type == 'navigation') {

       $offset = 1;
       $categories = count($content_tables);
       $sql = '';
       $option = '';

       // Get the value of the cookie if set

       if(isset($_COOKIE['navmenuitem'])){
           $menuvalue = $_COOKIE['navmenuitem'];
       }else{
           $menuvalue = 'pages';
       }

       foreach ($content_tables as $contenttype) {

           // Build the option for the content type

           $option .= '<button onclick="setnavitem(\''.
                   $contenttype.'\');
          document.location.reload(true);">'.$offset.'.
           '.ucfirst($contenttype).'</button> &nbsp;';
           $offset ++;

       }

      // Build the SQL statement for the menu item selected

       $sql = "SELECT id,title,'".$menuvalue."' AS
                   contenttype FROM ".$menuvalue."
     WHERE status = 2 ORDER BY title;";

       // Get the result

       $result = mysql_fetchrows($sql);

       // Convert the array into HTML links

       $links = arraytolinks($result);

       $menu = '';
       $menu .= '<div class ="menu-' . $type . '">';
       $menu .= '<h2>' . ucfirst($type) . ' (' .
                   $menuvalue.') - '.$categories.'
     categories</h2>';
       $menu .= '<p>&nbsp;</p>';
       $menu .= $option;
       $menu .= $links;
       $menu .= '</div>';

       return $menu;
   }
}
```

Listing 9. *Changes to faq_tempate.inc*

```
render($theme['heading']);
render(menu('navigation'));
render($theme['content']);
```

Listing 10. *Modify global.css*

```
 .menu-navigation {
  background-color: #E5E6AD;
  border: 1px solid #DADADA;
  padding: 10px;
  float: right;
  margin-left: 10px;
  margin-bottom: 10px;
}

#news, #page, #faq {
  border: 1px solid #DADADA;
  margin-top: 190px;
  padding: 20px;
  min-height: 640px;
  overflow: auto;
}
```

Listing 11. *Add global menu support to News, FAQ and pages templates*

```
 Add at the beginning of each file (e.g. just before
                   render($theme['heading']);)

 render(menu('global'));
```

Listing 12. *Add to preload.js*

```
 function globalmenu(){

   $(function() {$( "#menu" ).menu();});

 }
```

Listing 13. *header.inc*

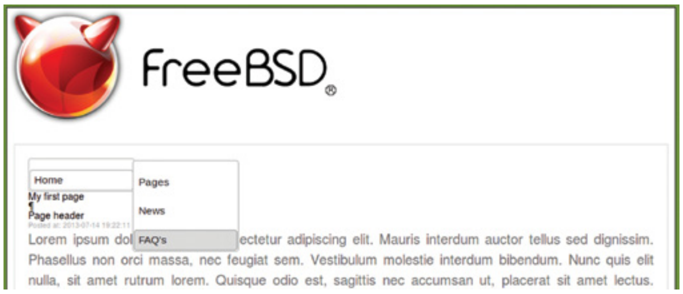
```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
<head>
<meta http-equiv="Content-type" content="text/html; charset='iso-8859-1'" />
<link rel="stylesheet" type="text/css" href="/stylesheets/reset.css" />
<link rel="stylesheet" type="text/css" href="/stylesheets/global.css" />
<link rel="stylesheet" type="text/css" href="/stylesheets/jquery-ui.css" />
<script src="/javascript/jquery-1.10.2.min.js" type="text/javascript"></script>
<script src="/javascript/jquery.cookie.js" type="text/javascript"></script>
<script src="/javascript/jquery-ui.min.js" type="text/javascript"></script>
<script src="/javascript/preload.js" type="text/javascript">
</script>
```

page should load the default menu type of Pages (Figure 5). The titles have also been cleaned up using the PHP `ucfirst()` function call to uppercase the first character of the selection, and we have added a sequential option number to each menu item.

One disadvantage of this method is the following piece of code as shown in (Figure 6). Each button has two pieces of Javascript attached, `setnavitem()` and `document.location.reload()`. The former sets the cookie via our function call in postload.js (and subsequently via the jquery.cookie.js script) and then refreshes the page. This causes the page to flicker every so often when the content is reloaded. A better way of implementing this would be to use Ajax, but for the time being, we will demonstrate a useful Jquery call – Fade in.

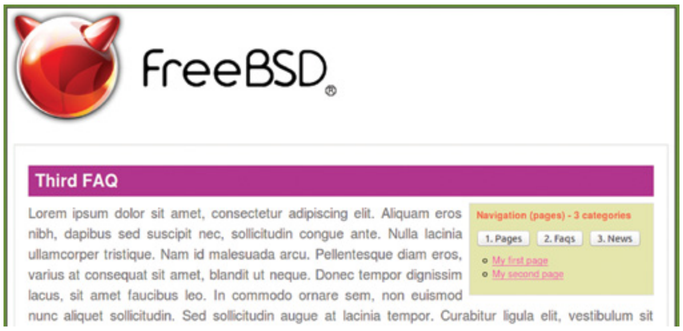
Add the following code to preload.js (Listing 3) and core.inc (Listing 4 and Listing 5).

This will halt the display of the page, allow the menu to be built etc. and the page will then fade in. The time can



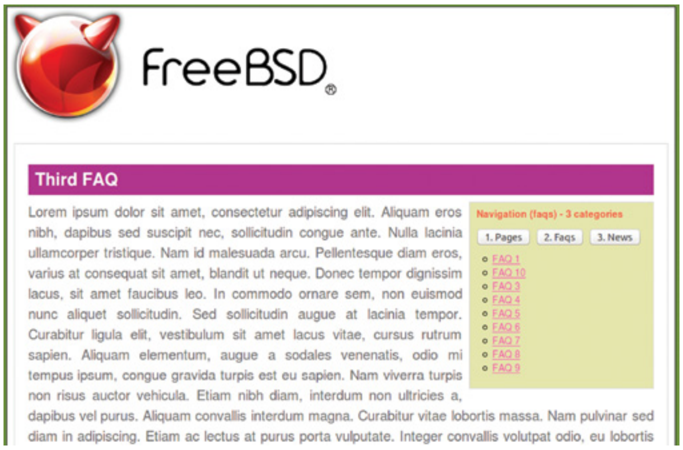
Lorem ipsum dol ectetur adipiscing elit. Mauris interdum auctor tellus sed dignissim. Phasellus non orci massa, nec feugiat sem. Vestibulum molestie interdum bibendum. Nunc quis elit nulla, sit amet rutrum lorem. Quisque odio est, sagittis nec accumsan ut, placerat sit amet lectus.

Figure 7. *FAQ page menu*



Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aliquam eros nibh, dapibus sed suscipit nec, sollicitudin congue ante. Nulla lacinia ullamcorper tristique. Nam id malesuada arcu. Pellentesque diam eros, varius at consequat sit amet, blandit ut neque. Donec tempor dignissim lacus, sit amet faucibus leo. In commodo ornare sem, non euismod nunc aliquet sollicitudin. Sed sollicitudin augue at lacinia tempor. Curabitur ligula elit, vestibulum sit

Figure 8. *FAQ faqs menu*



Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aliquam eros nibh, dapibus sed suscipit nec, sollicitudin congue ante. Nulla lacinia ullamcorper tristique. Nam id malesuada arcu. Pellentesque diam eros, varius at consequat sit amet, blandit ut neque. Donec tempor dignissim lacus, sit amet faucibus leo. In commodo ornare sem, non euismod nunc aliquet sollicitudin. Sed sollicitudin augue at lacinia tempor. Curabitur ligula elit, vestibulum sit amet lacus vitae, cursus rutrum sapien. Aliquam elementum, augue a sodales venenatis, odio mi tempus ipsum, congue gravida turpis est eu sapien. Nam viverra turpis non risus auctor vehicula. Etiam nibh diam, interdum non ultricies a, dapibus vel purus. Aliquam convallis interdum magna. Curabitur vitae lobortis massa. Nam pulvinar sed diam in adipiscing. Etiam ac lectus at purus porta vulputate. Integer convallis volutpat odio, eu lobortis

Figure 9. *FAQ news menu*

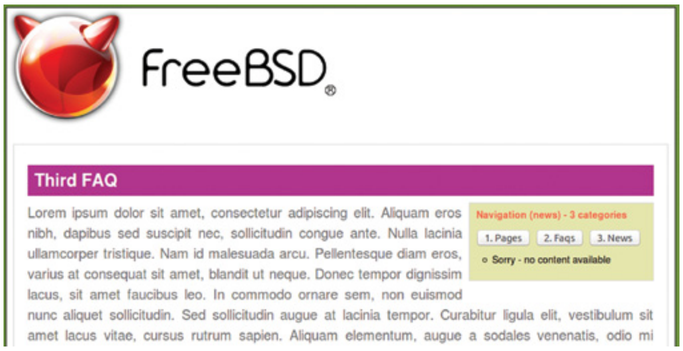

FreeBSD®
Third FAQ
Navigation (news) - 3 categories
1. Pages
2. Faqs
3. News
Sorry - no content available
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aliquam eros nibh, dapibus sed suscipit nec, sollicitudin congue ante. Nulla lacinia ullamcorper tristique. Nam id malesuada arcu. Pellentesque diam eros, varius at consequat sit amet, blandit ut neque. Donec tempor dignissim lacus, sit amet faucibus leo. In commodo ornare sem, non euismod nunc aliquet sollicitudin. Sed sollicitudin augue at lacinia tempor. Curabitur ligula elit, vestibulum sit amet lacus vitae, cursus rutrum sapien. Aliquam elementum, augue a sodales venenatis, odio mi

Figure 10. *Jquery multi-level menu*

be adjusted by incrementing or decrementing the `fadeIn()` parameter. While this is not an ideal solution, it does demonstrate the ease of integrating Jquery with a web page.

Step 2 – Displaying the links

Now we need to plug in the SQL result to our menu module. Add the following code (Listing 6-8).

We now need to make a few minor modifications at the theme and CSS levels, so change faq_template.inc to display the menu before the content (Listing 9).

Listing 14. *Additions to menu.inc*

```
  Add elseif at the end of the navigation block

        $menu .= '<div class ="menu-' . $type . '">';
         $menu .= '<h2>' . ucfirst($type) . ' (' .
                    $menuvalue.') - '.$categories.'
        categories</h2>';
        $menu .= '<p>&nbsp;</p>';
        $menu .= $option;
        $menu .= $links;
        $menu .= '</div>';

        return $menu;

    }elseif ($type == "global") {

         ?>

          <ul id="menu">
            <li><a href="/">Home</a>
           <ul>
             <li><a href="/page/1">Pages</a></li>
                <li><a href="/news/1">News</a></li>
                <li><a href="/faq/1">FAQ's</a></li>
              </ul>
            </li>
          </ul>

        <?php

    }
}
```

Listing 15. *Add to global.css*

```
  .ui-menu {
    width: 150px;
  }
```

> **Useful links**
>
> - Jquery UI source – *http://jqueryui.com/resources/download/jquery-ui-1.10.3.zip*
> - Jquery menu reference – *http://jqueryui.com/menu*

This will float the navigation menu on the FAQ page to the right and increase the height of our news, page, and FAQ content to accommodate the new menu.

See (Figure 7-9) for the final result. I added an extra "Ipsum Lorem" to pad the content out in FAQ 3. Note how the menu responds to user input decoupled from the content that the user is currently visiting.

Step 3 – Global website menu

Jquery provides an extensive library for the user interface. Rather than building the Javascript and CSS from scratch, we can install the CSS and JS libraries quickly into our CMS.

Download Jquery-ui-1.10.3.zip and extract Jquery-ui.css into the stylesheets directory and Jquery-ui.min.js into the javascript directory. Use MC, or extract the file into a temporary area using unzip.

Add the global menu to all of our content templates (`news_templates.inc`, `pages_template.inc` and `faqs_tempate.inc`) and add the Javascript function to preload.js. Add the Javascript and CSS files to the header.inc file and add a new menu option to menu.inc and finally tweak our CSS file to reduce the width of the menu (Listing 11-15).

Finally, visit the homepage of your site with your browser, refresh the page and voila, one multi-level menu (Figure 10).

In the next part

We will continue refining the menu system and start building the user interface.

ROB SOMERVILLE

Rob Somerville has been passionate about technology since his early teens. A keen advocate of open systems since the mid-eighties, he has worked in many corporate sectors including finance, automotive, airlines, government and media in a variety of roles from technical support, system administrator, developer, systems integrator and IT manager. He has moved on from CP/M and nixie tubes but keeps a soldering iron handy just in case.

XRY COMPLETE

IT'S NOT ABOUT DATA.

IT'S ABOUT MEANING.

If you think mobile forensics is just about extracting data – think again. Its not only what you get, but what you do with it that really makes the difference.

XRY has an intuitive GUI that's easier to use, has better display capabilities and superior analysis functionality.

MICRO SYSTEMATION

msab.com

PKGNG

The future of packages on FreeBSD and PC-BSD

This article will show how to install, upgrade and remove packages using pkgng. It will also discuss some of the improvements over pkg_*tools and demonstrate how pkgng will benefit end users. In addition, the upcoming functionality of pkgng will also be briefly discussed. Finally, it will show how to install Gnome 3 and Cinnamon on PC-BSD Rolling Release using Pkgdemon.

What you will learn…

- How to add a package site;
- How to install and upgrade packages;
- How to remove packages and all of the dependencies that were installed by the packages.

What you should know…

- Basic shell commands;
- How to install PC-BSD Rolling Release.

All of you want to know how to use a third party package repository. This article will use Pkgdemon with the PC-BSD Rolling Release as an example to demonstrate the process. This article will show the process of adding a package site and then installing packages on the PC-BSD Rolling Release. The same information can be applied to a fresh install of FreeBSD 9.1 onward, or a FreeBSD 9.1 installation where pkgng was used to install all previously installed packages. It will cover basic maintenance tasks. It will also demonstrate the advantages of using the new package system as well as its future potential.

Before PKGNG

Users of FreeBSD and derivative systems have always had primarily two ways of installing software. The first one is through the use of ports and building binaries from source code. The second one is by using packages. Anyone who has ever installed packages before on FreeBSD, will be familiar with the problems associated with using pkg_*tools. This may be one of the many reasons most users choose to stick with ports. While installation of software by ports is often ideal for servers and advanced users, it is not often ideal for a user who wants or needs a desktop that works out of the box.

A good example for me is when I need to provision a new machine for work purposes. I need to be able to get it installed quickly, so I can begin to look up customers and troubleshoot issues. I don't have time to wait around for a couple of days or have a half-usable system until all of the software is installed. So for me, installing from packages on a desktop is a perfect solution.

At my workplace, we have used FreeBSD on our servers for years and we love it for its rock-solid stability, security, and portability. A few of us experimented with the idea of using FreeBSD as a desktop a few times at work. However, it became clear that as much as we loved FreeBSD, none of us could afford the downtime. Then PC-BSD came along with a few really solid releases, starting with 8.0, and that really began to meet our needs.

PC-BSD has, of course, solved many of the problems associated with using packages by developing a third method called PBI. This self-contained method allows quick installation of many desktops and even certain server applications. It has allowed me and others to use FreeBSD at work on our desktops.

However, PC-BSD has also relied on using packages for its desktop interfaces quite often. The shortcomings of pkg_*tools are less noticeable, but nonetheless are still noticeable, when it comes to exceptionally long installation times and software upgrades. It would become even more noticeable if a user were to try to install software themselves through packages. Package conflicts could occur and a user could easily break their entire desktop.

For example, to upgrade from PC-BSD 9.0 release to 9.1, release packages would have to be manually deleted behind the scenes. In addition, the PC-BSD operating system had to know what packages a user had installed. If a user had installed their own packages or ports, they would have to be removed behind the scenes.

If the PC-BSD installer had not done this, the entire desktop could be broken. As a result, if a user has additional ports or packages that are not part of the PC-BSD installation, they will be deleted during an upgrade. This was not a huge issue, it was just an obvious limitation of pkg_*tools.

pkg_*tools and third party repositories

To give further examples of pkg_*tools issues, I will use my website: Pkgdemon. Pkgdemon is a 3rd party package repository for FreeBSD, GhostBSD, and PC-BSD. The purpose of Pkgdemon right now is to provide early access to easily-installable Gnome 3 and Cinnamon packages.

When I started Pkgdemon, these ports were not yet available in the ports tree, and I wanted to use them at work on my PC-BSD desktop. Knowing that I couldn't afford to have a lot of downtime at work, I devised a way to make the installation easy enough so I could install it on my work system.

After compiling ports into packages, there was one obvious problem. Installing Gnome 3 from using pkg_*tools is not easy if you have other packages installed which conflict with Gnome 3. Even if you know what the names of the packages are, it is still not easy. For example, for each pkg, you would have to list the entire name of the package. You are also likely to have to force the package to uninstall. Here is an example of one (out of the hundred) `pkg_delete` commands required for installation on PC-BSD 9.1 Isotope.

```
pkg_delete -f py27-dbus-0.84.0
```

You may be thinking that I could have recursively deleted Gnome 2 and all of its dependencies. However, this would break both my installation and the Nvidia drivers, as deleting Gnome 2 recursively would have uninstalled components required by Nvidia and Xorg. This was not an option for this script. This situation was somewhat eased later by pkgng as you will see later in this article.

I had to create executable shell scripts to remove the several hundreds of packages. In addition, occasionally other packages that were outside of my Gnome 3 repository were more recent versions than what Gnome 3 expected. For example, Gnome 3 expected that libpcre.so.1 and libpcre.so.3 were installed. I had to manually create symbolic links within the script. I had to do this for a few other libraries as well otherwise the package installation would fail and several applications would not open. This is no longer necessary as often, with pkgng.

In addition, I wanted to be able to host other desktops for installation besides Gnome 3. For now, the additional desktop is Cinnamon. In the future, I want to add Mate, possibly CDE, and some others. To make that easy, I had to create an interactive menu within the shell script to allow the user to choose which combination of desktops to install. This is also no longer necessary with pkgng.

When I had made changes to the ports tree, I forked Gnome 3 for packaging purposes. I needed to devise a way to give myself and others an upgrade path. This involved adding a list of packages that were installed by Gnome 3 or Cinnamon and making sure those were removed as well. If I had forgotten a package it would become obvious as the script would fail during installation. This was also drastically eased by pkgng.

Listing 1 is what the script would look like in a terminal. Without this script, users would have to manually remove hundreds of packages by hand in order to install Gnome 3 using packages created by pkg_*tools. The source code for this script showing the commands can be viewed here: *https://github.com/pkgdemon/freebsd-pkgdemon/blob/master/PCBSD9.1-x64-pkgdemon-v1.01.sh*

If you need the x86 version of the script, please go to the Pkgdemon github url: *http://www.github.com/pkgdemon/freebsd-pkgdemon*

PKGNG

PKGNG is referred to as the next generation package manager. It uses a SQLite database to store all of the info about the installed packages on your system. The location of the database still resides in /var/db/pkg, but now as a single file. This is in contrast to otherwise what would be many folders and files containing information about the installed packages.

Storing the information about packages in a database makes searching for information about installed packages fast. It also helps make installation and removal fast, and it allows for easier upgrades.

I could go on about the technical details of what makes pkgng so great, however, nothing will show it as well as using it. To do that, I must introduce PC-BSD rolling release and try to explain briefly what it is.

PC-BSD Rolling Release

Some time ago, Kris Moore announced that PC-BSD would be adding a rolling release model in addition to the current stable releases. The current stable release is PC-BSD 9.1 Isotope. As part of this release announcement, it was mentioned that PC-BSD would begin using pkgng, and that all of the packages would be converted to pkgng.

In addition to this, it was mentioned that a pkgng repository would be created for upgrades of system packages. This meant that if a new version of KDE or a new version of Xorg were released into FreeBSD ports that, if you were running the rolling release, you could upgrade to them as they periodically rebuild the packages for the repository. That is as opposed to having to wait for the next stable PC-BSD release. This currently only happens about every year or so when a new release of FreeBSD occurs. To get a rough idea of what types of commands, how many packages might have to be removed, and what might be required to upgrade to a new version of KDE or Xorg using pkg-tools, please take a look at script from Listing 1. With pkgng, it is just one command.

```
pkg upgrade
```

If that isn't a huge reason to switch to pkgng over pkg_*tools right now, what is?

Adding a third party repository

Now I am going to show the installation process for Pkgdemon using PC-BSD Rolling Release. I was told that eventually there will be a special page on the PC-BSD website for the Rolling Release with download links. A few download links will be included at the end of the article for Rolling Release.

Before following this guide please remove XFCE, LXDE, and Gnome. If you are doing a fresh install, do not select either of these or the following commands will remove these packages.

Listing 1. *Script's GUI*

```
[root@psBSD8920] ~# fetch http://www.pkgdemon.com/downloads/scripts/PCBSD9.1-x64-pkgdemon-v1.01.sh

PCBSD9.1-x64-pkgdemon-v1.01.sh                              100% of 14 1:13 459 kBps
[root@psBSD8920] ~# csh ./PCBSD9.1-x64-pkgdemon-v1.01.sh
################################################################################
# By installing this software you are in agreement that I am not in           #
# anyway responsbile for what may happen to your computer.                    #
# Please do not run this script unless you have made a backup and know        #
# what you are doing.                                                          #
# I am not afilliated with FreeBSD, or PCBSD.                                  #
# Please do not contact them for support if you run this script.              #
################################################################################

################################################################################
# Gnome 3 Installer for PCBSD 9.1 Isotope Edition                              #
################################################################################
********************************************************************************
 Phase 1 will download aprroximatley 581mb of packages from Pkgdemon
********************************************************************************
Press enter to continue or control c to abort
```

```
pc-metapkgmanager del GNOME
pc-metapkgmanager del GNOME-Accessibility
pc-metapkgmanager del GNOME-Games
pc-metapkgmanager del GNOME-Net
pc-metapkgmanager del GNOME-Utilities
pc-metapkgmanager del XFCE
pc-metapkgmanager del XFCE-Plugins
pc-metapkgmanager del LXDE
pkg autoremove
```

The pc-metapkgmanager commands are specific to PC-BSD; however, the `pkg autoremove` command is specific to pkgng. In this example, pc-metapkgmanager will only remove a single package, not all of its dependencies. Executing pkg autoremove will remove all of the dependencies required by Gnome 2, XFCE, and LXDE. Before, this would have also removed things required for the Nvidia driver and Xorg. This is no longer the case with pkgng, unless those packages are also removed before executing pkg autoremove.

After PC-BSD is installed, it is recommended you login to either the Fluxbox or KDE window manager and open a terminal. The first step will be to add the pkg repo key, which is used for verification.

To use the package signing key you will need to fetch it into `/usr/local/etc/`. For example:

```
cd /usr/local/etc
fetch http://www.pkgdemon.com/downloads/pkgdemon.cert
```

After this, you will need to edit `/usr/local/etc/pkg.conf` and comment the following lines like this example.

```
#packagesite: http://pkg.cdn.pcbsd.org/9.1-RELEASE/amd64
#HTTP_MIRROR: http
#PUBKEY: /usr/local/etc/pkg-pubkey.cert
```

Add the public cert key for `pkgdemon` to `pkg.conf`.

```
PUBKEY: /usr/local/etc/pkgdemon.cert
```

Then, add the appropriate line below to pkg.conf, depending on which architecture you used to install PC-BSD rolling release. PCBSD Rolling Releases are now only available for 64 bit systems.

64 bit

```
packagesite: http://www.pkgdemon.com/freebsd:9:x86:64/latest
```

There are still a few packages, however, that will have to be removed manually as the versions of these packages cannot be removed automatically by pkgng. The following packages are not yet built to be upgrade aware. Therefore they must be removed manually first. The -f flag is used to force removal. If -f is not specified, the packages will not remove as they are required by pcbsd-base.

```
pkg delete -fy at-spi
pkg delete -fy zenity
pkg delete -fy gnome-session
pkg delete -fy metacity
pkg delete -fy gdm
pkg delete -fy evolution-data-server
pkg delete -fy gnome-desktop
pkg delete -fy gnome-panel
pkg delete -fy gnome-settings-daemon
pkg delete -fy libgnomekbd
pkg delete -fy gnome-power-manager
pkg delete -fy gnome-keyring
pkg delete -fy libgweather
pkg delete -fy gnome-menus
pkg delete -fy libwnck
pkg delete -fy gnome-control-center
pkg delete -fy farsight2
```

The `-y` flag will automatically confirm the removal of each package so that you are not prompted to type y to remove each package.

You should notice a couple of things here. One is that the list of packages to delete is dramatically smaller than it would be with pkg_*tools. In fact, hundreds of packages less than when the script was removing the conflicting desktops for you and there were still hundreds of packages to remove. The second thing you should notice is that when you type *pkg delete*, you no longer have to include the exact version.

Upgrade packages

```
pkg upgrade -fy
```

It is safer to use -f flag here to force the upgrade of these packages. This will force an upgrade of the entire package set to ensure compatibility with Gnome 3. Otherwise certain packages necessary for this example may not successfully install and Gnome 3 will crash when launching. The `-y` flag is used again to automatically confirm to make installation easier.

Install Packages

After the packages have been upgraded, run the following command to install Gnome 3.

```
pkg install gnome3
```

Installing Cinnamon is also fairly straightforward.

```
pkg install cinnamon
```

You may notice that I am not going to tell you to create a symbolic link with my rolling release instructions. Pkgng is smart enough to figure out what version of pcre Gnome 3.4 wants.

Package removal

The way package removal works with pkgng is much better. If you wanted to remove my Gnome 3 and Cinnamon packages and go back to using what's provided by PC-BSD, it's fairly easy to do.

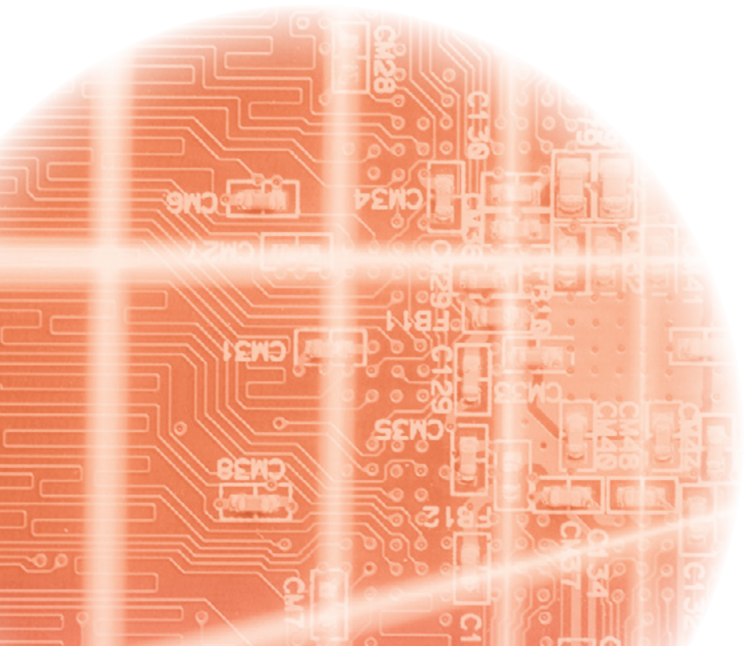
```
pkg remove gnome3
pkg autoremove
pkg remove cinnamon
pkg autoremove
```

Then just set your packagesite and cert file back to what was provided by PC-BSD. If you just commented the lines, you can uncomment them.

```
packagesite: http://pkg.cdn.pcbsd.org/9.1-RELEASE/amd64
HTTP_MIRROR: http
PUBKEY: /usr/local/etc/pkg-pubkey.cert
```

Now comment or delete the lines for Pkgdemon.

```
#packagesite: http://www.pkgdemon.com/freebsd:9:x86:64/latest
#PUBKEY: /usr/local/etc/pkgdemon.cert
pkg upgrade -fy
```



On The Web

- PC-BSD Official FTP Site for Rolling Release: *ftp://ftp.pcbsd.org/pub/mirror/9.1-RELEASE/*
- Pkgdemon official Web Site: *http://www.pkgdemon.com*
- Pkgdemon Guide for Rolling Release: *http://www.pkgdemon.com/support/install-pcbsd-testing*
- Pkgdemon Github to view scripts: *http://www.github.com/pkgdemon/freebsd-pkgdemon*
- Dedicated server hosting for Pkgdemon provided by Sumner Communications: *http://www.sutv.com*

Future pkgng features

It has been announced that pkgng will soon be getting some additional features, not entirely covered here. It essentially provides a solid support to a graphical user interface so that installing packages can become available for starters.

Currently, using my installation guide and the rolling release, you will have to replace PC-BSD's repo with my repo in order to install packages from Pkgdemon. An upcoming feature will allow upgrading across multiple repositories. This means you should be able to reference both PC-BSD's repository and my repository to get the latest packages from both Pkgdemon and PC-BSD.

Conclusion

Pkgng is a 21st century package system for FreeBSD. It will soon allow using multiple repositories concurrently for upgrading and software installation. It will prove to be beneficial for end users by providing easy installation and upgrades of software for FreeBSD using the packages system.

JOE MALONEY

Joe Maloney lives in the United States with his family. He works as an Assistant Network Administrator for an Internet Service Provider at Sumner Communications. He likes to hang out on IRC on the Freenode network in the #PCBSD, GhostBSD, and FreeBSD-Gnome channels as malco_2001.

He can be reached online at http://www.pkgdemon.com, pkgdemonteam@gmail.com, or by IRC.

Faster.

Better.

Reliable.

Trusted by over 500 ISPs worldwide.

Hyper is the first multimedia cache fully developed in Brazil, by Taghos. With Hyper, ISPs can save on network bandwidth while increasing content-delivery speeds, resulting in end-customer satisfaction.

Features:

- 24x7x365 always-on support
- Active monitoring
- Automatic updates
- Appliance or license
- Easy deployment
- Configuration and reports via web interface

hyper

Remote Install

Using your hardware

Model	Traffic	RAM	Cache	SSD
T15	Up to 15 Mbps	8 GB	1x 1 TB	-
T50	Up to 50 Mbps	8 GB	2x 1 TB	-
T100	Up to 100 Mbps	8 GB	2x 1 TB	1x 160 GB
T150	Up to 150Mbps	16 GB	3x 2 TB	1x 160 GB
T300	Up to 300 Mbps	16 GB	5x 2 TB	1x 240 GB
T500	Up to 500 Mbps	32 GB	7x 2 TB	1x 480 GB
T1000	Up to 1 Gbps	64 GB	10x 1 TB	1x 480 GB
T2000	Up to 2 Gbps	96 GB	24x 1 TB	3x 480 GB
T3000	Up to 3 Gbps	128 GB	32x 1 TB	5x 480 GB

Visit us at **www.taghos.com** and start saving bandwidth today!

Headquarters:
San Jose, CA

855.GREP.4.IX | Contact Us

99% Compatibility

online now...

IXSYSTEMS AND YOU ARE THE PERFECT MATCH

SHARED INTERESTS

- ✓ Enterprise Storage Solutions
- ✓ Personalized Customer Service
- ✓ Bold New Information Technology

I'm a	Storage Reseller
In	The EU
Looking for	Storage Solutions to Sell
	A Technology Partner
	More Technical Experience
	New Business Opportunities

Visit Today!

iXsystems

Technology Partner Seeking Resellers/Integrators for TrueNAS™ Storage Appliance

WWW.IXSYSTEMS.COM/PERFECTMATCH