BSD MAGAZINE

FOR NOVICE AND ADVANCED USERS

Free Pascal on BSD

HOW TO WRITE CODE ON BSD THAT RUNS CROSS-PLATFORM

HOW TO INSTALL NGINX, PHP AND PHP-FPM (FASTCGI PROCESS MANAGER) ON NETBSD

GETTING TO GRIPS WITH THE GIMP

HOW TO MANAGE LARGE SOFTWARE DEVELOPMENT PROJECTS USING RCS TOOLS

VOL.8 NO.04
ISSUE 04/2014(57)
1898-9144

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

✓ Rock-Solid Performance
✓ Professional In-House Support

FREENAS MINI STORAGE APPLIANCE

IT *SAVES* YOUR LIFE.

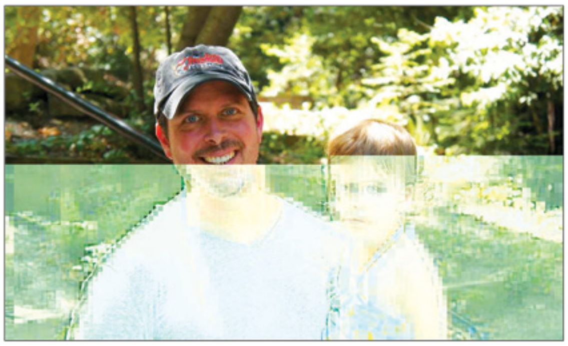
HOW IMPORTANT IS YOUR DATA?

Years of family photos. Your entire music and movie collection. Office documents you've put hours of work into. Backups for every computer you own. We ask again, *how important is your data?*

NOW IMAGINE LOSING IT ALL

Losing one bit - that's all it takes. One single bit, and your file is gone.

The worst part? **You won't know until you absolutely need that file again.**



Example of one-bit corruption

THE SOLUTION

The FreeNAS Mini has emerged as the clear choice to save your digital life. **No other NAS in its class offers ECC (error correcting code) memory and ZFS bitrot protection to ensure data always reaches disk without corruption and *never degrades over time*.**

No other NAS combines the inherent data integrity and security of the ZFS filesystem with fast on-disk encryption. No other NAS provides comparable power and flexibility. The FreeNAS Mini is, hands-down, the best home and small office storage appliance you can buy on the market. **When it comes to saving your important data, there simply is no other solution.**

The Mini boasts these state-of-the-art features:

- 8-core 2.4GHz Intel® Atom™ processor
- Up to 16TB of storage capacity
- 16GB of ECC memory (with the option to upgrade to 32GB)
- 2 x 1 Gigabit network controllers
- Remote management port (IPMI)
- Tool-less design; hot swappable drive trays
- FreeNAS installed and configured

http://www.iXsystems.com/mini

Intel, the Intel logo, Intel Atom and Intel Atom Inside are trademarks of Intel Corporation in the U.S. and/or other countries.

FREENAS CERTIFIED STORAGE

With over six million downloads, FreeNAS is undisputedly *the* most popular storage operating system in the world.

Sure, you could build your own FreeNAS system: research every hardware option, order all the parts, wait for everything to ship and arrive, vent at customer service because it *hasn't*, and finally build it yourself while hoping everything fits - only to install the software and discover that the system you spent *days* agonizing over **isn't even compatible**. Or...

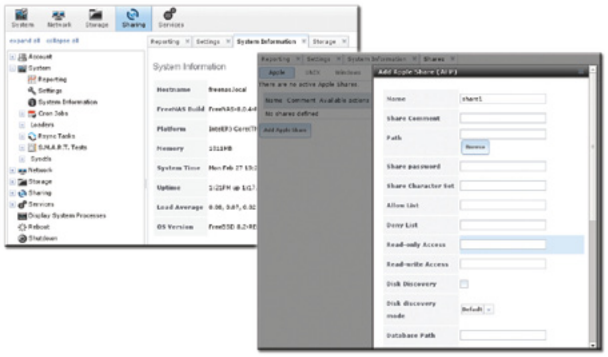
MAKE IT EASY ON YOURSELF

As the sponsors and lead developers of the FreeNAS project, iXsystems has combined over 20 years of hardware experience with our FreeNAS expertise to bring you FreeNAS Certified Storage. **We make it easy to enjoy all the benefits of FreeNAS without the headache of building, setting up, configuring, and supporting it yourself.** As one of the leaders in the storage industry, you know that you're getting the best combination of hardware designed for optimal performance with FreeNAS.

Every FreeNAS server we ship is...

» Custom built and optimized for your use case
» Installed, configured, tested, and guaranteed to work out of the box
» Supported by the Silicon Valley team that designed and built it
» Backed by a 3 years parts and labor limited warranty

As one of the leaders in the storage industry, you know that you're getting the best combination of hardware designed for optimal performance with FreeNAS. **Contact us today for a FREE Risk Elimination Consultation with one of our FreeNAS experts.** Remember, every purchase directly supports the FreeNAS project so we can continue adding features and improvements to the software for years to come. **And really - why would you buy a FreeNAS server from *anyone* else?**



System Information

Hostname	freenas.local
FreeNAS Build	FreeNAS-8.0.4-
Platform	Intel(R) Core(T
Memory	1011MB
System Time	Mon Feb 27 13:2
Uptime	1:21PM up 1:1
Load Average	0.00, 0.07, 0.02
OS Version	FreeBSD 8.2-RE

There are no active Apple shares.

Name Comment Available actions
No shares defined

Add Apple Share (AFP)

Name	share1
Share Comment	
Path	
Share password	
Share Character Set	
Allow List	
Deny List	
Read-only Access	
Read-write Access	
Disk Discovery	☐
Disk discovery mode	Default
Database Path	

FreeNAS 1U

- Intel® Xeon® Processor E3-1200v2 Family
- Up to 16TB of storage capacity
- 16GB ECC memory (upgradable to 32GB)
- 2 x 10/100/1000 Gigabit Ethernet controllers
- Redundant power supply

FreeNAS 2U

- 2x Intel® Xeon® Processors E5-2600v2 Family
- Up to 48TB of storage capacity
- 32GB ECC memory (upgradable to 128GB)
- 4 x 1GbE Network interface (Onboard) - (Upgradable to 2 x 10 Gigabit Interface)
- Redundant Power Supply

intel inside™ XEON®

http://www.iXsystems.com/storage/freenas-certified-storage/

Intel, the Intel logo, the Intel Inside logo and Xeon are trademarks of Intel Corporation in the U.S. and/or other countries.

Dear Readers,

I hope you had a great time with your family during the Easter holiday and now, you have renewed energy to work and to create new projects. It is always hard to be creative when we are tired and overwhelmed by our everyday duties. The most useful method is to learn how to cope with our daily stresses and responsibilities. I do not have the exact recipe for how to do that but I would like to recommend the article: "Revision Control Systems and Configuration Management. Part 1" written by Jose B. Alos, in which you will find very useful tips for managing large software development projects. Jose will show you what the RCS system is and how to cope with this tool.

I hope that you find many practical and easy tips which will lead you to overcome many obstacles and help you solve many problematic issues in your work.

For those of you who have dealt with NetBSD, I would like to invite you to read the article written by Diego Montalvo. In the second part of his article, you will find details on how to install the NetBSD Packages Collection (pkgsrc) which contains 3rd party software such as: databases, programming languages, text editors, and more. Then, you will discover how to install NGINX, PHP and PHP-FPM (FastCGI Process Manager) on NetBSD.

In the April issue of BSD magazine, we would like to tell you more about the Pascal programming language. Michael Van Canneyt will try to rectify the image of Pascal as an outdated language. He will show you how to write code on BSD that will run cross-platform. I think that it may be a good lesson for all of you who are looking for just such a solution which can be provided when you start programming in Pascal. I will add this article to my must-read list.

It is time to play with Gimp. The 3rd part of the article by Rob will demonstrate how to create a pastiche. Just read and use your imagination. The topic is yours.

To round things off, I would like to encourage you to read our column and conference report as it is always good to know what is going on in the world and what others think.

At the end, I would like to thank you Authors, Reviewers, Proofreaders, BSD fans, Friends, and Readers for your invaluable support and contribution.

Best regards,
BSD Team

MAGAZINE BSD

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

Contributing:
Michael Shirk, Andrey Vedikhin, Petr Topiarz, Charles Rapenne, Anton Borisov, Jeroen van Nieuwenhuizen, José B. Alós, Luke Marsden, Salih Khan, Arkadiusz Majewski, BEng, Toki Winter, Wesley Mouedine Assaby

Top Betatesters & Proofreaders:
Annie Zhang, Denise Ebery, Eric Geissinger, Luca Ferrari, Imad Soltani, Olaoluwa Omokanwaye, Radjis Mahangoe, Mani Kanth, Ben Milman, Mark VonFange

Special Thanks:
Annie Zhang
Denise Ebery

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

Publisher:
Hakin9 Media SK
02-676 Warsaw, Poland
Postepu 17D
Poland
worldwide publishing
editors@bsdmag.org
www.bsdmag.org

Hakin9 Media 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.

All trademarks presented in the magazine were used only for informative purposes. All rights to trademarks presented in the magazine are reserved by the companies which own them.

Free Pascal on BSD

Pascal is not the moribund language many people make it out to be. It is very much on par with modern languages such as Java, Python or Ruby. It runs on a wide variety of CPUs and OSes, including BSD.

What you will learn…

- Why Pascal is not dead.
- How to write code on BSD that will run cross-platform.

What you should know…

- The basics of programming.

Pascal was designed long ago, around the same time as C. Since then, C is or has become the language of choice for many open source projects, especially for low-level routines and system libraries. Object Pascal – to the surprise of its supporters, is not. In this article we'll attempt to rectify the image of Pascal as an outdated language.

On the surface of it, the C language has not fundamentally changed since it was originally developed (the introduction of C++ being a notable exception). Pascal – or its modern day version, Object Pascal – has run a more turbulent course. Originally made popular on DOS/Windows by Borland's Turbo Pascal and Delphi product, Pascal has evolved quite a lot and now possesses all the features that C or C++ have combined, and more:

- Procedural or Object Oriented programming.
- Easy string handling.
- Templates (called generics)
- Operator overloading
- Enumerators
- Threading – thread local storage is part of the language.
- Exception handling.
- Inline assembler
- Conditional compilation
- Built-in support for localization.
- RTTI (known as Introspection in some other languages)

At the same time, it knows none of the drawbacks of C. For example, the buffer overrun is a virtually unknown problem in Pascal. The strong static typing is a help in writing bug free code. The lack of macros can be seen as an advantage, although C programmers will most likely disagree with this sentiment. The Object Pascal Run-Time Library – the equivalent of libc – comes with a rich set of default routines, is highly configurable, and ensures Pascal can be used in any environment, from small embedded systems to web-based programs. On unix systems, the Object Pascal run-time can run on top of libc, or can avoid libc altogether and issue direct kernel calls.

Free Pascal

On Windows, the commercial product Delphi has always known success as an RAD environment. An almost equally old Open Source project exists which implements the same language features: Free Pascal (or FPC).

Free Pascal aims to be a Delphi compatible compiler, that runs on as many OSes and CPUs as possible. It succeeds very well in this goal: from the Intel 8080, over m68000, all 32 and 64 intel CPUs, SPARC, PowerPC (32/64 bit) and ARM: FPC supports all these CPUs. It even outputs Java Bytecode. The range of supported OSes is equally wide: DOS, OS/2, Windows, Gameboy, Solaris, Atari, Linux, Mac OS, iOS and – obviously – the various flavours of BSD.

Currently at version 2.6.4, Free Pascal binaries can be downloaded for all supported systems, including several versions of BSD (from the project's website: *http://www.freepascal.org/*). Free Pascal is present in the BSD ports system; at least the *fpc-base*, *fpc-units* and *fpc-utils* packages must be installed.

The Free Pascal project contains many routines, organized in so-called packages, that give access to popular C libraries and provide a unified database access system that gives access to all popular SQL based databases:

Open source (Firebird, MySQL, PostGres, SQLite) or commercial (Oracle, MS-SQL) alike. Installing from the ports system will take care of all the necessary dependencies. During the installation, you will be prompted to install various other packages as well, mostly bindings to useful C libraries will cause installation of these libraries. Free Pascal also comes with full documentation of the language, and compiler usage and RTL routines; all in all over 3000 printed pages.

When not using the ports system (not recommended unless you are comfortable setting up dependencies), installation can be done from the binary installer or from source. Free Pascal is self-contained, meaning the compiler itself is written in Pascal, making it hard to bootstrap if no prior installation of FPC is available. Therefore, binary installation packages are provided for FreeBSD i386/x86_64, versions 8, 9 and 10. OpenBSD and NetBSD packages are also provided for the same CPU architectures. The sources for the released versions are available as tarballs, but the daily version can be retrieved through subversion:

```
svn co http://svn.freepascal.org/svn/fpc/trunk fpc
```

will download the latest sources in an *fpc* directory. To install from source, a working installation of Free Pascal is needed, and GNU make. With these in place, a simple

```
gmake all
gmake install
```

in the fpc source directory will recompile and install Free Pascal. The sources are always guaranteed to compile with the latest officially released Free Pascal version (2.6.2 or 2.6.4). You can check the current version with the `fpc -i` command.

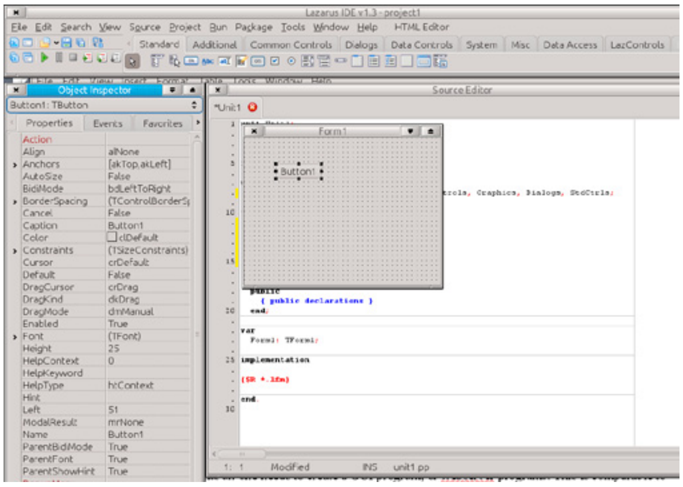
The behaviour of the compiler – search paths and so on – is controlled through a configuration file, `/usr/local/etc/fpc.cfg`. The ports installation will set up sensible defaults and, for most practical purposes, there should be no need to edit this file, but the file format (in essence a copy of the command-line options) is completely documented.

Lazarus

Pascal programs can be written using any editor; the Free Pascal compiler is a regular command-line program. Object Pascal – especially Delphi – programmers are used to a bit more comfort.

Enter the Lazarus IDE, a natural companion to Free Pascal. It is an IDE much like KDevelop, Eclipse or, on Windows, Visual Studio and Delphi. In addition to being a highly intelligent Object Pascal editor, it is also a Rapid Application Development (RAD) tool specifically designed for use with Free Pascal, and runs on almost all platforms that Free Pascal supports. In contrast with the compiler, it requires X-Windows to be present and will not run on a console. It does offer a console build tool (*lazbuild*) that can be used to automate builds.

Not only does Lazarus offer an extremely intelligent code editor with syntax highlighting, code completion and 'intellisense', but it also allows a user to visually design and debug GUI programs or web-server programs: a true RAD environment for those that prefer to code like that. A screenshot of Lazarus in action is in Figure 1.


Lazarus IDE v1.3 - project1
File Edit Search View Source Project Run Package Tools Window Help HTML Editor
Standard Additional Common Controls Dialogs Data Controls System Misc Data Access LazControls
Object Inspector
Button1: TButton
Properties Events Favorites
Source Editor
*Unit1
Form1
Button1
1: 1 Modified INS unit1.pp

Action	
Align	alNone
Anchors	[akTop,akLeft]
AutoSize	False
BidiMode	bdLeftToRight
BorderSpacing	(TControlBorderS
Cancel	False
Caption	Button1
Color	☐ clDefault
Constraints	(TSizeConstraints)
Cursor	crDefault
Default	False
DragCursor	crDrag
DragKind	dkDrag
DragMode	dmManual
Enabled	True
Font	(TFont)
Height	25
HelpContext	0
HelpKeyword	
HelpType	htContext
Hint	
Left	51
ModalResult	mrNone
Name	Button1
ParentBidiMode	True
ParentFont	True
ParentShowHint	True

```
  trols, Graphics, Dialogs, StdCtrls;

  public
    { public declarations }
  end;

var
  Form1: TForm1;

implementation

{$R *.lfm}

end.
```

Figure 1. *The Lazarus IDE*

Programs written in Free Pascal and Lazarus can be compiled to run on Windows, Mac, Linux and obviously BSD, unaltered. Indeed, one even has the choice of using Qt or GTK on Unix-based systems and not a single line has to be altered, changing the widgetset to use for the final program is just a setting in the IDE. Lazarus tries to live up to its motto: "write once, compile everywhere". Cross-compilation is supported, but is not recommended for the faint of heart.

All this is accomplished through the LCL – Lazarus Class Library: a set of high-level classes that implements all one needs to create a GUI program, or webserver programs. This library of classes is comparable to wxWidgets or Qt and GTK themselves. Lazarus can be downloaded from *http://www.lazarus.freepascal.org/.*

Lazarus has already 2 spin-offs, and even some competition: X++ coder, MSE-IDE, CodeTyphon.

The installation of Lazarus is slightly more complicated than Free Pascal itself. The default widget set is GTK2, meaning that a complete development environment for GTK2 (and hence X11) must be installed. The same holds true for the Qt version. If all dependencies are satisfied, a simple

```
gmake all
```

or

```
gmake bigide
```

will compile the IDE; the second command adds some extra functionalities to the IDE. The various dependencies make using the ports system a safer option. To function properly, the Lazarus IDE needs its own sources and the Free Pascal sources. This will let the code editor do all its magic such as identifier completion, code completion and much more. It will also function without the sources, but functionality will be restricted to that of a simple editor with syntax highlighting. It is therefore recommended to leave the extracted sources available on the system, and the Lazarus IDE will ask you where the sources are located on startup: just point to the directory where the sources are, and the IDE will do the rest.

Pascal programming

Pascal was designed for teaching, so it is rather verbose, but easy to understand. The smallest possible Pascal program is shown in Listing 1. This is a valid program, which will run, but it is not very useful, so we'll change it to the "hello world" program. Hello, world! in its Pascal version is shown in Listing 2. Compiling this is a matter of (assuming you saved the sources as *hello.pp*)

```
fpc hello.pp
```

this will result in a ready-to-go program. When executed, it will produce the familiar greeting on the console.

Pascal has all the usual constructs such as loops, if then, case (switch) and subroutines, as shown in Listing 3. This is not what makes Pascal different from C.

Listing 2 shows one of the aspects where Pascal does differ from C: Pascal is a case insensitive language. This is largely a matter of habit. The only thing where this really matters is when creating identifiers: there can be no 2 identifiers that differ in name in the same scope. Pascal programmers solve this using some conventions: for example, type names are usually prefixed with a T, and the majority of Pascal programmers use CamelCase for identifiers.

Another aspect where Pascal differs from C is the use of := to assign values.

Listing 1. *The Pascal program*

```
begin
end.
```

Listing 2. *Hello World*

```
begin
  writeln('Hello world');
end.
```

Listing 3. *The usual constructs of Pascal*

```
Procedure WriteHello;

begin
   writeln('Hello world');
end;

Begin
  if (ParamCount>0) then
    writehello;
End.
```

Listing 3a.

```
Var
   a : Integer = 0;
Begin
  A:=Random(5);
  if (A=2) then
     Writeln('Got 2');
End.
```

Listing 3b.

```
if (a=2) {
  printf("Got 2\n");
}
```

Listing 4. *The Unit Keyword*

```
Unit myunit;

interface

Procedure DoSomething;

implementation

Procedure DoSomething;

begin
  writeln('Doing something');
end;

end.
```

There probably isn't a Pascal programmer that wrote supposedly equivalent C code as Listing 3a and didn't subsequently curse the C compiler for not giving him an error...

Units and Namespaces

Pascal has a concept of software modules at the language level, called units. A unit is a collection of routines, grouped together in a single source file which form a logical whole. Listing 4 shows such a unit. A unit (indicated by the *Unit* keyword) contains 2 parts:

- The interface (started by the *interface* keyword), which contains the public declarations of the module. It can be compared to a C .h header file.
- The implementation (starting with the *implementation* keyword) contains – the name gives it away – the implementation of the publicly declared routines and structures. It can contain routines which are not declared in the interface section.

When compiling a unit, the Free Pascal compiler will create 2 files: an object file (*.o*) and a *.ppu* file. The ppu file can be compared to a pre-compiled C header file. It is a binary file which describes to the compiler what the unit interface contained. Without the associated .ppu file, the object file is useless to the compiler. This system is the reason why the Free Pascal compiler typically compiles much faster than C compilers: it doesn't need to parse the header files over and over again.

Units are also much like libraries: a Pascal programmer can simply distribute the *.ppu* and *.o* files to another pascal programmer, and that one will be able to use them in his program without needing the original sources (if they were compiled using the same version of the compiler).

To use a unit, a uses statement (comparable to "import" in java, or "using" in C#) is added to a program, as shown in Listing 5. Units can use each other, making it possible to construct a complex hierarchy of routines.

Units also form a namespace: In C, all identifiers of a program and all used libraries are put together, so it is possible that there are name clashes between public symbols of different libraries or object files. In Pascal, each unit automatically forms a namespace. It is therefore possible to have and use identifiers with the same symbol but declared in separate units. To differentiate between the 2 symbols, it suffices to prepend the symbol name with the unit name in source code referencing the symbol.

Pascal and OOP: Classes

Pascal has evolved over the years: in the 80s OOP features were added to the language; the most used variant of this was probably the then popular Turbo Pascal dialect by Borland. The Free Pascal compiler has supported this style of programming (using the *object* keyword) up to today: the objects were allocated by default on the stack, but could also be allocated on the heap, using specialized constructors. In the early 90s, Borland unveiled the Delphi RAD environment, and it used a somewhat upgraded version of OOP features, this time using the *Class* keyword. This is the Pascal OOP style that is still used today although various features have been added since the first version. Free Pascal allows you to create units in one or another of the various dialects – but only 1 dialect can be used for a unit. For this, it defines a directive which can be added to the sources:

```
{$mode XYZ}
```

The XYZ must be replaced by one of the following values:

- fpc (for strictly procedural programming)
- tp (for Turbo Pascal programming)
- objfpc (for classes programming)
- delphi (for classes programming, with some less strict checking, compatible to Delphi)
- macpas (for compatibility with the Mac OS variant of pascal)

The directive can be specified only once in a source file, best before the unit keyword. Upon encountering this

Listing 5. *How to use unit*

```
uses myunit;

begin
  DoSomething;
end.
```

Listing 6. *A sample class declaration*

```
TMyClass = Class(TObject)
Private
  Function GetProperty : Integer;
Protected
  Procedure AnAbstractMethod; virtual; abstract;
Public
  Constructor Create;
  Destructor destroy; override;
  Procedure SomeMethod:
  Property MyProperty : Integer Read GetProperty;
end;
```

directive, the parser of the compiler will put in a modus that recognizes only the language structures relevant to the chosen dialect. This feature allows use of the Free Pascal compiler to compile sources that are 30 years old. The run-time libraries contain the necessary units mimicking the units that were delivered with the original Borland Turbo Pascal products, as well as units that mimic the units delivered with Delphi.

In effect, it should be possible to compile code written a year ago, as well as code that was written 30 years ago.

To create a class that is straightforward, a sample class declaration is shown in Listing 6.

The example shows several key aspects of the OOP model implemented in Object Pascal:

- The existence of a constructor and destructor. They are marked with the special keywords constructor and destructor. Other than that they behave like normal methods. The constructor name can be chosen; the destructor name must be Destroy, and must be virtual.
- Polymorphism: methods must explicitly be marked virtual, otherwise they are static, and cannot be overridden in descendant classes. To override a virtual method, a keyword (override) must be used.
- Classes can contain abstract methods. These are methods for which no implementation is present in the class, but for which an implementation is expected to be provided by descendant classes. The compiler will warn if you construct a class that contains abstract, not implemented, methods.
- Object Pascal does not allow multiple inheritance, as found in C++. It does allow the use of interfaces: an interface is a well-defined set of methods grouped together, without implementation. Classes can implement multiple interfaces. This simply means that it implements all methods defined in these interfaces.
- Visibility of identifiers can be specified.
- Properties are used like fields, but optionally allow to specify a getter and a setter routine. The compiler will transform access to a property in a call to the getter or setter routine. That means that `A:=MyProperty`; will be (behind the scenes) transformed to a call to the private `GetProperty` routine.

Roughly, Object Pascal offers 4 visibilities:

- Private (identifiers only visible in the class itself).
- Protected (visible in the class and its descendants).
- Public (visible everywhere).
- Published (equal to published, but has RTTI associated with it, usable for introspection).

The above class could be used as in Listing 7.

GUI programming

The Lazarus IDE makes use of this style of object Pascal. It features a large class library (dubbed LCL), suitable for any GUI task. The classes are spread over lots of units: in fact, the names of the units and classes are based on the ones used in Delphi, so that code written for Delphi usually compiles seamlessly in Lazarus. This approach has made Lazarus a popular alternative for Delphi in the Pascal community.

Listing 7. *The class usage*

```
Var A : TMyClass;

begin
  A:=TMyClass.Create;
  A.SomeMethod;
  Writeln(A.MyProperty);
  A.Destroy;
```

Listing 8. *The Lazarus class library*

```
program helloworld;

uses
  Classes, Interfaces, Forms, controls, stdctrls;

Type
  TMyForm = class(TForm)
    Constructor Create(AOwner : TComponent);override;
  end;

Constructor TMyForm.Create(AOwner: TComponent);
begin
  inherited Create(AOwner);
  Caption:='Hello World';
  With TLabel.Create(Self) do
    begin
    Parent:=Self;
    Caption:='Hello World';
    end;
end;

begin
  Application.Initialize;
  With TMyForm.Create(Application) do
    Show;
  Application.Run;
end.
```

The manual way to create a ‘Hello world’ program using the Lazarus class library is relatively short, and is presented in Listing 8. The standard `TForm` class represents a top-level window in an application. In the constructor, we set the form’s caption – which will be displayed by the window manager – and construct an instance of `TLabel` (a simple text displaying widget). The widget is instructed to place itself on the form by setting the `Parent` property and by setting its `Caption` to “Hello World”, we tell it to display the hello-world caption.

Compiling and running the application in Lazarus will display the friendly greeting in an X11 window.

The `Application` instance is of class `TApplication`. In its `Run` method, it takes care of all the gory details like running an event loop to handle X11 events.

The ‘Owner’ parameter to the constructor call is used to introduce an owner-owned relationship between instances: the application instance owns the form instance. When the application instance is removed from memory, it will also remove the forms it owns from memory. That makes memory management – normally a manual affair – easier to do. The result of this code can be seen in Figure 2.

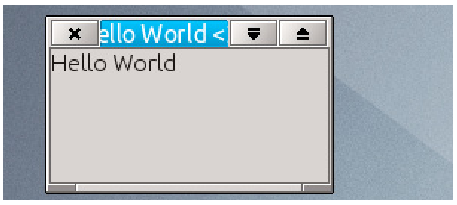
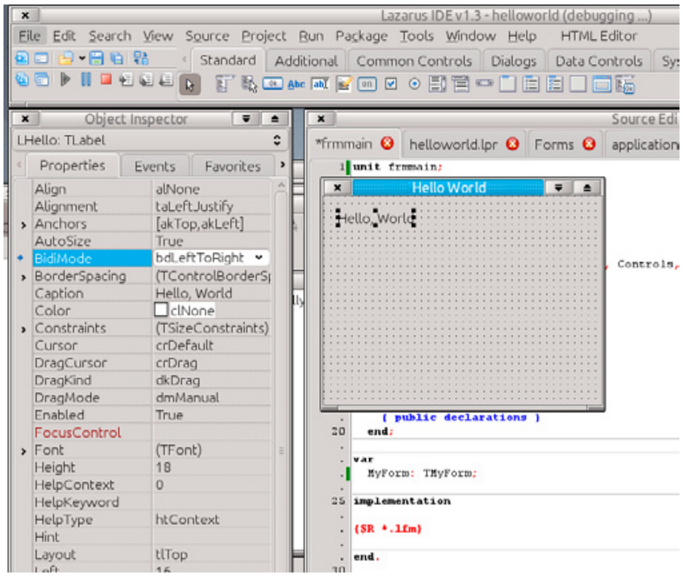


Figure 2. *A visual “Hello, World”*

While this is easy to do and code, the real power of Lazarus is not there. The real magic starts when this form is designed visually: Lazarus is a true RAD environment. No coding is required in order to produce a working hello-world example. Using a simple point-and-click paradigm, the Hello-World example can be created without writing a single line of code: the Lazarus IDE will create and maintain all code for you.

When creating a new GUI project in Lazarus, the IDE will automatically create an empty form (a window) for you in the designer. Each form in the project is a descendent of the `Tform` class, and Lazarus creates 1 unit per form (or window). The properties of the form can be manipulated with the mouse and keyboard. New controls (the pascal name for a widget) can be dropped on it from the component palette. Figure 3 shows the process: the left area is the Object Inspector, a key component of the IDE. The top window is the main IDE window, it contains the component palette: this is the list of visual (and non-visual) controls that can be used when designing a form. In the middle, a ‘Hello world’ form is displayed.


Lazarus IDE v1.3 - helloworld (debugging ...)
File Edit Search View Source Project Run Package Tools Window Help HTML Editor
Standard Additional Common Controls Dialogs Data Controls Sy
Object Inspector
LHello: TLabel
Properties Events Favorites
Source Edi
*frmmain helloworld.lpr Forms application
Hello World
Hello, World

Align	alNone
Alignment	taLeftJustify
Anchors	[akTop,akLeft]
AutoSize	True
BidiMode	bdLeftToRight
BorderSpacing	(TControlBorderS
Caption	Hello, World
Color	clNone
Constraints	(TSizeConstraints)
Cursor	crDefault
DragCursor	crDrag
DragKind	dkDrag
DragMode	dmManual
Enabled	True
FocusControl	
Font	(TFont)
Height	18
HelpContext	0
HelpKeyword	
HelpType	htContext
Hint	
Layout	tlTop

```
unit frmmain;
                                          Controls,
    { public declarations }
  end;

var
  MyForm: TMyForm;

implementation

{$R *.lfm}

end.
```

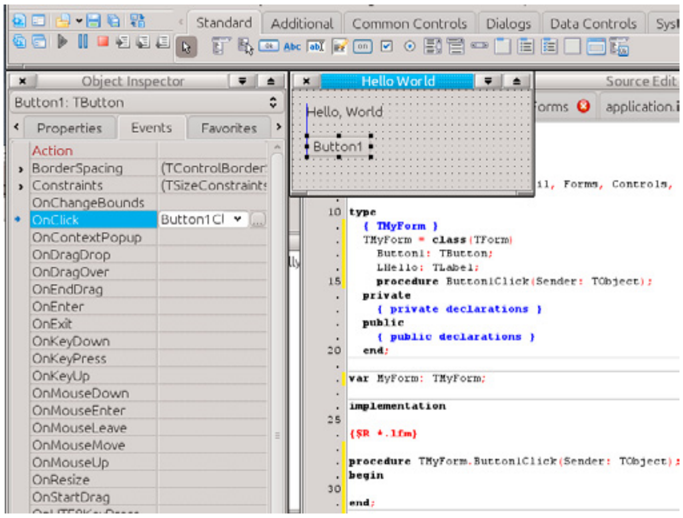
Figure 3. *Visually designing a form*

The Object Inspector plays a central role in the designing of windows. As one drops controls on the form, their properties are displayed in the object inspector, and these can be manipulated at will.

This is achieved through the use of RTTI, already hinted at in a paragraph above. Run-Time Type Information is extra information included in the binary, allowing inspection of the contents of published properties and fields of classes: a technique known as introspection in C# or Java. This allows the Lazarus IDE to generate and maintain a textual description of a form definition (stored in a so-called *.lfm* file). This textual description is loaded at run time, and is used to re-create the form.

GUI programming is mainly event-driven: clicks, typing, mouse movement. To react on these, Lazarus provides callback functions, called Event Handlers. These are achieved through a kind of delegation. Figure 4 shows how this functions: the ‘Events’ page of the object inspector shows the possible events for the selected control. There are 2 ways to attach a callback (event handler) to a control: selecting an existing callback from the list, or pressing the ellipsis button to create a new one.

When this is done, the IDE will create a new method in the form’s class definition, creates an empty method in the implementation section, and the code for the event handler can be typed at once. The name of the method can be chosen at will (obviously it must be a valid Pascal identifier), but the IDE will construct a default name based on the name of the event and the control to which the event is attached. The same event handler can be attached to multiple controls: at run time, the control from which the event originated is passed to the event handler in the *Sender* parameter.



Action	
BorderSpacing	(TControlBorder
Constraints	(TSizeConstraint
OnChangeBounds	
OnClick	Button1Cl
OnContextPopup	
OnDragDrop	
OnDragOver	
OnEndDrag	
OnEnter	
OnExit	
OnKeyDown	
OnKeyPress	
OnKeyUp	
OnMouseDown	
OnMouseEnter	
OnMouseLeave	
OnMouseMove	
OnMouseUp	
OnResize	
OnStartDrag	

```
il, Forms, Controls,

type
  { TMyForm }
  TMyForm = class(TForm)
    Button1: TButton;
    LHello: TLabel;
    procedure Button1Click(Sender: TObject);
  private
    { private declarations }
  public
    { public declarations }
  end;

var MyForm: TMyForm;

implementation

{$R *.lfm}

procedure TMyForm.Button1Click(Sender: TObject);
begin

end;
```

Figure 4. *Creating an event handler*

The IDE maintains all the code required to make this work: except for the content of the event handler, no code has been written by the programmer. This makes RAD and designing GUI programs easy: the programmer is relieved from the boilerplate code that is needed to make a responsive GUI.

Database access

No programming language can be complete without routines for database access. Free Pascal and Lazarus are no exception. They provide a unified access mechanism to most popular SQL databases, commercial or open source. This support is present in the db and sqldb units.

The small command-line program in Listing 9 will dump the contents of a table to standard output:

The same code can be used to connect to any other SQL supported database: the only difference would be the used connection class: the code in Listing 8 uses *TPQConnection*, which connects to PostGres, but changing this class to e.g. *TIBConnection* would connect to a firebird database instead.

Listing 9 demonstrates several other aspects of Object Pascal, such as exception handling: the use of Try finally (and try except) is part of the language. Throwing an exception (called Raising an exception in Pascal) for error conditions is used in all standard provided classes. Also shown is the use of enumerators. The statement:

```
For F in Q.Fields do
```

will enumerate the fields in the query result; the field class is returned in the variable F.

The Lazarus IDE and Free Pascal classes take these concepts further: the database classes can also be dropped on a form and manipulated in the Object Inspector, they are non-visual components. A nice upshot of this is that the result of an SQL query on a database can be viewed 'live' in the Lazarus designer, and the content is shown in the very controls that you have created for it. Those that prefer a pure object oriented approach (not RAD) can also choose from several persistence frameworks that implement an object/relational mapping.

Web programming

Operating systems such as BSD and Linux are often used as webservers. Pascal is suitable in this domain too: the HTTP protocol, HTML and such are largely text based. Easy string manipulation has always been a strong point of Pascal and, because of this, traditional

Listing 9. *The Object Pascal aspects*

```
uses db,sqldb,pqconnection;

Var C : TSQLConnection;
      Q : TSQLQuery;
      F : TField;
begin
  C:=TPQConnection.Create(nil);
  try
    // Set up connection
    C.UserName:='myuser';
    C.Password:='mypassword';
    C.DatabaseName:='mydatabase';
   // Set up transaction
    C.Transaction:=TSQLTransaction.Create(C);
    // Set up SQL statement.
    Q:=TSQLQuery.Create(C);
    Q.Database:=C;
    Q.SQL.Text:='SELECT * FROM MyTable';
    // Fetch the data
    Q.Open;
    // Dump the data
    While not Q.EOF do
      begin
      For F in Q.Fields do
        Writeln(F.FieldName,' : ',F.AsString);
      Q.Next;
      end;
  finally
    // Always free the connection
    C.Free;
  end;
end.
```

problems such as buffer overruns are not something that easily occurs in Pascal.

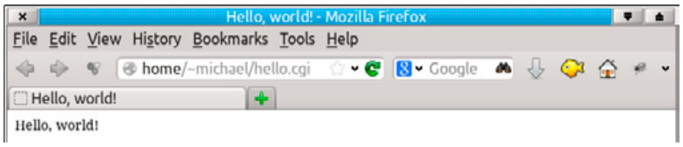
Free Pascal and Lazarus come with the necessary classes to make web programming really easy.

CGI, FastCGI, Apache loadable modules or running your own webserver: it is all possible. Moreover, the business logic of the application does not need to be changed for this: the same code can be used in all 4 environments.

Listing 10 shows the 'Hello-world' program for the web.

The basis of the request handling logic is that the web support classes will examine the URL, and based on the path of the URL will call the correct handler for the request. A request handler can be associated with each path.

The code shown in Listing 10 will register a handler (a descendent class of `TCustomHTTPModule`) for the URL path 'HelloWorld' in a CGI program. Since in this example there is only one registered handler, any URL (even


Hello, world! - Mozilla Firefox
File Edit View History Bookmarks Tools Help
home/~michael/hello.cgi
Google
Hello, world!

Hello, world!

Figure 5. *The web-based hello-world program*

Listing 10. *How to register a handler*

```
uses
  fpCGI, httpdefs, fpHTTP;

Type
  THelloWorld = class(TCustomHTTPModule)
    Procedure HandleRequest(ARequest : TRequest;
   AResponse : TResponse); override;
  end;

Procedure THelloWorld.HandleRequest(ARequest:
   TRequest; AResponse: TResponse);
begin
  AResponse.Content:='<HTML><TITLE>Hello, world!</
   TITLE>'+
                '<BODY>Hello, world!</BODY></HTML>';
  AResponse.SendResponse;
end;

begin
  RegisterHTTPModule('HelloWorld', THelloWorld,True);
  Application.Title:='Hello world!';
  Application.Initialize;
  Application.Run;
end.
```

On the Web

- Free Pascal: *http://www.freepascal.org/*
- Lazarus: *http://www.lazarus.freepascal.org/*

an empty one) will invoke the 'HelloWorld' handler: it acts as a fallback. After compiling, all that needs to be done is to copy the CGI binary to a location in the DocumentRoot directory of an Apache server that accepts CGI programs, pointing the browser to it, and the greeting should appear as in Figure 5.

The statements that call the various methods of an application class (`initialize`, `run` etc.), resemble the ones found in the GUI program source code: this is not a coincidence. The various application types are all descendent classes of a common TCustomApplication class. What this class does depends obviously on the type of application. In general, this is running some kind of event loop.

Changing this code to work with FastCGI, an apache module, or embedding it in a binary that acts as a webserver is simply a matter of changing the uses statement at the top of the file: for instance, changing it to `uses fpFastCgi` will turn it into a fastcgi application. That's all there is to it.

Listing 10 was the manual way to code a Hello World application. In the visual environment of Lazarus, with its easy handling of events when programming GUIs, handling web requests comes just as natural. Several frameworks have been developed in Free Pascal for use in web backends; some of them focus on the visual aspect of the DOM tree in the browser, others focus more on providing support for AJAX or JSON-RPC techniques.

Conclusion

It is not possible to give a complete overview of the possibilities of Object Pascal or to explain all language features. Therefore, this article has attempted to show that Object Pascal is suitable for any programming task by giving some examples of what can be done. A wide range of possibilities is at the disposal of the Pascal programmer: whether one prefers to program as one would in plain C, using the traditional POSIX interfaces, or one prefers to use powerful OOP classes, Object Pascal has it.

MICHAEL VAN CANNEYT

The author has been involved in Free Pascal since the early start, making the first unix (well, linux) port of the compiler. He has written almost all documentation singlehandedly and is responsible for maintaining a large part of the Free Pascal classes as well as some of the tools provided with Free Pascal.

Revision Control Systems and Configuration Management. Part 1

Software Configuration Management has been a traditional concern since the beginnings of the IT era. BSD development teams have traditionally used CVS as the main version control system for all their projects, including kernel development. However, CVS has some major drawbacks that can be superseded using more modern tools that outclass CVS in order to achieve the same goal.

What you will learn…

- How to manage large software development projects using RCS tools
- Make the right choice depending on the nature of your programming project
- Understand the weaknesses and strengths of the different RCS paradigms in use
- How to use and administer these RCS tools for BSD-Unix systems

What you should know…

- Intermediate UNIX OS background as end-user and administrator
- Some experience with CVS environment for version control
- Experience with Ports system package
- Software development experience

One of the more popular CVS tools was a system called RCS, which is still distributed with many computers today. Even the popular Mac OS X operating system includes the rcs command when you install the Developer Tools. This tool basically works by keeping patch sets (that is, the differences between files) from one revision to another in a special format on disk; it can then recreate what any file looked like at any point in time by adding up all the patches.

Notice that there are other Open Source systems for RCS not using a changeset approach but snapshots like Git, originally developed in C and Perl, which use SHA-1 digests instead of numbers to control the different versions stored in the distributed repositories.

Repository Models. Centralized vs. Distributed

The next major issue that people encounter is that they need to collaborate with developers on other systems. To deal with this problem, Centralized Version Control Systems (CVCSs) were developed. These systems, such as CVS and Subversion, have a single server that contains all the versioned files, and a number of clients that check out files from that central place. For many years, this has been the standard for version control (see Figure 1-2).

This setup offers many advantages, especially over local VCSs. For example, everyone knows to a certain degree what everyone else on the project is doing. Administrators have fine-grained control over who can do what;

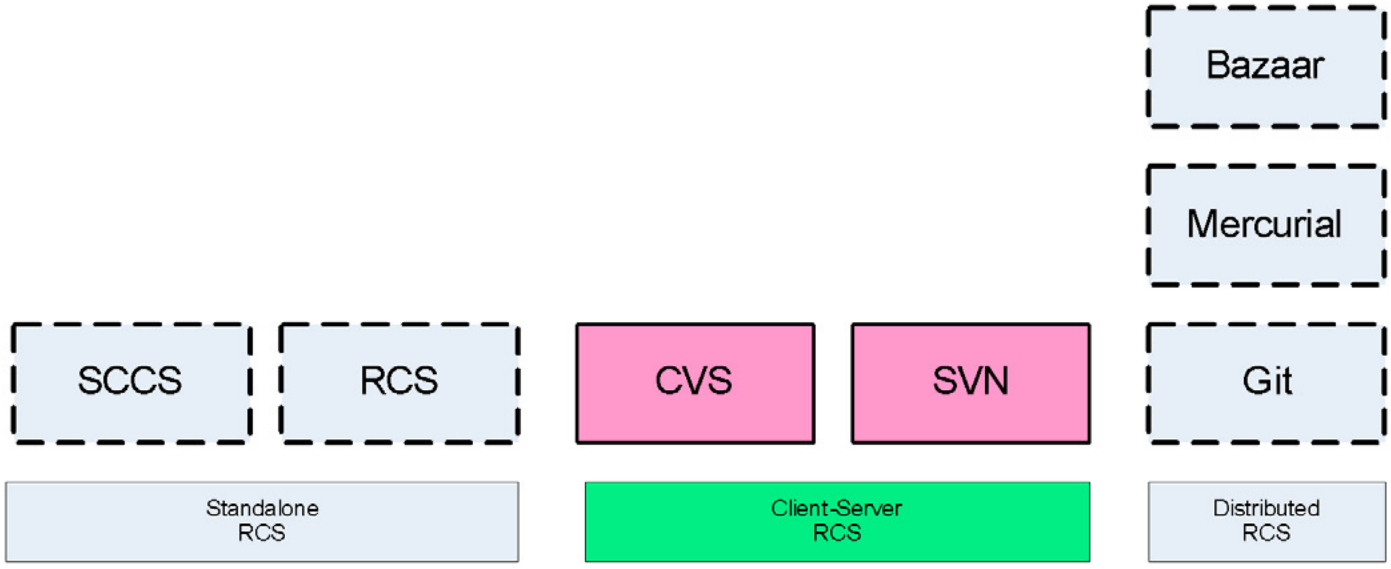


Figure 1. *Repository Model for common Revision Control Systems in use*

and it's far easier to administer a CVCS than it is to deal with local databases on every client.

However, this setup also has some serious downsides. The most obvious is the single point of failure that the centralized server represents. If that server goes down for an hour, then during that hour nobody can collaborate at all or save versioned changes to anything they're working on. If the hard disk the central database is on becomes corrupted, and proper backups haven't been kept, you lose absolutely everything – the entire history of the project except whatever single snapshots people happen to have on their local machines. Local VCS systems suffer from this same problem – whenever you have the entire history of the project in a single place, you risk losing everything.

This is where Distributed Version Control Systems (DVCSs) come in. In a DVCS (such as Git, Mercurial, Bazaar or Darcs), clients don't just check out the latest snapshot of the files: they fully mirror the repository. Thus if any server dies, and these systems were collaborating via it, any of the client repositories can be copied back up to the server to restore it. Every checkout is really a full backup

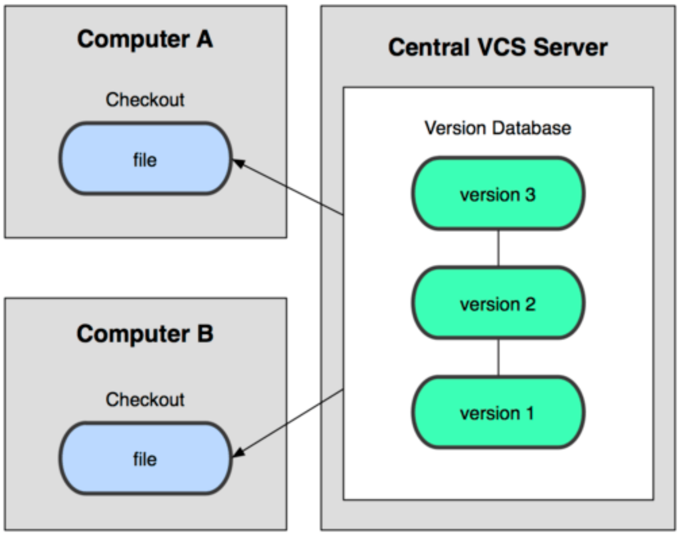


Figure 2. *Central VCS server*

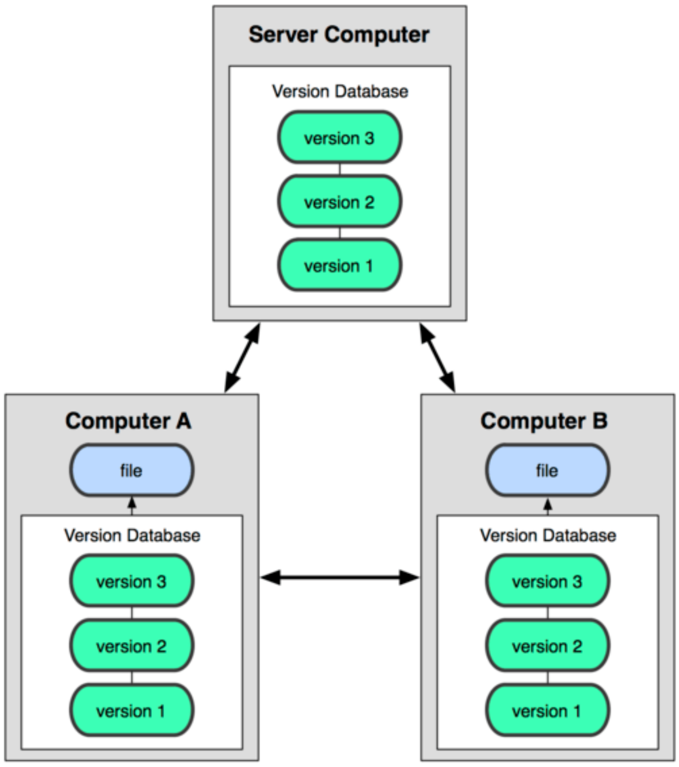


Figure 3. *Server computer*

of all the data (see Figure 1-3). For this reason, one of the most important drawbacks for distributed repositories is the poor performance for initial repository loads. This is the price to be paid for higher availability and failure protection using this redundant approach.

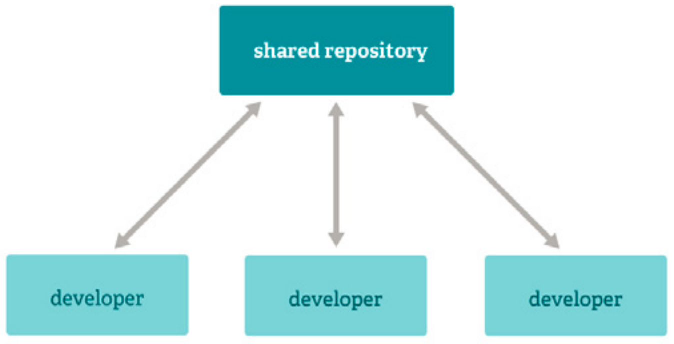


Figure 4. *Subversion workflow*

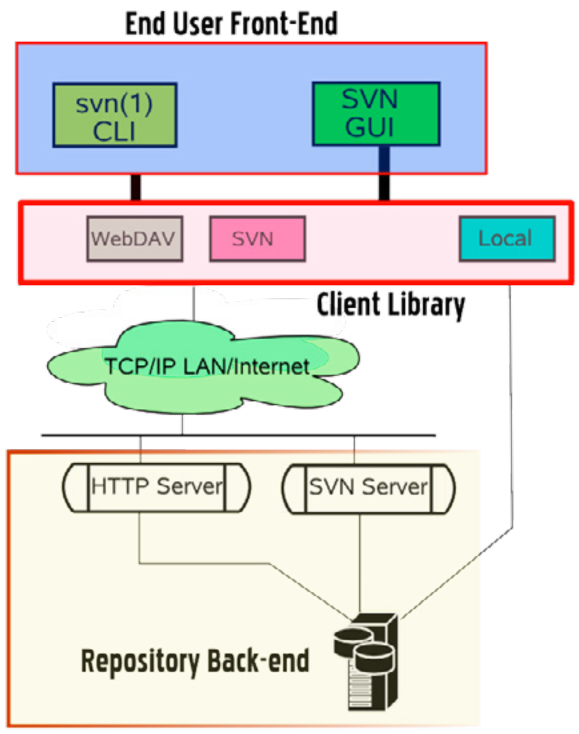


Figure 5. *Subversion client-server architecture*

Client-Server SCM Applications. Subversion

Before starting with this section, we assume the reader has a certain knowledge about CVS client-server system, as the main aim of these paragraphs are to explain by comparison the different approaches and strategies followed by both SCM systems. Since 80s CVS is the "de facto" revision control system used for most development projects, for instance BSD OS flavors.

Subversion explained

Subversion is an Open-Source client-server based version control system released by the Apache Software Foundation and it has become the "de facto" standard for Software Development version control after CVS. It is used to maintain current and historical versions of files such as source code and documentation. That is, Subversion manages files and directories, and the changes made to them, over time.

This allows you to recover older versions of your data or examine the history of how your data changed. Subversion can operate across networks, which allows it to be used by people on different computers.

Subversion platform architecture consists of a client-server architecture. The clients request files and modifications to a central Subversion repository. In this document we will focus on a specific subversion deployment, which is summarized in the following Figure 5.

Current deployment of Subversion consists of a central repository located on a development server named `ECD11461` and the access to the repository is done by HTTP protocol. All repositories are accessible on the URL: Table 1

```
http://ecd11461//svn/<name_of_repository>
```

To be able to see a browseable list all projects hosted in ecd11461 refer to:

```
http://ecd11461/svn
```

`ECD11461` provides a special repository named *sandbox.* The use of this repository is just for testing purposes; anyone can commit in the sandbox repository. This repository will be erased periodically.

Table 1. *Subversion Access protocols*

URL Schema	Access Method
file:///srv/svn	Direct Access provided by local disk
http://ecd11461/svn	Remote Access by WebDAV protocol provided by Apache HTTP
https://ecd1146/svn	Remote Access by WebDAV+SSL cypher
svn://ecd11461:/srv/svn	Remote Access to a svnserve server
svn+ssh://ecd11461:/srv/svn	Remote Access through SSH tunnel to a svnserve server

Clients to use with Subversion

In this document, we propose two different Subversion clients to be used in current deployment. One is a graphical client and the other is a command line client. A graphical client has the advantage of being "visual" but lacks some of the SVN client features, but it is suitable for regular use. The command line client will support the graphical one when needed.

Graphical SVN Clients

Despite the ease of use of plain SVN commands, to provide an ease-to-use interface for the end-user community using MS Windows platforms requires counting on graphical SVN clients to perform all activities related to the revision control system for source code and configured items by means of SVN. Currently, several alternatives are available within the FOSS community:

- RapidSVN. This is a multiplatform SVN client still in the development phase and attached to an SVN version, available at *http://rapidsvn.tigris.org*.
- SubclipseSVN is an Eclipse plugin that allows direct access to the SVN repository for Eclipse users.
- KDEsvn. This is the KDE client for Subversion, available for a wide variety of Unix-like platforms.

Whatever the client to be used, the use of the command-line `svn(1)` tool is highly recommended for development tasks.

Most relevant features for these SVN clients are:

- Shell integration: TortoiseSVN integrates seamlessly into the Windows shell (i.e. the explorer). This means you can keep working with the tools you're already familiar with, and you do not have to change into a different application each time you need functions of the version control.
- Icon overlays: The status of every versioned file and folder is indicated by small overlay icons. That way you can see right away what the status of your working copy is.
- Easy access to Subversion commands: All Subversion commands are available from the explorer context menu. TortoiseSVN adds its own submenu there.
- Graphical implementation of some Subversion commands: Such as diff or conflicts resolution.

Most relevant features for these SVN clients are:

- Shell integration: TortoiseSVN integrates seamlessly into the Windows shell (i.e. the explorer). This means you can keep working with the tools you're already familiar with, and you do not have to change into a different application each time you need functions of the version control.
- Icon overlays: The status of every versioned file and folder is indicated by small overlay icons. That way you can see right away what the status of your working copy is.
- Easy access to Subversion commands: All Subversion commands are available from the explorer context menu. TortoiseSVN adds its own submenu there.
- Graphical implementation of some Subversion commands: Such as diff or conflicts resolution.

Command-Line client

SVN Command line client provides non-dependent platform access to SVN server repositories granting the whole power of SVN features and options to the user. Given the proper client version (according to the server) a user can deal with all the server features. Compatible clients for the current Subversion server can be found in *http://www.sourceforge.net.* Most of them have been released under the General Public License (GPL).

Available subcommands are:

- add
- blame (praise, annotate, ann)
- cat
- changelist (cl)
- checkout (co)
- cleanup
- commit (ci)
- copy (cp)
- delete (del, remove, rm)
- diff (di)
- export
- help (?, h)
- import
- info
- list (ls)
- lock
- log
- merge
- mergeinfo
- mkdir
- move (mv, rename, ren)
- propdel (pdel, pd)
- propedit (pedit, pe)
- propget (pget, pg)
- proplist (plist, pl)
- propset (pset, ps)

- resolve
- resolved
- revert
- status (stat, st)
- switch (sw)
- unlock
- update (up)

Basic Work Cycle
Subversion has numerous features, options, bells, and whistles, but on a day-to-day basis, odds are that you will use only a few of them. In this section, we'll enumerate the most common things that you might find yourself doing with Subversion in the course of a day's work.

The typical work cycle looks like this:

- Update your working copy. svn update: Bring changes from the repository into the working copy
- Make the changes. svn add: Put files and directories under version control, scheduling them for addition to the repository. They will be added in the next commit. svn delete: Remove files and directories from version control. svn copy: Duplicate something in the working copy or repository, remembering history. svn move: Move and/or rename something in the working copy or repository.
- Examine your changes. svn status: Print the status of working copy files and directories. svn diff: Display the differences between two revisions or paths.
- Possibly undo some changes. svn revert: Restore pristine working copy file (undo most local edits).
- Resolve conflicts (merge others' changes). svn resolve: Resolve conflicts on working copy files or directories.
- Commit your changes. svn commit: Send changes from your working copy to the repository.

Strategies of Subversion Usage

Usage for Cooperative development
A usual tactic widely used in OpenSource projects that might be followed here, is based upon the use of a CVS/SVN repository to provide updated working copies for developers. These copies can be modified later on and these modifications will be sent to the SVN/CVS repository administrators in order to make the decision on including these updated in further versions.

To achieve this goal, the process to be executed can be summarized in the following steps, that can be extrapolated for SVN/CVS usage.

Get a working copy of the repository. In the case of using CVS, it is required to indicate the authentication process defined on this repository.

```
$ cvs -d $CVSROOT login
$ cvs -d $CVSROOT checkout my_project
```

For those developers using an Apache-based SVN interface, it is enough to perform a checkout. The HTTP server will be in charge of managing the authentication and authorisation processes:

```
$ svn checkout http://svn.remote.org/my_project
```

Update if it is required, the changes performed on a working copy. Notice that these operations are similar for both CVS and SVN servers.

```
$ cvs update -d
$ svn update
```

Perform the required changes on the different files and directories present in the working copy. Send the changes based on the previous version to the CVS/SVN repository.

```
$ cvs update -d
$ svn update
```

And eventually, generate the patch to be applied containing the different lines between both versions.

```
$ cvs diff -u > myproj.patch
$ svn diff tag1 tag2 > myproj.patch
```

Keep in mind that they shall only contain the changes related to the previously approved change.

Verify the contents of this patch generated in the previous step, coming back to the previous version of our working copy.

```
$ cvs update -C
```

and apply the patch:

```
$ patch -p0 < myproj.patch
```

Once these changes have been executed, it is possible to deliver the patch to the repository administrators or, alternatively to the project managers by electronic mail.

If, on the contrary, in the case of persisting the use of CVS, we wish to keep a copy of the repository, but keeping a backup copy of all modified files, the easiest way to achieve it is by means of the command:

```
$ cvs update -d -C
```

This process may be extended to every development project involving a large community of developers and it is only applied to all operations related to the support and management of source code.

Usage for branching development

The common structure underlying the SVN top-level repository is depicted in Figure 6 and shall be applied to all projects managed by Subversion. This structure allows fulfilment of the three main requirements for complex-project development:

- Multiple project allocation
- Mainline versioning control supported
- Branching and variant development supported

This structure allows project isolation so that only authorized users can access their respective projects under SVN control.

The use of the branching feature requires `svnmerge` to be installed before proceeding with the remaining instructions in this section.

- Create a new SVN branch – A new branch can be mainly created from the mainline of development by copying a given version to a branch named `my_branch`:

```
$ svn copy svn://ecd11461/svn/<project>/trunk \ svn://
   ecd11461/svn/<project>/branches/my_branch -m "Creating
   MY_BRANCH"
```

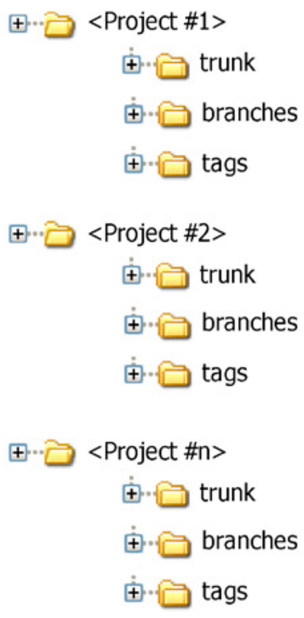


Figure 6. *Recommended SVN Repository Structure*

- Check-out a branch – The contents of the new branch can be downloaded for further amendment by checking-out the contents to a local directory:

```
$ svn checkout svn://ecd11461/svn/<project>/branches/
   my_branch
```

Also externals must be modified to point to the new A-T branch by issuing the following command:

```
$ svn propedit svn:externals .
```

- Initialise `svnmerge` tracking in the branch. If you plan to pull trunk into the branch for a project, this step shall be performed.

```
my_branch/ $> svnmerge init
property 'svnmerge-integrated' set on '.'
my_branch/ $> svn ci -F svnmerge-commit-message.txt
```

- Initialise `svnmerge` tracking on the trunk.

```
trunk/ $> svnmerge init svn://ecd11461/svn/<project>/
   branches/my_branch
property 'svnmerge-integrated' set on '.'
trunk/ $> svn ci -F svnmerge-commit-message.txt
```

Eventually it is perfectly possible to merge changes from one branch to another by following the steps written down below:

- Work from a clean checkout of the branch you want to merge TO and check the availability of changes to merge.

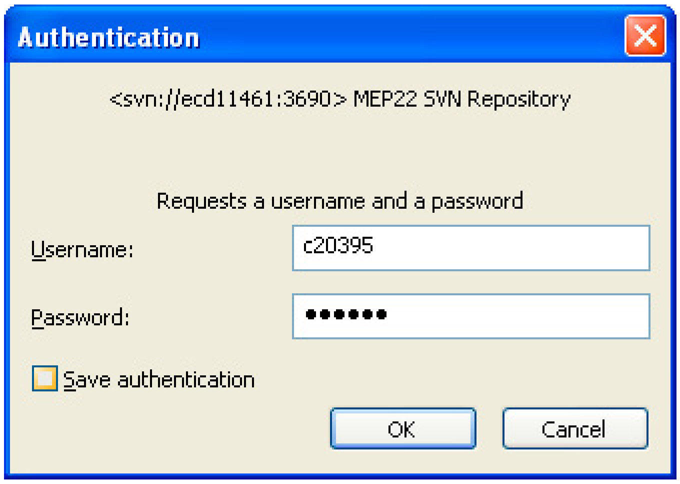
```
$> svnmerge avail -b -l

-----------------------------------------------------------
r584362 | gozer | 2007-10-12 21:00:47 -0700 (Fri, 12 Oct
   2007) | 1 line
Changed paths:
   A /perl/modperl/branches/mybranch (from /perl/modperl/
   trunk:584361)

creating mybranch
-----------------------------------------------------------
r584363 | gozer | 2007-10-12 21:05:32 -0700 (Fri, 12 Oct
   2007) | 3 lines
Changed paths:
   M /perl/modperl/branches/mybranch
```

Initialize merge tracking via `svnmerge` with revisions 1-584361 from `svn://ecd11461/<project>trunk`.

Then, merge the ones you wish to merge. For instance, if you want to merge changes from 584362 to 585363, issue the command:

```
$> svnmerge -r 584362-584363
```

Once the conflicts arisen as a result of changes made have been fixed, check in the merged version:


Authentication

<svn://ecd11461:3690> MEP22 SVN Repository

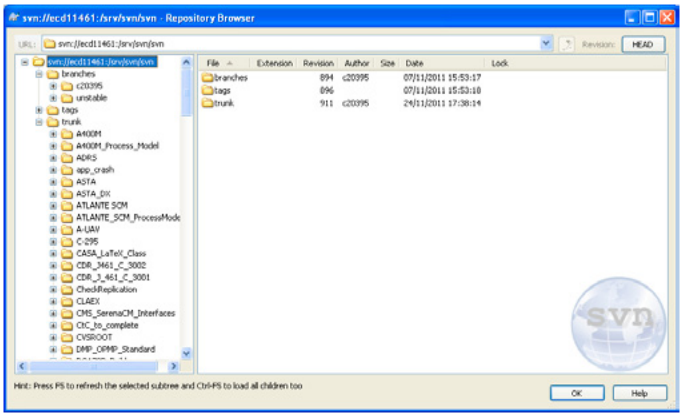
Requests a username and a password

Username: c20395

Password: ••••••

☐ Save authentication

Figure 7. *SVN Repository simple authentication*


svn://ecd11461:/srv/svn/svn - Repository Browser

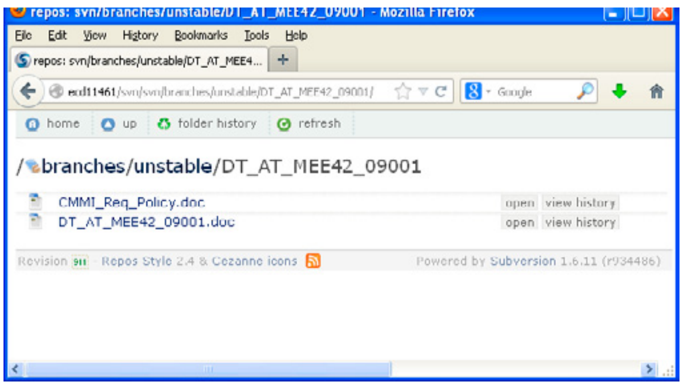
URL: svn://ecd11461:/srv/svn/svn Revision: HEAD

```
svn://ecd11461:/srv/svn/svn
  branches
    c20395
    unstable
  tags
  trunk
    A400M
    A400M_Process_Model
    ADRS
    app_crash
    ASTA
    ASTA_DX
    ATLANTE SCM
    ATLANTE_SCM_ProcessMode
    A-UAV
    C-295
    CASA_LaTeX_Class
    CDR_J461_C_3002
    CDR_J_461_C_3001
    CheckReplication
    CLAEX
    CMS_SerenaCM_Interfaces
    CtC_to_complete
    CVSROOT
    DMP_OPMP_Standard
```

File	Extension	Revision	Author	Size	Date	Lock
branches		894	c20395		07/11/2011 15:53:17	
tags		896			07/11/2011 15:53:10	
trunk		911	c20395		24/11/2011 17:38:14	

Hint: Press F5 to refresh the selected subtree and Ctrl-F5 to load all children too

Figure 8. *SVN Browse Repository Example*


repos: svn/branches/unstable/DT_AT_MEE42_09001 - Mozilla Firefox

File Edit View History Bookmarks Tools Help
repos: svn/branches/unstable/DT_AT_MEE4...
ecd11461/svn/svn/branches/unstable/DT_AT_MEE42_09001/
Google
home
up
folder history
refresh

/branches/unstable/DT_AT_MEE42_09001

CMMI_Req_Policy.doc open view history
DT_AT_MEE42_09001.doc open view history

Revision 911 · Repos Style 2.4 & Cezanne icons Powered by Subversion 1.6.11 (r934486)

Figure 9. *SVN Repository Browse using WebDAV Protocol*

```
$> svn ci -F svnmerge-commit-message.txt
```

The process concludes at this point with merging changes generated in the development branch `my _ branch` to the mainline or trunk for the selected project.

Basic Operations

This section will introduce the use of Subversion in a normal day's work. How to get a working copy of the repository, deal with modifications and put all the new stuff back into the repository. In order to provide a better understanding, let us assume we use a MS Windows client platform to operate with SVN server.

Browsing SVN Repository

There are two main ways to browse the contents of a given SVN repository. The first possibility, widely used by MS Windows-lovers, is the use of SVN Tortoise client (RepoBrowser option), which requires previous authentication to access: Figure 7.

The common structure for controlling changes and their time evolution is given in Figure 8.

The second one is to use HTTP+DAV support as shown in Figure 9. This is platform-independent and does not require an external tool, just a compliant HTTP browser such as MSIE or Mozilla among others.

Getting a working copy from SVN repository

To start with Subversion, the first step is to get a copy of an empty repository to be populated with the desired directories structure for our development project.

```
$ cd $HOME
$ svn checkout http://ecd11461/repository
```

The process of creating new files and directories within our empty repository in our local computer is simple. Firstly, three directories (trunk, branches, and tags) are used to allocate the main branch of development, the secondary one and the labels folder, respectively. The way to proceed is as shown below:

```
$ cd svn_repository && mkdir trunk branches tags
$ svn add trunk branches tags
$ svn commit -message "Estructura inicial repositorio SVN"
```

These three directories will be used for all projects created and managed under the SVN repository, and allow simplication of both branching and tagging operations pertaining to its predecessor CVS.

Committing changes to SVN server

Let's assume a development project included in the directory `$HOME/myprog`. The incorporation of this project into our SVN repository requires the following commands:

```
$ cp -r myprog $HOME/svn_repository/trunk
$ cd $HOME/svn_repository/trunk
$ svn add myprog
$ svn commit -message "Added myprog project to the repository"
```

A granted member of a development team in an SVN repository may get a copy of the project from this repository by means of a *checkout*:

```
$ svn checkout http://ecd11461/svn_repository/trunk/myprog
```

In the same way as occurred with CVS, Subversion does not allow individual *checkout* of files to be performed. Checkout operations are only allowed with directories.

End-users Interaction. Cooperative work

The incorporation of further changes, performed in a working copy by a team member onto SVN repository use, is as dangerous as it is necessary. For this reason, it is encouraged to have a depth of understanding about all involved operations in order to identify which differences are to be merged.

```
$ svn status -u
M *       12  Makefile
M         12  program.c
  *       12  program.h
Head revision:     13
```

Hence, the status of changes performed in an SVN server is illustrated by the `svn status` command. Looking at the first column, the letter `M` indicates that a file has been modified in a working copy and the symbol `*` in the second column, reflects the fact that another developer has also been modified and performed a *commit* between the period from which the copy is obtained and the current moment in time. The third column shows the revision number of the file. That means that the fact of performing a *commit* on `program.c` will not be a source of problems for this file, but it will not be the case for the last two remaining files.

```
$ svn update
M *       12  Makefile
M         12  program.c
  *       12  program.h
Head revision:     13
```

In this case, the letter `U` indicates that the update of this file has been done successfully while the letter `C` gives evidence of a conflict.

By listing the files in this directory, it can be realized that SVN has been much more than filling up the original `Makefile` with lines containing the consecutive differences by means of the command `diff(1)`:

```
$ ls
Makefile       Makefile.r12   program.c
Makefile.mine  Makefile.r13   program.h
```

Now, there are four different versions of `Makefile`, the first one, without any extension is filled up as it would happen if we had used CVS. However, the three remaining versions are copies of different versions. Thus `Makefile.mine` is nothing else than the current copy available at a working copy prior to update; `Makefile.r12` is the previous update of this working copy and `Makefile.r13` is the copy of a new file in the SVN repository. That allows the user to choose the code required according to their needs in a manual way. Once this conflict has been fixed for our `Makefile`, simply type:

```
$ svn resolved Makefile
```

The latter command allows SVN to enable a *commit* operation on this file.

Branching and merging in SVN

According to the directories structure, the creation of a new branch for a specific Project in SVN is as simple as copying the whole Project within the `branch` directory. This can be done by issuing the command:

```
$ svn cp trunk/my_projectbranches/myproject_branch_1
```

For those people unfamiliar with the terminology, the process of joining this branch with the main development branch is termed *merging*. This process consists of incorporating the contents of the directory `svn_repository/trunk/myprog` into the new branch `svn_repository/branches/myprog-branch` created:

```
$ cd svn_repository/branches/myprog-branch
$ svn merge -r 12:HEAD http://ecd11461/
```

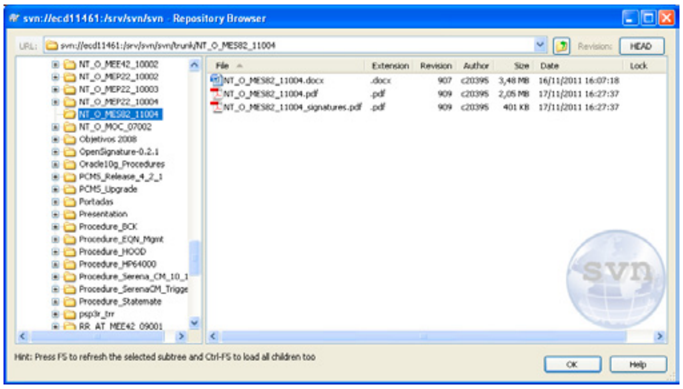
The last command joins (*merges*) the differences between version 12, which was previously used when the branch was created and the current revision in the main development branch for the Project `myprog`.

Getting the software or part of it

In order to get a working copy of the full repository or part of it, a checkout has to be performed. This is performed selecting "SVN Checkout..." in the TortoiseSVN explorer context menu. In the checkout window, you can specify some checkout options:

- URL of repository: The apache URL where the repository is accessible.
- Checkout directory: Local directory for working copy.
- Checkout Depth: Always set to Fully Recursive since this option grabs the whole repository from the URL.
- Revision: Specify the specific revision you would like to work with (HEAD is a special label that always points to the latest version). If you need help selecting some revision, a "show log" button is provided.

The process of selection for a module and/or version stored in the SVN repository is quite simple by using SVN Tortoise and is depicted in Figure 10, Figure 11 and Figure 12.


svn://ecd11461:/srv/svn/svn - Repository Browser

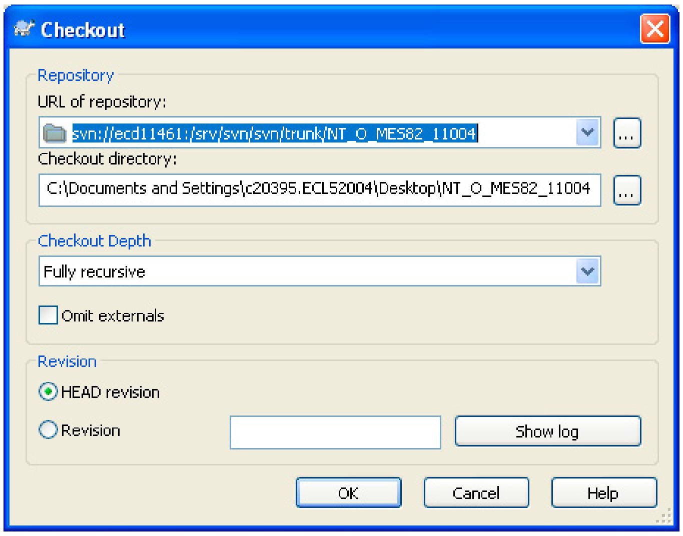
URL: svn://ecd11461:/srv/svn/svn/trunk/NT_O_MES82_11004   Revision: HEAD

NT_O_MEE42_10002
NT_O_MEP22_10002
NT_O_MEP22_10003
NT_O_MEP22_10004
NT_O_MES82_11004
NT_O_MOC_07002
Objetivos 2008
OpenSignature-0.2.1
Oracle10g_Procedures
PCMS_Release_4_2_1
PCMS_Upgrade
Portadas
Presentation
Procedure_BCK
Procedure_EQN_Mgmt
Procedure_HOOD
Procedure_HP64000
Procedure_Serena_CM_10_1
Procedure_SerenaCM_Trigge
Procedure_Statemate
psp3r_trr
RR_AT_MEE42_09001

File	Extension	Revision	Author	Size	Date	Lock
NT_O_MES82_11004.docx	.docx	907	c20395	3,48 MB	16/11/2011 16:07:18	
NT_O_MES82_11004.pdf	.pdf	909	c20395	2,05 MB	17/11/2011 16:27:37	
NT_O_MES82_11004_signatures.pdf	.pdf	909	c20395	401 KB	17/11/2011 16:27:37	

Hint: Press F5 to refresh the selected subtree and Ctrl-F5 to load all children too

Figure 10. *SVN Checking out a module (I). Select module*


Checkout

Repository
URL of repository: svn://ecd11461:/srv/svn/svn/trunk/NT_O_MES82_11004
Checkout directory: C:\Documents and Settings\c20395.ECL52004\Desktop\NT_O_MES82_11004

Checkout Depth
Fully recursive
☐ Omit externals

Revision
[✓ HEAD revision]
Revision

Figure 11. *SVN Checking-out a module (II). Select check-out details*

Once you have your local copy checked out, you can start working on it as if they were regular files. You can edit and change it, move it around, even delete the entire working copy and forget about it.

The Subversion repository will be composed of a basic layout of three root directories on each repository:

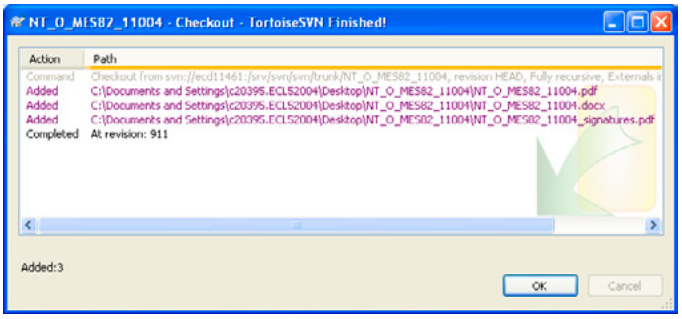
- `branches`: This directory will be used to store the different project branches.
- `tags`: Will store all the different tags created from trunk or from branches.
- `trunk`: Will store the developmental day-to-day work. It is the main project code and documentation storage. Speaking in version control language, this directory will be the HEAD of the repository where work gets done.

Getting your data into the repository

Once you have made changes into a local copy as shown in Figure 13, you will need to store them in the repository. This operation is performed by the commit command. This command tells the repository what has changed and sends modifications to the repository.

In order to commit your changes, you will have to right click into your local working copy directory (or file) containing your changes and select "SVN Commit..." in the TortoiseSVN explorer context menu.

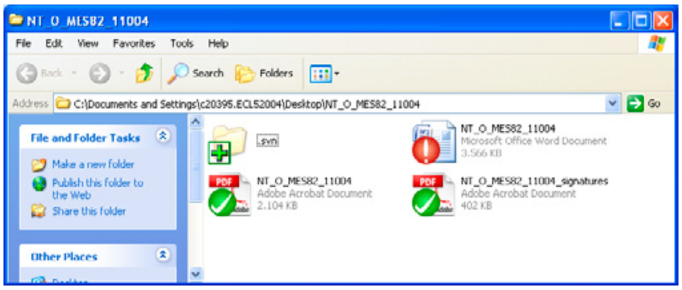
In the commit window given by Figure 14, you can select a proper message identifying the commit and select the modified files to be committed (all modified by default).


NT_O_MES82_11004 - Checkout - TortoiseSVN Finished!

```
Action     Path
Command    Checkout from svn://ecd11461:/srv/svn/svn/trunk/NT_O_MES82_11004, revision HEAD, Fully recursive, Externals i
Added      C:\Documents and Settings\c20395.ECL52004\Desktop\NT_O_MES82_11004\NT_O_MES82_11004.pdf
Added      C:\Documents and Settings\c20395.ECL52004\Desktop\NT_O_MES82_11004\NT_O_MES82_11004.docx
Added      C:\Documents and Settings\c20395.ECL52004\Desktop\NT_O_MES82_11004\NT_O_MES82_11004_signatures.pdf
Completed  At revision: 911
```

Added:3

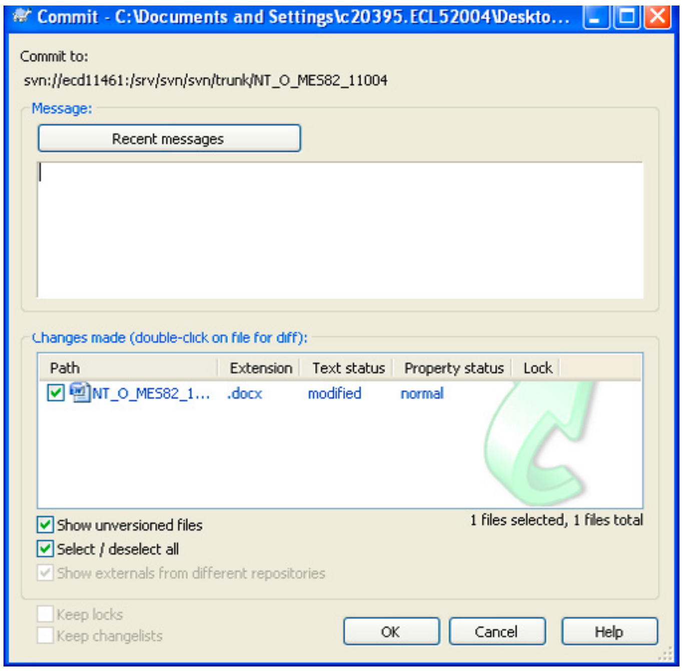
Figure 12. *SVN Checking-out a module (III). Output log*


NT_O_MES82_11004

Address: C:\Documents and Settings\c20395.ECL52004\Desktop\NT_O_MES82_11004

.svn
NT_O_MES82_11004 — Microsoft Office Word Document — 3.566 KB
NT_O_MES82_11004 — Adobe Acrobat Document — 2.104 KB
NT_O_MES82_11004_signatures — Adobe Acrobat Document — 402 KB

Figure 13. *Local changes in SVN checked-out files*


Commit - C:\Documents and Settings\c20395.ECL52004\Deskto...

Commit to:
svn://ecd11461:/srv/svn/svn/trunk/NT_O_MES82_11004

Message:

Changes made (double-click on file for diff):

Path	Extension	Text status	Property status	Lock
✓ NT_O_MES82_1...	.docx	modified	normal	

1 files selected, 1 files total

- [✓] Show unversioned files
- [✓] Select / deselect all
- [✓] Show externals from different repositories
- [ ] Keep locks
- [ ] Keep changelists

Figure 14. *SVN Commit local changes*

If the log message is not properly formatted, the commit is not allowed and an error window will appear. Proper log messages must contain between square brackets the ECR/DR/PR that triggered the change. If not applicable you can use [NA] (Not Applicable) or [II] (Initial Issue).

References and abbreviations

This section lists the specifications, standards, manuals, and other documents, including policy directives, referenced or used as source material for this plan.

- IEEE Trans. on Soft. Eng. 1975 SE 1-4 – The Source Code Control System (SCCC)
- *http://scm.tigris.org* – Introducción a los sistemas SCM.

Specific Documents

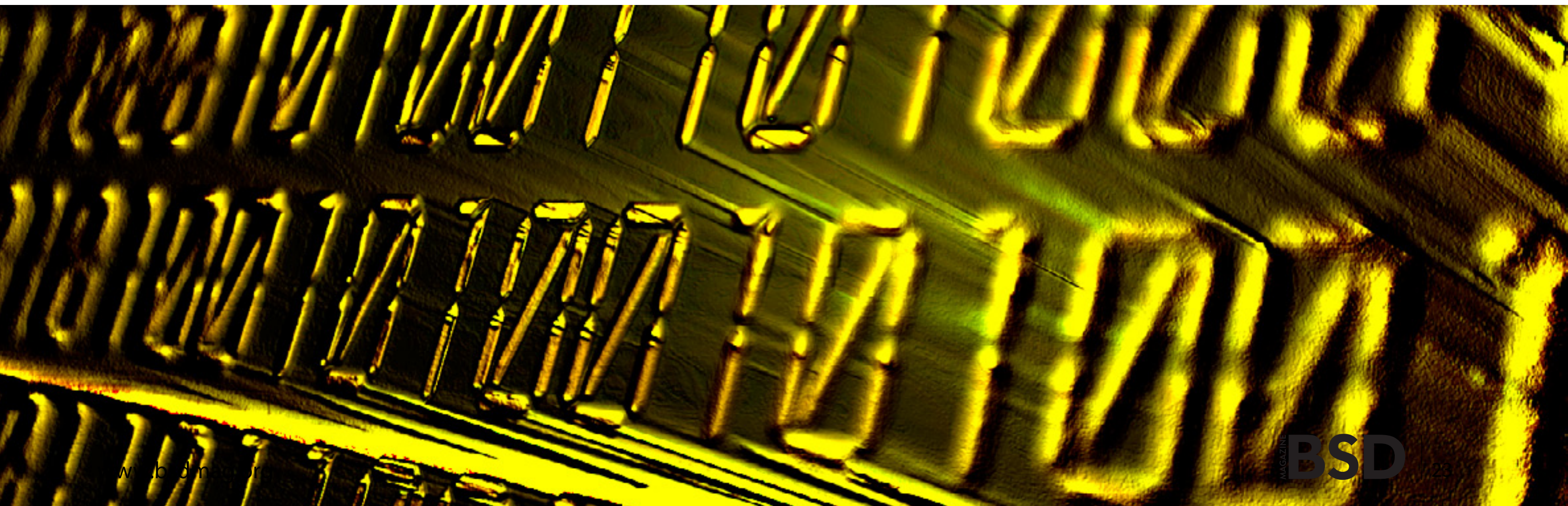
- *http://subversion.tigris.org* – Subversion Open Source Software Engineering Tools
- *http://www.webdav.org/neon/* – Project Specific Configuration Management Plan
- *http://cvsbook.red-bean.com/OSDevWithCVS_3E.pdf* – Open Source Development with CVS. M. Bar y K Fogel
- *http://svnbook.red-bean.com* – Version Control with Subversion. Ben Collins-Sussman, Brian W. Fitzpatrick, C. M. Pilato
- DP-SIS-001 – Software Quality System Management Plan
- *http://subclipse.tigris.org* – Subclipse. SVN Plugin for Eclipse
- *http://tortoisesvn.net* – The Coolest Interface to Subversion Version Control

Acronyms and abbreviations

SVN – Subversion
CVS – Control Version System
RCS – Revision Control System
HTTP – Hypertext Transfer Protocol
SSL – Secure Sockets Layer
DAV – Distributed Authoring and Versioning
COTS – Commercial Off-The-Shelf.
CSCI – Computer Software Configured Item
DAL – Development Assurance Level

JOSÉ B. ALÓS

José B. Alós has developed an important part of his professional career since 1999 as an EDS employee, as a UNIX System Administrator, mainly focused on SunOS/Solaris, BSD and GNU/Linux. Five years ago he joined EADS Defense and Security, nowadays CASSIDIAN as the person responsible for providing support for end-users in aircraft engineering departments for long-term projects. That is the main reason underneath this article as VAX/VMS systems still play a paramount role in today's aerospace industry for a wide variety of embedded RT systems conceived for mission and flight operations. He was also Assistant Professor in the Universidad de Zaragoza (Spain), specialized in the design of High Availability solutions, and his academic background includes a PhD in Nuclear Engineering and three MsC in Electrical and Mechanical Engineering, Theoretical Physics and Applied Mathematics.



Deploying NetBSD on the Cloud using AWS EC2: Part 2

In the last article, I took you through the steps for deploying a NetBSD cloud server on Amazon EC2 Web Services. In this tutorial, I will first cover the necessary steps for installing the NetBSD Packages Collection (pkgsrc) which contains 3rd party software such as: databases, programming languages, text editors and more. After the packages collection has been properly installed, I will show you how to install NGINX, PHP and PHP-FPM (FastCGI Process Manager) on NetBSD.

What you will learn…

- NetBSD Basics
- How to install and configure NGINX
- How to run PHP scripts on NGINX

You need…

- Amazon AWS account
- Terminal
- OpenSSL

I am assuming you have read the previous article and have already setup your NetBSD EC2 server, so I will move on to installing packages on your server.

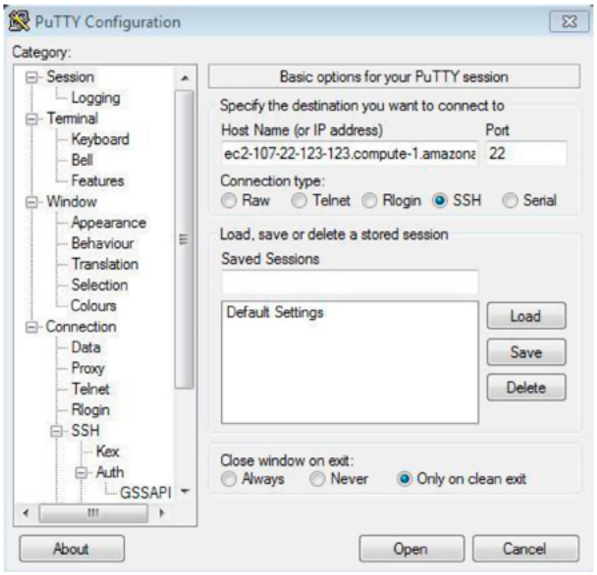
1) Login into your VPS using terminal or PuTTY (Figure 1)

```
# ssh -i  your_aws.pem root@ec2-123-12-123-123.compute-1.
    amazonaws.com
```

2) Once you are logged into your server you will begin downloading and updating *pkgsrc*. I recommend you download "current" software packages which are auto-generated daily. (Figure 2)

```
# ftp ftp://ftp.NetBSD.org/pub/pkgsrc/current/pkgsrc.tar.gz
```

Once the pkgsrc.tar.gz file is downloaded, extract it using the command below, which will create `/pkgsrc` inside



Specify the destination you want to connect to
Host Name (or IP address): ec2-107-22-123-123.compute-1.amazona
Port: 22
Connection type: Raw Telnet Rlogin [✓ SSH] Serial
Load, save or delete a stored session
Saved Sessions: Default Settings
Close window on exit: Always Never [✓ Only on clean exit]

Figure 1. *Login into your VPS using terminal or PuTTY*

the `/usr` directory. All the package sources will be stored under `/usr/pkgsrc/`

```
# tar -xzf pkgsrc.tar.gz -C /usr
```

It is very important to keep `pkgsrc` up to date for both stability and security reasons. Next you will be updating via the CVS method.

```
# cd /usr/pkgsrc && cvs update -dP
```

If you get a *No CVSROOT specified* error use the command below.

```
# cd /usr/pkgsrc && env CVS_RSH=ssh cvs up -dP
```

Once `pkgsrc` has finished updating, you can begin installing packages.

Installing Packages

Installing NGINX

```
# pkg_add ftp://ftp.netbsd.org/pub/pkgsrc/packages/NetBSD/
   amd64/6.1.3/All/nginx-1.5.12nb3.tgz
```

Installing PHP

```
# pkg_add ftp://ftp.netbsd.org/pub/pkgsrc/packages/NetBSD/
   amd64/6.1.3/All/php-5.5.10nb2.tgz
```

Installing PHP-FPM

```
# pkg_add ftp://ftp.netbsd.org/pub/pkgsrc/packages/NetBSD/
   amd64/6.1.3/All/php-5.5.10nb2.tgz
```

Creating a Web User and Public Web Directory

Create a user using the following command

```
# useradd -m webdude
```

Next, create a public web directory in the `webdude` home directory.

```
# cd ~webdude
# mkdir public_html
# chown webdude public_html
```

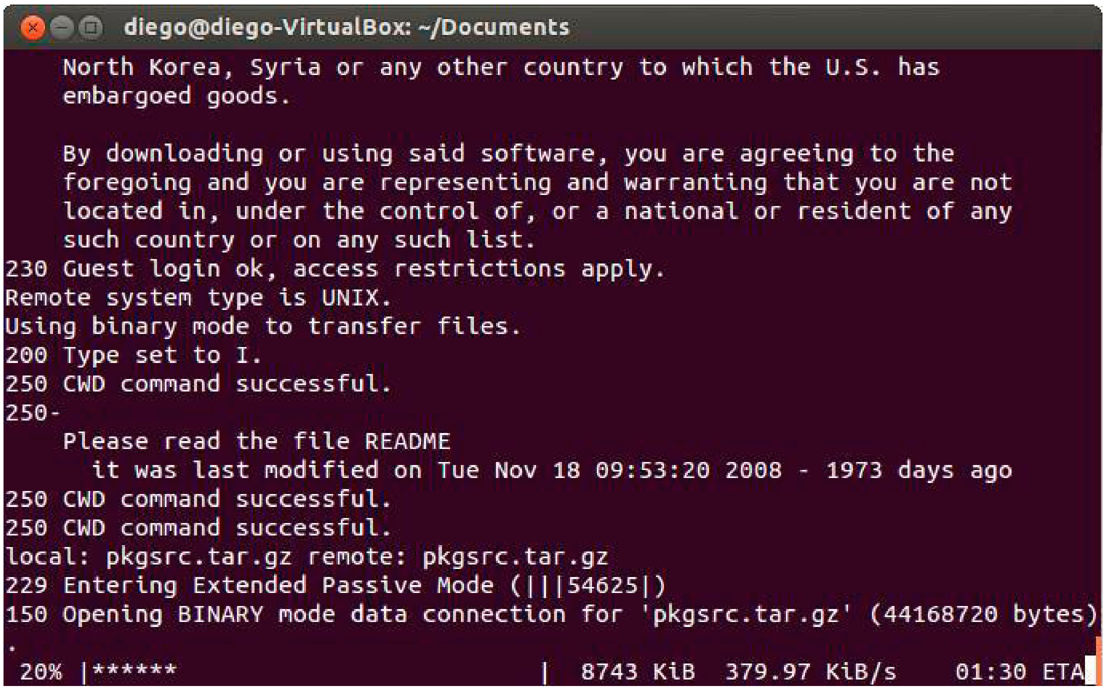
In the next step, you will be changing the default NGINX root directory to your `webuser` path and configuring PHP-FPM in the `nginx.conf` file using your favorite editor.

```
# vi /usr/pkg/etc/nginx/nginx.conf
```

Inside `nginx.conf`, change the root line to the following:

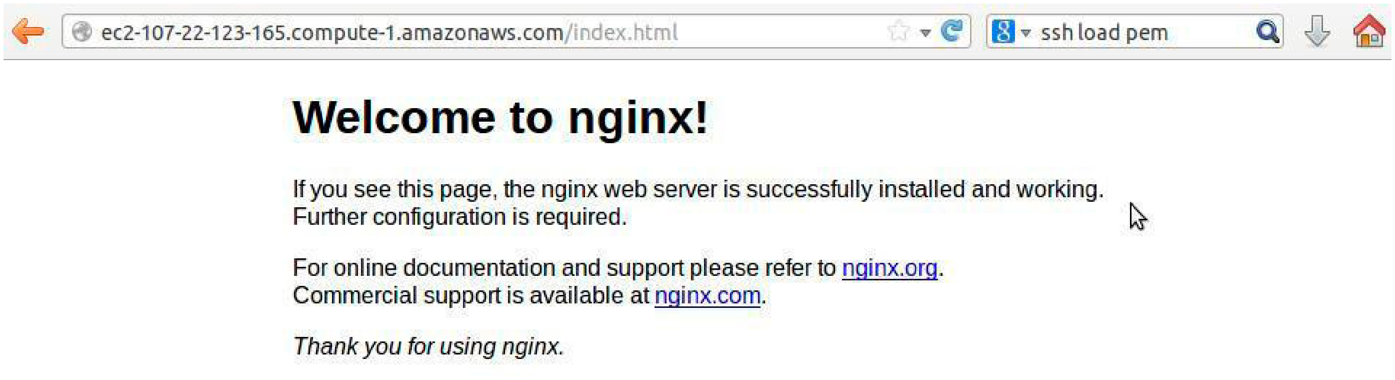
```
root /home/webdude/public_html
```

Next uncomment the following location block:



```
    North Korea, Syria or any other country to which the U.S. has
    embargoed goods.

    By downloading or using said software, you are agreeing to the
    foregoing and you are representing and warranting that you are not
    located in, under the control of, or a national or resident of any
    such country or on any such list.
230 Guest login ok, access restrictions apply.
Remote system type is UNIX.
Using binary mode to transfer files.
200 Type set to I.
250 CWD command successful.
250-
    Please read the file README
      it was last modified on Tue Nov 18 09:53:20 2008 - 1973 days ago
250 CWD command successful.
250 CWD command successful.
local: pkgsrc.tar.gz remote: pkgsrc.tar.gz
229 Entering Extended Passive Mode (|||54625|)
150 Opening BINARY mode data connection for 'pkgsrc.tar.gz' (44168720 bytes)
.
 20% |******                                |  8743 KiB  379.97 KiB/s    01:30 ETA
```

Figure 2. *Software packages*


ec2-107-22-123-165.compute-1.amazonaws.com/index.html

ssh load pem

Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

Thank you for using nginx.

Figure 3. *Welcome to NGINX" screen*

```
location ~ \.php$ {
            #root            html;
            fastcgi_pass   127.0.0.1:9000;
            fastcgi_index  index.php;
            fastcgi_param  SCRIPT_FILENAME  /home/webdude/
   public_html$fastcgi_script_name;
            include        /usr/pkg/etc/nginx/fastcgi_
   params;
        }
```

Note

You will have to copy web files from `/usr/pkg/share/examples/html` into your `/home/webdude/public_html` directory.

You are now ready to start-up NGINX and PHP-FPM

```
# php-fpm
#   nginx
```

Note

If you would like to start PHP-FPM and NGINX on boot, you will have to append the following lines to `/etc/rc.local`.

```
# vi  /etc/rc.local

  /usr/pkg/sbin/php-fpm
  /usr/pkg/sbin/nginx
```

Resources

Amazon Web Services – *http://aws.amazon.com*
AWS Free Tier Program – *http://aws.amazon.com/free/*
The NetBSD Project – *http://www.netbsd.org/*
NGINX – *http://nginx.org/*
#Pozr Dead Simple Cloud Servers – *https://www.pozr.in*

Successful Setup

On a successful setup you will be prompted with the "Welcome to NGINX" screen (Figure 3).

Having read through this article, you should have a basic understanding of the NetBSD Packages Collection, NGINX and how to run PHP scripts on your newly installed web server. If you have any questions or comments, feel free to drop me a line at *diego@pozr.in*. Keep it Moving!

DIEGO MONTALVO

Diego is the chief architect at #pozr. When he is not coding or building web technologies, Diego is ranching and skateboarding. He currently resides in both Hebbronville, Texas and San Diego, California. If you have any questions or comments you can contact him at diego@pozr.in.

Developing for Amazon Web Services?

Attend Cloud DevCon!

Cloud DevCon

June 23-25, 2014

San Francisco

Hyatt Regency Burlingame

www.CloudDevCon.net

Attend Cloud DevCon to get practical training in AWS technologies

- Develop and deploy applications to Amazon's cloud
- Master AWS services such as Management Console, Elastic Beanstalk, OpsWorks, CloudFormation and more!
- Learn how to integrate technologies and languages to leverage the cost savings of cloud computing with the systems you already have
- Take your AWS knowledge to the next level – choose from **more than 55 tutorials and classes,** and put together your own custom program!
- Improve your own skills and your marketability as an AWS expert
- Discover HOW to better leverage AWS to help your organization today

Register Early and SAVE!

A BZ Media Event

CloudDevCon

Amazon Web Services and AWS are trademarks of Amazon.com, Inc.

Getting to Grips with the Gimp – Part 3

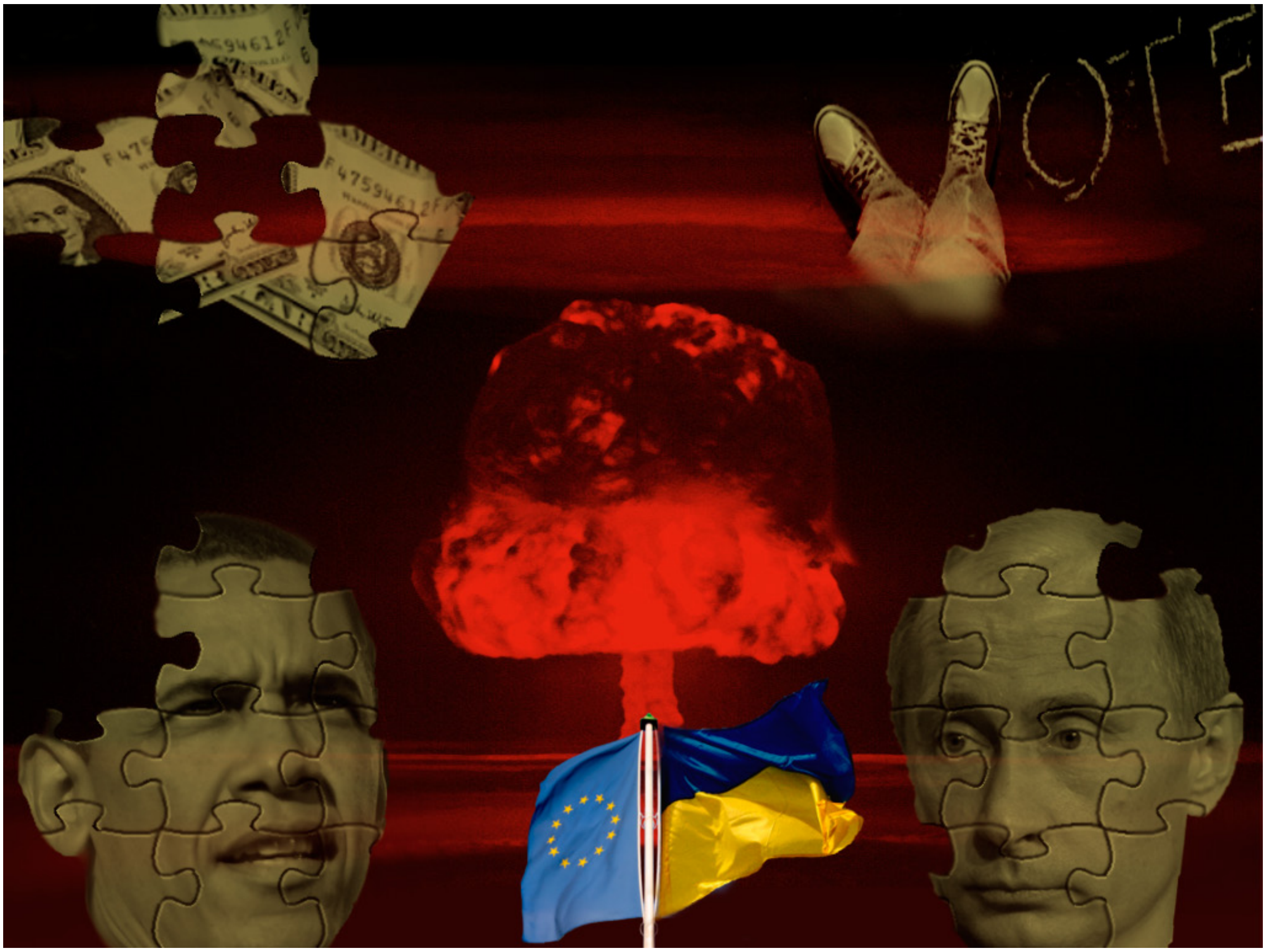
In the third in our series on the Gimp, we will create a pastiche that depicts the current political crisis in the Ukraine.

What you will learn…

- How to manipulate images like a design pro

What you should know…

- General PC administration skills



There is nothing new under the sun. Inspired by a pastiche in today's Sunday newspaper, we will create a graphic that portrays the current crisis in the Ukraine.

Step 1

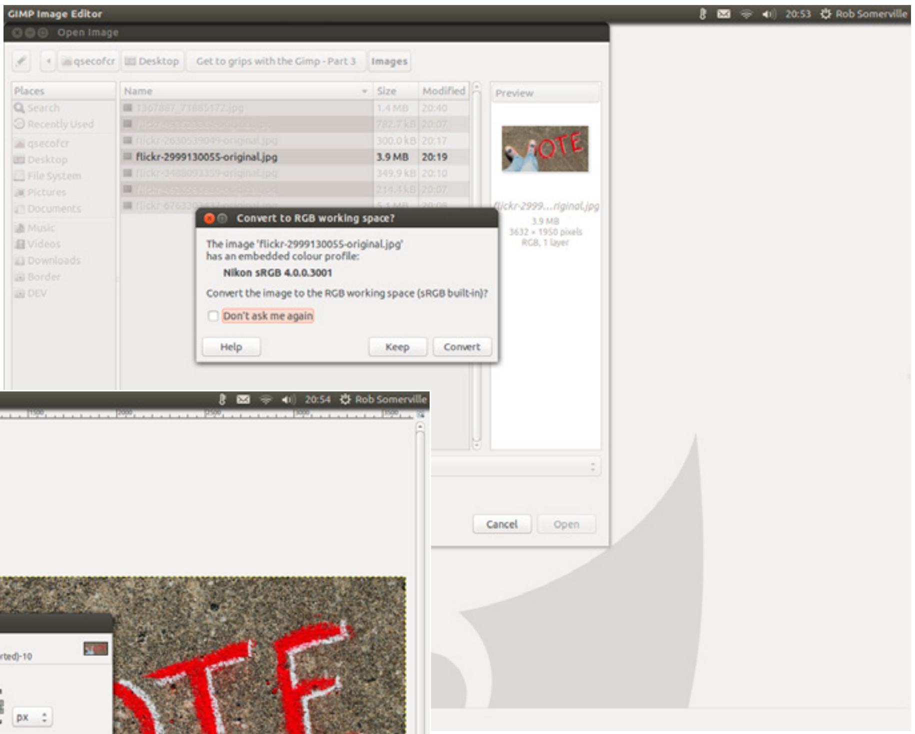
Download the images listed in Table 1.

Table 1. *Details and credits*

Image	URL	Details and credits
EU Flag	http://www.freeimages.com/photo/1367887	European flag in the wind Uploaded by Ayla87
President Putin	http://www.fotopedia.com/items/flickr-3488093359	photo by World Economic Forum on Flickr
President Obama	http://www.fotopedia.com/items/flickr-6763303437	photo by Intel Photos on Flickr
US dollar	http://www.fotopedia.com/items/flickr-2630539049	photo by iChaz on Flickr
Ukraine flag	http://www.fotopedia.com/items/flickr-493523361	photo by LancerenoK on Flickr
Nuclear explosion	http://www.fotopedia.com/items/flickr-4926596880	photo by The Official CTBTO Photostream on Flickr
Vote now	http://www.fotopedia.com/items/flickr-2999130055	photo by Theresa Thompson on Flickr

Step 2

Let's start with the "Vote Now" image. As we will use this on the right hand side of the picture, we will have to remove the US stars from the sneaker. Open the file in the Gimp and resize to a width of 640px [Figure 1 and 2].



Name	Size	Modified
flickr-2999130055-original.jpg	3.9 MB	20:19

flickr-2999...riginal.jpg
3.9 MB
3632 × 1950 pixels
RGB, 1 layer

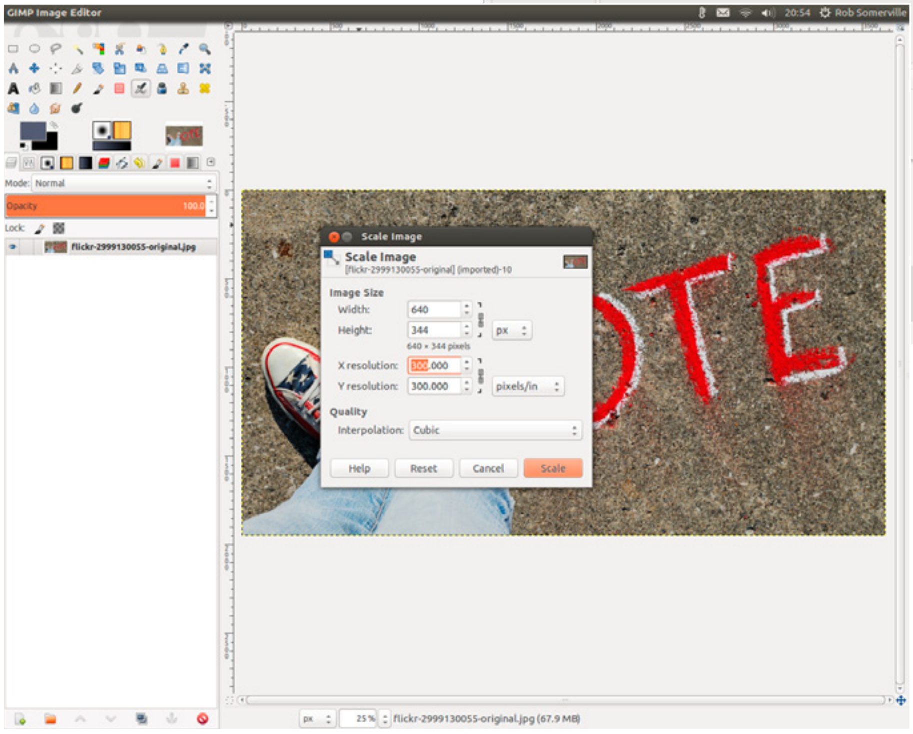
Convert to RGB working space?

The image 'flickr-2999130055-original.jpg' has an embedded colour profile:

Nikon sRGB 4.0.0.3001

Convert the image to the RGB working space (sRGB built-in)?

☐ Don't ask me again



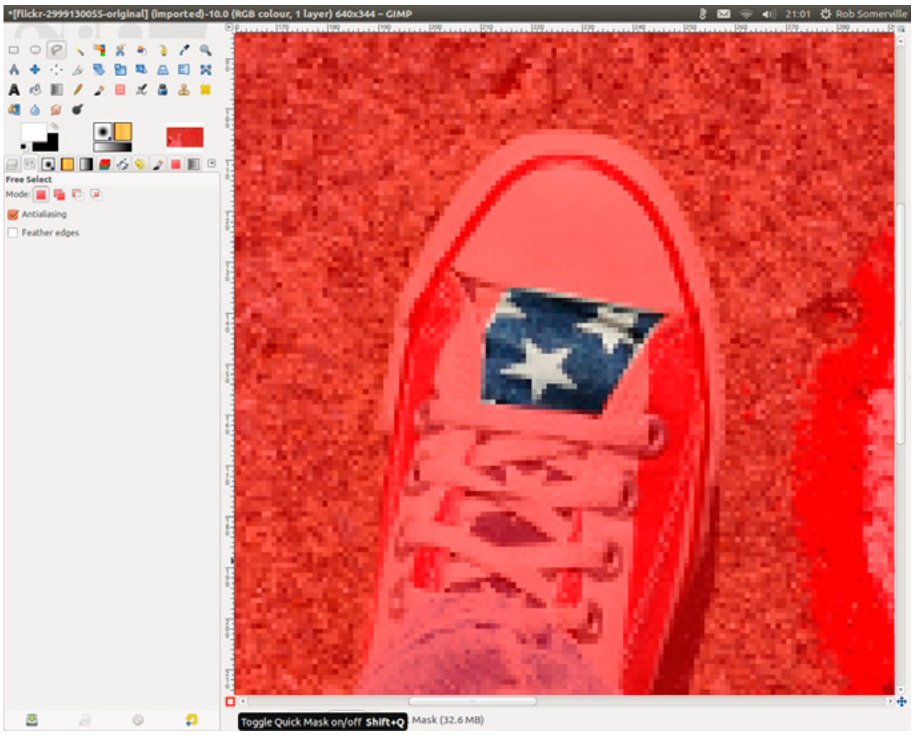
Scale Image
[flickr-2999130055-original] (imported)-10

Image Size
Width: 640
Height: 344 px
640 × 344 pixels
X resolution: 300.000
Y resolution: 300.000 pixels/in

Quality
Interpolation: Cubic

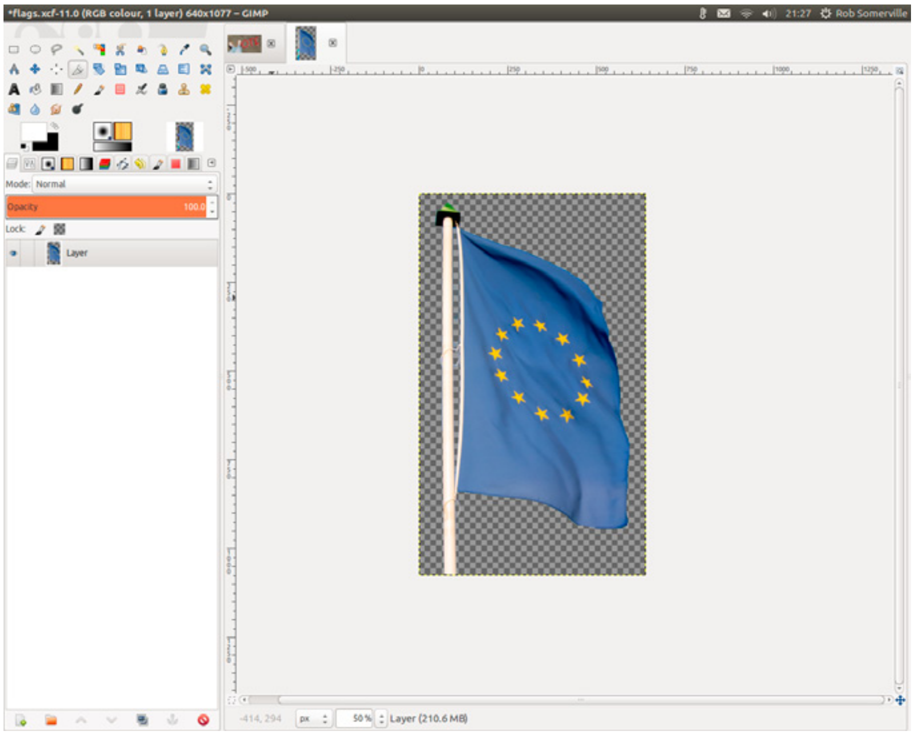
Step 3

Zoom in and, using the freehand select tool, select the blue / stars area from the sneaker. Paint the area white with the paintbrush tool or press. Deselect the area, and using the smudge tool, blend the greyer area of the top of the sneaker into the area you just painted. Repeat with the other foot. As the marching worms is not clear, I have turned on the mask to highlight the area that will not be affected by our paint process. Save the image as shoes.xcf [Figure 3].


*[flickr-2999130055-original] (imported)-10.0 (RGB colour, 1 layer) 640x344 – GIMP
21:01
Rob Somerville
Free Select
Mode:
Antialiasing
Feather edges
Toggle Quick Mask on/off Shift+Q
Mask (32.6 MB)

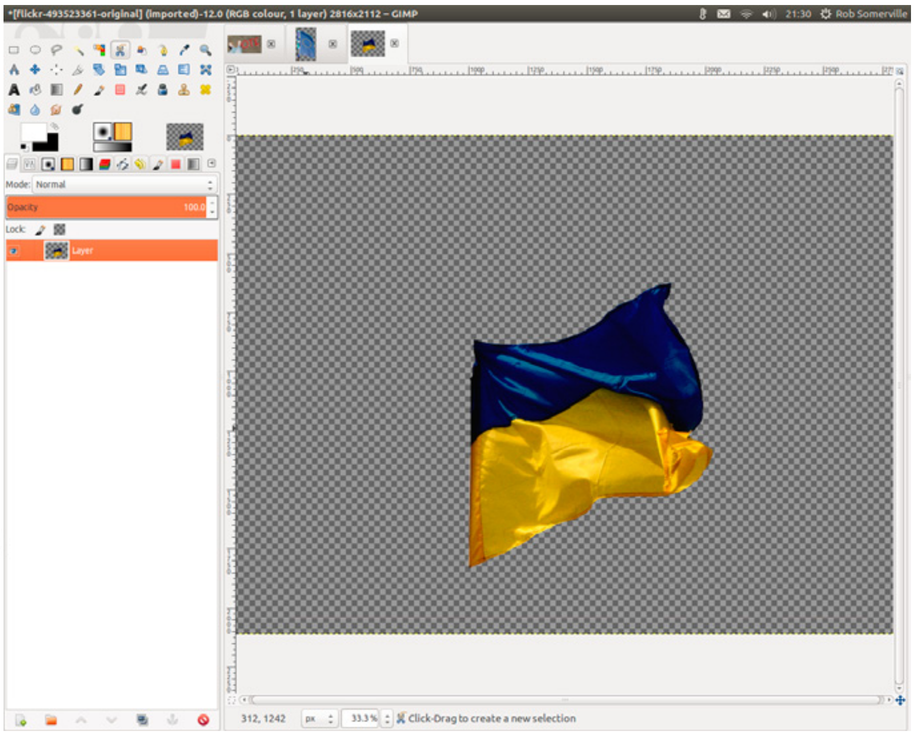
Step 4

Open the image of the European flag. Using the Scissors select tool, click around the flag and the flagpole and join the nodes up at the bottom once you have gone round the flag. Click on the middle of the flag to finish the selection, copy the area, create a new transparent and paste and anchor the selection. Delete the original layer. Using the fuzzy select tool, select and delete the sky areas between the flag and the flagpole. Crop, and save as flag.xcf [Figure 4].



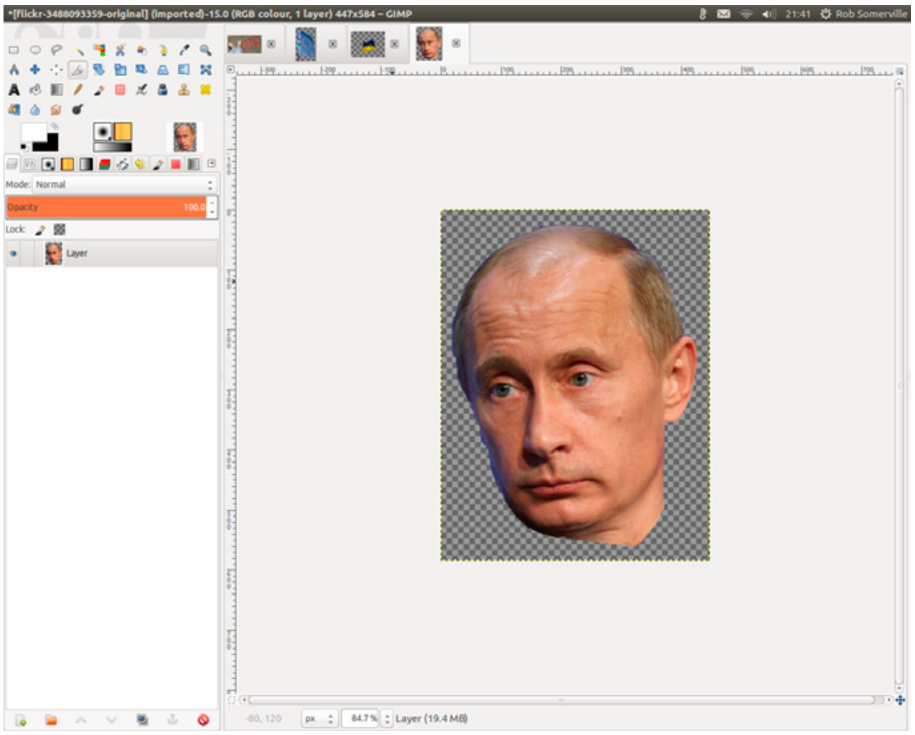
Step 5

Repeat step 4 with the Ukrainian flag, but omit the flagpole. Save as ukrainianflag.xcf [Figure 5].


*[flickr-493523361-original] (imported)-12.0 (RGB colour, 1 layer) 2816x2112 – GIMP
21:30 Rob Somerville
Mode: Normal
Opacity 100.0
Lock:
Layer
312, 1242 px 33.3 % Click-Drag to create a new selection

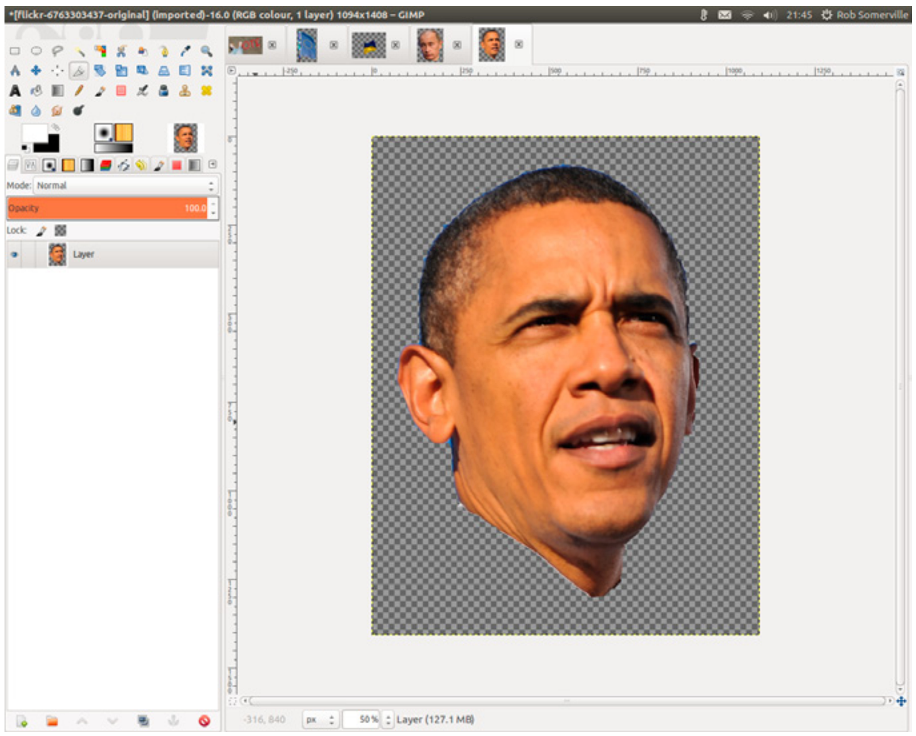
Step 6

Open the image of the Russian President, and repeat step 4 so that you have just the leader's face. Crop and save as putin.xcf [Figure 6].


*[flickr-3488093359-original] (imported)-15.0 (RGB colour, 1 layer) 447x584 – GIMP
21:41 Rob Somerville
Mode: Normal
Opacity 100.0
Lock:
Layer
-80, 120 px 84.7 % Layer (19.4 MB)

Step 7

Open the image of the US President, and repeat step 4 so that you have just the leaders face. Crop and save as obama.xcf [Figure 7].



*[flickr-6763303437-original] (imported)-16.0 (RGB colour, 1 layer) 1094x1408 – GIMP

Mode: Normal
Opacity 100.0
Layer

-316, 840 px 50% Layer (127.1 MB)

Step 8

Open the nuclear explosion image and crop the area of the mushroom cloud in the middle. Resize to width 1024px, and adjust the canvas size to 1024 x 768. Ensure the chain is unclicked, as we only want to increase the height. Adjust the layer to image size, and fill the bottom of the image with red picked from the bottom edge of the explosion. Using the smudge tool, blend the bottom and top of the image so that the hard line is erased. Save as image_001.xcf [Figure 8].

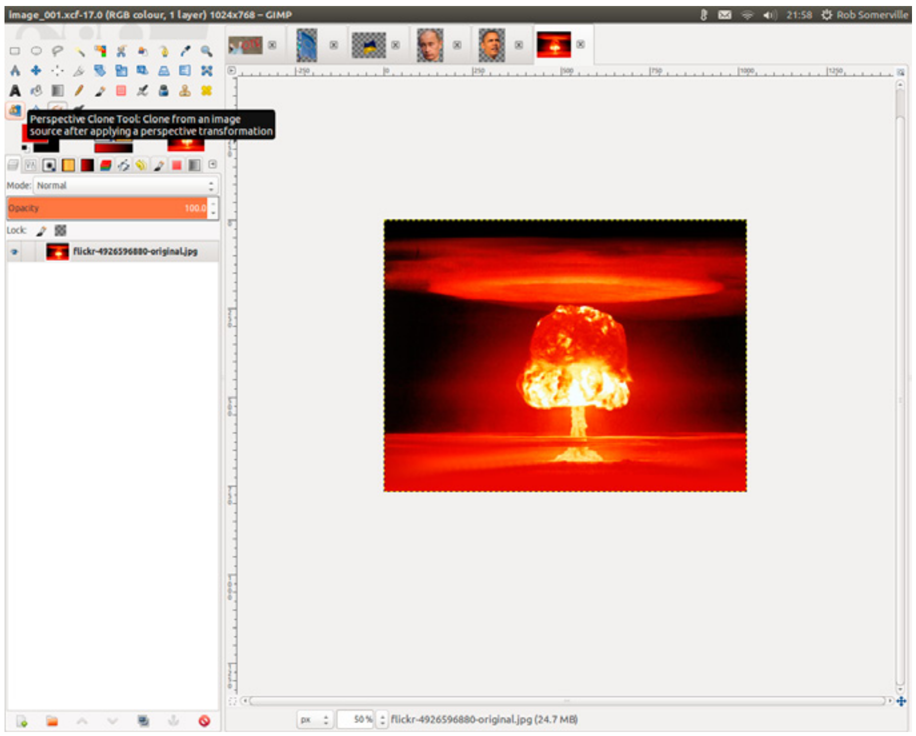


image_001.xcf-17.0 (RGB colour, 1 layer) 1024x768 – GIMP

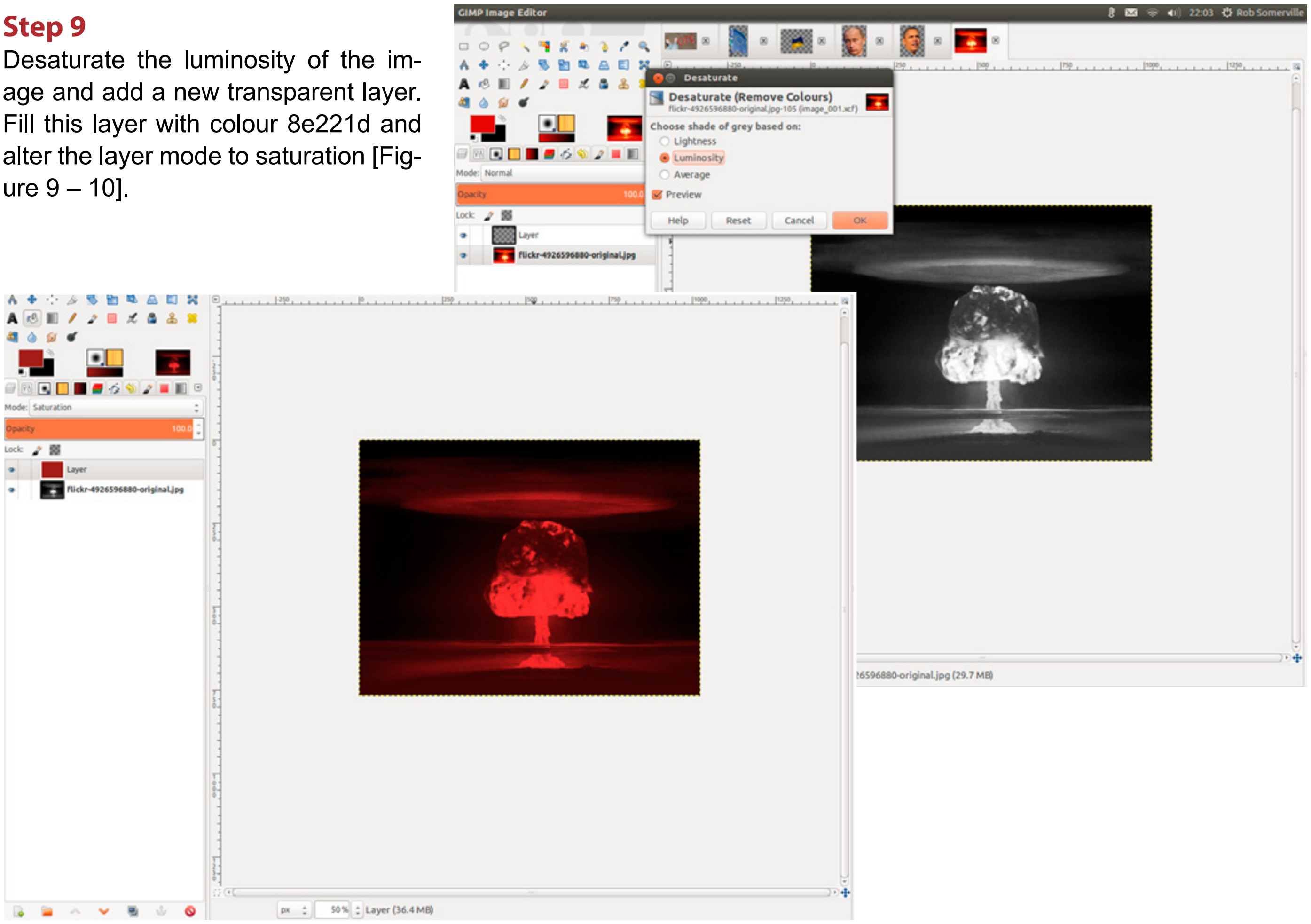
Perspective Clone Tool: Clone from an image source after applying a perspective transformation

Mode: Normal
Opacity 100.0
flickr-4926596880-original.jpg

px 50% flickr-4926596880-original.jpg (24.7 MB)

Step 9

Desaturate the luminosity of the image and add a new transparent layer. Fill this layer with colour 8e221d and alter the layer mode to saturation [Figure 9 – 10].



GIMP Image Editor

Desaturate

Desaturate (Remove Colours)
flickr-4926596880-original.jpg-105 (image_001.xcf)

Choose shade of grey based on:
( ) Lightness
(●) Luminosity
( ) Average
[✓] Preview

Step 10

Create a new transparent layer. Open the image of the US dollar, and scale to 150px. Select all, copy, and click back onto the nuclear background tab. Ensure you are on the new layer you created and paste the resized image of the dollar near the top left hand side of the image. When you are happy with the position of the layer, anchor it. Duplicate the layer, and using the rotate and move tool, reposition another copy of the dollar. Repeat until you have 3 dollars. When you are happy with the positioning, right click each dollar layer and merge down. Rename the single layer "Dollars", and scale and reposition until the layer dominates the top left hand corner of the image [Figure 11].

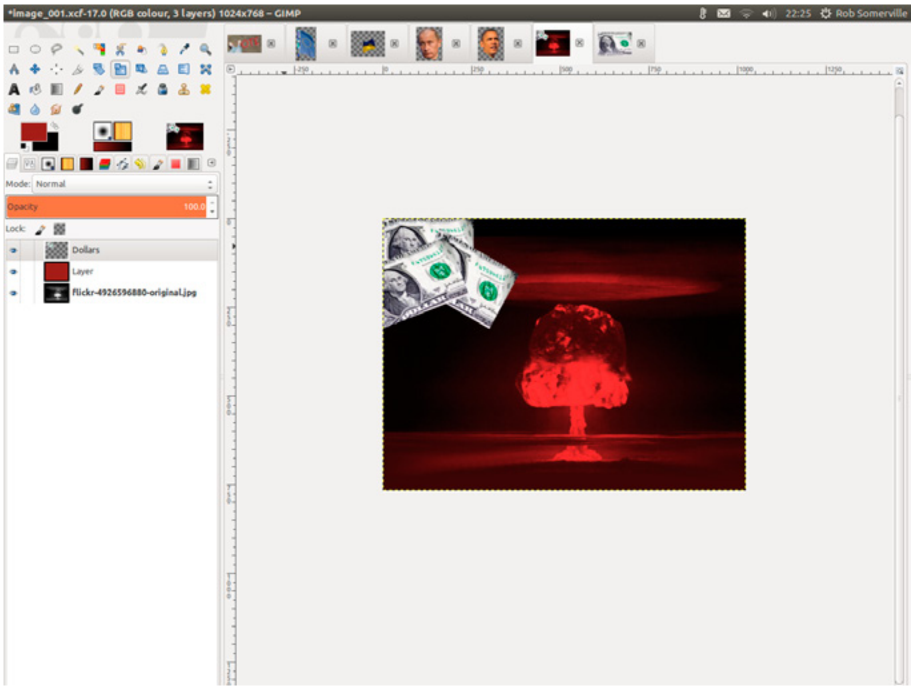


*image_001.xcf-17.0 (RGB colour, 3 layers) 1024x768 – GIMP
22:25 Rob Somerville
Mode: Normal
Opacity 100.0
Lock:
Dollars
Layer
flickr-4926596880-original.jpg

Step 11

Open obama.xcf and select all, copy. Create a new layer and paste the picture of the US president into the middle. Using the scale tool and pressing the Ctrl key, constrain the scale until you are happy with the dimensions. Locate the president at the bottom right hand side on the image. Use the smudge tool to remove any ragged blue edges. Rename the layer Obama [Figure 12].

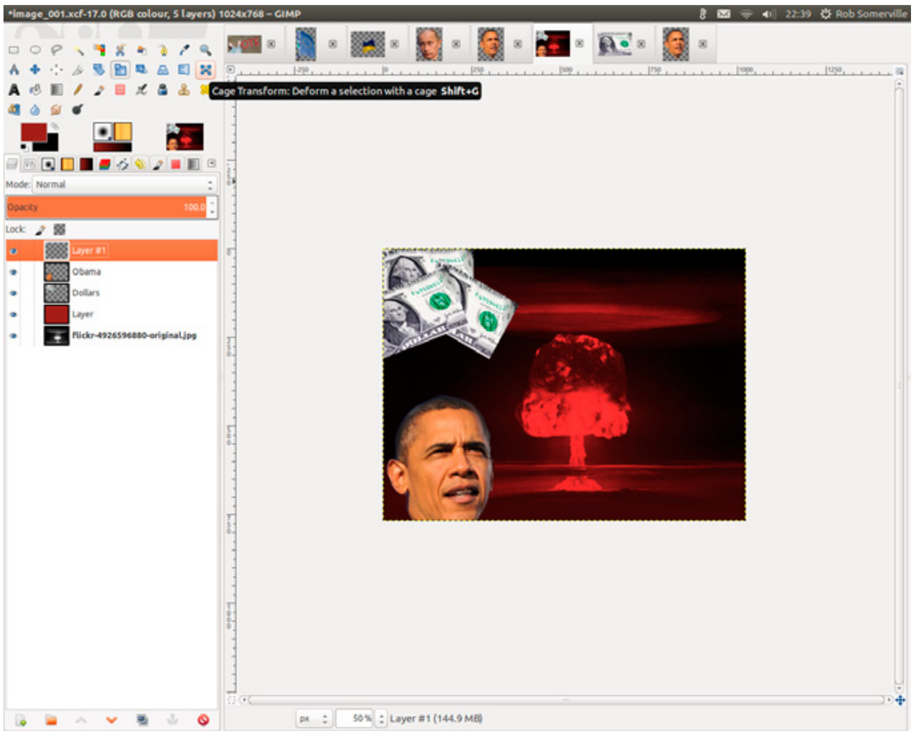
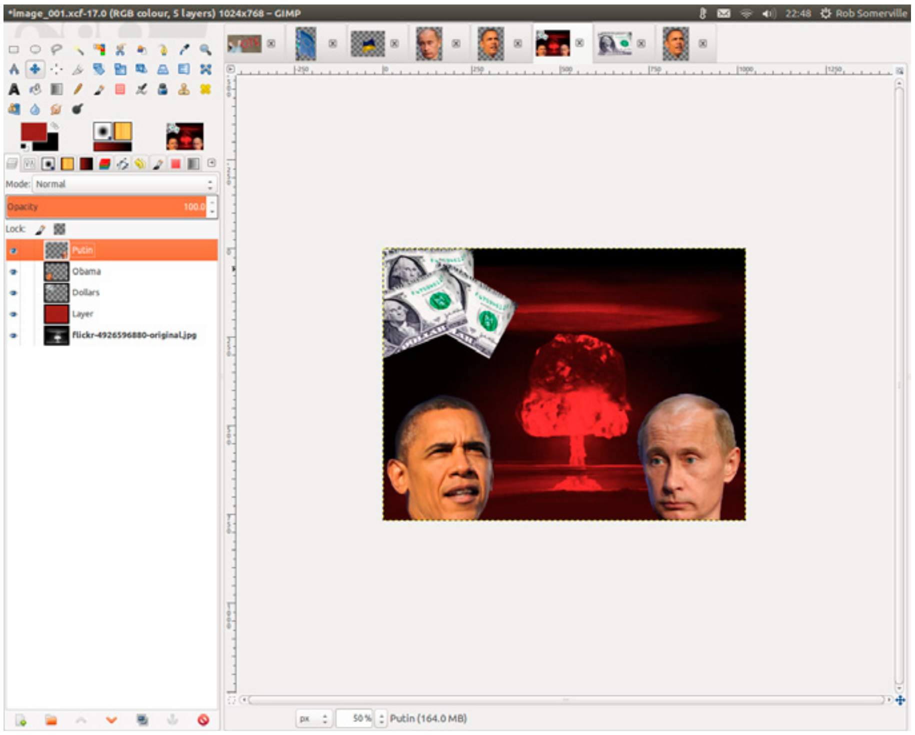


*image_001.xcf-17.0 (RGB colour, 5 layers) 1024x768 – GIMP
22:39
Rob Somerville
Cage Transform: Deform a selection with a cage Shift+G
Mode: Normal
Opacity 100.0
Lock:
Layer #1
Obama
Dollars
Layer
flickr-4926596880-original.jpg
px
50 %
Layer #1 (144.9 MB)

Step 12

Repeat step 11 with the picture of the Russian leader and place at the bottom right hand side of the image. Rotate the layer so that the eyes are horizontal [Figure 13].



Step 13

Repeat with shoes.xcf but instead of constraining the scale tool, increase the height. Smudge and erase the legs to get a fade effect. Change the layer mode to hard light and delete any speckles that show through [Figure 14].

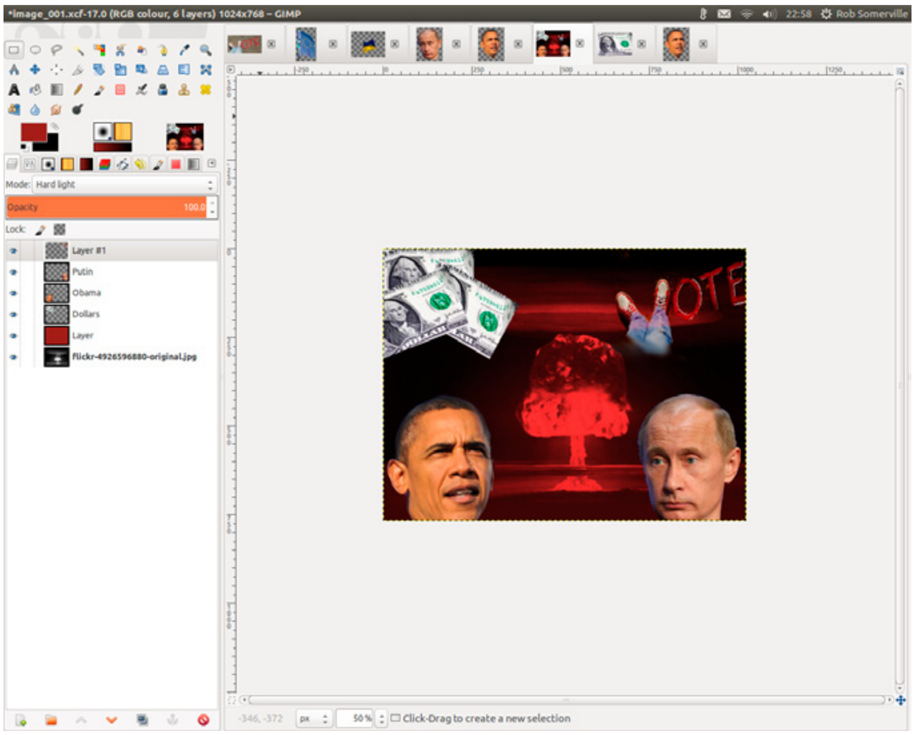


*image_001.xcf-17.0 (RGB colour, 6 layers) 1024x768 – GIMP

Mode: Hard light
Opacity 100.0

Layer #1
Putin
Obama
Dollars
Layer
flickr-4926596880-original.jpg

-346, -372 px 50 % Click-Drag to create a new selection

Step 14

Open flag.xcf, and copy and paste into a new layer on our pastiche. Duplicate the layer, flip it vertically, and line up the flagpole so it lines up perfectly over the other layer. Add a new transparent layer, open the Ukrainian flag and paste and scale over the right hand side of the flagpole. Use the smudge tool to align the edge of the flag against the curved part of the flagpole. Merge down the three layers, move the bottom of the pastiche, and layer to image size [Figure 15].

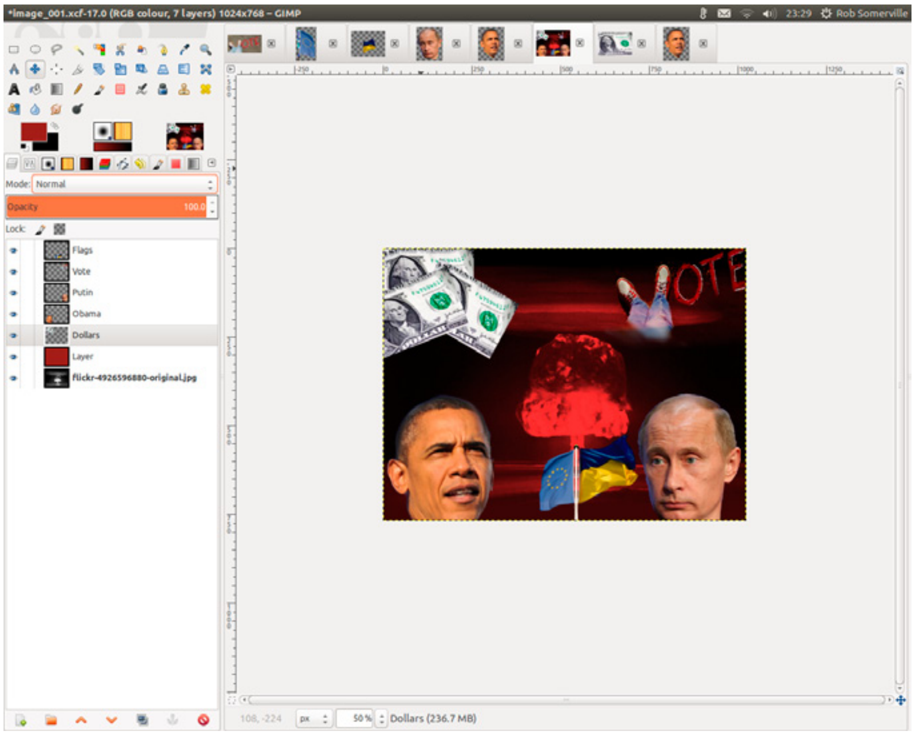


*image_001.xcf-17.0 (RGB colour, 7 layers) 1024x768 – GIMP

Mode: Normal
Opacity 100.0

Flags
Vote
Putin
Obama
Dollars
Layer
flickr-4926596880-original.jpg

108, -224 px 50 % Dollars (236.7 MB)

Step 15

Move the flag layer just above the red layer. Desaturate the Obama, Putin, Vote and Dollar layers in turn. Create a new transparent layer at the top of the layers stack, and fill with c9c26e. Change the layer mode to multiply [Figure 16].

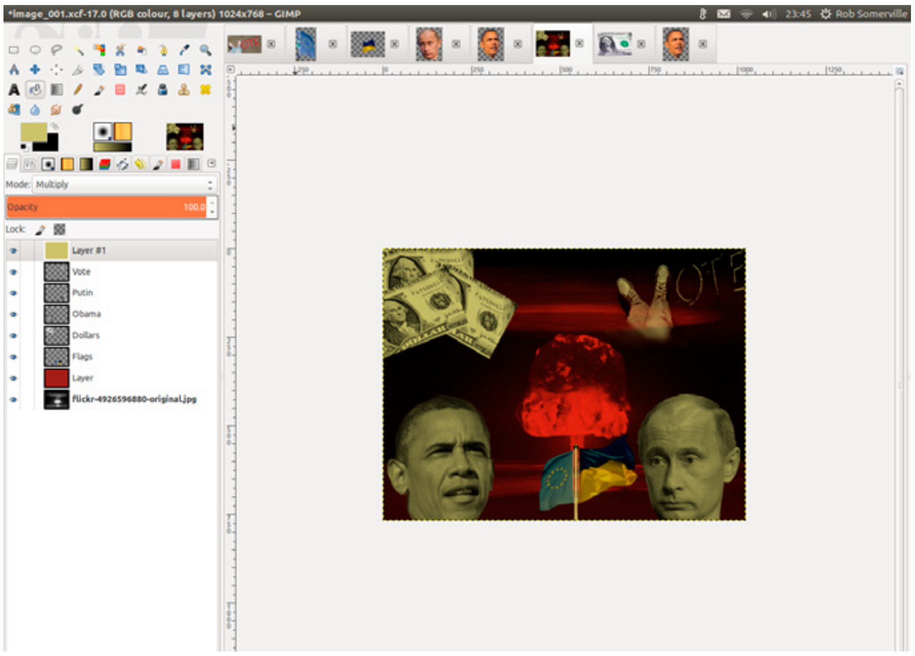


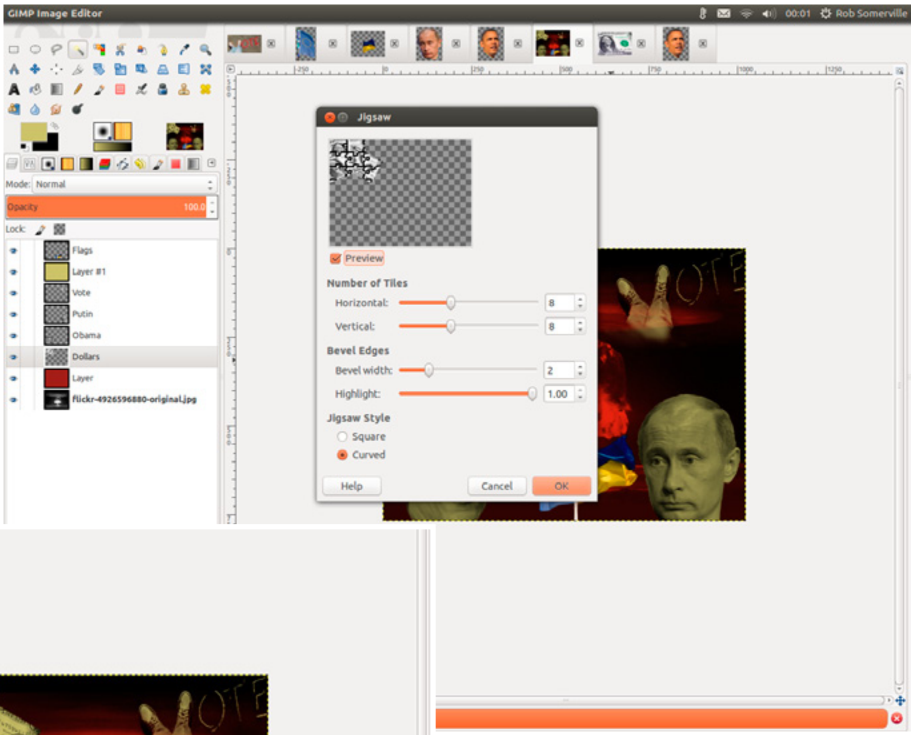
*image_001.xcf-17.0 (RGB colour, 8 layers) 1024x768 – GIMP

Mode: Multiply
Opacity 100.0

Layer #1
Vote
Putin
Obama
Dollars
Flags
Layer
flickr-4926596880-original.jpg

Step 16

With the exception of the vote layer, apply filter → render → pattern → jigsaw to Putin, Obama, and Dollars [Figure 17 – 18].



GIMP Image Editor

Mode: Normal
Opacity 100.0

Flags
Layer #1
Vote
Putin
Obama
Dollars
Layer
flickr-4926596880-original.jpg

Jigsaw

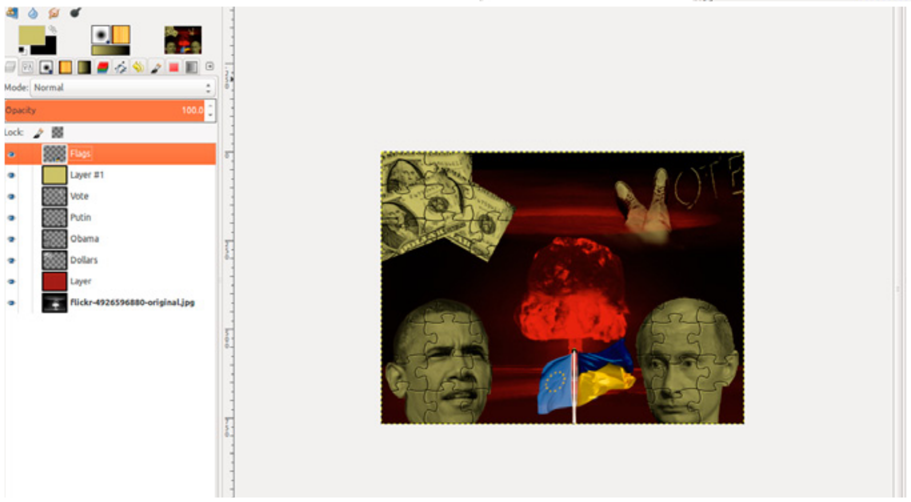
Preview

Number of Tiles
Horizontal: 8
Vertical: 8

Bevel Edges
Bevel width: 2
Highlight: 1.00

Jigsaw Style
Square
[✓ Curved]

Help Cancel OK



Mode: Normal
Opacity 100.0

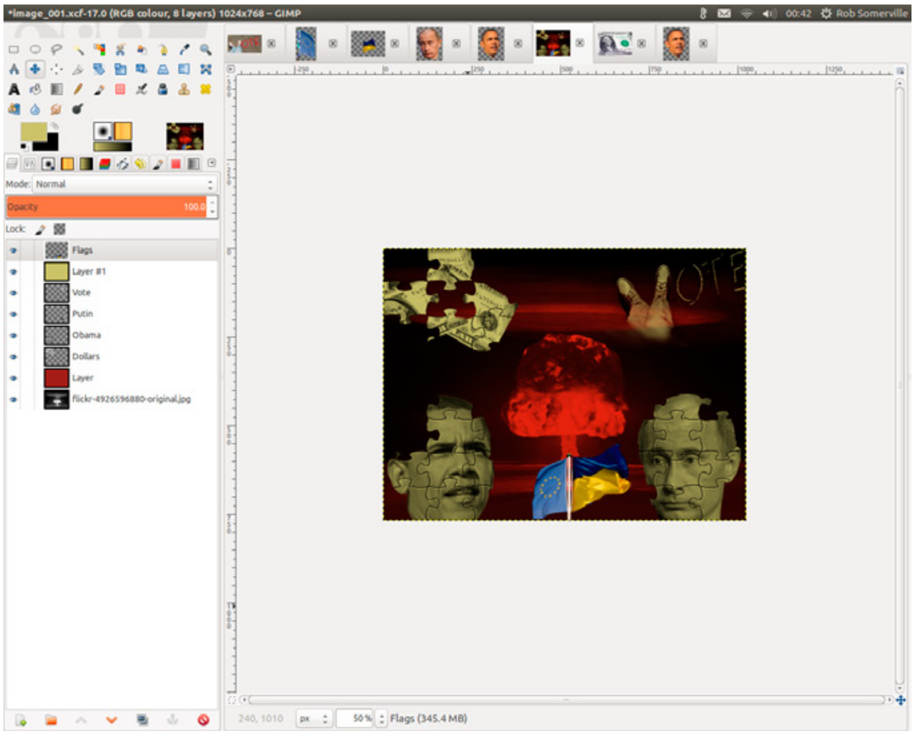
Flags
Layer #1
Vote
Putin
Obama
Dollars
Layer
flickr-4926596880-original.jpg

Step 17

Using the select tool or erase tool on the layers in step 11, delete parts of the jigsaw pieces. Tidy up the rough edges of the jigsaw with the blur tool.

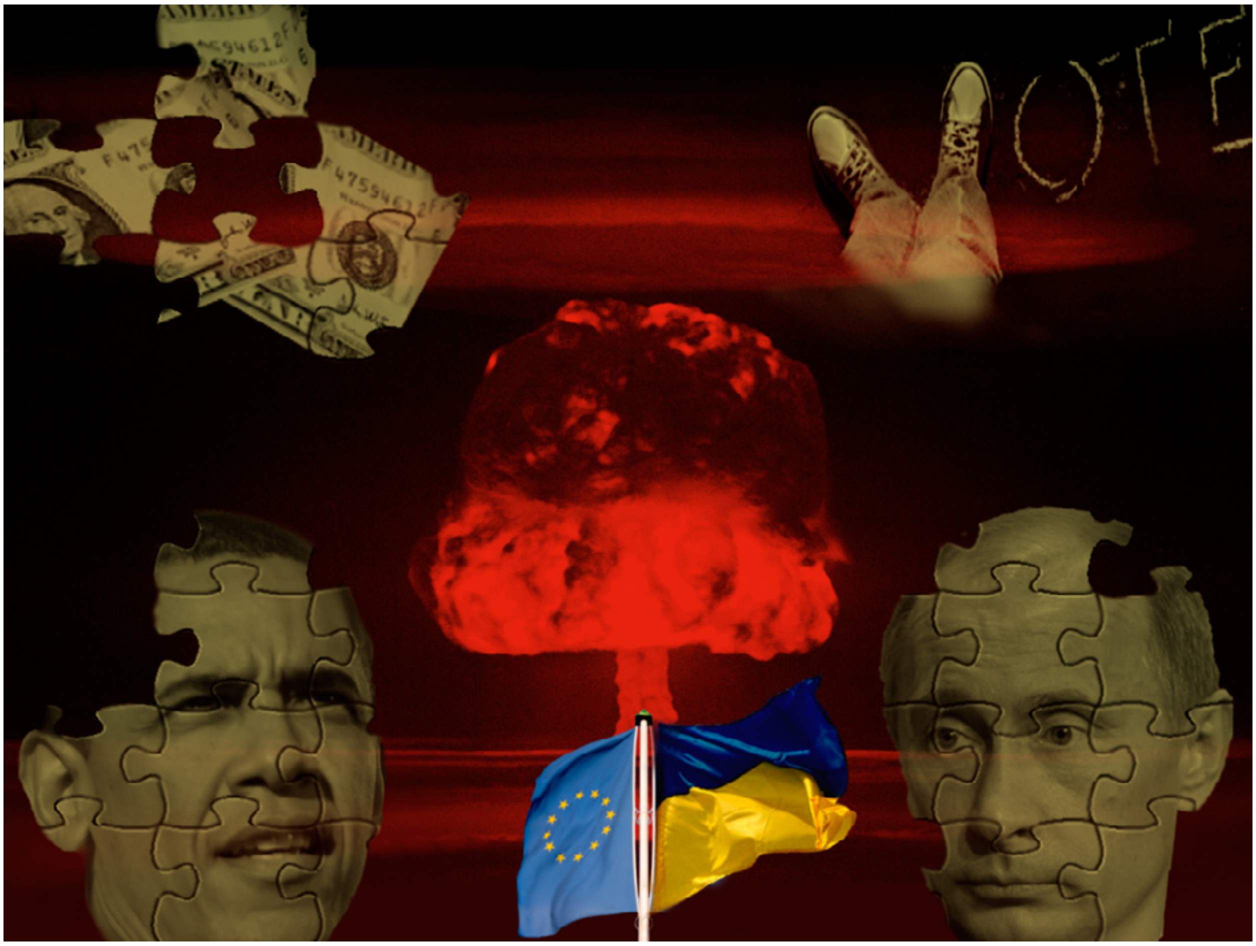
Step 18

Select the last layer and increase the scale width and position so that the mushroom cloud is in the centre. Realign the flag layer as required [Figure 19].



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.



Credit Card Fraud is a Thing of The Past

I shop at Target – alongside a hundred million other people, it would seem.
The difference is, that I've never used a credit card there, which made me sympathise with all of the people who had their card details stolen.

What you will learn…

- Credit card fraud issue

What you should know…

- Basics of security

This made me think about how easy it would be, to make sure that, if anyone stole all the credit card details from a retailer, those details would be totally useless.

First, I thought about the dangers of modern shopping.

When you fill in your card details at a Point-of-Sale terminal, access your bank's web page, or visit your ATM, there are many evil devices waiting to rob you.

A hidden camera records the PIN code that you type, embedded malware records every keystroke, while another device reads the magnetic stripe on your card.

Meanwhile, a network snooper on the line will pick up your PIN, your user ID and credit card number so that some sociopath can rob you of everything in the future.

If any of these devices misses anything, an internal security breach exports the entire customer details database, which will be sold to criminals around the world.

The whole issue comes down to one thing. How do you prove that you are really you?

Being a rational, hard-working person, you don't have a problem doing so, but the things your bank is asking you to do, to achieve this, are becoming a bit user-hostile.

Since you don't have a criminal record, and have no desire to have some hacker create one for you, you're not particularly keen to supply fingerprints to your bank.

You don't work for the CIA, so you also would rather not have your retina scanned, nor to have your DNA tested, nor to supply a voice-print. Anyway, biometric data is transmitted digitised, so the hackers can intercept it, copy it, and use it again.

Besides, imagine the length of the queues at Target, if everyone had to wait to be scanned.

Your frightfully modern and forward-thinking bank is making noises about introducing 'multi-factor authentication' – whatever that might be, which they say will fix the security issues. The 'multi' bit sounds like you're going to have to write more than one password on the back of your ATM card, since you'll never remember more than one.

Your friend's bank uses EMV smart cards, and you want to ask her how that works, and then you remember that she lost her card last month, and really can't remember buying all of the stuff that's appearing on her bank statements. Perhaps smart cards aren't that smart, especially since you tend to lose more things than your friend.

Okay, let's look at a possible alternative solution.

Instead of communicating numbers to the bank or credit card company, all of which can be intercepted, decrypted and used by the criminals, what would be really good, would be some kind of telepathic password, which only existed in your head.

Since it could be difficult to transit this password telepathically, how about a compromise, where you transmit to your credit card company, or bank, information about your telepathic password, which only your bank understands?

Yes, but the camera, and the malware, would record what you typed, and use it to get into your account.

Okay, then, how about, if what you typed only worked once. Then, using the same keystrokes a second time would be useless. That would work, but how does the bank know that, what you typed the second time, represented the same telepathic password? Also, you certainly wouldn't want to contact your bank every day, to get a new method of transmitting your telepathic password, which we'd better call a 'keyword', instead.

How about this, then? Each time you want to access your account, a popup shows you an alphabet, with a number under each letter, and you type the numbers, instead of the letters?

Okay, that's obviously bad because the camera would pick up the numbers but, what if the numbers were all scrambled? That's better, but the camera would still get you, and the malware would still send them back to some sociopath who, after a few months, would be able to guess your password, from the patterns of the numbers.

What about, if there were only two numbers and, what if there were two alphabets, in upper and lower case? Then your keyword would be represented by a selection from 52 letters, each letter identified by one of two random digits.

If the pattern of the digits changed randomly, with each access, then your keyword of "gobbledeygook" could be "1111010101110" the first time you accessed your account but, the second time, it could be "1110010001101".

Instead of calling this a 'series of digits', let's invent a name, and call it a 'SteelCode', which has a nice impregnable sound to it.

Now we're getting somewhere. Perhaps we should call it a 'matrix', and it could look something like shown on Figure 1. The camera sees you entering a pattern of 1's and 0's, each of which could correspond to any one of 20 or 30 letters. The network snooper sees the numbers, but not the letters, and the malware sees both, but doesn't know what they mean.

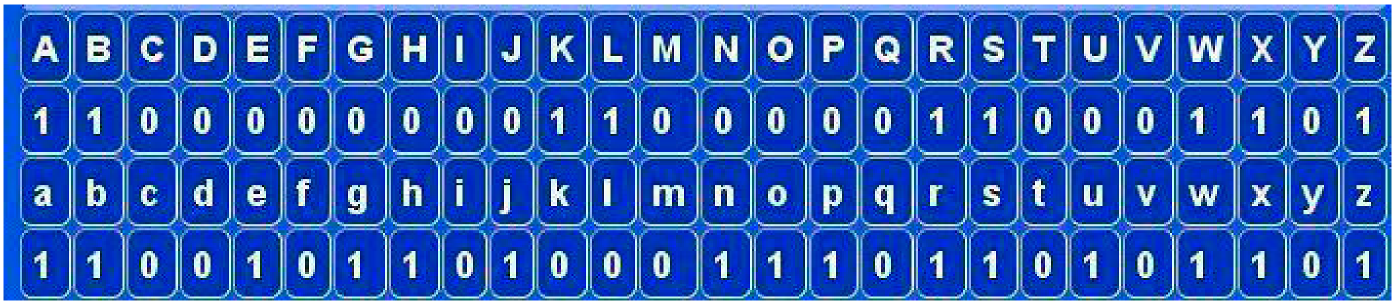
Luckily, you took maths in college, and spend a lot of time in the casino, so you know how to calculate odds,

a d v e r t 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 ...



A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
1	1	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	1	1	0	0	0	1	1	0	1
a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
1	1	0	0	1	0	1	1	0	1	0	0	0	1	1	1	0	1	1	0	1	0	1	1	0	1

Figure 1. *Matrix*

and you can see they're now in your favour, but you still want them to be better, because you have access to the company's payroll account, and would like to make sure that only the employees get the cash.

What if you had two keywords, and added the numbers from the first to the numbers of the second? That would make it even tougher for the hackers to work out your keywords.

Something like "gobbledeygook" and "GOBBLEDEYGOOK", would mean adding "1111010101110" to "0011100000001" which would mean that you would type "1122110101111".

Even at 2 o-clock in the morning, you could manage to do this arithmetic without a calculator, assuming that you had consumed fewer than ten beers, or so.

This is now getting really interesting, as you consider more possibilities.

You could further confuse the hackers by, for instance, adding or subtracting '1' from every digit, or from every other digit, adding an incrementing number to every digit, and a host of similar tricks. The only limiting factor would be how well you think your brain normally works at two o'clock in the morning.

Now, there is only one piece of information left, that the hacker knows about your keyword. He knows its length.

Having come this far, you're not going to be beaten by a trivial setback like that. When you define your password, it's a simple matter to also define, that a certain number of leading and a (perhaps different) number of trailing digits are dummies.

Now, when you enter your digits, you add, maybe, two random digits in front, and, maybe, three random digits at the end. Now the hacker will be trying to crack the wrong length password.

Let's say that the credit card company installs an authentication server, which uses the methods we've discussed. So how does that help the retailer?

Picture the scene, where you're standing at the point-of sale terminal at Target, about to pay for the present you bought for your wedding anniversary. What happens next?

They hand you the little terminal device, and you swipe your card. This is a new kind of card, where the magnetic stripe only contains your user ID on the authentication server, and a code, describing the credit card company.

The retailer's POS server sends your user ID to the credit card company, identified by the code on your card. The user ID is received by the authentication server, which sends out a matrix. You fill in the SteelCode, which is transmitted back to the credit card company.

If the authentication server finds that the SteelCode is correct, it passes this information to the credit card company's usual server, which makes sure that your card hasn't expired, and that you have enough credit. Only then, does it send approval to the retailer.

So, what did all of the hacker's devices pick up?

They know your user ID, credit card company and what was in the matrix. They also know the SteelCode that you typed – and all of this is the only information left in the retailer's transaction logs.

The SteelCode is useless, since it was only valid in conjunction with the matrix of the time. The next purchase you make will have a different matrix, and a different SteelCode.

I guess the hackers could wait patiently, and collect matrix and SteelCode data over a period of a few months, in the hope that you'd be buying stuff every day. When they had a few hundred samples, they could narrow your keyword down to a few hundred possibilities. By the time they got it right, one of two things would have happened.

First, they'd be drawing a retirement pension and, second, you'd have changed your keyword.

Try the solution at *www.designsim.com.au.*

MARK SITKOWSKI
Design Simulation Systems Ltd
http://www.designsim.com.au

Consultant to Forticom Security
http://www.forticom.com.au

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

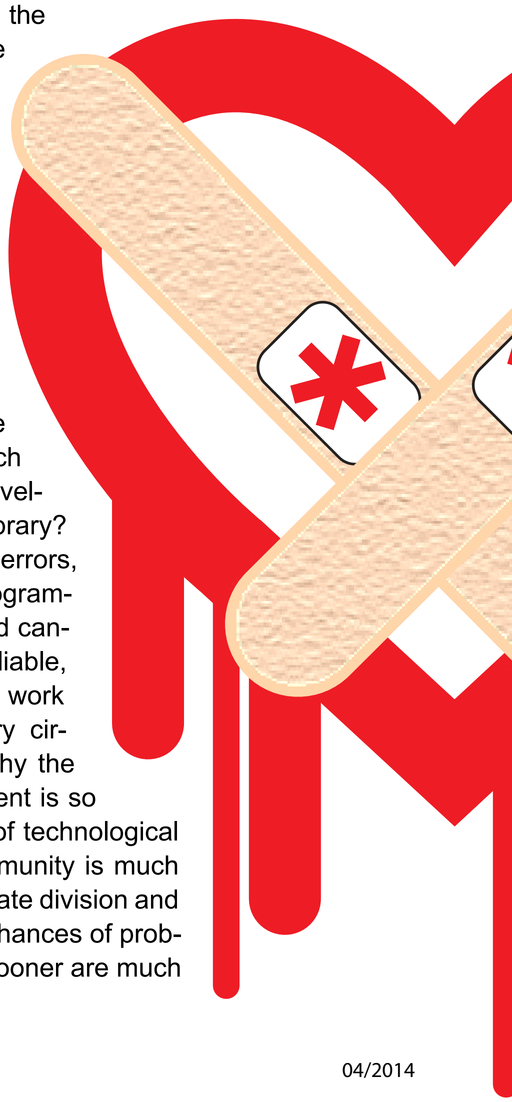
Visit us at www.taghos.com and start saving bandwidth today!

With the Recent Announcement of the Widespread Heartbleed SSL Vulnerability is it Time to Reconsider who the Troublemakers Really are?

Unless you have been away from the newspapers, television or the internet for some time, the widespread media coverage (and to some degree panic) of this particular coding error in the OpenSSL library will not have escaped your attention. Due to a fairly trivial coding oversight, an attacker can compromise any server running SSL and extract X509 keys, cookie data, usernames, passwords or even potentially documents. This affects OpenSSL from versions 1.0.1 through to 1.0.1f – a vulnerability window of 2 years in the wild.

I'll be the first to admit then that my optimism in last month's column about security matters improving was probably premature taking into account this latest announcement. I think in the future I will return to the cynical position that the only secure computer is one encased in concrete and dumped in the bottom of the Western Pacific. What is more disturbing is the announcement by Bloomberg News [1] that the NSA was aware of this exploit for two years and did not alert the software security community or OpenSSL. The NSA naturally denied this, but to paraphrase the immortal words of Mandy Rice-Davies who was embroiled in the UK Porfumo spy scandal in the 1960's – "They would, wouldn't they?"

So let us look at the the anatomy of a bug, particularly insidious ones like Heartbleed. To start with, the more complex a piece of software, the greater the chances there will be errors in it. If more than one programmer is working on a project, the greater the possibility that errors will accumulate. I say "Aluminium" where as State side, it is "Aluminum". One coder will use a for .. each loop, another will use a do … while loop. Differences in writing style, design and logic will always plague large projects, even with strict controls in place, as people have a natural or cultural approach to coding these syntactical and logical errors will creep in. And that is before we even get down to the nitty-gritty of genuine bugs, where code has been tested thoroughly and found to be fit for purpose. The infamous Therac-25 radiation therapy bug that killed patients undergoing radiotherapy would only arise when the operator pressed an obscure sequence of keys. The other lesson that was not learned is that previous versions of the software were reused and modified. Unfortunately, the earlier version was dependent on a hardware interlock preventing the fatal scenario. In later versions, the software acted as the interlock, and as the previous fail-safe masked the true nature of the bug mistakes were made.



And so it is today with the programmers' dependence on libraries. Best practice always says "Don't reinvent the wheel" yet how much confidence can a developer have on any library? Human beings make errors, and even the best programming team in the world cannot produce 100% reliable, bug free code that will work flawlessly under every circumstance. That is why the Open Source movement is so critical in these days of technological complexity – the community is much larger than any corporate division and with peer review the chances of problems coming to light sooner are much greater. From a security and code review perspective, Openness is good, Opacity is bad. That is why, if true, the failure of the NSA to notify the developers concerning Heartbleed is so morally and ethically repugnant. While I fully appreciate the difficulty that is the ethical minefield of National Security, to me – naively I know – behaving in this way seems a total betrayal of why the agency is there in the first place. A cynical colleague once suggested that a lot of viruses and malware were designed by the Anti-virus software manufacturers themselves to keep their businesses viable. He may have been closer to the truth than he thought.

So who are the real troublemakers? If a tree falls in a forest and there is no-one there to observe it does it make a sound? Without an attack or extensive research by the security community, a lot of these errors will remain hidden. All software has bugs and vulnerabilities. Period. From the O/S right through to the user interface, the thread of error runs. While some are more diligent than others in writing quality systems, we will not reach perfection in this lifetime no matter how hard we try. The developer using a third party library trusts the third party. The end user trusts the IT department to source the best software for the purpose at hand. The customer trusts the company or organisation. Like banking, the IT industry is built upon layers of trust, and where this is betrayed, confidence is reduced. Unlike banking, the industry is still young and developing, so there is hope that we will grow and mature accordingly. However, our hands are tied by the disconnect between the technologists, black hat hackers and the establishment. There are stringent laws in place with punitive sentences for those that breach the computer misuse act. If I were to probe a third party SSL server without the owners' permission to definitively prove whether or not it was vulnerable to Heartbleed I could potentially leave myself open to prosecution in the UK, and no doubt the USA as well. That seems fair until you realise that > 66% of the servers out there are probably vulnerable to this bug. Will all these sys-admins patch their code? What about organisations that are undergoing fiscal cuts and do not have the technical resources? To deal with Heartbleed will take a lot of resources and ideally the internet as a community should be the one to deal with it by identifying this silent and pernicious weakness and lending a helping hand. No wonder some people on forums are saying "Stay off the Internet". If all you have is someone's word, without empirical evidence, it is a lot to ask when your credit card or personal details are at stake.

The black hats on the other hand will be having a field day. The call has already gone out to build an army of Heartbleed honey traps, but more troubling is the potential data loss that may have quietly occurred unseen in the previous two years. There are known knowns. There are known unknowns. There are unknown unknowns. If I was a Black hat, and discovered such a vulnerability, I would make sure I kept very quiet indeed and make hay while the sun shines. Now that the attack is out in the open, all bets are off. It will be a straight race between the Black and White hats with those that are not technologically aware no doubt accidentally tripping both sides up from time to time.

As an industry, we need to grasp the nettle of ethical disclosure, and help each other out but unfortunately this will be a tough call. Walking home one night, I saw a car parked outside my house with the driver's door unlocked and the keys in the ignition. I didn't recognise the car, and being an area renowned for joy-riding and car theft, I checked the car to see if I could find any clue to the ownership details. Finding none, I locked the car and phoned the police to say I had the keys, and would they like me to drop them off at the police station? Thanking me for my community spirit, they got in touch with the owner. Unfortunately, it was my next door neighbour, who was fixing the car for a friend and was duly reprimanded by the owner for his carelessness. I didn't even get a thank you – just a very gruff "Keys" and an angry glare for my efforts when he knocked on my door late that night. As the old saying goes, no good deed ever goes unpunished.

References

- *http://www.bloomberg.com/news/2014-04-11/nsa-said-to-have-used-heartbleed-bug-exposing-consumers.html*
- *http://www.theguardian.com/technology/2014/apr/12/us-government-nsa-denies-aware-heartbleed-internet-bug*

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.

AsiaBSDCon2014 Report

AsiaBSDCon (http://www.asiabsdcon.org/) is a conference for users and developers on BSD based systems. During 13-16 March 2014, the 9th AsiaBSDCon was held at the Tokyo University of Science in Kagurazaka, Tokyo, Japan.

Kagurazaka is the old-established town near the Edo castle, and 6 minutes by train from Akihabara cyber electronic town. During the Edo period, many significant rendezvous are held in Ryoutei – Japanese traditional restaurant. It is a good place to get some thinking done. Now you walk up the long sloping street and find the conference room in narrow back streets.

This year, over 140 attendees (33% from overseas) joined the conference.

Keynote Speech

- Bambi Meets Godzilla: They Elope – Open Source Meets the Commercial World by Eric Allman
- An Overview of Security in the FreeBSD Kernel by Dr. Marshall Kirk McKusick

FreeBSD

- The Future of LLVM in the FreeBSD Toolchain by David Chisnall
- Deploying FreeBSD systems with Foreman and mfsBSD by Martin Matuška
- Modifying the FreeBSD kernel Netflix streaming servers by Scott Long
- Transparent Superpages for FreeBSD on ARM by Zbigniew Bodek
- How FreeBSD Boots: a soft-core MIPS perspective by Brooks Davis

ZFS

- OpenZFS ensures the continued excellence of ZFS on FreeBSD, Linux, and Illumos by Matthew Ahrens
- Snapshots, Replication, and Boot-Environments – How new ZFS utilities are changing FreeBSD & PC-BSD by Kris Moore
- ZFS for the Masses: Management Tools Provided by the PC-BSD and FreeNAS Projects by Dru Lavigne

bhyve

- Visualizing Unix: Graphing bhyve, ZFS and PF with Graphite by Michael Dexter
- Nested Paging in Bhyve by Neel Natu and Peter Grehan

FreeNAS

- Introduction to FreeNAS Development by John Hixson

NetBSD

- NPF – progress and perspective by Mindaugas Rasiukevicius
- Developing CPE Routers based on NetBSD: Fifteen Years of SEIL by Masanobu Saitoh and Hiroki Suenaga
- Carve your NetBSD by Pierre Pronchery and Guillaume Lasmayous

OpenBSD

- VXLAN(4) and Cloud-based networking with OpenBSD by Reyk Floeter

OSX

- Adapting OSX to the Enterprise by Jos Jansen

Networking and development

- Netmap as a core networking technology by Luigi Rizzo
- OpenBGPD turns 10 years – Design, Implementation, Lessons learned by Henning Brauer
- Implementation and Modification for CPE Routers: Filter Rule Optimization, IPsec Interface and Ethernet Switch by Masanobu Saitoh and Hiroki Suenaga
- Bold, fast optimizing linker for BSD by Luba Tang

BSD Associate Exams:

- Analysis of BSD Associate Exam Results by James P. Brown

BSDA Certification Requirements have been translated to Japanese and the 1st Japanese version of the BSDA certification exam was held at this conference (*http://www.bsdcertification.org/news/bsda-certification-requirements-now-available-in-japanese*).

Tutorial

- Networking from the Bottom Up (Packet Processing Frameworks) by George Neville-Neil
- IPv6 Basics by Massimiliano Stucchi and Philip Paeps
- Kerberos service operation by Hiroki Sato
- An Introduction to the FreeBSD Open-Source Operating System by Dr. Marshall Kirk McKusick
- Testing on FreeBSD by Julio Merino

Work-In-Progress session

- FreeBSD Journal
- Porting DTrace to NetBSD/arm
- Improving MII PHY
- Building PKG binaries for #FreeBSD/#RaspberryPi
- *https://wiki.freebsd.org/Teams/clusteradm/generic-mirror-layout*
- Simple Shell Script to build a big company's system
- Writing the ZFS Chapter of the FreeBSD Handbook
- FreeBSD Benkyokai
- FreeBSD Test Suite
- OSv on Bhyve

Developer Summit:

- FreeBSD Developer Summit
- NetBSD Developer Summit
- *BSD Vendor Summit
- NetBSD BOF (*https://www.facebook.com/events/727500413946864/*)

BhyveConf: (http://peatix.com/event/30480)
Bhyve masters arrived for AsiaBSDCon weekend on Wednesday. Before AsiaBSDCon Day 1, BhyveConf was held in SAKURA Internet Research Center in Shinjyuku,Tokyo.

- Introduction to Bhyve
- Bhyve Future developments
- Bhyve provisioning and monitoring
- How ScaleEngine Replaced its last CentOS box
- OSv on Bhyve / ruby-virtualmachine
- Introduction to Qubes OS
- mocloudos

AsiaBSDCon is organized by Hiroki Sato and BSD Research (*http://www.bsdresearch.org/*). There are three points to drive home regarding this conference.

- Speaker should write the paper, not just give the technical lecture.
- Keep a creative atmosphere between Speaker and developer.
- Need more sponsorship and supporters.

AsiaBSDCon2015 will be back in March 2015, Kagurazaka, Tokyo. If you have a plan to spend the Easter holidays in the East, join the conference and have a great time!

Please contact *secretary@asiabsdcon.org* if you need support and/or have a specific question about AsiaBSDCon.

JUN EBIHARA

Dr.Web 9.0 for Windows — the rapid response anti-virus

1. Reliable protection against the threats of tomorrow
2. Reliable protection against data loss
3. Secure communication, data transfer and Internet search

© Doctor Web 2003 — 2013

www.drweb.com

Free 30-day trial: https://download.drweb.com

New features in Dr.Web 9.0 for Windows: http://products.drweb.com/9

FREE bonus — Dr.Web Mobile Security:
https://download.drweb.com/android

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

$49.95 Save a BUNDLE!
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

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