news@UK

The Newsletter of UKUUG, the UK's Unix and Open Systems Users Group
Published electronically at `http://www.ukuug.org/newsletter/`

Volume 14, Number 3 ISSN 0965-9412 September 2005

Contents

News from the Secretariat

Jane Morrison

The Linux 2005 event, held in Swansea between the 4th and 7th of August, was well attended and a great success. Reports of the conference are available on the PingWales site:
`http://www.pingwales.co.uk/software/linux-2005-day1.html`

For those members who were unable to attend please find enclosed a copy of the conference CD. This is a benefit of membership which entitles you to receive all CDs produced by UKUUG.

Planning has started now started for Linux 2006!

The AGM this year will be held on 22nd September at University College London. Details including Agenda etc. have recently been sent to all members. All details are also on our web site. The AGM at 6.15 p.m. will be followed by a technical talk.

Looking further ahead to next year you should find enclosed the Call For Papers for (what was previously known as the Winter conference) Spring 2006. This will be held on 22nd and 23rd March 2006 at Durham University, a venue which provides good facilities and has excellent rail, road and bus links. Please put the event dates in your diary now.

The next Newsletter will be the December issue and the copy date is: 22nd November.

LinuxWorld Exhibition and Conference

The LinuxWorld 2005 Exhibition takes place at Olympia2 on October 5th and 6th 2005.

UKUUG will be represented at the Exhibition in the ".Org Village".

Alongside the Exhibition there will be a conference. Speakers include Klaus Knopper, Alan Cox, Michael Tiemann, Paul Everitt, Mark Shuttleworth, Andrew Eddie and Mike Banahan.

Full details of the events are available at
`http://linuxworldexpo.co.uk/`

AUUG 2005

We have received notification from David Purdue of the AUUG 2005 conference.

This will be held at the Carlton Crest Hotel in Sydney between Wed 19th and Fri 21st October.

The conference brochure is available at
`http://www.auug.org.au/events/2005/auug2005/AUUG-2005-Conf-Brochure.pdf`

BCS Open Source Specialist Group

We have received notice from Gary Lloyd of forthcoming meetings of the BCS Open Source Specialist Group.

The next four meetings are as follows:

27th October 2005: Southampton - Open Source Software Quality in Practice

The speakers will be Mark Taylor of the Open Source Consortium and Dr Graham Oakes.

10th November 2005: Oxford - Open Source Technology Stack

Alan Lenton will speak on the LAMP technology stack.

12th December 2005: London - An Introduction to Open Source for Developers and Managers

The speakers will be Barry Cornelius of OSS Watch and Mark Taylor.

10th January 2006: London - An overview of Open Source Licensing

The speaker will be Andrew Katz, a solicitor with specialist legal firm Moorcrofts LLP.

Full details of these events are available at
`http://ossg.bcs.org/`

Free Culture UK launch announcement

We have received the following announcement of the launch of Free Culture UK in July this year.

2005-07-23: Free Culture UK Launched

Free Culture UK launches at OpenTech today with the aim of building a grassroots movement to promote an open, participatory culture. We will lobby politicians about threats to free culture; create new and promote existing spaces in which free culture can thrive; and work to engage the public in the future of our cultural domain.

Our first campaigns concern returning the length of copyright protection to a more balanced level; the loss of the public domain, which we hope to reverse with an open database of public domain works; and the promotion of Creative Commons, a positive alternative to ‘all rights reserved’ copyright. Working both nationally and locally, offline and on the Internet we will build a broad movement to reassert the freedoms and rights of the public, and halt the advance of monopolies.

Free Culture is launched by a group of people including Rufus Pollock (Open Knowledge Foundation and Friends of the Creative Domain), Tom Chance (Remix Reading), Ed Griffiths-Jones (Remix Brighton), and David Berry (Music Commons). It is part of the Open Knowledge Foundation Network and affiliated with Remix Reading, Remix Brighton, Friends of the Creative Domain and LiquidCulture. We are committed to an open an inclusive approach and welcome the collaboration or participation of similar organisations.

More information on Free Culture UK can be found on the web at
`http://www.freeculture.org.uk`

Eduforge

We have received the following announcement from Richard Wyles of Eduforge.

Eduforge has recently been upgraded with the help of the eXe Editor Project.
`http://www.eduforge.org/`
`http://www.exelearning.org/`

Eduforge is an open access environment designed for the sharing of ideas, research outcomes, open content and open source software for education. You are welcome to use our community resources or start your own project space. Registration is free. Eduforge offers a wide range of collaborative tools as well as project spaces for the development of educational software, content or to facilitate collaborative research and discussion.

Eduforge was founded in early 2004 as part of the New Zealand Open Source Virtual Learning Environment (NZOSVLE) project
`https://eduforge.org/projects/nzvle/`

The NZOSVLE is a major collaborative education project funded by the New Zealand Tertiary Education Commission, an arm of the NZ Government. The project's goals are to adopt and contribute to open source solutions for education. The consortium is composed of polytechnics, universities, and a private training organisations.

Part of the vision for Eduforge is to create an environment that is robust enough to support large scale collaborative software development, but friendly enough to be used by non-programmers who want to collaborate on a range of projects. Projects may include learning materials design, application testing, and research as well as contributing documentation, tutorials, and help files to software development projects.

In addition to reducing the economic and organisational barriers associated with large scale distributed collaboration, Eduforge is envisioned to reduce the technology usability barrier often confronted by non-ICT professionals. We've recently added Planet Eduforge, a blog aggregator, and integrated a fully featured blogging tool.
`http://planet.eduforge.org/`

Eduforge is developed using FOSS - GForge, Serendipity, and PHPWiki. While this started with somewhat of a NZ flavour due to the projects on it, the intent has always been a global shared resource. Recently we shifted the hosting to the US to improve latency for many of the international users. We would very much welcome participation – either in using the resources or in helping us develop the resources available on Eduforge. Or simply feedback, we're committed to continuous improvements.

BSD Certification Roadmap

We have received an announcement of the BSD Certification Group's release of their Certification Program Roadmap. Full details are available from:
`http://www.bsdcertification.org/`

SELinux conference: call for papers

We have received a call for papers and details of the second Security Enhanced Linux Symposium:
`http://www.selinux-symposium.org/`

The event takes place between February 28th and March 2nd, 2006, at the Wyndham Hotel, Baltimore, Maryland, USA.

Full details are available at the URL shown above.

Second Open Source World Conference: Extremadura

We have received notification of this conference organised by the Regional Government of Extremadura which takes place at MEEErida on the 25th and 26th October 2005.

Full details are available on the conference web site:
`http://www.opensourceworldconference.com/merida05/en/modules/news/`

Information Accessibility Initiative

We have received the following call for participation from the Information Accessibility Initiative (IAI).

The Information Accessibility Initiative (IAI) was launched in November 2004 to promote the use of open formats and information standards. It is actively seeking individuals and organisations as members and partners to assist in driving this essential work forwards.

Open formats are ones which are truly free, allowing users a choice of different software and systems to access the information. They do not have barriers to access, such as proprietary computerised access software or patented reader technology. For example, the XML file formats of OpenOffice.org are open formats, while the binary formats of Microsoft Office are not.

The Initiative will provide information including an open format registry, coordinate and initiate campaigns, and provide criteria for certifying information accessibility. Assistance with all of these areas is welcome.

Open formats are vital to help remove the digital divide from the "information revolution", and to ensure both physical and social accessibility of information. In this way, the IAI complements other projects to broaden access to information by lowering costs of access, such as the Open Access Initiative.

However, the IAI does not directly concern itself with the cost of information: it is possible to charge for production or supply of information, yet still produce that information in an open format.

Promoting open formats will also help a wide range of free software and creative commons projects. The IAI complements other European groups working on copyright and patent use, such as the Free Software Foundation (FSF-Europe) and Foundation for a Free Information Infrastructure (FFII).

The Information Accessibility Initiative's web site is at
`http://www.okfn.org/iai/`

The Information Accessibility Initiative is a member of the Open Knowledge Foundation Network: see
`http://www.okfn.org/`

Proposed UK digital rights organisation

We have received details of a proposal at Pledgebank for a UK digital rights organisation (intended as a UK equivalent of EFF). A pledge has been placed there stating:

"I will create a standing order of 5 pounds per month to support an organisation that will campaign for digital rights in the UK but only if 1000 other people will too."

Further details are at
`http://www.pledgebank.com/rights/`

SANE 2006: call for papers

We have received the following call for papers from the organisers of SANE 2006.

The Programme Committee for the 5th System Administration and Network Engineering Conference (SANE 2006) seeks original and innovative papers about the applications, architecture, implementation, performance and security of modern computing systems and networks. Paper submissions are due October 24, 2005. Organised by Stichting SANE and co-sponsored by

Stichting NLnet, USENIX, and SURFnet, SANE 2006 takes place May 15-19, 2006, Delft, The Netherlands.
`http://www.sane.nl/sane2006/`

FLOSSIE conference report

Leslie Fletcher

Along with about 50 others, I attended the second day of a two-day FLOSSIE (Free/Libre Open Source Software In Education) meeting organised by Schoolforge UK at the Bolton Technical Innovation Centre:
`http://www.uktic.org/`

Although it is hard to tell from its website, the Centre has a major commitment to open source so was a most appropriate venue for the event. The programme for the two days can be found at
`http://www.schoolforge.org.uk/flossie/conference200507.html`

Speakers' slides will eventually appear on the Schoolforge UK website.

The first talk of the day was given by Dr Brian Iddon MP, Chair of the Board of Bolton Technical Innovation Centre Limited, Member of Parliament for Bolton South East and Member of the House of Commons Science and Technology Select Committee. I was a colleague of Brian before he became an MP and I was delighted to be able to renew an old acquaintance. It is a pity that one of the other Bolton MPs did not put in an appearance!
`http://www.theyworkforyou.com/mp/ruth_kelly/bolton_west`

The day's keynote speech was given by Liz Grant, Business Development Executive, Public Sector, IBM Global Services. She began with the rather provocative comment that "Research shows that learners do not associate ICT with school". Children and young people appreciate and respond to the technology around them in the form of mobile phones, games consoles etc, but find mere computers in schools much less relevant and exciting. This needs to change if the potential of ICT to enhance learning is to be fully realised. She highlighted the success of Kemnal College, Bromley
`http://www.ktc.bromley.sch.uk/`

to which IBM provides a managed service, and the innovation at Orwell School, Felixstowe, well-known in the open source community for its pioneering efforts
`http://www.schoolforge.org.uk/index.php/Orwell_High_School,_Felixstowe`

Liz finished her presentation by commending the government's "Building Schools for the Future" (BSF) funding as a not-to-be-missed opportunity for schools to improve their ICT facilities. During the question session I voiced my strongly-held views that BSF is a serious threat to the use of open source in schools and that IBM is looking the other way while this happens.

Following the coffee break, Ian Lynch gave the best talk I have heard on the enormous educational potential of open source. He illustrated this using INGOTS, INternational Grades in Office TechnologieS
`http://www.theingots.org/www/index.php`

and OOoAuthors, the volunteer group writing documentation for OpenOffice.org
`http://www.oooauthors.org/`

Ian's title – "Strategies to get from zero to Linux" – was a long way from doing his talk justice. The main point I took from what Ian had to say was that the open source philosophy delivers not only technical excellence but, much more importantly for education, a way of working. Ably assisted by Daniel Carrera from OOoAuthors, Ian argued that open source projects can help children and young people get involved in live international projects. In ways dramatically more effective than the conventional ICT curriculum, this gives them invaluable skills in

- Finding things out
- Developing ideas and making things happen
- Exchanging and sharing information
- Reviewing, evaluating and modifying work as it progresses

to quote the ICT National Curriculum Themes.

Steve Lee and Simon Judge spoke next on “Open Source Assistive Technology Software”, which supports people of all ages with special needs. This is a very specialist, though growing, area. Software which users can “just use” and complete reliability are very important when day-to-day living is dependent on it.

If I can blow a Manchester trumpet, the importance of ICT in special education is very well illustrated in the OFSTED report for Piper Hill School, Manchester, which is an absolutely outstanding special school:
`http://www.ofsted.gov.uk/reports/105/105611.pdf`

The final talk before lunch was given by Richard Voaden, IBM’s North Region Linux Software Manager. The conference was considerably behind schedule by this stage so he had the unenviable task of keeping delegates from their lunch! Speaking to the title “Why Linux”, Richard expanded Liz Grant’s talk earlier in the day in a tour de force around the whole Linux scene, as perceived by IBM, of course! He spoke very persuasively of the growing impact of Linux worldwide and of IBM’s commitment. He was particularly interesting on Linux in the UK public sector, where he felt that local government was leading the way. Central government attention is so much distracted by other changes that it is unable to respond to the opportunities and challenge which Open Source presents.

The one talk after lunch, “Parrs Wood School - A pioneer of FLOSS!”, involved me, so others should comment on it!
`http://www.schoolforge.org.uk/index.php/FLOSSIE_2005_Report`

I found the whole day very stimulating. However, I had a definite feeling of deja vu and that FLOSSIE had not moved very far since the previous conference in February 2004
`http://www.schoolforge.org.uk/flossie/conference200402.html`

Open source in schools is still heavily dependent on a small number of enthusiasts. Most schools and LEAs remain unaware that there are more-than-viable alternatives to the standard, expensive and restrictive proprietary offerings. All of us who are committed to Open Source and education need to think really hard about how the message can be spread to a wider audience.

GDB Pocket Reference
Arnold Robbins
O’Reilly and Associates

ISBN 0-596-10027-2
80 pp.
£ 6.95

reviewed by John Collins

GDB is the GNU debugging package which may be used to debug C, C++ Objective C, Java compiled to machine code with GCJ and Fortran. Best support is in conjunction with the GNU Compiler Collection, GCC.

The book gives a brief but reasonably clear description of the options, configuration files command syntax and a summary of how to do things before launching into an alphabetical list of commands and brief syntax instructions. The book has a 7-page index – 10% of the pages – which makes it the only pocket reference I've seen with one and I've got ten others in the same series.

I don't think there is actually too much to say about a debugger and this book says very well all you'd really want to read in such a reference. Nothing really beats actually using it in anger, getting used to the facilities which help you get the job done and this will help you do that.

I'm sure this book will be of great assistance to anyone doing any serious development work in any of the supported languages.

SSH The Secure Shell: The Definitive Guide

Daniel J Barrett, Richard E Silverman and Robert G Byrnes
O'Reilly and Associates

ISBN 0-596-00895-3
400 pp.
£ 28.50

reviewed by John Collins

SSH is a suite of servers and clients which replace telnet, FTP and rlogin with much more secure equivalents. Not only is the authentication required to establish a connection more secure, but all the data traffic, including to and from X servers, is transparently encrypted.

SSH, after its free beginnings as a project by Tatu Ylonen, a researcher at Helsinki, has split into two versions, OpenSSH, which continues to be Open Source, now in release 4.1 (although this book describes 3.9) and is supplied with most Linux distributions and a number of Unix distributions, and a commercial version called SSH Tectia (abbreviated to Tectia). The command-line options and many of the keywords in configuration files are subtly different between the two versions. The commercial version can be obtained from **`www.ssh.org`** and the Open Source version downloaded from **`www.openssh.com`** (I find this contrast slightly amusing!).

This book attempts to describe both versions, with tables in places explaining the differences. I rather had the feeling that the authors preferred Tectia to OpenSSH although I am sure that almost all the readers and all of the writers of this review will have no intention of buying Tectia. The appendices are primarily focused on Tectia and its options and benefits.

That said, I did learn quite a lot about SSH which I didn't know before from this book, and one or two things I was actually doing wrong. It covers all aspects of SSH in great detail. Starting from an overview and discussion of basic usage, it talks about internals, compiling and installing the sources, configuring servers, key management, advanced options for clients and servers, port forwarding, recommended setups, case studies of particular requirements and troubleshooting. It then talks about a variety of non-Unix implementations from various sources, with particular attention to Tectia under Windows. The five appendices deal with options and features, although mostly emphasising Tectia.

I think that this book is well-written and comprehensive despite it talking so much about the Tectia product with its subtly different options, although most of the time it does make clear where it is talking about Tectia rather than OpenSSH. If you are going to be doing a lot of work with SSH and want to be sure you've covered security issues, port forwarding and so forth properly, I'm sure this book will be extremely helpful to you.

Mac OS X Tiger for Unix Geeks

Brian Jepson and Ernest E Rothman

O'Reilly and Associates

ISBN 0-596-00912-7
415 pp.
£ 24.95

reviewed by Graham Lee

Released to coincide with version 10.4 of Apple's operating system, this is the third edition of "Mac OS X for Unix Geeks". As with the previous editions, the scope of the book is wider than its 395 pages would suggest. It sets out to describe not only the differences in implementation between Darwin – the open source Unix layer in Mac OS X – and other Unix flavours, but also features unique to Mac OS X.

The definition Jepson and Rothman seem to use for "Unix Geek" is "developer, systems and network administrator who is also a power user and likes dual-booting"; chapters on directory services, package managers, source compilation and multimedia give some example of the breadth. The effect of squeezing so many topics in is that the depth of coverage is highly variable. It can seem that this book was written with 800 pages and then had sections arbitrarily chopped out to make it fit. Some of the topics which are briefly covered or neglected relate to new features in Tiger which are not found in any other Unix or indeed earlier Mac OS X, and should be prime pickings for this kind of text. Examples of this are `launchd`, Apple's replacement for `init`, which is given a quick look in the chapter on OS X's boot sequence, and Apple System Logger, used in Tiger instead of syslog, which is not mentioned at all. Similarly the chapter on searching and metadata discusses Spotlight and its command-line tools in great detail, but doesn't mention the `*xattr()` family of metadata access functions.

Where a topic is covered in depth, the book genuinely excels in its description. One such example is the Directory Services chapter, which includes information relevant to both developers and SAs. It shows an example of a traditional Unix way to get a password, explains why this fails on Mac OS X, then describes a solution. It explains the structure of Mac OS X's Directory Services setup and how it relates to `/etc` files on other systems. Two chapters are devoted to information for developers, which are similarly detailed in their coverage of Apple's GCC, porting considerations, and the Darwin dynamic link editor. Cocoa and Carbon – Mac-specific APIs for creating Objective-C and C++ applications respectively – are not treated in this section but are not of immediate relevance to the "Unix Geeks" of the title.

The interoperability of Mac OS X with other Unix is rightly given plenty of space, so setting up open source databases, scripting with Perl and Python and using X11 are all discussed as well as using Macs with CUPS and NFS. Providing Macintosh services on other platforms, for instance with the netatalk package or Unison are also mentioned but not in as much detail. This seems slightly at odds with the view of the audience as Unix-savvy and wanting to know how Mac OS X differs and what else it has to offer, and in my view reinforces the notion that the book could potentially cover much more.

Mac OS X Tiger for Unix Geeks is a useful introduction for Unix sysadmins new to Mac OS X, although some topics receive such brief attention that it may provide no more than a list of search engine keywords in some cases. This is unfortunate given the high quality of treatment given to other subjects; if the book were divided into two volumes, one for SAs and one for developers, each of length equal to the current edition then they would make an excellent reference. Apple are famed for the pace of their operating system's development, and that could limit the longevity of the book's currency. The Panther edition was only on the shelves for 15 months.

Ship It! A Practical Guide to Successful Software Projects

J Richardson and Will Gwaltney

Pragmatic Bookshelf

ISBN 0-974-51404-7
200 pp.
£ 20.95

reviewed by Harry Newton

This is a book about the other half of software development: the half that isn't writing code but which, if done badly, can cripple a project. For want of a better phrase, development process – how developers should work to deliver good robust software. The kind of things established engineering disciplines do well, but software development does badly, and all too often atrociously. The subtitle of the book is "A Practical Guide to Successful Software Projects" and this gives a clue to the nature of the book. It is essentially a handbook of best practice as collected by the authors over their lengthy careers and varied roles. The real strength of the book is the pragmatic approach taken by the authors.

The book has four main chapters: "Tools and Infrastructure", "Pragmatic Project Techniques", "Tracer Bullet Development" and "Common Problems and How To Fix Them". The eight appendices are synopses of the tools available for the stages discussed in the first chapter, a list of development methodologies and recommended reading. The first chapter, "Tools and Infrastructure", is a straightforward recipe for building, testing and managing code. Topics covered include: sandbox development, automatic building, issue and feature tracking and test harnesses. The second chapter, "Pragmatic Project Techniques", is in a similar vein, but concerned with effective team working rather than each individual's working practice. It covers the role of the technical lead, the needs for effective team communication, code reviews and change notifications, and work tracking. The third chapter, "Tracer Bullet Development", is a discussion of the development methodology favoured by the authors. In this method, the project is divided into blocks of related functionality (e.g. client, database), interfaces established between them, and each block assigned to a different team. The smallest amount of code is written to glue everything together, the blocks being mock objects at this moment. The stubbed out blocks are then filled with functional code. The fourth chapter, "Common Problems and How To Fix Them", is exactly what it says it is. The problems discussed are mainly at a project level (e.g. "We're on a 'Death March' Project", or "Help! I've Inherited Legacy Code") and, as with a lot of the book, the suggestions are common sense and practical.

All this may sound obvious, but it clearly is not. The authors quote a recent survey in which 40% of US software houses were found not to use any form of source control. Another survey shows that 70% of software houses have no daily build process, let alone a continuous integration system. The book is not dryly prescriptive. Although there are a series of recipes, its real benefit is in the experience of the authors. They are prepared to backup their suggestions with examples from their careers: they demonstrate and persuade rather than simply prescribe. This is what impressed me about the book. I didn't see it as reference, but more as a discussion that prompted me to consider the way I work.

Who should read it? Most people I think. It's aimed at technical leads certainly, but all developers would benefit by reading it. I believe that project managers would benefit from having high-level familiarity with the book's ideas. I thoroughly enjoyed the book: I found it a pleasure to read and was particularly impressed with the layout and organization of the material. Recommended.

Understanding Open Source and Free Software Licensing
Andrew M St Laurent
O'Reilly and Associates

ISBN 0-59600-581-4
224 pp.
£ 17.50

reviewed by Michael Fraser

The picture on the cover of Understanding Open Source and Free Software Licensing by Andrew M. St. Laurent is a 19th century engraving of a shootout at a railway in the American West. What early conclusions should we draw from that less than innocent image? Leaving aside men with guns in the Wild West, Understanding Open Source is an in-depth study of software licences commonly used with the release of open source or free (as in speech) software. The book is written by a US lawyer but not necessarily for lawyers (though I am certain there are lawyers who would benefit from reading this work) nor just for US citizens (ditto).

There are over fifty approved licences listed by the Open Source Initiative (OSI) [1], together with a review process for further licences. Clearly, it would be a work of weight (and dare I say not terribly interesting) that examined each one in detail. A quick look at SourceForge.net, home to the largest number of open source software projects [2], reveals projects to have been licensed under most of the OSI-approved licences with by far the most popular being the GNU General Public License (GPL) (41067 projects), GNU Library or Lesser General Public License (LGPL) (6596 projects), and Berkeley Software Distribution (BSD) License (4271 projects). The Apache Software License, the Artistic License, and MIT License all have in the region of 750-1500 projects each. Following this pattern Understanding Open Source provides a close reading of these together with the Academic Free License and Mozilla Public License.

In addition to the software licences, St. Laurent has included a chapter dedicated to the QT, Artistic and Creative Commons licences and a further chapter on non-OSS licences which incorporate some OSS-like elements. The latter mainly comprise the Sun Community Source licence [3] and the Microsoft Shared Source Initiative. However, Microsoft's contribution is only discussed briefly and cast aside as, 'it is, at least at this time, little more than a branded extension of Microsoft's current commercial licensing practices' (p.145).

The bulk of the book, therefore, deals with individual licences (four chapters out of seven) with the remaining three chapters providing: a basic overview of copyright, licences in general and the definition of open source; the legal implications of entering into software contracts based on open source/free software; and finally, a chapter dedicated to issues particular to software development.

I approached Understanding Open Source as Co-ordinator of the Research Technologies Service (RTS) [4], a section which contains a significant number of projects piloting new technologies and which has a commitment to open standards and the appropriate use of open source software. Of course, I should note that the RTS (as are others in a similar position within the UK) are ably assisted by OSS Watch [5], the JISC-funded open source software advisory service, in our decision-making concerning the development and deployment of open source software. There still remains the need, however, for detailed guidance on single issues, which is what this book aims to do for licensing.

Given that the audience of the book is not necessarily lawyers, I also approached it with three questions which are based on actual cases recently encountered:

- A nationally-funded project states that it will release software outputs as open source. But under which licence? [6]

- A company producing ’closed source’ software wishes to include material from the aforementioned project within one of its products. What does it need to know to make that decision?
- A university computing department selects an open source product for one of its enterprise systems and intends to modify the code for its own needs. What might its obligations be in this regard?
- Oh, and finally, Understanding Open Source & Free Software Licensing is licensed under the Creative Commons Attribution-NoDerivs License 2.0. What does this mean for me the reader?

There are some caveats with this approach. First, as with many such questions the initial answer is always going to be ‘it depends’. But at least having an overview of the dependencies is useful. Secondly, even if you do work out an answer, don’t always expect absolute truth. Licences, like the Bible, may give the impression of having been God-given but actually they’re the work of human hands, attempting to disambiguate language in order to convey common meaning, but often failing. In which circumstance the lawyers will often direct us towards that other semi-divine institution, the judgement of the courts. And it is the court which will ultimately interpret its meaning in law. This applies as much to the use of licences or contracts originally developed in the US (as most open source licences are) - and used within the UK - as it does to licences written with English law in mind.

It is not the purpose of this review to provide the answers to any of the above questions, but rather to give the reader some sense of whether the book under review will help to provide the answers.

Start at the end of the book if you are trying to choose an open source licence. Sections in chapter 6, on the legal impacts, outline some of the issues particular to open source/free software licensing (e.g. Violation of a licence risks nullifying any benefits accrued from improving licensed code; or how to join together material from two or more programs released under different licences). Chapter 7 provides a relatively brief discussion of the issues inherent in open source licences relating to software development, including a section on the risks and opportunities of project forking and an outline of the nature of the choices faced by a project manager or software developer. Perhaps, not unexpectedly (and in keeping with the no-easy-answers theme): the conclusion drawn is that ‘while a certain license is the best for a given project, particularly when a substantial amount of work has already been done under that license, such decisions depend largely on circumstance and on the taste of the project developer’ (p.176). Elsewhere, it’s noted that the taste might not always be pragmatic, ’The thesis is that the licensor’s choice to use the GPL license is, in some sense, a political one, and that choice should be protected and defended against encroachment’ (p.45 on §7 of the GPL). However, a deeper consideration of rights management within a software development environment would have been useful at this point.

Having to proceed to the end of the book is a symptom of the book’s format which does not lend itself to such linear decision-making. The detailed commentary on each of the selected licences is useful if you already have an idea of which licence you wish to employ for your own software (or have modified and plan to distribute an already-existing open source application) and wish to understand the implications of a particular clause. However, if you are still at an earlier point in the decision-making process then the inclusion of a flowchart would have been helpful in order that a project, for example, might determine answers to some of the more basic questions, e.g. Who actually owns the software in whole or part – you, your employer, a consortium? Are you sure you know? Is the software wholly original or derivative in part or whole? Do you care if others modify and re-distribute the software under a different licence, even a closed licence?

What disclaimers of liability or warranty do you want or are permitted to include? How much of this would you be prepared to stand up and claim in court? Having answers to starter questions such as these are crucial to both the development of software in an open source context or the modification and distribution of someone else's open source software.

Commentary on complex and often misunderstood licences like the GPL and LGPL, for example, is as clear as one could expect. As with many attempts to explain legal clauses for the general reader there is always a risk that one over-burdened sentence is replaced by a paragraph equally weighty in meaning. Whilst some of the commentary suffers from not enough re-writing in simple terms, the author himself, fortunately, does not hold back from drawing attention to inconsistencies or ambiguity within a licence. For example, on the Academic Free License, 'There are problems with this first sentence. First, it is not immediately clear that the licensor intends that the provisions of this license also govern the derivative works created by the licensee and derivative works created by the licensee's licensees and so forth' (p.27). Or, concerning the LGPL, 'This bar on the creation of derivative works other than libraries from LGPL-licensed works makes the LGPL essentially useless as a license for such works. Creators of such works should look to the GPL or another open source license' (p.53). Or, finally, the Mozilla Public License 1.1, 'As with many provisions of this license, its [§3.4(c)] legal effects are unclear at best' (p.72).

Knowing the substantial differences between the licensing models helps to answer the second question relating to inclusion of material from open source software within a 'closed' or proprietary package. Clearly the company needs to read the terms of the licence. For example, what may be permitted in the BSD and Apache licences (so long as the copyright notices are reproduced) becomes somewhat more complicated in the GPL or LGPL where the meaning of 'to include material' is crucial (perhaps it's actually 'mere aggregation' or a 'linked' software library). The author is also eager to dispel myths on this subject, 'Contrary to the beliefs of some, the GPL does not require that software running on a GPL-licensed operating system be licensed under the GPL' (p.82). (As he notes elsewhere, the GPL is not contagious like a cold. Software doesn't catch it by mere proximity - it needs to derived from or integrated with GPL-licensed code.)

Finally, there is a section within the book which discusses the Creative Commons licences which although not intended for software, certainly draw upon the open source way of life [7]. By comparison, the Creative Commons licences are models of clarity (and also probably the only open source-like licence to have a version adapted for UK usage [8]). It happens that the entire Understanding Open Source & Free Software Licensing is licensed under the Creative Commons Attribution-NoDerivs License. The commentary in the body of the book relates to the 'Attribution-ShareAlike' 1.0/2.0 version of the licence. However, the 'Attribution-NoDeriv[ative]s' version is reproduced as an appendix. As you may imagine this neatly answers my final question and clarifies the rights I have:

- to copy, distribute, display and perform the work
- to make commercial use of the work
- but not to alter, transform or build upon the work

while at all times giving the original author credit and making clear the terms of the licence in any subsequent distribution.

Understanding Open Source is a text-rich work. There are no attempts to visualise the issues through any other means than writing about it. The use of comparative tables, flowcharts and other illustrations would have helped mitigate the risk of missing an important observation simply because it was buried in the discussion of section 2.1(b) of a licence in which you assumed you had no interest. Having said that, making the effort to read the book in a linear fashion before using it as a reference work certainly increased my understanding of the practical implications of the freedoms and benefits granted under open source/free licences. I can certainly recommend this book to anyone who has a responsibility for reading software licence agreements whether prior to clicking 'I accept' on an installer, or as part of the process of creating, building on and distributing open source software. I am sure institutional legal services or technology-transfer units will find the book to be of interest, especially if the queries about open source software are only now beginning to arrive in their inbox.

And if you want to write your own open source software licence? Read the book and then consult a lawyer.

References

(1) Open Source Initiative (OSI)
`http://www.opensource.org/`

(2) SourceForge.net: OSI-approved open source
`http://sourceforge.net/softwaremap/trove_list.php?form_cat=14`

(3) The OSI has recently approved an open source licence submitted by Sun Microsystems, the Common Development and Distribution licence (CDDL) based on the Mozilla Public License 1.1. Solaris 10, Sun's operating system, is being released as an open source product. See further
`http://www.sun.com/cddl/`

(4) Research Technologies Service
`http://www.oucs.ox.ac.uk/rts/`

(5) OSS Watch - the JISC-funded open source software advisory service
`http://www.oss-watch.ac.uk/`

See further "Government policy on the use of Open Source Software within the UK government"(2.0) which includes the principle, "Publicly funded R&D projects which aim to produce software outputs shall specify a proposed software exploitation route at the start of the project. At the completion of the project, the software shall be exploited either commercially or within an academic community or as OSS".
`http://www.govtalk.gov.uk/policydocs/consult_subject_document.asp?docnum=905`

(6) Creative Commons: choose license.
`http://creativecommons.org/license/`
`http://creativecommons.org/worldwide/uk/`

Michael Fraser is the Co-ordinator of the Research Technologies Service at Oxford University
`http://users.ox.ac.uk/~mikef/`

This article first appeared in Ariadne, Issue 42, January 2005 and is reproduced by permission.
`http://www.ariadne.ac.uk/issue42/`

Mapping Hacks

Schuyler Erle, Rich Gibson and Jo Walsh

O'Reilly and Associates

ISBN 0-596-00703-5
304 pp.
£ 20.95

reviewed by Gavin Inglis

See the combined review below.

Web Mapping Illustrated
Tyler Mitchell
O'Reilly and Associates

ISBN 0-596-00865-1
367 pp.
£ 28.50

reviewed by Gavin Inglis

The short Preface to Tyler Mitchell's "Web Mapping Illustrated" is nicely judged. He recalls the magic of finding his Scout camp and detailed features of his home town on a topographic map, and how that map inspired him to get out and explore the area. This evocative introduction really captures the author's enthusiasm for mapping and motivates what is at heart a technical book.

This volume takes a very applied approach, but assumes no background in geographic information systems (GIS) on the part of the reader. What it does assume is a basic competence in software installation and perhaps web server administration. The text runs from top level concepts right down to the detail of how to compile MapServer and use the likes of **`grep`** and **`sed`** to extract information from map data files. This makes it a good purchase for the reader who only wants to buy one book on the subject. If there is a theme, it is that understanding your geodata and its purpose is the key to effective mapmaking.

The focus is on using open source software to create maps in a variety of useful ways. Various packages are covered, from the popular workhorse MapServer to the OpenEV viewer and a range of useful libraries. 3D visualisation receives a basic treatment. There are chapters on sourcing map data, with plenty of ideas and URLs to get you started. Chapter 9 concerns creating and editing personal map data, and steps the reader through a complete data project, showing a large fire around the Canadian city of Kelowna in the summer of 2003, and how its extent threatened a particular home. This and the subsequent chapter on creating static maps is likely to fire the reader's enthusiasm for starting a new project.

This is all preparation for the later chapters which deal with mapping through the web. First we consider combining MapServer with Apache. Enter this chapter with a bit of background in web forms and CGI and you should leave it confident that you can implement a clickable, zoomable online map. From there it's a short step to web services; querying those which are out there and setting up your own. Final chapters concern the PostGIS spatial extension to PostgreSQL and programming the MapServer API with examples in Python.

Throughout, the book is very attractive, with rich colour maps on about half of the 350 pages. Its Open Source approach means that the emphasis is on using Linux, but the door is left open for Windows users with brief advice on most packages. UK readers may find the map data a little too oriented towards the US and Canada. However the principles are transferable.

For a comprehensive conceptual introduction to GIS there are likely to be better choices, but "Web Mapping Illustrated" is a solid buy for the web mapping beginner with an established technical background.

The visionary introduction of Mitchell's book is ratcheted up a notch in the Preface to "Mapping Hacks". This envisions a near future where location-specific data of every kind – environmental, cultural, community (and spam) – is integrated into a geoinformation space which reacts to precisely where you and your mobile devices are, whether you like it or not. The humanising concept is of stories told using maps, a narrative approach which runs through the entire book.

Like other entries in the O'Reilly "Hacks" series, the bulk of the book is made up of self-contained little projects which can be completed in no more than a single evening. These range from using online services to create simple street maps to mapping an entire fictional planet.

Chapter 1, "Mapping Your Life" has nice suggestions about giving geographical context to your holiday snaps, tracing flights using Sherlock and creating a distance grid in Excel. The delightful "Will The Kids Barf?" compares point-to-point distance with actual road miles to evaluate just how curvy and hilly a particular route might be.

The scale then expands to the neighbourhood level, showing how to map wifi hotspots, health code violations, and how the neighbours are voting. Zoom further out, and we calculate distances between points on the globe using trigonometry, and experiment with projections to visualise the world in different ways. By the end of chapter 3 we're raytracing to create a 3D rendering of other planets in the solar system.

As soon as gadgets enter the picture, of course, all bets are off. Mac users can connect their GPS to Terrabrowser to allow monitoring of their position on an aerial photo in real time. Windows users can convert their tracking logs to a 3D visualisation of their route. And Linux users can add a geostamp to their audio recordings. Perhaps the ultimate hack in the book is number 62: "Build a Car Computer - A project that will consume all your time and money, but make you the envy of your nerd friends."

The final chapters encourage collaboration, and contributing to the "geospatial web". Many of the ideas here are mind-expanding.

Again there is a US emphasis, but this time the reason is explained: the US government is legally compelled to make its data available at only the cost of reproduction, whereas the Ordnance Survey in the UK is compelled to charge users for its data. The argument that this will slow down the development of geodata services in the UK is convincing.

As with other books in the series, one or two of the hacks seem more like sidebars than actual hacks. However with 100 of them compiled at a reasonable price, "Mapping Hacks" represents excellent value. Map geeks will delight in this catalogue of appealing ideas and projects. The rest of us will be educated and impressed.

Perl Best Practices

Damian Conway

O'Reilly and Associates

ISBN 0-596-00173-8
544 pp.
£ 28.50

reviewed by Alain Williams

Summary: experienced perl hackers will find this worth reading even if some of it is dubious.

I have been writing perl for some 15 years and so looked in this book for things that I might be getting wrong or could do better. What did I find ?

The structure of this book is of a piece of advice, expressed as an imperative rule, followed by a discussion explaining why you should follow the rule. These a grouped by topic starting with code layout and what conventions to use for variable names. Much of this makes sense, eg: 'Place a comma after every value in a multi-line list' and 'Code in paragraphs'; other parts are recipes for arguments 'Use four-column indentation levels'.

After this Damian talks about how to write constants: 'Don't pad decimal numbers with leading zeros' (because they become octal numbers) is fair enough, but we are also exhorted to not use `''` or `""` for an empty string but to use `q{}` instead, and for a single space to not use `' '` but use `q{ }` instead (since the former can, apparently, be confused with an empty string).

In the chapter on variables he nicely explains how to use localisation of perl magic variables (which can change how perl does things) to prevent the unexpected elsewhere in the program and how to make your program faster by not using the regex match variables except when you really need them. The use of `$_` can have unexpected side effects as it can be an alias for another variable – something that I was hazily aware of and this firmed me up on.

Control structures receive attention next, again much that makes sense, but some advice I found not just 'not me' but puzzling. For instance we should not use `unless` or `until` but `if !` and `while !` since many developers will not be familiar with these from other languages; I must admit that one of the reasons that I like perl is for these very keywords that allow me to not negate a condition.

The chapter on documentation is one that I wish many people to read, even if all that they learn is that writing documentation is not hard and won't give the programmer a fatal disease. He gives some templates for POD and some good advice such as: 'Comment anything that has puzzled or tricked you'.

There follow chapters on: built-in functions, user defined subroutines, I/O, references, regular expressions, exceptions, command line (option) parsing and processing, object orientation, classes, modules and debugging. These contain the same mixture of advice - good and, errm, interesting.

The justification for much of the advice is to write code that later on can be read and understood with a minimum of mistakes. Many of the recommendations and reasoning I am happy with, but some not – even after allowing for different styles. The author does acknowledge this early on commenting that a style applied consistently is far more important than the details of the style.

I found it good to read something like this to use as a reasoned input into my own perl style. Although I did not agree with everything that was said, it made me think, and I will try to do some things differently. This is the sort of book that you can read an occasional chapter at a time and think about what is written before moving on.

I found that the book provoked thought of what I knew well and made me sharpen up on some things that I was a bit fuzzy on. I suspect that many of you, like myself, learned perl some time ago and use a subset of the language – this book can encourage you to try some new things.

Learning Perl - 4th Edition

Randal L. Schwartz Tom Phoenix and brian d foy

O'Reilly and Associates

ISBN 0-596-10105-8
704 pp.
£ 28.50

reviewed by Greg Matthews

A few years ago I was given the task of maintaining and improving a bunch of in-house scripts written in Perl and was given the first edition copy of "Learning Perl" (the Llama) along with "Programming Perl" (the camel) and told to get on with it. I never seemed to have the time to devote to really studying this language and everything that I have learned is the result of having to fix someone else's code or bolting on extra functionality to an existing tool.

Initially, the fourth edition of the Llama doesn't seem that different to the first, but further investigation shows that a lot of thought has gone into the progression of the book and its topics, the subject matter of the chapters is much more clearly delineated. It no longer plunges into multi-page scripts in the Introduction.

Early chapters are concerned with introducing Perl as a language and an interpreter, getting the reader used to the perl vocabulary of scalars, arrays and hashes. Then on to meatier topics of subroutines and I/O. All new material is exemplified with code snippets and each chapter ends with a handful of exercises (reflecting the materials origins as course notes). Chapter 6 introduces “hashes” (these used to be called associative arrays but as they were so useful and used so much, the perl developers decided they needed a shorter name to lessen the risk of RSI) and then we are on to three whole chapters devoted to regular expressions. Perl and regular expressions are inseparable so it’s great to see them covered in detail. Some people may think they look like line noise but in fact regular expressions are not very difficult, and although I am not fluent, I know that they are logically constructed and if I quell the rising sense of panic at seeing something like:

```
s/\w<([^>]+)>/\U$1/g
```

I can start at the beginning and apply the rules of regular expressions and work out exactly what it means. Learning how to read and use regular expressions will stand you in good stead in all sorts of situations. As Perl’s great strengths are in parsing, manipulating and reporting data it makes perfect sense to devote 34 pages of the book to deal with this topic.

Next we are introduced to more control structures, these are standard toolbox commands such as conditionals, incrementers and loops which are many and varied. Interestingly, at the end of this chapter, the exercise is one that appeared as an example in the Introduction of the first edition – 10 chapters into the fourth edition! More chapters deal with file and directory handling.

Further information follows on process management and even brief sections on forking and signal handling – I would have like more on these topics but each of these requires a more in-depth analysis and there is no shortage of information available for anyone who has graduated past the scope of “Learning Perl”; Appendix B “Beyond the Llama” is 20 pages of references to more detailed information. Its worth noting from this, that a vast amount of documentation is included with standard installations of perl; a quick **`man perl`** will get you going, or **`perldoc perltoc`**.

This is expected to be the last Llama book published before the release of Perl v6 but Perl v6 is hardly mentioned, new users of Perl would be interested to know of future developments. My main concern was the lack of coverage for CPAN, the online Perl archive, absolutely crucial to anyone writing Perl code. This edition doesn’t have Larry Wall’s original foreword but it continues the tradition of copious footnotes, most of which are either exceptions to the rules – which are many – or bad jokes. The style is light and open which is no mean feat for a language that has competitions for obfuscated code. This book can be summed up as a solid introduction to Perl v5.8. There’s no quick way to learn a language but finding time to work through this book will put you in good stead. Anyone past the basics of the language would be better off splashing out on “Perl Cookbook” or “Learning Perl”.

Make: Technology on Your Time (volume 3)

Mark Frauenfelder (Editor)

O’Reilly and Associates

ISBN 0-596-10080-9
224 pp.
£ 9.95

reviewed by Greg Matthews

When this book first dropped into my in-tray for review, I was overjoyed. I had recently discovered this series on the O’Reilly book store at the UKUUG conference in Swansea. After poring over them I had decided to buy volumes 1 and 2; one for me and one for a friend (after I’d read it of course). Now, volume 3 is in my clutches too.

First of all, this isn't exactly a book so much as a quarterly magazine. Only three editions so far but each one has been jam packed with fascinating articles. The problem I have now is to give you an idea of its content.

Make is aimed at the sort of people who are driven to take stuff apart and put it back together again, preferably in new and bizarre ways. If the warning “No user serviceable parts inside” just makes you even more eager to open it up then this magazine is for you. Finding new uses for old and new technology is the main thrust, as well as articles on DVD copy protection, Open Source, tools, welding, electronics, anything that could interest the hand those that caught my fancy in this issue are the alternative uses for old VCRs (in this case, a cat food dispenser) and of course, the “Night Lighter” potato canon which can fling spud plugs up to 200 yards! One of the other projects is actually a whole series of ideas and howtos for hacking on your car, such as making it into a WiFi hot spot or connecting your MP3 player to the stereo. This was less interesting to me as I don't use a car much, but the instructions on brewing your own bio-diesel were fascinating and went into some detail on the actual chemistry involved.

The projects don't take up all the room in this weighty volume (almost 200 pages). Lots of it is given over to short articles and reader contributions. This issue is worth buying if only to read about the scientists from all over the world who are still convinced that there is something to learn from the Pons-Fleischmann effect (cold fusion), and their attempts to recreate it in back-yard laboratories. Also noteworthy are the Canadian students that attached a petrol engine to a shopping trolley, and a short article on how to connect a disposable digital camera to your computer for an ultra cheap, lightweight reusable camera (ideal for kite photography - see volume 1).

I can't tell you how much I like this magazine. Of course I have the odd complaint; the articles have a US slant so prices are all in dollars, measurements are in feet and inches and many of the suggested materials suppliers are unlikely to ship to Europe. Also, the computer references tend to be a bit Mac-centric although Linux is often mentioned too. On the other hand, instructions are detailed, explanations are clear and technical details are rarely glossed over. I even like the adverts, for instance; the first advert of this issue is for an RFID (Radio Frequency Identification) tag reader, and the third is advertising a USB powered Linux server the size of a matchbox as well as offering bounties for people porting applications to it. That said, there is very little advertising and I wonder if that will have to change.

There isn't enough room here to fully describe the contents of the this magazine, I hope I've given a flavour. It costs £9.95 from O'Reilly and subscriptions are available at $49.95 (promotional codes don't work for overseas customers). Alternatively, you can subscribe to an online only version for $26.95. The era of disposable technology can't last, this magazine shows you that it can be reused and recycled by applying your imagination. Get making.

Classic Shell Scripting

Arnold Robbins and Nelson Beebe

O'Reilly and Associates

ISBN 0-596-00595-4
300 pp.
£ 24.95

reviewed by Mick Farmer

I first used UNIX in the mid 1970s, so it must have been version 6, or possibly version 7. The Bourne shell was so radically different from anything I'd used before (VAX Control Language? and something similar on a PDP-15) that I happily wrote scripts before I'd learnt to program in

C. Unfortunately, I can't find any of those early scripts, so it's nice to get a book that I can use to check whether I'm still up-to-date in this area.

The first seven chapters cover the basics (Background, Getting Started, Searching and Substitution, Text Processing, Pipelines, Variables and Control Structures, Input & Output) very thoroughly with short examples. A brief introduction to Internationalization (i18n for short) via locales quickly taught me about this important growth area.

I was nodding through these short scripts when one example leapt up and showed me how restricted the UNIX tool set really is for today's world. The tools excel at handling text on a line-by-line basis. Anything with an XML structure (such as HTML) and you're on your own – the script could only handle tag pairs written on the same line!

In Chapter 8 (Production Scripts) we begin to see some solid work and it was good to notice that option processing is taken seriously, as is error handling with the liberal use of functions.

Chapter 9 is an introduction to awk and chapter 10 was an overview of file handling.

Chapter 11 consists of the first serious example, that of merging the password files on two systems. It is important to get unique UIDs and that the ownership of files is correctly handled. This allows two computers to share their files over NFS. The script is built up piece by piece and the chapter finished with issues that might have to be addressed in the real world.

Chapter 12 covers everything you ever wanted to know about spell checking. It starts with the early UNIX prototypes built from pipelines, continues with a discussion about aspell and ispell, and finishes with a fully-functional spell checker written in awk. An interesting retrospective points out that the authors' 190 lines of awk compares extremely favourably with the GNU ispell (13,500 lines of C) and the GNU aspell (29,500 lines of C and C++).

Chapter 13 is about controlling UNIX processes and points out how the different shells handle processes in subtle, different ways.

This leads on nicely to Chapter 14, which deals with shell portability issues. It also describes the various extensions that have been imported into some shells. My impression is that the shells are converging towards one high-powered scripting language, perhaps one to rival Perl!

Chapter 15, the final one, contains a number of tips for writing secure scripts. I was pleased to see that I knew about most of the tips and even put some of them into practice.

There are three appendices (Writing Manual Pages, Files and Filesystems, Important Unix Commands) which briefly cover those topics. However, I do wonder how many people are going to write manual pages these days.

To summarise, an enjoyable read, but it doesn't contain any information that isn't available elsewhere. I like seeing a script (or any program for that matter) built up, step by step, with a good explanation of what each step is trying to achieve. Use this book, like me, to see if you're still keeping your script writing skills up to scratch.

Contacts

Ray Miller
Council Chairman; Events; Newsletter
Oxford
Tel: 01865 273 200
`ray.miller@ukuug.org`

Mike Banahan
Ely
`mike.banahan@ukuug.org`

James Youngman
UKUUG Treasurer
Manchester
`james.youngman@ukuug.org`

Sam Smith
Website
Manchester
`sam.smith@ukuug.org`

Alasdair Kergon
Events
Reading
`alasdair.kergon@ukuug.org`

Alain Williams
Watford
`alain.williams@ukuug.org`

Roger Whittaker
Schools; Newsletter
London
`roger.whittaker@ukuug.org`

Newsletter
`newsletter@ukuug.org`

Jane Morrison
UKUUG Secretariat
PO Box 37
Buntingford
Herts
SG9 9UQ
Tel: 01763 273 475
Fax: 01763 273 255
`office@ukuug.org`