news@UK

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

Volume 16, Number 3 **ISSN 0965-9412** **September 2007**

Contents

News from the Secretariat

Jane Morrison

This year July and August were exceptionally busy for the UKUUG office as all attention was focused on the LinuxConf Europe event held in Cambridge at the beginning of September. Bookings just kept coming in and we eventually had some 250 attendees. As Cambridge is quite local to the office it was easy for me to attend, and it was nice to meet people who I had only previously exchanged emails with. The event was a great success and UKUUG is very grateful to the event's sponsors: Intel, Google, Bytemark, Collabora and The Positive Internet Company. Linux Emporium very kindly produced the event CD and our Gold sponsors Novell and Sun Microsystems allowed us to budget for some add on extras. There were two social events, a Dinner in Churchill College on the first night of the Conference and on the next evening a Social event which included a tour of Duxford (Air Museum) with a buffet meal. The papers and slides from the event can be found on our web site at:
`http://www.linuxconf.eu/2007/programme/timetable.shtml`

On Tuesday 16th October we have two events organised, both at the same venue (Imperial Hotel, Russell Square, London): RT Tutorial and a Seminar on Databases. Please see full programme details on the web site at:
`http://www.ukuug.org/events/rt/`
`http://www.ukuug.org/events/seminars/databases/`

We are also planning another single day seminar at the end of November with the topic of 'Storage and Backup' full details will be available shortly and sent to every member.

Looking further ahead to next year we have already circulated the Call For Papers for the Spring Conference being held from 31st March to 2nd April 2008.

The event will be held in Birmingham at the Birmingham Conservatoire, and will include papers and tutorials on the major dynamic languages, including Perl, Python, Ruby and Smalltalk.

The next Newsletter will be the December issue and the copy date is 23rd November. As usual any articles, letters etc. can be sent for inclusion to:
`newsletter@ukuug.org`

News about Liaison

Sunil Das

"The Linux Desktop, Present and Future" is a light-hearted presentation and demonstration about the desktop that I have arranged on Wednesday 26th September 2007 at University College London: 6.45pm for a 7pm start (to follow the AGM). Michael Meeks, Distinguished Engineer at Novell, will prod some of the new toys coming into the desktop from promising technologies to curvaceous applications. More information is at
`http://www.ukuug.org/events/agm2007/`

"Databases and the Web" is the first in a series of UKUUG seminars. The event will be held on Tuesday 16th October 2007 at the Imperial Hotel, London. I have put together an exciting programme of six speakers covering diverse topics which include PostgresSQL, MySQL, Oracle, DB2, XML, Linux and AIX. More information at
`http://www.ukuug.org/events/seminars/databases/`

For our newsletter, my friend, the author Peter H. Salus has submitted another excellent "Letter from Toronto". I've been in regular contact with our colleagues in Usenix and this has resulted in publication of "ext4: The next Generation of the ext3 File System" and the obituary of John Backus.

Please continue to help with the liaison activity by email or telephone. Initial contact can be made using
`sunil.das@ukuug.org`

Conference announcements

We have received notification of the following events:

AUUG Annual Conference 2007

The AUUG Annual Conference will be held in Melbourne, Australia, 13-14 October 2007. The conference will be preceded by one day of tutorials, to be held on 12 October 2007. This year's theme is 'Quality'.

Further details are at:
`http://new.auug.org.au/AUUG2007/`

USENIX/SAGE LISA '07 Announcement

The 21st Large Installation System Administration Conference which will take place in Dallas, Texas, USA between 11th and 17th November 2007, and will include a large number of tutorials and technical sessions.

Full details can be found at
`http://www.usenix.org/lisa07/`

NLLUG 25th anniversary conference

NLUUG will celebrate its 25th anniversary with a very special autumn conference to be held on 7th November 2007 in Amsterdam.

Full details of the programme and on the special anniversary website at:
`http://www.nluug25.nl/`

As with all NLUUG conferences UKUUG and other NaLUUG members will get a special discount for this conference.

LinuxConf Europe 2007

Roger Whittaker

The annual summer Linux Conference took place in Cambridge at the beginning of September. The event took a slightly different form this year, as it was jointly organised with the German Unix Users Group (hence the European title), and was held immediately before (and partly overlapping) the 2007 Linux Kernel Developers' Summit which took place at the same venue.

Despite the different title, the format and atmosphere of the conference were similar to previous years: this was very recognisably an Alasdair Kergon production.

The relationship with the Kernel Summit guaranteed a very good programme and a high turnout of delegates: the number of attendees was around 250.

The conference was held at the University Arms Hotel in Cambridge. Delegates had the choice of staying in the hotel or in college accommodation very close by. The rooms provided by the hotel for the conference sessions suitable and of a good size: the only minor criticism I heard was that the rooms used for the different conference tracks were not adjacent.

There were three tracks: throughout the conference proper (Sunday 2nd to Tuesday 4th September) one room was occupied by tutorials for most of the time. These included a tutorial on System Tap and kdump by Richard Moore of IBM and tutorials on RDMA and on programming for the Cell processor.

The other two tracks took place in a large room downstairs and a rather smaller room on the first floor. As always, there were one or two occasions when one felt torn between the two tracks, but the programme was well thought out, and in general the subject matter of the two tracks was contrasting rather than clashing.

Another room was set aside for registration and exhibitors: exhibitors included O'Reilly, Sun, Novell, Google and the Linux Emporium.

I found that I spent more of my time in the larger room downstairs.

Many of the talks were impressive: particularly memorable were Jonathan Corbet's contributions ("The kernel report" and "How to work with the kernel development process"), Lars Marowsky-Brée's high availability talk, Chris Mason's presentation on his new btrfs filesystem, and Michael Meeks' talk on his IO/grind performance analysis tool.

The two talks (by Arjan van de Ven and Matthew Garrett) on Linux power consumption were also very impressive: the use of the powertop tool in tracking down applications which unnecessarily prevent the processor from going into idle mode in desktop use was demonstrated.

Dirk Hohndel of Intel spoke interestingly and persuasively on how to encourage hardware vendors to work with the Open Source community.

With that in mind, it was very good that the Kernel Summit was where the news came out that AMD will open the specifications for their ATI graphics cards.

The last two sessions of the conference proper were "How to manage patches with Git" given by James Bottomley followed by an "Advanced Git BOF session" which did not have a speaker's name shown in the programme. No-one was particularly surprised, however when Linus Torvalds stood up and ran that session.

On Wednesday 5th September, there were further tutorials (on RPM package building, chip development and writing Ruby on Rails extensions).

On the Sunday evening there was a conference dinner at Churchill College after which Dr David Hartley spoke about the history of computing in Cambridge.

On the Monday there was a trip to the Duxford air museum (sponsored by Positive Internet), where food was served before delegates split up into groups to look at historical aeroplanes.

The hotel provided very good food and refreshments (sponsored by both Intel and Google) and there were as usual many other opportunities for ad-hoc socialising.

Overall, this conference was a great success. Details and abstracts of all the talks and slides and papers for many of them are on the conference web site:
`http://www.linuxconf.eu/2007/`

LCFG workshop report

Sean McGeever

This year's UKUUG Spring Conference featured a talk from Paul Anderson on "Mass System Configuration". Interest in this aspect of system administration has grown with the increasing affordability of and requirement for large scale deployments; and also from the growing maturity of both free and commercial tools that help to manage such deployments. Paul designed one of the earliest (and now most mature) of these tools, LCFG – which has been managing a deployment of more than 1000 nodes in his own School for more than a decade. As LCFG adoption gathers pace, the pool of contributors widens, delivering LCFG support for a wider range of platforms and applications.

The University of Edinburgh held a one day tutorial workshop on System Configuration and LCFG on 13 June 2007 in Edinburgh.

This 1 day workshop targeted would-be adopters of LCFG with a practical, cost-effective "bootstrap", aimed at getting novices over the well-known obstacles to adoption: Namely, the notoriously steep learning curve; patchy and sometimes outdated documentation; and the lack of simple examples to enable and support worthwhile and efficient small-scale deployment.

As this workshop was the first of its kind on LCFG, the team were grateful to receive support and guidance from UKUUG during planning and preparation. This helped ensure that attendees from across the country experienced a successful and productive *practical* introduction to LCFG.

The workshop started with an abridged version of Paul's UKUUG talk, in which Paul described the context and rationale for large-scale system configuration tools. Paul highlighted some of the most common constraints on effective and efficient large-scale system management. In particular, *change* and *relationship* management emerged as critical factors in the success of these efforts.

Following Paul's high-level context-setting, attendees were led through a series of structured tutorials using VMPlayer images of a typical LCFG "client" (representing a Fedora Core 5 host). This approach proved both helpful and efficient, by safely and quickly enabling attendees to practice the basics of the "LCFG way" of system configuration. Each tutorial built on its predecessors, giving a logical progression from the trivial and small-scale to the more complex and large-scale.

Kenny MacDonald (Information Services, University of Edinburgh) led the first of these sessions, an introduction to LCFG Basics. This introduced most of the key parts of any LCFG deployment:

- the "component" abstraction (each component is defined by a text file containing a set of key/value pairs representing some kind of system service or resource, such as a web server);
- the LCFG compiler (which generates, from a defined set of component files, a profile for each LCFG-managed machine);
- the LCFG client/server model (which enables an LCFG server to control the installation and updating of LCFG clients by distributing generated or modified profiles).

The VMPlayer image conflated this client/server model to present a "self-hosting" LCFG client (that is, one whose LCFG server was, in fact, itself). Kenny used this image to walk attendees through the LCFG lifecycle:

- modifying a component;
- generating a new profile reflecting this change;
- automatically distributing this new profile to affected clients.

Subsequent practicals introduced one of the core LCFG components ("File", delivered by Stephen Quinney); and how to design, build and test new components. On completion of these practicals, attendees could deploy a new LCFG-managed machine and could create simple LCFG components of their own.

The success of this practical-based approach was only slightly spoiled by the lack of time for two afternoon talks from Alastair Scobie (author of the Linux port of LCFG) on advanced topics (Prescriptive Management and Site Management). However, the consensus of attendees was that the workshop was both a practical and efficient first step for new adopters, and also a convincing demonstration of the power and flexibility of the LCFG approach to "Mass System Configuration".

EuroBSDCon 2007

Sam Smith

EuroBSDCon in Copenhagen in September was an extremely enjoyable conference, with a wide variety of topics related to BSD from Europe and beyond. As ever, it was a chance for BSD developers, users and friends to gather, meet each other, and make connections. With a strong programme, and broad base of support from all the BSDs, the high promise of the conference didn't disappoint.

Highlights of EuroBSDCon included Kirk McKusick giving an overview of his 28 years' work on FFS and Robert Watson giving a rapid overview of all the novel security features in FreeBSD, many of which Robert at least worked on. Steve Murdoch from the Cambridge Computer Security Lab gave a lively and highly descriptive talk of fingerprinting hosts through clock skew, and the impacts and mitigation required by anonymisation services like Tor on which he works. This led to and involved other timing impacts from temperature across “air-gap” boundaries and environmental impacts (including the amusing aside that in a building at Cambridge, if you leave the windows open overnight, the building management programme will turn the heating on in the morning to warm the building in case the sun doesn’t come up). A very entertaining session with immediate and significant real world impact, threats and benefits.

Other sessions covered the host failover daemons in OpenBSD, ZFS and the problems that come from trying to put wireless links in remote countries across 300Km. Unfortunately, Sam Leffler was ill and unable to travel to the conference for this talk, so the organisers arranged for him to record his talk in Berkeley, and it was downloaded and played on the two large screens (with slides being displayed on the third). A novel solution to a common problem, which worked surprisingly well.

BSD Conferences normally have a “State of the Daemon” session where the projects get a few minutes to inform on what they’re doing and to cross pollinate ideas that aren’t covered in talks. Taking a different tack this year, it took the form of a Question and Answer session where the questions were asked of all the projects, and they all gave their perspective.

The big issue of the moment is the impact of GPL version 3. Briefly, the BSD projects have no idea what those impacts will be. The anti-TIVOisation clauses have significant impacts on the potential use of GPL v3 even in the ports collection, as what is meant by “distribute” is no longer clear. Packages from the ports trees (BSD consists of a set of base parts which are developed in a single CVS tree, and then a “ports collection” which is a set of make files which lets you build and package any third party software) may well be in contravention of that license; and that’s without consideration of the legally enforced security standards explicitly forbidding what the GPL v3 says you must do. As with any potential problem of this nature, there are opportunities as well as significant threats. While losing all GPL v3 software (which is certainly anything controlled by the FSF, plus additional software that chooses to switch), a comment from George Neville-Neil (part of FreeBSD core) shows the potential opportunities, in that while GPL v2 was not the preferred license for the BSDs, it was “good enough” that it wasn’t generally targeted or resources prioritised across the projects for proactive removal from the base source trees. GPL v3 will almost certainly change that. The upside of this is that there will need to be a BSD licensed compiler (pcc is now being actively worked on by OpenBSD, and there are potential replacements for all tools) and related tools which are BSD licensed. This will reduce the monoculture of compilers and the other tools where GPL’ed software is currently the only free choice. A system which doesn’t rely on GNU’s gcc specific extensions will have a wider range of compiler choices. The most obvious impact is some temporary reductions in the speed of compiled code, although applications who place a higher value on run speed will be more likely to be able to choose from the range of commercial compilers (e.g. Intel’s icc) and potentially get even better performance than from gcc. One advantage of pcc is that although it produces executables that are slightly slower in some cases than gcc, it produces those executables in around 10% of the time. This significant time saving will directly increase the productivity of programmers on BSD. These licensing changes focusing on embedded uses matter to the BSD projects as embedded devices (at this point an iPhone, which runs Mac OS X, was waved in the air) are the future of the vast majority of unix devices – and while Mac OS X and it’s BSD heritage makes it the most successful measurable UNIX, it is completely dwarfed by the embedded space where systems don’t have a keyboard or a monitor, and you have no idea what they run. Like BSD on the desktop (or the iPhone), they just work.

Decisions on inclusion of GPL v3 software within ports are going to need to be made within weeks (possibly by the time you read this they will already be public – at time of writing (Sept 15), they’re not back from the lawyers). It is extremely unfortunate that both the FreeBSD and NetBSD Foundations are having to spend non-trivial sums of money on lawyers to get this advice rather than spending their resources on more directly supporting the projects’ aims. A rear guard action of squabbling while covering their legal backsides from legal attacks from people who profess to have the same goals, benefits only the lawyers. Previously, the projects might have gone to the Software Freedom Law Centre, run by Eben Moglen, although their recent opinion (in the case of two solely BSD-licensed files in the Atheros driver) suggesting that it is legally permissible to take a BSD licensed piece of code, and wrap a GPL round it without the changes required to make it a derivative work, puts a very significant hole in any claim of impartiality towards GPLv3. At the time of writing, the alleged thieves are refusing to back down, and a formal lawsuit is possible, although not necessarily soon, as the author whose code was relicensed without permission has until 2047 to file it. While the Atheros driver is developed by a number of people often under dual-licensing regimes in single files, the two files in question here were written by one person, who very repeatedly over a number of years as he was asked on sometimes a weekly basis by individuals who didn't like the previous answer, always stated that he was not interested in dual licensing his work – it was BSD only (as is the OpenBSD way).

While this licensing disagreement over two files in one driver may seem trivial, it threatens the heart of all the BSDs and all BSD licensed projects. If it is permitted for Linux or other GPL projects to take a piece of BSD code and simply relicense it under the GPL without it meeting the legal definition of a derivative work, then all BSD licensed code can go down that street and be relicensed. However, that road is one way – those changes, improvements and fixes can not be legally fed back in to the original projects. This potentially makes impossible any bilateral sharing between open source projects; but has the Linux/GPL developers saying thank you for your work and offering a single finger handshake as "thanks". Even if that was legally permissible through a novel reading of the law and licenses, it could be considered more than a little morally dubious. While there has been lively debate over the best license to reach our similar aims, until now, the choices made by an individual for their own code have always been respected.

Given the concerns over GPL v3, and this issue, licensing is currently getting a significant amount of attention, and while the BSD projects may not like the GPL v3 licenses, the embedded market (which FSF explicitly targeted, the BSD projects are just in the cross fire) pro-actively can not use it. This may bring in significant amount of potential new resources at a time when they're needed. One concern expressed in the room is that there isn't enough new blood to do the work, but there is historic BSD code which can be cleaned up, updated and enhanced, that may provide a very significant starting point. If you doubt this approach is feasible, there's a little project called OpenSSH which followed this route. You may have heard of it. The collaboration and success of OpenSSH is a testament of what a small group of people can do when motivated. For more on this developing situation (which has changed significantly in the 24 hours since I returned from EuroBSDCon). See the OpenBSD Journal at
`http://www.undeadly.org/`

With three annual regional BSD Conferences now very well established (Europe, Asia and BSDCan for Canada/USA), there's significant cross-pollination of ideas around the globe. This year's conference was extremely well organised by EurOpen/DKUUG, and BSD-DK, with special thanks to Paul Henning-Kamp and Sidsel Jensen for all their work. With 2008's conference being in Strasbourg, the 2009 conference will be returning to the UK and Cambridge in 2009, led by Robert Watson and myself. While the question "are you mad?" was asked more than once, we look forward to the next 2 years. UKUUG's involvement will be substantial, and there will be many opportunities for you to get involved and help if you're interested.

If you are interested either specifically or generically please contact me on
`sams@ukuug.org`
The interim group organising the event consists of Robert Watson (FreeBSD code and the University of Cambridge), George Neville-Neil (AsiaBSDCon) and myself who can be contacted on
`eurobsdcon@ukuug.org`

In Memoriam: John W Backus 1924–2007

Alex Aiken

Computing has been such an integral part of everyday life for so long that many people alive today don't remember a time when it wasn't so. In the 1950s, digital computers were only a few years old, and very little of what we take for granted about modern computers had been invented. One thing was already apparent, however: the new, powerful hardware needed equally powerful software if it was to be of use to anyone, and software was turning out to be surprisingly difficult, timeconsuming, and expensive to write.

A big part of the problem was that software at the time was written directly in the machine's native assembly language. As anyone who has ever written in assembly knows, it takes a lot of error-prone coding to get anything done that way.

John Backus, who was working at IBM at the time, had the idea that programming could be greatly improved if the programmer could write more abstract and less machine-specific commands and if the program could be organised in a way that seemed more natural for humans. A special program called a compiler would take care of translating from the higher-level language into the machine's language. The obvious problem with this idea, a difficulty that led some to predict it would fail, was the concern that the translation would end up being much less efficient than a carefully written assembly program; since machines were very expensive, computer time was a valuable commodity.

But John was literally a visionary; he had a vision of what could be. He led a team that spent three years (1954–1957) designing and implementing the programming language and its compiler.

Within a few months of its first release, FORTRAN (for FORmula TRANslation) was a huge success for IBM, dramatically improving software productivity. FORTRAN was an even greater intellectual triumph, for it changed forever the way people thought about programming computers and led directly to a surge of interest in the design and implementation of high-level computer languages.

John was also a prototype of the modern computer science research manager, an oxymoron if there ever was one. John always claimed that his contribution to the FORTRAN project was only breaking up the chess games after lunch, that really he hadn't done anything himself.

His characteristic modesty aside, his management style was one of almost no management at all, taking only the role of articulating the vision of where the project should go, both to inspire those working with him and to convince his backers to continue funding the project.

The key was to let the project be guided by the technical constraints of the problem they wanted to solve, which was poorly understood at the time, and not by a manager's early guess at what the solution should be. It's a great trick, but really quite difficult to pull off, since projects of FORTRAN's ambition succeed only after many failures. It requires both great technical ability and a gift for connecting with people to keep an effort like that on track, but John made it look easy.

After FORTRAN, John went on to invent BNF (or Backus-Naur Form), the notation used universally today to describe the syntax of programming languages. He also was a major force behind the early development of functional languages, championing even higher-level programming as necessary to move beyond languages of the FORTRAN era.

For his work he won the Turing Award and the Draper Prize, among many other awards, but he never stopped being a researcher, always interested in new ideas and ways to make programming easier and more useful.

*Originally published in **;login: The USENIX Magazine,** 32, #3 (Berkeley, CA: USENIX Association, 2007). Reprinted by permission.*

Help everybody love Free standards and Free Software!

Marco Fioretti

Free as in Freedom file formats, computer protocols and software are an excellent idea at many levels: ethical, moral, educational, economical and so on. I am convinced of this. Whenever I can, I only use Free Software and write articles explaining how to use and customise it. I also believe that mainstream usage of such technologies is an essential component of a Free and advanced society.

At the same time, however, I am worried every year more that the traditional way to explain, justify and advocate Free as in Freedom software has reached a communication plateau or, if you will, some built-in limit.

There is nothing wrong in the Gnu Manifesto or in almost all practical proposals of Richard Stallman, and of the Free Software Foundation, both of whom I greatly respect. I find, however, that their language and arguments are as intrinsically right as not applicable nor relevant to the great majority of computer users of today.

Aren't Free Digital standards and software a matter of ethics, civil rights and equal opportunities? Sure. Nevertheless, talking of how beautiful it is to study, improve and share source code was enough when almost every computer user was an ICT student or professional.

Today 95 out of 100 computer users do little more than play games, file holiday pictures or (because they are **forced** to do so) write office or school reports. There are very little things that these people find more boring than programming or having source code available, and there is nothing changeable, or wrong, in this.

Everybody who could be convinced with just the GNU Manifesto has already read it. This, however, is an age where we also need as soon as possible the votes (in the booth or as consumers) of all those other people to make Free, open standards and software a basic, mainstream part of a more free society. But we'll never convince them by assuming that they, too, should program, file bug reports or write documentation.

These thoughts are what led me to write pieces like "The seven things we're tired of hearing from software hackers" or "A Free Software Manifesto for all of us", both available in the Opinion section of Digifreedom.net. Those were just starting points, of course. The solution I suggest, and the main subject of this article, is another.

The Family Guide to Digital Freedom

I consider the recent campaign of the Free Software Foundation to reach out social activists as a proof that they share these concerns of mine and, generally, a very worthy initiative. I see two limits in it, however.

One is that software, Free or not, is just one of the many components, closely linked in ways still ignored by the general public, of a better world. Mandating the usage of Gnu/Linux in schools, for example, would do nothing to fix the current abuses of the copyright system. Making OpenDocument-compatible applications the only ones acceptable for Public Administrations is essential, but if certain documents aren't published online, how can citizens effectively control if everything is right?

The other limit of the FSF campaign is the definition of "social activists": from what I've seen so far, it seems limited to "the ICT staff or maybe also the employees, of formally organised groups, big enough to have their own full time ICT infrastructure". Again, nothing wrong with this, and the FSF can do it much better than me: still, this definition leaves out what is the largest and most ubiquitous category of social activists of all times, everywhere: parents.

This is why, in the last year, I have published a book and started a website to complement the current efforts. The Family Guide to Digital Freedom is not another Linux Guide for beginners, but something pretty different: it is a book that gives all parents (and teachers) the real reasons why they should start to care about Linux and many other things which I call the Digital Dangers: from copyright to biometrics, e-voting, software induced pollution, RFID, Net Neutrality, DRM, online activism and much more!

In the Guide, things like source code, licenses and similar are deliberately mentioned almost by accident. The book, which counts just above 200 pages, is divided in 50 very short chapters, which prove, through very practical examples, how all the issues above are interrelated and how ignoring them directly hurts the budget, civil rights, education opportunities and entertainment possibilities of every family.

The problem that we FOSS activists have today is not to teach how to use Gnu/Linux: is how to make people understand why they should ever bother to listen to such talks in the first place. This is why I have written the Guide and why, by the way, it is possible to order copies with a custom back cover which promotes your LUG or pro-FOSS activity.

The Guide is not a book for FOSS activists: I have done my best to write the perfect gift that every FOSS activist could give to non-geeks parents, partners, friends, room-mates, coworkers and so on, to make them finally understand why we consider a weekend spent promoting Free Software a worthwhile activity.

The other half of this project, that is the companion of the Family Guide to Digital Freedom, is the Digifreedom.net website: I am building it to be the first place where all parents and teachers, starting from those who still hope that computers had never been invented, can come to figure out why it is bad to ignore the existence of DRM, Free Software and all the Digital Dangers.

Besides excerpts of the book, the visitors of Digifreedom.net will be able to meet other parents and teachers who want to fight those dangers or just share related experience. Above all, visitors will find a lot of useful news and resources to better understand these problems and fight them.

Right now, such resources include a database of Digitally Free Schools, one of Trashware (associations which refurbish old computers with Free Software for families, schools and non-profits) and other miscellaneous links to relevant websites. Over time I plan to add at least a database of Bad ICT information, a series of flyers temporarily called “The Fridge copyright violation cheatsheet” and very short tutorials (1 or 2 pages maximum) on several arguments.

The first of those tutorial will probably be “How to try Free Software and migrate to it without getting hurt”. A possible outline, just to give an idea of what the Digifreedom approach is meant to be, is the following:

- Start using Free Software for Windows.
- Once you’re comfortable with it, choose the best version of Gnu/Linux, that is anyone, as long its online user community is tolerant and friendly with beginners.
- Try a live CD of that distribution, without installing anything.
- If, and only if, you like what you see, install that distribution: Otherwise, stick to Windows, but please never ever send proprietary attachments in email or create web pages unusable with Gnu/Linux.

The Bad ICT information pages will try to explain how to recognise incompetent reporting in mainstream press. They will link to articles which give misleading or plain wrong information about FOSS, P2P and so on, briefly explaining what’s wrong with them. An article titled “Hackers attacked public servers and made them crash”, for example, will earn a caption explaining the difference between hackers and crackers. One boasting that “File sharing is illegal”, instead, will get a “only if the file license forbids it” note attached.

The Fridge cheatsheets will be lists of short answers and questions, one for each country, where a FOSS activist of that country explains what is legal and what not with digital content and where to complain, something like:

Q: Can I make a backup copy of my own legally purchased CDs?

A: Not in this country, if the law proposal number XYZ is passed

Q: If I don’t like the law, how can I stop it from being approved?

A: Write to you Parliament Representative or sign the petition at www.somewhere.org

Even in this case, the purpose is to help people to finally **see** what is happening and make informed decisions about it.

I always welcome feedback about the book, the website and proposals for future initiatives. Please let me know what you think at
`marco@digifreedom.net`

Google’s Position on OOXML as a Proposed ISO Standard

Introduction

Google is concerned about the potential adoption of Microsoft’s Office Open XML (OOXML) format as an ISO standard. Google supports open standards and the Open Document Format (ODF), an existing ISO standard that has been a driver for innovation. We do not think it is beneficial to introduce an alternative standard when the Open Document Format already meets the common definitions of an open standard, has received ISO approval and is in wide use around the world. Google’s concerns about OOXML include, but are not limited to:

- The limitations on the openness of OOXML format;
- The lack of proper review as compared to other ISO standards;
- The continued use of binary code tied to platform-specific features; and
- Unclear licensing terms for third-party implementers.

The following is a Q&A to help clarify Google's position on the ISO standardisation of OOXML.

Aren't multiple document standards good? We have PDF and HTML, so why not ODF and OOXML?

Multiple standards are good, but only if they are designed to address different problems. HTML is a very simple mark-up language designed for rendering within browsers, while PDF is a display-only format designed for high-fidelity print output. ODF and OOXML are both designed as a format for editable documents. As such they both address the same problem and almost completely overlap. The current state of file formats for editable documents makes life very difficult for consumers and vendors of office productivity software, and is a looming disaster for long-term document storage. Having two mutually incompatible formats for editable documents will allow the current non-interoperable state of affairs to continue.

Microsoft has been arguing that OOXML is a good thing as it gives vendors and customers choice. Multiple incompatible standards are a bad thing for customer choice, as purchasers of Betamax video recorders discovered to their cost. Multiple implementations of a single standard are good for both the industry and for customers.

If Microsoft wishes to create a document format that is better able to address the problems of the many editable legacy documents created in their older proprietary formats Google welcomes them to help extend the existing ODF ISO standard, in order to add the capabilities they require. Allowing OOXML to become a parallel ISO standard will propagate the current legacy situation into what is supposed to be a solution to the problems of long-term document storage.

OOXML is a perfectly good ISO standard. Isn't this just complaining by other vendors?

In developing standards, as in other engineering processes, it is a bad idea to reinvent the wheel. The OOXML standard document is 6546 pages long. The ODF standard, which achieves the same goal, is only 867 pages. The reason for this is that ODF references other existing ISO standards for such things as date specifications, math formula markup and many other needs of an office document format standard. OOXML invents its own versions of these existing standards, which is unnecessary and complicates the final standard.

If ISO were to give OOXML with its 6546 pages the same level of review that other standards have seen, it would take 18 years (6576 days for 6546 pages) to achieve comparable levels of review to the existing ODF standard (871 days for 867 pages) which achieves the same purpose and is thus a good comparison.

Considering that OOXML has only received about 5.5% of the review that comparable standards have undergone, reports about inconsistencies, contradictions and missing information are hardly surprising.

Isn't this standard needed to support the millions of existing Microsoft Office documents?

OOXML is a brand new format, different from the existing .DOC, .XLS and .PPT formats that are widely used by Microsoft Office. In order to move to an XML-based format these documents will have to be translated anyway. There is no wide use of OOXML format documents on the Web. Counting the number of documents found by doing Web searches for different document types the older Microsoft Office formats dominate, but the second most widely used format is the existing ISO standard ODF. As translation is needed anyway it would make more sense to convert to ODF, the existing ISO standard for editable document types.

In addition, if OOXML were necessary to faithfully convert these legacy documents to an XML format, it would have to contain the complete specification of these older document formats. Without this OOXML would be incomplete in its descriptions for an ISO standard. No specifications for older document formats exist in the OOXML descriptions, and so any argument that OOXML is needed for their accurate translation is false. Such legacy documents may just as easily be translated to ODF (as can be seen in the way some existing ODF implementations handle the import of the legacy Microsoft Office file formats).

Doesn't OOXML already have wide industry adoption?

Many companies have announced they will support OOXML, and several have announced translators for the new formats. This is only to be expected, as Microsoft is a major vendor in the office automation space. Wide industry support doesn't necessarily make a good ISO standard, although it definitely helps. What matters more for a good interoperable standard is multiple implementations. On this score ODF is very well served, with around twelve different implementations of software that can read and write ODF files. Most of the OOXML implementations are from partners of Microsoft who have contractual agreements to implement OOXML software.

Multiple independent implementations help a standard mature quicker and become more useful to its users. It fosters a range of software choices under different licensing models that allow products to be created and chosen whilst still faithfully adhering to the ISO standard.

Isn't OOXML safe to implement by anyone?

NB. This section is not legal advice from Google. For a full analysis of the OOXML licensing conditions, please consult a lawyer.

Microsoft has offered an Open Specification Promise covering OOXML which they claim would cover third party implementations of the standard. See
`http://www.microsoft.com/interop/osp/default.mspx`

There is considerable legal uncertainty around the scope of this promise, which appears only to cover the exact version of the specification currently published, but not any future revisions or enhancements. The legal uncertainty surrounding the scope of this license grant weighs heavily against the propriety of ISO acceptance of the OOXML standard. The existing ODF ISO standard is covered by Sun's "OpenDocument Patent Statement", which does not suffer from these issues. See
`http://www.oasis-open.org/committees/office/ipr.php`

This description of Google's position on the proposed OOXML standard is reprinted by permission.

---

ext4: the next generation of the ext3 file system

Avantika Mathur, Mingming Cao and Andreas Dilger

Last year, a new Linux file system was born: ext4. A descendant of the ext3 file system, ext4 will soon replace ext3 as the "Linux file system". Ext4 provides greater scalability and higher performance than ext3, while maintaining reliability and stability, giving users many reasons to switch to this new file system. The primary goal for ext4 is to support larger files and file systems. Once it is mature, the developing ext4 file system will be suitable for a greater variety of workloads, from desktop to enterprise solutions.

Why Ext4

Among the many file systems Linux offers today, ext3 is the most popular, with the largest user base and development community. Having been designed with stability and maintenance in mind makes ext3 a very reliable file system with relatively good performance. Thus it has been the default file system in many commercial Linux distributions for many years.

However, the conservative design of ext3 limits its scalability and performance. One of the hard limits faced by ext3 today is the 16-TB filesystem size maximum. This limit has already been reached in large installations and will soon be hit by desktop users. Today, 1-TB external hard drives are readily available in stores, and disk capacity can double every year.

Last year, a series of patches were sent to the Linux kernel mailing list, to address filesystem capacity and to add extents mapping to ext3. The extent patches would cause intrusive changes to the on-disk format and break forward compatibility for file systems that used them. In order to maintain the stable ext3 file system for its massive user base, it was decided to fork the ext4 file system from ext3 and address performance and scalability issues in this new file system.

Why not use a file system such as XFS for this? The answer is that the XFS code in Linux is very complex, being burdened with an extra layer of compatibility code for IRIX. Even with the addition of the features described here, ext4 is still much smaller and more understandable than XFS (25k lines vs. 106k lines). Also, there is a considerable investment in the stability and robustness of the ext3 and e2fsck code base, most of which will continue to be used for ext4.

What's New in Ext4

Ext4 was included in mainline Linux version 2.6.19. The file system is currently in development mode, titled ext4dev, explicitly warning users that it is not ready for production use. There are many new features in the ext4 road map under development and testing. Some of the features in progress may continue to change the filesystem layout. Any future changes, as with the current ones, will be protected by appropriate feature flags so that existing kernels and `e2fsprogs` will be able to know whether it is safe to mount a given ext4 file system. Once the layout is finalised, ext4 will be converted from development to stable mode. At that point, ext4 will be stable and available for general use by all users in need of a more scalable and modern version of ext3.

The initial version of the ext4 file system includes two new key features: extent support and 48-bit block numbers. Combined, these features support larger file systems and better performance on large files.

Extent support

The ext3 file system uses the traditional indirect block mapping scheme, which is efficient for small or sparse files but causes high metadata overhead and poor performance when dealing with large files, especially on delete and truncate operations. To address this issue, ext4 uses extent mapping as an efficient way to represent large contiguous files.

An extent is a single descriptor that represents a range of contiguous blocks. A single extent in ext4 can represent up to 128 MB. Four extents can be stored directly in the inode structure. That is generally sufficient for small to medium contiguous files, but for very large or highly fragmented files, an extents tree is created to efficiently look up the many extents required.

Extents mapping improves performance on large files, such as mp3, DVD, video, or database files, as it is tuned toward allocating large contiguous blocks. Extents bring about a 25% throughput gain in large sequential I/O workloads when compared with ext3. A similar performance gain was also seen on the Postmark benchmark, which simulates a mail server, with a large number of small to medium files. With extents the metadata is more compact, causing greatly reduced CPU usage.

Although the indirect block and extents mapping schemes are incompatible, both are supported by ext4, and files can be converted between the two formats. This is discussed further in the migration section.

Large file system support

The first issue addressed in ext4 is the filesystem capacity limit. Because ext3 uses 32 bits to represent block numbers and has a default 4k block size, the file system is limited to a maximum of 16 TB. Ext4 uses 48-bit block numbers. In theory this allows it to support 1-EB (1-million-TB) file systems. This change was made in combination with the extents patches, which use 48-bit physical block numbers in the extents structure. Other metadata changes, such as in the super-block structure, were also made to support the 48-bit block number.

In order to support more than 32-bit block numbers in the journaling block layer (JBD), JBD2 was forked from JBD at the same time that ext4 was cloned. There is also work underway to add checksumming to the journal in JBD2 to validate this critical metadata at recovery time. Currently, JBD2 is only used by ext4, but eventually both 32-bit and 64-bit Linux file systems will be able to use JBD2 for journaling support.

One may question why we chose 48-bit block numbers rather than the round 64 bits. Although it is possible to design for 64-bit ext4 file systems, this is impractical today because of reliability and serviceability problems. Having full 64-bit physical and logical block numbers would have meant that fewer extents could fit within the inode (2 vs. 4) for only very theoretical gains. It would take 119 years at today's speeds to run e2fsck on even a 248-block file system. The ext4 developers will focus their efforts on improving the reliability aspects before worrying about a theoretical size limit.

What's Next

The increased file system capacity created by the 48-bit block numbers and extent mapping features that are currently in ext4 will provide many new features in the road map. These features are focused on enhancing ext4 scalability, reliability, block placement, and performance.

An ext4 git tree is hosted at
`git://git.kernel.org/pub/scm/linux/kernel/git/tytso/ext4` The tree contains the series of patches in line for ext4. Up to date information on new ext4 features, patch sets, and development discussion can be found at the ext4 wiki page,
`http://ext4.wiki.kernel.org/`

Block allocation enhancements

Fragmentation is a key factor affecting filesystem performance. The following features in the ext4 road map attempt to address performance by avoiding or decreasing fragmentation. This is essential for the efficient use of extents and maximising performance on modern high-bandwidth storage.

Persistent preallocation

Applications such as large databases often write zeros to a file for guaranteed and contiguous filesystem space reservation. Persistent preallocation in ext4 allocates a contiguous set of blocks for a file without the expensive zero-out. The preallocated extents contain a flag specifying that the blocks are uninitialised. These uninitialised extents are protected from undesired exposure of their contents through read operations. A new system call, `sys_fallocate`, is planned to be added to the Linux kernel, and the existing `posix_fallocate` library is being modified. These interfaces can be used by users to specify the portion of the file to preallocate.

Delayed allocation and multiple block allocation

The ext3 block allocation scheme is not very efficient for allocating extents, as blocks are allocated one by one during write operations, so ext4 will use delayed allocation and multiple block mechanisms to efficiently allocate many blocks at a time and contribute to avoiding fragmentation. With delayed allocation, block allocations are deferred to page flush time, rather than during the write operation. This avoids unnecessary block allocation for short-lived files and provides the opportunity to queue many individual block allocation requests into a single request.

The multiple block allocation feature uses a buddy data structure to efficiently locate free extents and allocates an entire extent referencing multiple blocks, rather than allocating one at a time. Combined, delayed allocation and multiple block allocation have been shown to significantly reduce CPU usage and improve throughput on large I/O. Performance testing shows a 30% throughput gain for large sequential writes and 7-10% improvement on small files (such as those seen in mail-server-type work-loads).

Online defragmentation

Even though there are techniques in place to attempt to avoid file fragmentation, with age, a file system can still become highly fragmented. The online defragmentation tool is designed to defragment individual files or an entire file system. The defragmentation is performed by creating a temporary inode, using multiple block allocation to allocate contiguous blocks to the inode, reading all data from the original file to the page cache, then flushing the data to disk and migrating the newly allocated blocks over to the original inode.

Scalability enhancements

Besides enlarging the overall file system capacity, there are many other scalability features planned for ext4.

Although ext3 has support for different inode sizes, the default inode structure size is 128 bytes, with little extra room to support new features. In ext4, the default inode size will be enlarged to 256 bytes. This will provide space for the new fields needed for the planned features, nanosecond time stamps, and inode versioning. The latter is a counter incremented on each file modification that will be used by NFSv4 (network file system) to keep track of file updates.

By using a larger inode size in ext4, the EA-in-inode feature can be enabled by default. This feature is also available in ext3, but because of the smaller default inode size it is not widely used. EA-in-inode stores extended attributes (EAs) directly in the inode body, making EA access much faster. The faster EA access can greatly improve performance, sometimes by 2-3 times, for those using SELinux, ACLs, or other EAs. Additional EAs, which don't fit in the inode body, are stored in a single filesystem block. This limits the maximum EA capacity per file to 4k. In ext4, there is a plan to remove this limit by storing large EAs in a file.

To address directory scalability, the directory indexing feature, available in ext3, will be turned on by default in ext4. Directory indexing uses a specialized Btree-like structure to store directory entries, rather than a linked list with linear access times. This significantly improves performance on certain applications with very large directories, such as Web caches and mail systems using the Maildir format. Performance improvements were often by factors of 50-100, in particular for directories with more than 10,000 files. An additional scalability improvement is eliminating the 32,000 subdirectory limit in ext4.

Support for larger files, extending beyond the 2-TB limit, are in the road map. There is interest in efficiently supporting large numbers of files, surpassing the 4-billion limit, which implies 64-bit inode numbers and dynamic inode tables. Because of the potential intrusive on-disk format changes involved, plans and design aspects are still on the drawing board.

Faster repair and recovery

A file system is not very useful if it cannot be repaired or recovered in a reasonable amount of time. As the filesystem size grows, the e2fsck time becomes unbearable, taking from minutes to years depending on the filesystem size. This issue is more prevalent for ext4, as it can support larger file systems. Although the ext4 journaling support tries to avoid the need for file system recovery as much as possible, in the event of a disk error e2fsck is still necessary.

The e2fsck tool scans the whole file system, regardless of whether only a small part of it is being used. With a little more information about the file system, e2fsck could skip those unused block groups or inode structures. The uninitialised block groups feature uses a flag in the block group to mark whether it is uninitialised. Once it is written to, watermarks are used to keep track of the last used inode structures. Any uninitialised block groups or inodes are not scanned by e2fsck, which greatly reduces the run time. This feature can speed up e2fsck dramatically, reducing run times to 1/2 to 1/10 of the original run time, depending on how the file system is used. The flags marking a block group uninitialised and the high watermark are checksummed. If there is ever corruption, e2fsck will fall back to the slower, but safer, full scan for that block group. As new features are added to ext4, the user tools, e2fsprogs, will be updated correspondingly.

Migration from Ext3 to Ext4

Compatibility between ext4 and existing ext3 file systems has been maintained as much as possible. Even though ext4 has changed some parts of the on-disk format, it is possible to take advantage of many of the ext4 features, such as extents, delayed allocation, multiple block allocation, and faster e2fsck, without requiring a backup and restore.

There is an upgrade path from ext3 that allows existing file systems to start using ext4 immediately, without requiring a lengthy downtime. Users can mount an existing ext3 file system as ext4, and all existing files are still treated as ext3 files. Any new files created in this ext4 file system will be extent-mapping-based. There is a flag in each individual file to indicate whether it is an ext3 file (indirect mapped) or ext4 file (extent mapped).

There is also a way to do a systemwide migration. A set of tools is under development to migrate an entire file system from ext3 to ext4, which includes transferring indirect files to extent files and enlarging the inode structure to 256 bytes. The first part can be performed online in combination with online defragmentation. This tool will perform the conversion from ext3 to ext4 while simultaneously defragmenting the file system and files, and it will have the option of converting only part of the file system. Resizing the inodes must be performed offline, and this can be done in conjunction with converting files to extent mapping. In this case the whole file system is scanned, and proper backup is done to prevent data loss if the system crashes during the migration.

Users who are hesitant to migrate to ext4 immediately can optionally format their ext3 file system with large inodes (256 bytes or more) to take advantage of the EA-in-inode feature today and nanosecond timestamps if they migrate to ext4. This would avoid the need to do an offline migration step to resize the inodes.

There is also a downgrade path from ext4 to ext3, with a method to convert the extent files back to indirect mapping files. In the case that users prefer to go back to ext3, they can mount the ext4 file system with the `noextents` mount option, copy the extent-based ext4 files to new files, rename these over the old extents, use `tunefs` to clear the `INCOMPAT_EXTENTS` flag, and then remount as an ext3 file system.

Conclusion

As we have discussed, a tremendous amount of work has gone into the ext4 file system, and this work is ongoing. As we stabilise the file system, ext4 will become suitable for production use, making it a good choice for a wide variety of workloads. What was once essentially a simple file system has turned into an enterprise-ready modern file system, with a good balance of scalability, reliability, performance, and stability. Eventually ext4 will replace ext3 as the default Linux file system.

*Originally published in **;login: The USENIX Magazine,** 32, #3 (Berkeley, CA: USENIX Association, 2007). Reprinted by permission.*

Innovation

Peter H Salus

Brief judicial note: On 10 August 2007, Judge Dale Kimball ruled that "the court concludes that Novell is the owner of the UNIX and UnixWare copyrights" (i.e. not the SCO Group). This effectively blows the entire SCO v Novell case out of the water (though there are a few bits still wriggling) and guts the SCO v IBM case. Unfortunately, I am certain that appeals will be forthcoming, as neither the officers of the now-bankrupt company nor the legal team are eager to face the various suits (from stockholders where both groups are concerned or from the officers where the bar is concerned). [I am writing this in mid-August. Stay tuned for more legal flailing. (Oops! I meant "legal filing".) – PHS]

For 300 years, from the Statute of Anne on, copyrights and patents have intended to assure that the creative endeavours of individuals are protected.

In *The Economist* for 4 August 2007, I read:

[I]nnovation and the knowledge-based economy are all the rage. Since June innovation has been enshrined, along with universities and skills, in the formal title of a ministerial department.

It started me thinking.

On a purely etymological level, to innovate is to bring in something new. Innovation, then, is the act of bringing in something new. While *innovare* occurs in Latin, it only comes into use in English in the 16th Century.

Though employed, it was rare. H.L. Mencken (*The American Language*, 1936) thinks it still "on probation", not fully a part of the language.

Over the past decade or so, the word “innovation” has been employed ever more frequently, but with no gain in clarity. In fact, I’m not quite certain any more what the word means and what it refers to.

David Katz wrote me

> *I distinguish between invention and innovation as follows:*
>
> *Invention is the creation of something new, either a thing or an idea that did not exist before, or a new arrangement of previously existing elements that serves a new purpose. Innovation is the incremental improvement of an invention. Following these principles, an invention is deserving of (limited) intellectual property protections, while innovations are not.*
>
> *Writing a song is an invention; creating a new arrangement of the song is an innovation. Writing a novel is an invention; re-editing it or reissuing it in a different format is an innovation. Lotus 1-2-3 was an invention; Excel is an innovation.*

Henry William Chesborough (*Open Innovation: The New Imperative for Creating and Profiting from Technology*, 2005), writes about running R&D organisations in what he thinks of as a more open way. That is, balancing internal R&D with the acquisition of the results of external (= others’) R&D. He seems to think, for instance, that IBM invented “open innovation” with the beginning of the Internet in the 1990s. Chesborough also seems to think that PARC was an R&D failure.

A decade earlier, Eric von Hippel’s *The Sources of Innovation* (1994) seems to assert that innovation is “process innovation” and that it is something managed by manufacturers. Innovation, he argues, will take place where there is greatest economic benefit to the innovator. (As I’ll point out later, von Hippel’s point of view has changed over the years.)

I’ll get to several recent books on innovation later in this essay, but it’s important to understand that whereas Francis Bacon (1561-1626) seems to believe that innovations just come about – “As the births of living creatures at first are ill-shapen, so are all innovations, which are the births of time.” – Charles Babbage (1791-1871) believed that innovation was the implementation of invention. [Yes, *that* Charles Babbage.] And Norbert Wiener (1894-1964) believed that creativity and invention were the spurs to innovation. (Clearly Babbage and Wiener would agree with Katz.)

On a more contemporary level, we have Steve Jobs: “Innovation distinguishes between a leader and a follower” and Woody Allen: “If you’re not failing every now and again, it’s a sign you’re not doing anything very innovative.”

But neither Steve nor Woody helps us as to what innovation is. Might Bill Gates do better? On 3 March 1998 he said: “Innovation depends on freedom to move constantly from one frontier to the next”, to the US Senate Judiciary Committee. And two years later (7 May 2000) he declared:

> *The symbiotic nature of software development may not be obvious outside the industry, but it is a phenomenon that has produced enormous consumer benefits. Windows and Office – working together and drawing on each other’s features and innovations – have improved personal computing for millions.*

Of course, we may have a few problems with this, for on 1 March of this year, the EU issued a statement

> *In acknowledging the work of its designated trustee, Dr. Neil Barrett, the EU said it examined 160 Microsoft claims to patented technologies, and concluded that among those, only four may deserve to claim “a limited degree of innovation.”*

Ah, well. But Gates’ successor may help us here.

On 27 July 2007, CNET reported Steve Ballmer, the President of Microsoft, as saying “We are hell-bent and determined to allocate the talent, the resources, the money, the innovation to absolutely become a powerhouse in the ad business." I am certain this was intended to mean something. But I have no idea what devoting innovation "in the ad business" might involve.

Recently, the US Supreme Court (in 550 US 04-1350 KSR Int'l Co. v Teleflex Inc.; the decision is dated April 30, 2007) that:

> *We build and create by bringing to the tangible and palpable reality around us new works based on instinct, simple logic, ordinary inferences, extraordinary ideas, and sometimes even genius. These advances, once part of our shared knowledge, define a new threshold from which innovation starts once more. And as progress beginning from higher levels of achievement is expected in the normal course, the results of ordinary innovation are not the subject of exclusive rights under the patent laws.*

In this, I think the US Supreme Court reveals itself to be closer to Bacon and Wiener than to the more mercantile authors/speakers.

Though I realise that this piece is far more laden with citations that most of my writings, I'm hoping that I've not driven every reader away. For I have a handful of recent books I want to mention – all with "innovat-" in their titles. And some are quite good, though they may not have much "innovation" in their contents.

Though I admire much of his work, Clayton M. Christensen is a prime employer of today's vocabulary item. In 1997, his *The Innovator's Dilemma* appeared; in 2003, he was co-author of *The Innovator's Solution*; in 2005, he wrote the foreword to *Fast Innovation*.

The Innovator's Dilemma is about corporate adaptation to "disruptive technology". (Though he employs the term quite frequently, Christensen nowhere credits Joseph Schumpeter's creation of "creative destruction" in 1942 - the process of transformation that accompanies innovation.) *The Innovator's Solution* concerns "creating and sustaining successful growth". I've not read *Fast Innovation*.

2004 brought me *Innovation and its Discontents* by Adam B. Jaffe and Josh Lerner, a fascinating and insightful examination of the patent process and just how "broken" it is. Jaffe and Lerner believe that the patent system is what made American industry great in the 19th and early 20th centuries. They are right. But, unfortunately they confine themselves to the US patent system and ignore copyright as well as most of the world outside the USA. (They do mention Clinton's abandonment of WIPO when opposed by the patent bar.) But I remain unconvinced that the "system" endangers "innovation and progress".

In 2005, Eric von Hippel brought out *Democratizing Innovation*, a truly interesting brief look at the effects of the free- and open-software movements as well as some "physical products". I think that von Hippel is right as to the importance of what he calls "user innovation". But I'd prefer to think of it as "user development".

And this (finally, you sigh) brings me back to my first point: we over-use "innovator", "innovation", and "innovative" beyond reasonable bounds. A too-frequently handled coin becomes worn and loses its features. So too the words of the language.

But *The Myths of Innovation* by Scott Berkum (2007) employs innovation in its title and frequently in its text, and renders the word nearly featureless. But Berkum's "myths" are of a substance nearer to jelly than to a solid.

Berkum devotes time and energy to debunking the myth of Newton and the apple. He might have glanced at E.N. da C. Andrade's *Sir Isaac Newton*, first published in 1954. Or he might have written of Canute and the tide. Or Washington and the apple tree. But he (entertainingly) sets up his straw men and knocks them down. (It's interesting to note that while Berkum documents the "true" story of the founding of eBay, he fobs off the reader with "the tale of Newton's apple owes its mythic status to the journalists of the day", with no citation nor reference.)

Berkum's other "straw men" include Franklin, Whitney, Fulton, Edison, Ford, Carnegie, and Steve Jobs (after Newton, non-Americans were barred from this game).

I attribute much of Berkum's attitude and exposition to his past employment: from 1994-1999 he "was a member of the Internet Explorer team at Microsoft". Certainly a great place to acquire "myths of innovation".

For I don't really think that "innovation" is more than a vocabulary item to Microsoft's executives. Over a period of over 20 years, their business model has been one of theft and purchase wherever feasible and of "embrace, extend, extinguish" elsewhere. I didn't originate this. In a brilliant *New York Times* article John Markoff wrote, "Rather than merely embrace and extend the Internet, the company's critics now fear, Microsoft intends to engulf it". (16 July 1996)

Remember, Bill Gates told us that "Windows and Office – working together and drawing on each other's features and innovations – have improved personal computing for millions". and Dr. Neil Barrett, who examined nearly 200 of Microsoft's claims, found but four that might reasonably contain some innovation.

In the recent past, the USPTO (following the overturning of several patents by the courts) has revoked several patents. (The Public Patent Foundation has been prominent in trying to limit abuse of the patent system – both where computing is concerned [e.g. the Microsoft FAT patent, which was rejected in 2004] and in other areas [e.g. the Pfizer Lipitor patent, rejected in 2005].)

Most computer developments, whether hardware or software, are mere innovations and undeserving of much protection.

On a theological level, of course, "there is nothing new under the sun" [Ecclesiastes 1:9]. But we needn't heed that.

I'd like to express my thanks to Paul Gooch, President of Victoria University (Toronto), for his patient discussion of this topic with me.

Devices of the Soul

Steve Talbott
O'Reilly Media
ISBN-10: 0-596-52680-6
ISBN-13: 978-0-596-52680-1
281pp.
£ 15.99
Published: 18th May 2007

reviewed by Roger Whittaker

Steve Talbott became well known as the author of *The Future does not Compute* which, when it was published in 1995, was a fairly rare example of sceptical writing among the avalanche of hype about the liberating power of the Internet and the personal computer which was current at that time.

The full text of *Future does not Compute* is available online:
`http://netfuture.org/fdnc/index.html`

The material in *Devices of the Soul* was originally written as a set of essays. These have been woven into a book, and as a result this book does not possess a very clear linear argument. It is none the worse for that, however.

The subject matter varies quite widely, but the unifying theme is a strongly humanistic approach to the nature of knowledge, of learning, and of engagement with the world. Talbott discusses among other things education, disability, science, ecology, the Internet, robotics, baby-walkers, community and marriage.

In each case, he argues for the vital importance of real human engagement as opposed to tempting alternatives, and outlines how technology and modern modes of thinking can mitigate against this.

So for instance, in discussing science education he talks about the need for children to get close to and engage physically with the realities they are studying. He dissects examples of modern "good practice" and discusses the ways in which they fail to engage students' imagination and hence fail in their educational aim.

Talbott believes that technologies such as the personal computer and the Internet have a two-fold negative effect.

Firstly there is the way in which the nature of the technology limits the ways one deals with the real matters one uses it for (for instance the use of a spreadsheet to constantly view the business "bottom line" blinds one to other less tangible matters which do not appear in the spreadsheet but can be controlled and which do affect a company's success). By making particular aspects of reality visible and hiding others, the technology shapes our view of reality.

But the second and more important negative effect for Talbott is the way in which we begin to model our view of our own activities on our understanding of the technology that we use. Thus education (about which he writes very passionately) is seen as a matter of transferring information from one place to another; management of the environment is seen as simply adjusting certain inputs to obtain the desired results; well-being is seen as measurable through an aggregate of economic or other numerical indicators. Worse, he claims that our internal model of ourselves becomes based on our model of the technology that we use: this impacts on our ability to understand the world.

> *When we lose awareness of all but the machine-like in ourselves, we also lose the ability to conceive the world as anything but a machine. Those whose intellectual horizons are encompassed by digital machinery tend to see the world computationally, just as their predecessors saw the world in terms of clocks, cameras, steam engines, telegraph lines, and movie projectors. There is security in believing the world is like the things one knows best, and intellectual ease in draping one's well-practiced ideas like a veil over the Great Unknown. [p 179]*

The chapter entitled "Educational Provocations" consists of a list of bullet points about the question of computers in education, a subject about which Talbott clearly feels strongly. He challenges the reader to deny any of the particular statements in this list, all of which are arguments against the use of computers in the classroom, particularly below secondary level.

It would be wrong to describe Talbott as a Luddite: he is criticising the effects of the technology from the point of view of a person who has been a close observer of its development. But he is always on the look-out for unintended consequences and his critique of the ways that technology can shape its users in its own image is a profound one. In some ways he reminds me of Ivan Illich, who made the same kind of points in the 1970s not only about motor vehicles (fairly obvious and relatively uncontroversial) but also about schools and hospitals (shocking to most).

The Myths of Innovation

Scott Berkun
O'Reilly Media
ISBN-10: 0-596-52705-5
ISBN-13: 978-0-596-52705-1
192pp.
£ 17.50
Published: 22nd May 2007

reviewed by Roger Whittaker

This small book examines defines and examines ten widely held assumptions about where innovative ideas come from and how they are developed (the ten "myths of innovation" of the title: one for each chapter).

This most definitely is not one of those books that claims to give you a recipe for success: it examines (in straightforward language) the facts about how people historically have come up with creative and world-changing ideas and contrasts those facts with some of the comforting stories we tell ourselves about these things.

Although the book is deceptively simply (and playfully) written, Berkun has clearly spent a great deal of time researching, thinking about and distilling its content. The points he makes are sometimes fairly obvious and sometimes counter-intuitive, but the way he puts them together is thought-provoking and interesting.

One of his most important themes is that we like to delude ourselves about how innovation takes place: in the “myth of epiphany” (chapter 1) he stresses both the “shoulders of giants” theme and compares the production of an innovative idea to putting in the last piece of a jigsaw puzzle: something that can only happen if you have been working patiently on the puzzle for some time. The idea of a long period of incubation is something that he stresses with various historical examples.

In chapter 5 “the myth of the lone inventor” he examines various cases where we think that an invention or idea was the sole work of a particular inspired individual and shows that the reality is far more complex. Berkun takes some of the most well-known and important inventions (the light bulb, powered flight, the automobile) and shows that the stories we are commonly told about them are grossly simplified. But as with the “myth of epiphany” the common simplification somehow answers a psychological need in ourselves.

It was good to see Berkun quoting Mihaly Csikszentmihalyi on his concept of *flow*: one of the best descriptions of how the right kind of work (or play) can both be creative and enjoyable at the same time.

There are footnotes on almost every page, many of these will make you want to go off and investigate the fascinating stories they refer to: among others the history of “post-it notes”, the story of the Phillips screw versus the Robertson screw, the history of tyres and how the art of making concrete was lost and rediscovered.

Although not specifically related to the computer industry, many of Berkun’s examples will be familiar: he mentions Xerox PARC a number of times and discusses the conditions of work and management approach which made it so productive of innovation. He notes how Google and others have consciously tried to create the necessary conditions for creative and innovative work to flourish. Other familiar examples which he looks at to illustrate some of his points are Dan Bricklin’s development of the idea of the spreadsheet and Tim Berners-Lee and the World Wide Web.

The book cover tells us that Berkun was a Microsoft employee who worked on the development of Internet Explorer: discard any prejudices which that fact elicits – this book is well worth reading.

Manage It!

Johanna Rothman
Pragmatic Programmers
ISBN-10: 0-9787392-4-8
ISBN-13: 978-0-9787392-4-9
351pp.
£ 24.99
Published: 29th Jun 2007

reviewed by Raza Rizvi

The reader of this probably already does some project management alongside a hundred other activities that go towards delivering a software project or building a multi-person piece of code. In that sort of role one does the best that one can because there isn’t anyone else there to do it instead. In that case this is a perfect non-prescriptive guide to pretty much the entire remit of a project manager.

I say non-prescriptive because this book really does being with the basics of what might constitute a project in the first place. There is no single methodology forced on the reader and in fact things come over as being logical natural sense, one after the other in a very easy to understand flow. In that sense the book of course starts with an overview of projects and how one would use life cycles to understand the flow of information within a project, and the internal/external pressures that determine when the software product can actually be produced.

Project managers know the question they get asked the most is “when?”, and chapter 5 is devoted to the subject of estimation. This follows hand-in-hand with project 6 which very clearly and amusingly with the scheduling tactics employed by those who have to get the product out of the door and who therefore influence, cajole, and trick the project managers into adding that extra feature in the same timescale, or bring forward the release date. This chapter alone is testament to the experience of the author as she gives both the description of the technique used and how one might counter the argument validly. Having been in large-scale coding environments (for billing systems, though luckily not in recent years as a coder) I chuckled as each one reminded me of the past misdeeds of managers.

Keeping the project on time and the updates flowing form the middle portion of the book, as it tends towards the management of ever larger projects and programs before concluding with project completion. There is a comprehensive bibliography (with many of the authors own works cited...)

Overall, *Manage It!* does project management with a relaxed style that makes the read a pleasure rather than a chore. It is well illustrated with figures, diagrams, cartoons, side-bars, and practical case-snippets from real people in real projects. This latter feature really illustrates the author’s experience in teaching the subject as a practical (and essential) part of the development process rather than as a theoretical, academic exercise. I can see this book being of use to more than those involved day to day in software production. The advice is adaptable with relative ease to (non-software) product and process development, and the chapter on managing meetings deals with issues all businesses have of getting the most information in the smallest amount of time!

Learning Ruby

Michael Fitzgerald
O’Reilly Media
ISBN-10: 0-596-52986-4
ISBN-13: 978-0-596-52986-4
275pp.
£ 24.99
Published: 1st June 2007

reviewed by Lindsay Marshall

For a language that has been so prominent recently, there has been a surprising dearth of books aimed at novices learning Ruby. The original Ruby book in the Pragmatic Programmer series is not really suitable as a place to start (I don’t think it’s suitable for anything, bit that’s another review...), so it is good to see something that is not a reference manual and that does not have a huge chunk devoted to Rails.

The material is order in a sensible way (not objects first, thank goodness) and coverage is thorough though perhaps a little shallow: two or three pages on regular expressions really is simply not enough for beginners, but of course this is not a book on regular expressions. As a support text for a Ruby course I think this book will work fairly well, and I am going to try it out this coming semester to see how it goes.

Niggles? I think that it assumes perhaps a little too much experience and background knowledge, for instance, describing an Array as a “compact, ordered collection of objects” is entirely accurate but I am not convinced that it actually tells a beginner anything useful. It then later the author goes on to say that a Hash is “an unordered collection of key-value pairs”, and then that it is “similar to an Array”. Which is also true, and fine if you already understand the concepts but is not helpful if you don’t. Also not nearly enough material on exception handling.

Pluses: some nice examples and only two short mentions of Ajax.

I can definitely recommend this for someone starting out with Ruby who is not that experienced a programmer.

Adding Ajax

Shelley Powers

O'Reilly Media
ISBN-10: 0-596-52936-8
ISBN-13: 978-0-596-52936-9
399pp.
£ 24.99
Published: 3rd July 2007

reviewed by Lindsay Marshall

See the combined review below.

Ajax with PHP 5

Andrew Curioso

O'Reilly Media
ISBN-10: 0-596-51403-4
ISBN-13: 978-0-596-51403-7
56pp.
$9.99
Published: May 2007

reviewed by Lindsay Marshall

I have to confess that I feel a rant coming on. Possibly two rants. Recently it has looked to me very much as though the O'Reilly juggernaut is slowing down and perhaps wobbling a bit: there seem to be rather too many, to put it politely, "niche" books on their list at the moment. You know the sort of thing I mean: "Facebook mashups with ruby on rails", "JSON hacks for Jaiku". It does look like they have a program that perms topics together and then they get someone to write the book. Ajax with PHP 5 is exactly this kind of book. Now, don't get me wrong, the technical content is excellent, all bases covered, and the braces are in the right place in the code too. However, I just can't see the point. There are dozens of web pages covering most aspects of the material, and, what's actually more important, is that most people don't need to know the kind of low level detail that is presented.

The author develops Ajax things starting at the bare metal and working up. Which is fine if you want the minimum of code in your application, but, in reality, 95% of people are going to grab one of the packages like jQuery or prototype where someone has done all the hard spade work already and just use the facilities provided. It isn't as if the low level details are even that interesting! There are some slightly hairy error modes, but basically you ship some data off to your web server, wait for a reply and process any data returned. Yes, you can use JSON or XML to code it all, but a lot of the time this is overkill.

Basically I just can't work out who the audience for this Short Cut is, but it does what it says in its title so if it sounds like it is for you, I can recommend it.

My other rant is about the whole Ajax/Web 2.0 bandwagon. (BTW why do web technologies and cleaning products seem to share a namespace – Ajax, Flash – I feel an urge to introduce one called Vim or Omo). Again, I am not knocking the technology which is fine and is, after all, just technology. The endless hype is getting a bit tiresome though, especially as there have been almost no interesting advances recently. Dressing up your web page with a few fades and slideshows is not going to make it that much better (not that it isn't a lot of fun doing just that, but that is not the point).

The trouble with Adding Ajax is that it tries to cover far too much stuff in a fairly short book. The pages are dense with code (brackets in the wrong place, grrrr), HTML and CSS and so are just hard to read. There are discussions of several of the JavaScript packages but not in enough detail to make full use of them, and indeed as the author says in a footnote, this is an area where things are changing fast (note, I said **interesting** advances above, new packages are generally not inherently interesting). In fact there is almost no coverage of jQuery which is rapidly becoming very popular indeed (it's the one I use, so that proves it). Again, most people will pick an appropriate package and start with its documentation and sample code and go from there.

The author really has a good go at covering everything from basic Ajax to mashups with web services right throught to SVG and canvas, but I could feel my brain slowly melting as I read through it all. I just didn't need to know all this stuff nor to have to look at all that Courier set code. And once again I have to say, as I always seem to be saying in reviews recently, I just don't know who this is aimed at. A bit hard for the absolute beginner, but the more knowledgeable will have found most of the ideas on the web already.

O'Reilly publish nearly all of the essential books that any self-respecting developer owns (and keeps up to date too) but the deeper reaches of the catalogue are getting murky and I do think they need to tighten things up a bit.

Ferret

David Balmain

O'Reilly Media
ISBN-10: 0-596-52785-3
ISBN-13: 978-0-596-52785-3
91pp.
$9.99
Published: March 2007

reviewed by Roger Whittaker

This is an O'Reilly *Short Cut.* Short Cuts are a series of short booklets (mostly less than 100 pages in length, and mostly priced at $9.99) provided in PDF format as a paid-for download. A quick look at O'Reilly's web site seems to show that more than 80 of these are already available. The very earliest of was published in 2004, but almost all of those listed were published since last summer.

A couple of Short Cuts were reviewed in the last newsletter: this is the first time I've seen one. The review copy I received was actually supplied printed single-sided on A4 paper, looking rather ugly with a large type-face and small margins. Whether this kind of distribution for documents of this type will catch on remains to be seen: I notice that they seem to be only priced in US Dollars, and that they seem not to be available from outlets other than O'Reilly's web site (certainly not from Amazon's UK site). However, we shall certainly be seeing more reviews of this series here, so I felt it was worthwhile to explain exactly what they are.

Ferret is a native port to Ruby of the Apache Lucene library. It is a search library which allows the creation of a search index for text documents. With the use of appropriate filters it allows you to index other formats such as PDF, HTML, Microsoft Word, OpenOffice.org files and others.

Ferret claims fast permormance figures for both indexing and search, and seems to be already in fairly widespread use, particularly for internal search tools on sites that make use of the Rails framework.

Ferret was written by the author of this Short Cut, David Balmain, and is partly written in Ruby and partly as Ruby extension code in C. The documentation that comes with the software is fairly clear in the sense that it describes at a technical level the various classes and methods that are available: this document reads more like a tutorial at least at first, but seems to degenerate into a reference towards the end.

I found that by following the text it was easy to write some simple code that worked as expected by modifying the examples given. Having started out in that way, it would not be difficult to use this library to create your own search application for a specific purpose.

If I had simply read the documentation that is included in the package I would probably not have been in a position to do that without quite some deep thought and Googling.

So this Short Cut is useful for those who want to use the software, and it does not directly violate the principle that “free software should have free documentation” because Ferret certainly does have relatively complete documentation at least for those used to dealing with Ruby packages. But this Short Cut certainly makes getting started that little bit easier.

Contributors

Alex Aiken is Professor of Computer Science at Stanford University.

Mingming Cao has been a Linux kernel developer at the IBM Linux Technology Center for 7 years, mainly in the areas of IPC, block IO, and the ext3 filesystem. Her recent focus has been on bringing up the ext4 filesystem.

Sunil Das is a freelance consultant providing Unix and Linux training via various computing companies.

Andreas Dilger is a Principal Systems Software Engineer for Cluster Filesystems, Inc., designing and developing the Lustre distributed file system on some of the largest computers in the world. He started programming more than 25 years ago and has spent much of the last 10 years focused on Linux filesystem development.

Marco Fioretti is freelance writer for ICT magazines, the co-founder and current coordinator of the Rule and Eleutheros Projects and a member and contact for Italy of the OpenDocument Fellowship.

Lindsay Marshall developed the Newcastle Connection distributed UNIX software and created the first Internet cemetery. He is a Senior Lecturer in the School of Computing Science at the University of Newcastle upon Tyne. He also runs the RISKS digest website and the Bifurcated Rivets weblog.

Avantika Mathur is a Linux kernel developer in the IBM Linux Technology Center. Her primary focus is ext3 and ext4 filesystem development and testing.

Sean McGeever is Senior Computing Manager at EPCC, the Edinburgh Parallel Computing Centre.

Jane Morrison is Company Secretary and Administrator for UKUUG, and manages the UKUUG office at the Manor House in Buntingford. She has been involved with UKUUG administration since 1987. In addition to UKUUG, Jane is Company Secretary for a trade association (Fibreoptic Industry Association) that she also runs from the Manor House office.

Raza Rizvi is Head of Customer Technical Services for Opal Solutions until the end of September. Then he is having a rest.

Peter H Salus has been (inter alia) the Executive Director of the USENIX Association and Vice President of the Free Software Foundation. He is the author of “A Quarter Century of Unix” (1994) and other books.

Sam Smith has been on UKUUG Council for 4 years and is currently the treasurer, with many random interests in addition to OpenBSD and Mac OS X. He’s also active in the UK Online Democracy group mySociety and the Manchester Universities’ Gilbert and Sullivan Society.

Roger Whittaker works for Novell Technical Services at Bracknell and is the UKUUG Newsletter Editor.

Contacts

Alain Williams
Council Chairman
Watford
Tel: 07876 680256

Sam Smith
UKUUG Treasurer; Website
Manchester

Mike Banahan
Council member
Ely

John M Collins
Council member
Welwyn Garden City

Phil Hands
Council member
London

John Pinner
Council member
Sutton Coldfield

Howard Thomson
Council member
Ashford, Middlesex

Jane Morrison
UKUUG Secretariat
PO Box 37
Buntingford
Herts
SG9 9UQ
Tel: 01763 273475
Fax: 01763 273255
office@ukuug.org

Sunil Das
UKUUG Liaison Officer
Suffolk

Leslie Fletcher
UKUUG Spokesperson
Manchester

Roger Whittaker
Newsletter Editor
London