UKUUG

news@UK

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

Volume 17, Number 2 **ISSN 0965-9412** **June 2008**

Contents

News from the Secretariat

Jane Morrison

We are pleased to have worked together with Josette Garcia of O'Reilly to bring you a joint tutorial on Moodle. This will be held at the Imperial Hotel on Monday 30th June. At the time of writing places are still available. We hope this is the first of many tutorials we will be able to organise jointly with O'Reilly.

The OpenTech 2008 Event will be held at University of London Union on Saturday 5th July. I trust you have already registered your place as within days of advertising the event we were nearing capacity!

The UKUUG Annual General Meeting will be held this year on Thursday 25th September. Full details will be sent to members during August. If any member is interested in joining Council, please let me know.

The next Spring Conference is planned for March 2009, we are currently looking at venues. The call for papers will be available shortly.

Don't forget the Inland Revenue allows UKUUG members to claim their subscription amounts for tax allowance purposes. See further details on our web site.

I am very pleased to advise that Novell are continuing with their Gold Sponsoring membership for a further year. This gives us the opportunity to keep delegate fees as low as possible at future events.

IBM have also recently become Gold Sponsor members again and we trust that SUN Microsystems will renew their sponsoring membership in the next few months.

UKUUG has a 'consultative' email list – `advisory@ukuug.org`. If you feel you would like to be included on discussions on matters of policy etc. please let me know and I will add you name to the list.

Finally, no we haven't forgotton about Linux 2008 – we are currently looking at venues for the event this year. It will probably be held in Manchester in the Autumn. Of course as soon as we have confirmed information all members will be advised.

The next Newsletter will appear in September and has a copy date of 22nd August. Any interesting articles from members will be very welcome – all submissions should be sent to
`newsletter@ukuug.org`

UKUUG and BSI: Legal position

Alain Williams

The topic that is exercising myself and Phil Hands is the legal action that UKUUG has started against BSI.

What is this all about? At the end of March the BSI approved the Microsoft OOXML proposed standard (DIS 29500) for fast tracking to an ISO standard. This was much to everyone's surprise since BSI had raised some 635 objections in September 2007, few of which had been resolved. DIS 29500 would become a full standard unless there was a formal objection within 2 months.

The proposed standard is some 6,000 pages long and has many problems: internal inconsistencies, points not fully defined, lack of adherence to existing ISO standards. In its current state it is not possible for an independent implementation – this is important, what is the use of a standard if it can only be used by one organisation. It is possible, given enough time, that these issues may be resolved. What is key for me is that a half baked standard is not suitable for the ISO fast track process.

You may have noticed that there has been a lot of noise on this issue. National standards bodies around the world changed their vote to 'yes' – often to the accompaniment of protest from their technical committees and comments about voting irregularities. Several countries who appeared to join the ISO P membership just to vote 'yes' to OOXML have now dropped out.

The bottom line is, of course, Microsoft's attempt to maintain its strangle-hold on the computer desktop. This is one of its major income generators and so excluding competition is of prime importance to its long

term profitability. The ISO approved ODF format (supported by Open Office and others) is starting to make in-roads: this is not just about price, but a well defined file format with multiple implementations is of great interest. Many organisations keep documents for years, governments keep them for centuries. Have you tried reading a Microsoft Word file of 15 years age with their latest software?

What is UKUUG doing? We are seeking a Judicial Review of BSI with respect to its actions on OOXML. A Judicial Review is asking the high court to tell a government body or a quango to do its job properly and to follow the rules and procedures. A JR cannot tell the BSI that its decision was wrong – just that it went about things the wrong way and please go and do it properly this time. We hope that BSI will thus revert to its 'No with comments' decision of September 2007 and thus take OOXML off the fast track. The impact of this should be wider than the UK since BSI is a well respected national standards body. Within the UK the non approval of OOXML will affect implementation decisions.

We are not alone: as I write the South African standards body has filed a formal appeal against the fast tracking of OOXML. This has stopped the clock, OOXML cannot become a standard until their objections are addressed. Thank you South Africa.

We now have a new representative on the BSI IST/41 committee, you will find his report elsewhere in this newsletter.

At the time of writing BSI have submitted some arguments in their defence, along with the legal bill to date of some £24,500. This is not something that we will have to pay – unless we lose badly. Our own legal bill is much smaller. However: we will have costs to bear and any support that you can give us will be gratefully received – also get your friends and company to join UKUUG.

We will publish occasional developments at:
http://www.ukuug.org/ooxml/

You may also find these of interest:
http://www.noooxml.org/
http://www.groklaw.net/

UKUUG Representation at the BSI

Richard Melville

For those who have not heard, Mike Banahan, our representative on the IST/41 BSI Technical Committee, has stepped down owing to pressure of work, and I have taken his place. As well as being a new addition to the IST/41 committee I am also a new member of the UKUUG, having joined about four months ago.

The IST/41 committee has been charged by the BSI to advise on the technical merits of putative standards relating to XML. Currently, of major concern is the request by Microsoft to have its new OOXML document format agreed by ISO as an internationally accepted open document standard. In addition to this, Microsoft also wants its application to be put on the "fast-track" process.

I think that it is fair to say that, historically, the "fast-track" process has been reserved for applications that have, to all intents and purposes, already become a standard by their almost universal adoption by the community. That is, they have already become widely used and, as such, demand limited additional work in order to elevate them to an approved international standard. OOXML certainly does not fit into this category.

The UKUUG, therefore, does not support the view that OOXML should be fast-tracked. We believe that much more time should be allocated to enable a more detailed analysis of the format. It would appear that documentation already promised has not, thus far, been produced, and deadlines have been ignored. Furthermore, the UKUUG finds it difficult to accept that there is even a case for another open document format.

The OASIS group, of which Microsoft is a member, has already produced, and had ratified, ODF, the "Open Document Format". Surely it makes far more sense for Microsoft to adopt this international standard than to attempt to produce yet another.

The UKUUG believes that the final vote from the BSI, in favour of fast-tracking OOXML, did not appear to follow the correct procedures. At present, we are trying to persuade the BSI to change its mind.

Announcement: Using Moodle Tutorial

A tutorial on Moodle, organised jointly by UKUUG and O'Reilly, will be held on Monday 30th June 2008 at the Imperial Hotel, Russell Square, London WC1B 5BB. The tutor for this event will be Helen Foster.

The Using Moodle tutorial will cover how to:

- Create a course in Moodle and add content, such as worksheets or links to websites
- Use forums, chats and messaging for communication and collaboration
- Set up assessment activities, with the option of students receiving instant feedback
- Manage students and monitor their participation
- Give students additional roles, such as forum moderator or, if necessary to restrict access

There will be opportunities throughout the tutorial for delegates to ask questions, and you'll need to pay attention as questions may be asked too!

Delegates will be given a copy of the book "Using Moodle" plus supplementary materials describing how to make use of Moodle's roles and permissions system.

Helen Foster is a Moodle administrator for the site with the most users (over 400,000 with more than 500 new users daily) – moodle.org – and is a facilitator for the Using Moodle course, which provides free 24-hour support for Moodle users worldwide. Helen now works full-time for Moodle as Community Manager.

Please note places are limited and early booking is essential.

The deadline for "early bird" booking deadline is 13th June and the final booking deadline is 24th June 2008.

The cost for UKUUG members is £ 140 + VAT. For non-members the cost is £ 175 + VAT.

Standard rates apply after 13th June.

The above prices include FULL day tutorial, refreshments, lunch and a set of tutorial notes. For full details and terms and conditions, see the booking form.

Download the booking form pdf at:
http://www.ukuug.org/events/moodle/Moodlebooklet.pdf

Or book on-line at:
http://www.ukuug.org/events/moodle/booking/

Announcement: Open Tech 2008

The Open Tech 2008 event will take place at the University of London Union on Saturday 5th July.

The event is sponsored by BT Osmosoft (the open source innovation arm of BT).

Open Tech 2008, from UKUUG and friends, is an informal one-day conference about technology, society and low- carbon living, featuring Open Source ways of working and technologies that anyone can have a go at.

There will be 60 talks across 3 sessions, including:

- Open Rights Group - 2 years, 344 days on

- mySociety - WhatDoTheyKnow.com launch, and other goodies
- Overthrowing Government on a Budget, Keeping Track of the CIA's Rendition Flights, Tracking Arms Dealers with Python and Bits of String
- Ben Laurie and friends on network security
- Danny O'Brien's Living on the Edge
- AMEE, and Open Source Solar Heating
- Saving money and reducing carbon through Green IT
- Getting people involved with online media

Pre-registration is recommended as the event is likely to sell out. Entrance is £ 5.

Further details and a full schedule are available at:
http://www.ukuug.org/events/opentech2008/

On-line registration:
http://www.ukuug.org/events/opentech2008/registration

Sponsorship News

Sunil Das

We welcome the confirmation that Novell have agreed to continue their Gold sponsor membership of the UKUUG. In addition we are delighted that IBM have now joined this elite group. Gold sponsorship offers considerable benefits both to the entire UKUUG membership, to the sponsors themselves and helps UKUUG to continue its commitment to high value services for members.

For the UKUUG membership, sponsors offer many opportunities, including provision of top quality, cutting edge speakers at our conferences, seminars and workshops. Their contribution to newsletters ensures that UKUUG members are right up to date on key developments within the technical arena. Overall, sponsors provide a significant contribution to the Unix/Linux/Open Source/Open Systems communities. Their presence at UKUUG events ensures opportunities for members to meet with and discuss issues with high profile companies in an informal environment.

Gold sponsorship offers many benefits, but in particular companies have a consistent presence within, and the opportunity of actively supporting UKUUG events and activities. It gives access to the UKUUG membership, and provides a forum for meaningful dialogue with 'open systems' professionals. Gold sponsors have a presence at all UKUUG events via flyers, attendance, speakers etc. and actively contribute to the newsletter.

For both parties many opportunities present themselves in respect of recruitment possibilities, development, information exchange, problem solving, updating knowledge, and simply meeting people who understand.

If your company would like to know more about becoming a sponsor, please contact us for more information.

OpenMoko event report

Dave Crossland

Ole Tange gave a talk on OpenMoko on Wednesday 9th April at University College London. This was one of an occasional series of free evening events organised by UKUUG.

The slides for this talk are now available on the UKUUG web site at:
http://www.ukuug.org/events/openmoko/

The evening began with an introduction from Alain Williams in which he mentioned UKUUG's stand against the fast-tracking of the OOXML format standards process.

The following are notes taken during the talk together with some of the questions and answers that followed.

Ole Tange:

OpenMoko is a revolution for your mobile phone. I feel its as disruptive for phones as GNU/Linux has been for computers in general.

I'm a developer at Ange Optimization:
`http://www.ange.dk/`

I've been using GNU/Linux since 1992 and doing paid work on free software since 1996. I have some fame for a diagram about patents on web shops. I'm a OpenMoko wiki webmaster.

Topics:

- The Hardware: FIC Neo FreeRunner
- The Software: Ideas for software
- Price and availability
- Future developments
- A look inside the hardware

My dream: The **personal** computer; like glasses or a watch, a computer that is carried around all the time. A smaller amount of storage is okay if it has Internet access, and it should be about as capable as a laptop, but a lot smaller.

When I saw the first OpenMoko smart phone, I thought "This is it!".

What is the hardware? The FIC Neo FreeRunner has a 400Mhz ARM CPU, 128Mb RAM, and a MicroSD slot for up to 4Gb disks (well, 8Gb isn't tested yet, and the MicroSD controller might not handle that). The hardware is standardised as much as possible, you can get MicroSD cards almost everywhere. There is a 3D graphics chip, a 640x480 280dpi screen – that's like the old laser printers, a very high resolution, and it is a touch screen (single touch, not multitouch like Apple's iPhone).

USB can charge it, and transfer data to it or from it – all using stock USB ports so you can use the USB cables you can buy anywhere, unlike almost all other phones, whose cables are often very expensive for what they are, and hard to buy when you need them.

WiFi (Atheros AR6K). Two accelerometers – allowing the phone to sense how it is being moved in space, as seen in the Nintendo Wii controller. There is GSM 2.5G (no EDGE) and I don't really know what the top data speed is, I think around 100kbps, but its not megabits. And there is GPS too, so you can do location. And Bluetooth, of course.

Q: Battery life?
A: There is no battery in it! ;-) The software to control the power is not finished, at all. I heard that someone got 50 hour of standby, with everything turned off, but so you can receive calls for that amount of time.

Q: Why not 3G?
A: Not sure.

Q: Maybe because that uses a lot of power?

Q: Whats with the Atheros chipset for WiFi?
A: We'll get to that :-)

Q: Who did the case design?
A: Not me ;-) But the CAD files are online, so if you have a CAD machine you can make a new case if you like. [Thanks to Jon Philips for organising this.]

Okay, who knows FIC? First International Computer. Not many. But everyone probably used their computers; they make computers that are badged by other companies, like Dell and Toshiba. They make a 5th of all laptops made. They have been very open about delays, explaining things, and we'd rather solid hardware and delays than flaky hardware. Flaky software is fine initially because that can easily be changed, but hardware cannot.

The Apple iPhone has tried to mark down the modification of the iPhone software; this is just the opposite, they are inviting you to change the software. They are selling directly to consumers as they expect a better profit that way.

Many of us have been to Nokia's website, but where is the page on that site to show you how to take apart the phone? FIC do! That's only half the good news; they also have a page about how to put it back together. *If* you are going to take it apart, they help you. This signifies to me how open they are about everything, and I appreciate this as someone who likes to take things apart and see how they work.

OpenMoko is a GNU/Linux distribution; this is a base for the phone, and they see it as a base for other devices. My camera has a flash card for storage, a screen for interaction, it has a small OS in there. The Neo is the same; if I have to produce software for the phone, FIC see they might be able to use it to produce devices that don't currently run GNU/Linux, or even new kinds of devices.

The GNU/Linux distribution is custom, because there isn't a lot of disk space around. But you can install Debian on it. Today you can download an emulator and a developer phone is available, although it is sold out.

What is free-as-in-freedom software? Here's a simple diagram of a computer hardware. All the green part is free, the red is non-free: GSM firmware, GPS firmware. The GSM and GPS drivers are both free software. You cannot change the GSM software, but you can talk to the GSM firmware with AT commands. NMEA is like AT commands for GPS devices. After that, it is 100% free software.

Q: Will you be able to upgrade the firmware?
A: You can't upgrade the GSM firmware; you can think of the GSM unit as hardware, you have to go to an authorised repair shop to do it. Or at least, you'll need a special tool to do that, an expensive one, and its not documented how to do that.

Q: Will you include schematics?
A: No, its public what the chips are, but I haven't seen the circuit board published anywhere.

Q: So the software and hardware interfaces are free, but the hardware design isn't?
A: Yes. It starts with the chipsets; if you want to copy the PCB, you could scrape off the chips? And there is also a project to make a home made mobile phone; if that's what you want, its something you can look into, although it won't be as small as this ;-)

Q: Is there a GSM compliance issue?
A: Yeah, that's why its burned into hardware.

Okay, so the software applications are in two phases; phase one has a dialer, main menu, music player. Phase two has all the usual smart phone stuff; clocks, web browsers,

QTopia is software for TrollTech's GreenPhone; they made software for that device, and that software has been ported to The Neo1973, the developer's Neo hardware. It will run on the FreeRunner. It means you can use a Neo1973 to make calls now, but it won't accept incoming calls unless it is fully booted, which kills the battery fast. But it will improve.

So what is the big deal?

There are no limitations. It is a GNU/Linux computer; you can access things as root, do anything you like. Later we will add chroot and normal users, but initially everything runs as root.

Here are some of the ideas people have come up with when there are no limitations:

Location based calendar. Ever run out of toothpaste? Gone to a shop, done the shopping, and forgotten something? You could have a location-based calendar; you can say when you run out of toothpaste, "buy toothpaste when I enter this circle around a GPS co-ordinate." Then when I enter that area, the phone will

sound an alarm and remind me. But if I go to that area at night, I don't want that alarm. So I can enter the opening times of the shop at that GPS location! "Never send a human to do a machine's job" said Agent Smith in the Matrix.

Get off at bus stop. Point to where you want to get off, and ride the bus until your phone reminds you just before you get to that area. This is just the same problem as the last problem; give me an alarm when I get to a certain area.

GPS Friends. You send your location to your friends, so they can see where you are. You choose who your friends are, of course, and you can get an alarm if they are close. Say some friends are in town and I didn't know - I can drop them a line to hang out. But perhaps we are going to a free software conference in a strange country, we can know when we are there. And you can of course *lie*, such as if your boss is one of your friends.

Q: How can you tell where the friends are?
A: You could upload the information to a website, and we have crypto so that only the people you want to decode your location can do so. The location can be a rough estimate, every hour or so.

Q: You could use it to avoid people?
A: Sure ;-) The other example, is you can give a perimeter around your current location and if your friends – or your kids with Neo FreeRunners – leave that perimeter. This is the same just inverted.

Q: Could the police use this, like for house arrest?
A: Well, you could if you locked it down so they couldn't reprogram it ;-)

Closest WiFi. If a WiFi is nearby, you can upload its position to a central server using the WiFi – and this is how the connection is tested. Then you can show a map with the nearest WiFi is located, showing when each AP was last tested. And then you can easily make zero price VoIP calls. This might not be legal in Britain, but it is in Denmark.

Q: Using unknown WiFi is dangerous, you could have a tainted DNS server.
A: You can run things encrypted; you are right, you can't trust a AP to not listen in on everything, encrypting helps that.

Q: Does the phone support WPA?
A: WPA is done in software, and this is a computer, so yes.

Q: Is there hardware encryption?
A: No.

Q: A lot of people have Oyster cards. Could you put an RFID chip in there?
A: We could, and we might also put a RFID reader in there too, so you can read them – and copy them! ;-)

Q: Regarding WiFi, what about Fon? An opportunity for cross marketing?
A: I'm personally not too fond of it. FIC are open to bundling like that, I think.

Q: Is there a camera?
A: No, not on the 1973 or the FreeRunner. I started on the OpenMoko email list in November 2006. The first email everyone sent was "This is very cool! Why is there no WiFi?" and the first model didn't have it, but FIC listened and added it. Few people asked for a camera; people are coming up with ideas that use a camera, so the next model – a year or two away – probably will have a camera, I speculate.

Q: A USB host, so you can download the photos from your camera into the phone?
A: Yes – it can act as both a USB host and a USB slave. There are endless possibilities here.

Q: You can get CompactFlash cards with Bluetooth transmitters too.
A: Yes.

So, in cities with municipal WiFi, we can use that to make calls. And we can do this for anything! Petrol station prices, restaurant reviews, anything – and this would be uploaded next time the phone gets online via WiFi.

Navigation. If you want to navigate, you need a map. Maps are a problem; in Denmark a public office sits on the maps but its very expensive. There is a Wikipedia approach to make our own maps,

(`openstreetmap.org`) – and this phone can record a map automatically: you are moving at 50mph, so probably you are on a road or a train track. If everyone moves along the road the same way, its probably a one way street. We might get rush hour statistics useful for efficient routing.

Q: Privacy issue here?
A: To make it anonymous, we'll have no cookies or other identifiers, and we can upload it via tor by default. This opens the door to spammers, but its possible to filter that.

Q: Is the browser WebKit, so it support CSS3?
A: I don't know, but you could install such a browser. I know its not Mozilla, since we have 128Mb of RAM.

Bluetooth/WiFi gateway. Bluetooth devices use less power than WiFi; the phones could automatically elect a gateway OpenMoko with the most battery to run WiFi, and then everyone else can use Bluetooth to access the net.

Q: Are there studies to show how long Bluetooth/WiFi/GPS can run?
A: The power management software is changing all the time, so those studies would be premature. The trends are going up. Amusingly, the battery is the same form factor as a Nokia battery, but FIC are likely to sell spares soon too.

Graphics tablet. The screen is touch sensitive. The accelerometer could even be used as a pointer. There are two of them, so that the accelerometers accurate enough for that kind of thing.

Q: Can you do voice recognition? It could be a security aid.
A: Voice recognition is a hard problem, but that could be useful to stop you calling when drunk.

Profiles. Time and location based, and can have a timeout. You can have a silent mode come up when a meeting is in the calendar. The phone can tell when you leave the location of the meeting, so it will stay in silent mode if the meeting runs longer than expected. Or maybe you want all business calls to go to voicemail when you are at home. In the cinema, they tell you to turn off your phone, but we often forget to turn it back on afterwards. You can turn it off for a set amount of time, and then turn back on automatically.

VOIP. Voicemail on the phone; Asterisk. I think we will pay flat rates for mobile phone calls within a few years, and when that happens, we will see spam calls. I'd like a small puzzle, like a voice message saying "What is three plus ten?" and if they can't answer that, I don't want to talk to them ;-) And if a computer can figure that out, I do want to talk to it! ;-)

Firewalling. Time based; so when I am in Singapore, I get calls late at night. So I can record a message for late night calls that says, "If you really want to wake me up, press 123, or hold for voicemail." And I could have white-listing to let specified numbers through straight away.

Voice Text. I can make a physical gesture that is recognised by the accelerometer, and start a small recording, and then I can send those audio files to a typist to type them. That might be me in the future, a real secretary, or voice recognition software.

Q: No touch feedback with touchscreen phones
A: Yes, not good for less well sighted people.

My dad had an idea, when he goes golfing his heart might pack in, so it can tell if he stops moving when he is within the golf club grounds and call home after noticing he's been still, and sounding a loud reminder sound.

Q: You could have another Bluetooth devices so you could be alerted if you leave your phone behind, like in a a restaurant.

Q: A GPS-Friends style feature can tell you where you left it.
Dasher input. I'll demonstrate this later.

Distance measuring. The soundcard can do up to 90kHz, so if you can control the speaker and microphone, you can make a sonar with this. So you can see now, people are thinking very outside the box with this – you can do very interesting things with this device impossible with other phones. I don't know how useful this exact feature is.

Q: Can you adjust the ringtone volume to the ambient sound?
A: Sure! So its always loud enough, and never too loud.

Cheap Data Transfer. Different subscriptions give different ideas. If you have free voice, you can write a soft-modem, and send data pretty slowly, but still for zero price – perhaps sending SMS via a cheaper gateway online via an Asterisk machine you can call free. Free data means free VOIP calls, and free SMS messages could also be converted into other kinds of data – "IP over SMS" ;-) And when you call someone, you can see what number is calling, right? That number is controlled by the handset! So if we have a 16 digits for data there; start all of them with 111, which no number does, and then you can share data with the rest of the digits. I'm not sure how legal that is ;-)

Q: Simultaneous calls?
A: You can with VOIP; you can hold a call and make another though.

Games. I gave a talk at a university who makes computers games. I'm not a computer guy. They loved the accelerometer. They have already implemented a game called "drainers'n'gainers" and it requires Bluetooth. You scan the room for Bluetooth devices, and measure the signal strength to map how far away they are roughly. I assign positive and negative values to each of the devices, and some add to my value and some of them drain my value. So if you have a bunch of people in the room, there will be this mad dash around the room to stay close to people who gain you and away from people who drain you.

And many more advanced ideas are out there. BUT!

Software patents. Some of the ideas I have presented are patented for sure in the USA. Software, firmware and hardware patents. In the EU, a complete system can be patented – hardware running software.

So we need help to fight software patents: Join the EFF, FFII, FSF, FSFE and ORG. We have won several battles, but the war isn't over. Right now in the EU, we don't know which way it will go, we fought it off, but it might return. These organisations are working on this problem, help them so we are all able to write these ideas.

Q: You described a lot of ideas that need servers online to store data.
A: Some things are implemented, many are not – but you can take part and contribute. There is a great opportunity to change things, nothing is set in stone in this project.

Like any GNU/Linux system, OpenMoko has a packaging system. There is a Stable branch in the FIC repository, which is stable and tested and security patched by FIC. There is a Experimental Branch like Debian testing, which FIC is testing, and many development branches out there that other people host. You can easily recover if you brick the phone if you get the development hardware.

The Neo1973 came out on 12th July 2007, and was USD$300 (base model) and $450 including additional development hardware. The FreeRunner is due in about 6 weeks or so, and will be $450 base and $600 including additional development hardware. If you know someone with the development hardware, you probably don't need to get one yourself, you'll use it so rarely.

Q: Pulse, in Germany, are advertising themselves as an official FIC distributor.
A: Great!

Q: When will the software be good enough to make a phone call?
A: I made my first call 3 weeks ago! I couldn't make two phone call though, but the power management is being improved. When will it be like a normal phone? I don't know, but all the resources are into getting the hardware done – once the hardware is done and stable, you can work on the software being ready by Christmas.

Q: Which Christmas?

Q: I'm at o-hand.com and have a FreeRunner – here! – and you can make phone calls and so on. It will work by the time you buy a FreeRunner. There are about 2 or 3 in the UK, and the battery life is pretty good, say 12 hours. Its a lot better than the Neo1973, and the power management software will help a lot.

Q: Are other mobile hardware makers going to be make OpenMoko compatible hardware?
A: Lots of email addresses from those companies are subscribed to the lists, although they don't post much...

Q: Does it take SIM cards?
A: Yes, just one. People have said taking two SIM cards would be a killer feature, but no its not planned. Can you make a software SIM? Not any more.

Q: Can you swap the SIM while the phone is running?
A: The SIM is behind the battery, so no, and that's probably on purpose.

Q: Is there a danger that too much developer time is spend on nonsense stuff like we talked about, instead of the core stuff?
A: There are people paid to work on the core stuff, like o-hand guy, and all this other fringe stuff is done by the community.

Q: How does the OpenMoko community feel about Google Android?
A: Its basically a hardware Java Virtual Machine. When we get the source code for that, I think we can compile it for the Neo, and run Android on the Neo – and maybe even on the OpenMoko GNU/Linux platform. So we don't see it as a competitor.

Q: Someone mentioned that "moko" is Spanish for "snot". Will this change?
A: I did this talk in Spain, and it was funny, but no.

Q: Some people are addicted to the UI of certain phones. KDE puts things where they are expected to be for a Windows user. Will there be a Nokia UI skin?
A: This is a computer, so I expect people in the community to make Nokia UI clones, for sure. Menu structures, everything.

FIC see this as an opportunity to experiment; if you make a USB device and connect it to this phone, and everyone wants that as a combined unit, they will help you develop that product and share the profits. They want everyone to do this; maybe make a braille display in a larger case, and use the innards of this device to make it possible.

So, we've talked about the FreeRunner hardware, the OpenMoko software and ideas for what you can do, how much one costs and when you can get it, and the potential future developments. For more details see
`http://www.openmoko.org`
`http://wiki.openmoko.org`

Q: Bulk orders?
A: Yes, FIC will do that for orders of more than 10.

Okay, break for a bit, then we'll take it apart and demonstrate Dasher.

Postscript: When introducing the Neo FreeRunner hardware, and talking about the USB cable, Ole showed us another USB cable with a proprietary connector for his digital camera, noting that it is not possible to get a replacement. An hour after we all left UCL, he texted Alain Williams asking for a contact at UCL since he had lost the cable. He went back the following day but could not find it. This illustrates why anything proprietary creates artificial costs for us all.

UKUUG Spring Conference 2008: report

Roger Whittaker

The Spring Conference was held at the Birmingham Conservatoire on the 1st and 2nd April, with a day of tutorials the preceding day.

Traditionally the Spring Conference has a Systems Administration flavour, with a different particular dominant theme on each occasion. This year that theme was dynamic languages, and this was reflected in two of the three tutorials (on Perl and Python), and in several (but by no means all) of the individual talks.

For those who chose to arrive on the Monday, there were three one-day tutorials to choose from: a Perl Teach-in from Dave Cross, a Python Introduction from John Pinner, and a tutorial on openBSD's PF firewall tools by Peter Hansteen.

I attended John Pinner's excellent Python tutorial: this was mainly intended as a one-day introduction to Python for refugees from other languages. The tutorial was well-attended, and the very well-produced set of notes constitute good reference materials for future use.

An article describing the PF tutorial follows this one.

The conference proper commenced on Tuesday morning, with a short introductory session in the Recital Hall, which was the larger of the two rooms used for talks during the conference. Throughout most of the two days, the talks were divided into two tracks, one in the Recital Hall, and one in a somewhat smaller lecture theatre (more a large classroom really, as the seating in that room was not tiered).

At times it was difficult to decide which track to attend. There were certin times during the conference when there was an element of "lock-in" caused by three talks in one track taking up the same amount of time as two in the other: personally I would prefer a system where all talks are the same length.

Highlights of Tuesday's programme for me included the talk by Geriant North of Transitive about their cross-platform virtualisation technology that uses binary translation to run binaries compiled for one architecture on a different platform. Also impressive was Stephen Quinney's talk on system configuration using LCFG.

In the afternoon, Russel Winder's talk "The Great Language Debate" was a thought-provoking survey (not really a debate, though some sharp differences of opinion surfaced during questions) of the differences between languages and their features by an expert in the field.

Mike Whitaker of Yahoo! followed this with a talk entitled "Perl is", and then Alex Wilmer spoke on the use of Python for working with tabular data. The track concluded with talks about command line option processing in Perl and Python given by Jon Allen and John Pinner respectively.

I was sorry to miss a talk on "Advances in OpenSolaris Network Administration" that took place at the same time as these dynamic langauge talks.

On Wednesday the parallel PostgreSQL conference run by the PostgreSQL UK User Group took place alongside the UKUUG events.

Wednesday's highlights included Randy Appleton's talk looking at an analysis he and his students had done of the size and speed of the Linux kernel and certain applications, with some surprising results.

Also popular on Wednesday was George Shepherd's talk on ZFS, and consecutive talks by Jon Allen and David Jones on Functional Programming in Perl and Python respectively.

In the afternoon there was a plenary session with Dr Martin Wright of West Midlands Police entitled WARP. WARP stands for "Warning, Advice and Reporting Point", and WARPs are intended to provide help and information in the form of warnings and advice about security issues, with a particular focus on small and medium sized businesses. There was some discussion about whether and how UKUUG could become involved: Dr Wright's notes of the session are included elsewhere in this newsletter.

Peter Hansteen gave an interesting and useful talk on spam and malware protection, with particular reference to OpenBSD's `spamd` (no relation to that other `spamd` that many people know).

Some time at the end of the day was earmarked for lightning talks, and as usual on these occasions, several interesting and entertaining talks were given against the clock (literally). Of these, a few stand out in the memory: Geraint North demonstrated the generation of printed materials from DocBook source using XSL-FO and related tools. Andrew Stribblehill of Google demonstrated his `xapply` tool. Zeth Green demonstrated scipts to convert ODF documents to RST (restructured text) format and back again. And in an impressive demonstration of accurate speed-typing, David Jones gave an example of embedding Lua code into C code.

The day ended with some discussion about UKUUG's future and aims: discussion which it is hoped will continue through mailing lists, this newsletter and elsewhere, with an expansion of the membership and more activities being among the main objectives.

PF Tutorial Report

Khusro Jaleel

The superiority of OpenBSD when it comes to security is legendary. The OpenBSD community continuously do security audits of their codebase and their website proudly boasts of having only two remote holes in the default install, in more than 10 years! PF is the default packet filter used in OpenBSD from version 3.0 onwards.

Having setup OpenBSD firewalls using PF in the past, I was interested in expanding my knowledge and this tutorial provided the perfect opportunity for me to do so. The tutorial was given by Peter M Hansteen, who is a consultant, writer and sysadmin based in Bergen, Norway, and also the author of the excellent *The Book of PF*, published by No Starch Press.

Peter started off by answering some common questions that people might have about PF such as: "Can I run it on Linux" (Answer: "No, but some are trying"). He recommended *not* trusting any GUI tools, and simply using a text editor to edit `pf.conf` as that is simpler and faster. In addition, some tools claim to automatically convert "other" firewall rules to PF, but he recommended implementing a fresh PF config yourself.

The first firewall I ever tried to configure for my home network used iptables; when I had to implement one in PF, I found it to be a breath of fresh air. I have always found the concept of "chains" that iptables uses confusing. PF doesn't have any concept of a chain; you simply start your rules by first "blocking everything", then enabling the things you need, one line after the other. Although in principle this sounds exactly like what you are supposed to do with iptables, in practice the rules you generate are much simpler, and easier to understand.

Leading on from this, he showed examples of how PF should be set up in an environment where you need a "gateway" between 2 networks, and how to deal with problems faced by people who try to use FTP from behind a NAT firewall (ftp-proxy). Tables and filtering by services (http, ftp, etc) were introduced next. A table in PF is basically a list of IP addresses; listing them in a table makes it easier to apply a single rule to a collection of hosts.

Peter then moved on to the subject of dealing with the huge volume of spam that besieges us all. The two main concepts that he focused this section of the tutorial on were "tarpitting" and "greylisting". In tarpitting, when a blacklisted host connects to you, you send replies to them very very slowly, let's say around 1 byte at a time. When doing greylisting, you lie to unknown connecting clients using SMTP 45*n* errors (temporary local error). This usually thwarts spammers, who simply want to quickly connect, deliver their payload and leave. Many spammers don't attempt to reconnect after seeing this error, while legitimate clients will automatically retry after a short period of time. These legitimate hosts are then added to a whitelist, which means that the next time they try to connect, they will no longer be given a 45n temporary error, their mail will be accepted immediately.

We then got a look at how to thwart SSH bruteforce attacks by using rate-limiting, and a short introduction to wireless networking in OpenBSD. The next thing that Peter talked about, `authpf`, was quite interesting. Basically users need to authenticate to `authpf` first; once authenticated, only then is traffic generated by these users allowed to pass through the firewall. Special rules can be setup specifically for `authpf` users.

The next topic was load balancing and Peter showed how to configure a "web server pool" using PF. Requests to this pool were alternated using a form of round-robin. To solve a common round-robin problem where machines in the pool go down, you can use `hoststated`, which monitors the state (up/down) of the certain specified hosts and compensates accordingly. `hoststated` has been renamed to `relayd` in OpenBSD 4.3.

You can tag incoming packets, so you can quickly pass/block packets marked with a certain tag. Setting up a OpenBSD bridge was discussed next. A bridge in this context simply refers to a transparent firewall that sits between 2 or more networks and filters packets at the link level.

Using ALTQ (Alternate Queueing), you can do bandwidth allocation and traffic shaping. You can used class based queues (percent, kilobytes, or megabytes), priority based or hierarchical queues. In a class based queue, you can say for example that FTP is only allowed 20% of your bandwidth.

The last major aspect of PF that Peter discussed was CARP (Common Address Redundancy Protocol) and `pfsync`. Put simply, CARP and `pfsync` allow you to setup 2 redundant firewalls instead of 1, and in case one firewall fails, everything switches over to the other firewall automatically. pfsync is used to keep the rules between the 2 firewalls in sync.

Overall, I am very pleased that I attended this tutorial. I was familiar with some of the concepts, but things like `authpf`, `hoststated` and CARP were completely new to me. I will definitely use the things I learned here when considering any OpenBSD based firewalling solutions in the future.

UKUUG WARP

Dr Martin Wright

These are the notes made by Martin Wright after the session he ran about WARPs (and the subsequent discussion) at the Spring Conference in Birmingham.

1) On the 2nd April 2008 a discussion was initiated between West Midlands Police and the UKs Unix and Open Systems User Group (UKUUG) regards the development of a UKUUG WARP. The discussion was initiated following the presentation of a paper at the UKUUG Spring 2008 Conference, held in Birmingham. It was suggested that the Unix and open systems communities would both benefit from and make a significant contribution to online safety and security.

2) It is widely recognised that individuals, organisations and businesses are becoming more dependent on Information and Communications Technology (ICT). Broadband is now more widely available which has significantly encouraged online activities, for example working from home and facilitates the online dissemination of important information. If this networked online capability was to fail or be compromised then it could have a significant economic and societal impact.

3) The security of these ICT systems, including their availability, is therefore critical to those individuals, organisations and businesses. The problem is that there are huge volumes of information available on how to protect ICT systems, but it is often not clear to individuals just what, how and when action needs to be taken. Keeping their ICT security up-to-date can be time-consuming and costly, especially for small organisations, and consequently many do not attempt it and are left vulnerable to attack. Such vulnerabilities in turn increase the overall risks throughout the Internet.

4) The creation of a UKUUG WARP, where people can work together and share information on threats, incidents and solutions can address many of these problems very cost effectively. WARPs (Warning, Advice and Reporting Points) are being promoted by the UK governments Centre for Protection of the National Infrastructure (CPNI), since they contribute to the overall reduction of ICT vulnerabilities generally. See:

`http://www.warp.gov.uk/`

`http://www.cpni.gov.uk/`

5) A WARP is set up to provide a service to members of a particular community, usually of between 20 and 100 members (in order to preserve a sense of community). It is run by a WARP operator who uses a website, email, telephone, SMS, and occasional meetings (where possible) to send a personalised service of warnings and advice to the members. The operator will: filter relevant information and deliver it to the community; facilitate the sharing of advice and best practice within the members of that community; help build trust within the community thereby encouraging members to report incidents to each other; and anonymise these reports and where relevant, share them with other WARPs.

6) The Midlands WARP service is licensed by CPNI to provide WARPs within the geographical location of the Midlands but also across sectors, for example to the UKUUG. It is established as a not-for-profit organisation and is keen to work with UKUUG in facilitating the UKUUG WARP.

`http://necpc.org.uk/warps.html`

Further it is recognised that the unique knowledge of open systems held by UKUUG members would be of real benefit to the wider WARP and IT security communities in that members would be enabled to and provided with a structured approach to effective risk and insecurity information sharing.

Report: Richard Stallman talk at Manchester

Paul Robinson

If there is one name that familiar to anybody with opinions on software licensing it has to be Richard Stallman.

Founder of the GNU project and author of emacs, Stallman (or RMS as he is often known), gave a talk on the 1st May at Manchester University titled 'Free Software in Ethics and Practice'. Organised by the Free Software UK Manchester group and local branches of the BCS and IET, just over 300 people squeezed into a lecture theatre for the evening.

RMS last spoke in Manchester when the only people interested in listening to him were mostly academics and students: an audience traditionally receptive to radical thoughts within their field of expertise. Today, Manchester is one of the centres of digital media, technology development and a hub of entrepreneurial software development. The local GeekUp meetings are testament to the diversity of the local sector. This was an audience that makes money from software development, specifically through software licensing. Experience made me think some of them may want an argument.

His talk began by specifying what he means when he talks about 'free software'. It boils down to 'Four essential freedoms':

- Freedom 0 is the freedom to run the program as you wish.
- Freedom 1 is the freedom to study the source code and change it, so the program does what you wish.
- Freedom 2 is the freedom to help your neighbour. That's the freedom to distribute copies when you wish.
- Freedom 3 is the freedom to contribute to your community. That's the freedom to make and distribute copies of your modified versions, when you wish.

The GPL, he later explained, is designed as a software license to protect those four freedoms. Other licenses, such as the BSD license, whilst not explicitly protecting some of them, do at least make the four freedoms possible and therefore are also considered "free software" licenses. It is the protection of the freedoms the FSF and GNU project concern themselves with.

As RMS himself explained:

> *If you have all four freedoms, then the program is free software, because the social system of the program's distribution and use is an ethical system, respecting the user's freedom and respecting social solidarity.*
>
> *But if one of these freedoms is missing or insufficient then the program is proprietary software, non-free software, user subjugating software. Because, the social system of its distribution and use is unethical. To release a non-free program is not a contribution to society, it's an attack. It's an attempt to subjugate people. And we should hope it fails, and even better, it shouldn't be done at all.*
>
> *So, if you have a choice between developing a proprietary program and developing no program, it's better to develop no program. At least then you're not hurting society.*

His talk continued by expanding each of the four freedoms in more detail, explaining why they were moral essentials for a free and fair software society. The detail (link to transcript below), would be familiar to most.

What was perhaps more interesting was how political RMS appears to have become. The constant jibes at Bush, "Bliar" and "Gordon Clown" seemed to have been lifted out of a Mark Thomas routine, but this is part - I believe - of how the FSF and RMS himself intend to develop the argument. They do not see software freedom as something for geeks to talk about in a lecture theatre: it is a fundamental piece of democracy.

Some are left incredulous by the zeal with which the free software movement operates in this regard. In a world of over-population, starvation, climate change, fundamentalist beliefs at war with each other, it

all seems rather surreal to suggest that the most important challenge we face in society today is software licensing.

But even an enthusiastic first year undergraduate can quote Stroustrup: “Software runs civilisation”.

If that is the case, then civilisation needs to think carefully about what it means to produce and run software as an ethical and moral choice. Discussion and debate within the Free Software movement now resembles something a little like a CND or Greenpeace meeting: how people can take action; how to promote beliefs the freedoms; how to ensure you are not supporting the world of proprietary software.

After the talk were questions from the audience. Many of the questions were ones I was expecting from an audience who make their living writing software.

True freedom means I get the right to choose what software I run even if that means running proprietary software, right?
Answer: That’s not true freedom - you can’t edit the source.

If I write software for myself and you require me to release it as free software, how is that different from fascism?
Answer: If you don’t release it, it doesn’t matter, but if you do release it and do so under a proprietary license that’s an attack on society.

Some seemed to want to try and make RMS see an error in his logic. It was as if they thought a convincing enough argument would suddenly see the GPL banished and a move to BSD licenses for all GNU projects.

And so the evening rumbled on.

Books were signed, personal questions asked, RMS’ OLPC got disassembled and reassembled. By 10:30 we were packed up and a very small group of us headed to Chinatown for food - I was providing a sofa for the night, so for me it rumbled on a little longer trying to get an OLPC to talk to my WiFi access point, an access point I was hesitant to point out my proprietary OS X machine in the corner talked to fine.

Over the next few days, there was some fallout from the talk. Those who believe in free software found themselves in conflict with those on local geek mailing lists who found RMS’ attitude pompous, arrogant, and in some cases absurd if not outright insulting to commercial developers. Watching the debate and reading back the transcript (below) I came to a few conclusions.

RMS has, I think, started to move this to a wholly political debate. The rest of us are perhaps talking about the wrong thing in that we think it has something to do with the software industry.

This isn’t about what is a right or wrong answer, this is about an act of faith, what you believe in.

The ‘open source’ community who do not agree with RMS and the FSF are at heart free marketers. They perhaps believe Adam Smith and Darwin were right, things will work out for themselves. RMS and the FSF on the other hand are perhaps more like anarcho-syndicalists, believing it is in the hands of software developers to develop software that will direct civilisation, thereby improving it. They have a moral absolutism underpinning their actions, whilst those who more comfortable with say the BSD license believe in a more relativistic view of the World.

This debate isn’t over, but for me that night it became clearer as to what this debate really was, and as a consequence will now consider any argument pro- or anti-GPL as futile as an argument on the benefits of Marxism or Capitalism: this is all about personal belief.

A video of the talk is available:
http://tinyurl.com/64o9do

The is also a transcript:
http://users.bshellz.net/~bjwebb/rms-transcript

Five years and counting

Peter H. Salus

I enjoy "bad" science fiction movies.

In 1958, there was *The Thing that Couldn't Die.*

In 1962, *The Brain that Wouldn't Die.*

In 1974, *It's Alive!.*

So, I suppose I shouldn't be astonished that the various SCO Group law suits are still plodding along. I had hoped that the trial of April 29 through May 1 before Justice Dale Kimball might draw us close to a conclusion. But two weeks later, Kimball has not announced his descision. So there's no genuine news.

And, because the activities in the bankruptcy court are dependant upon the SCO v. Novell decision, nothing is happening there – except the ever-increasing requests for payment from legal firms and accountants to whom The SCO Group has "promised" cash which it does not have.

The Novell monies are not debts, in a legal sense. They are stolen goods. The agreements between Novell and SCO [pre-SCO Group] required SCO to collect fees, pass them on and receive 5% of the collected amount from Novell. The agreements also precluded SCO from engaging in licensing deals without Novell's agreement. The SCO Group (a) entered into licensing agreements (e.g. with Microsoft and with Sun) and (b) kept the money received and refused to give Novell an accounting.

Here's what Judge Kimball stated last August:

> *Finally, the court concludes, as a matter of law, that the only reasonable interpretation of all SVRX Licenses includes no temporal restriction of SVRX Licenses existing at the time of the APA [Asset Purchase Agreement]. The court further concludes that because a portion of SCO's 2003 Sun and Microsoft Agreements indisputably licenses SVRX products listed under Item VI of Schedule 1.1(a) to the APA, even if only incidental to a license for UnixWare, SCO is obligated under the APA to account for and pass through to Novell the appropriate portion relating to the license of SVRX products. Because SCO failed to do so, it breached its fiduciary duty to Novell under the APA and is liable for conversion.*
>
> *The court, however, is precluded from granting a constructive trust with respect to the payments SCO received under the 2003 Sun and Microsoft Agreements because there is a question of fact as to the appropriate amount of SVRX Royalties SCO owes to Novell based on the portion of SVRX products contained in each agreement.*

Well, that seems clear. Yet, on 30 April, Stuart Singer, one of SCOG's lawyers, said: "We read the court's August 2007 order as leaving open for determination in this trial as to whether we owe Novell money."

So, depending on whom you trust, The SCO Group has "converted" between zero and $36 million, and "owes" Novell between $20 and $40 million. Their situation is not good.

In their traditional fashion, The SCO Group has requested "rescheduling" of several matters in the bankruptcy court, thereby delaying those decisions as well.

Your intrepid correspondent will report again, when ...

On another front...

On May 3, Microsoft "withdrew" its offer to Yahoo! The following week, Carl Icahn, a billionaire, made an announcement that a "rival" slate of directors would be standing for election at Yahoo!'s annual meeting. Yahoo! has stated that its "trial" arrangement concerning advertising – with Google – had worked well. And ... [wait for it] ... on Sunday, May 18, "Microsoft released a brief statement on Sunday disclosing the renewed talks, a surprising reversal just weeks after it withdrew its $47.5 billion bid for Yahoo and said it had 'moved on.'" [*New York Times*]

Whether (a) the Icahn slate will be elected; (b) the renewed talks between Microsoft and Yahoo! will amount to anything; etc. Should be know by the time you read this. My personal take is that Microsoft's "renewed talks" are designed to do nothing except disrupt whatever arrangements are in process between Google (Microsoft's true target) and Yahoo!.

As Microsoft has demonstrated repeatedly over the past two decades, it is incapable of producing and developing anything. All it can do is pilfer or purchase, borrow or blackmail.

(For those interested, Microsoft's stock closed on May 16, 2008 at $29.99, almost exactly 50% of its peak in December 1999.)

Meanwhile ...

In his *The Wealth of Networks* (2006), Yochai Benkler (of Harvard) argues that blogs and other modes of participatory communication can lead to "a more critical and self-reflective culture", where citizens are empowered by the ability to publicize their own opinions on a range of issues. Benkler raises the possibility that a culture where information was shared freely could prove more efficient economically than one where innovation is too-frequently protected by patent or copyright law, since the marginal cost of re-producing most information is effectively nothing. (This is the thread that Clay Shirky elaborates upon in his *Here Comes Everybody* (2008).) Let's look at the economics a bit more.

Ryan Paul has posted a report: (http://arstechnica.com/news.ars/post/20080425-study-70-percent-say-red-hat-more-secure-than-windows.html) indicating that:

> *The Standish group recently completed an extensive study that examines factors influencing open-source adoption. Based on five years of research and analysis, the report provides intriguing insights into open-source adoption levels and the way that open source is reshaping the software industry. Individuals who participated in the Standish survey identified several key drivers for open source adoption, including lower costs, better security and reliability, and faster development speed. . . .*
>
> *Over 70 percent said that Red Hat Linux is less vulnerable to security issues than Microsoft's operating system.*

Paul's article concludes:

> *Despite adoption challenges, the Standish report concludes that open source software is a big win for businesses, which are saving billions and passing that savings along to consumers. "Open source dominates web server installations and is an integral part of most e-commerce websites", the report says. "The open-source movement is advancing because of feature-rich, secure, high-quality, reliable software with compelling economic benefits."*

It's worth looking at that Standish report more closely. In an article posted in late April 2008, Martin Broersma wrote

http://software.silicon.com/os/0,39024651,39201838,00.htm

> *Open source software is successfully displacing proprietary applications in many large companies and eating into the annual revenues of proprietary software vendors by $60bn per year, according to research.*
>
> *According to the study from the Standish Group called Trends in Open Source, released this week, the losses of proprietary software makers are disproportionate to the actual spend on open source software, which is a mere six per cent of an estimated worldwide spend of $1tr per year. The researchers put this difference down to the fact a large proportion of open source isn't paid for – an intended result of the open source licensing structure. . . .*
>
> *The study, the result of five years of research, states if open source products and services were calculated at commercial prices, open source as a whole would be equivalent to the largest software company in the world, with revenues exceeding the combined income of Microsoft, Oracle and Computer Associates.*

And that's it. Open source is a big win. The news media hammer away at Microsoft's near-monopoly on the desktop. But that "near-monopoly" is ephemeral. In search engines, Microsoft barely exists where Google or Yahoo! dominate. Where accuracy and reliability are required, we find Linux.

In Formula One auto racing, for example, Jonathan Neale, the managing director of McLaren Racing, says: "Formula One is a product excellence business that's all about innovation and technology. We're competing for first place in an environment where the difference between first and tenth is about 0.6 seconds, so we're constantly seeking fractions of a second in performance improvement."
http://www.itpro.co.uk/applications/features/192522/linux-and-formula-one.html

"On average we'll make a change to the car every 20 minutes during the course of a season, and to do that, simulation is vital in making efficient changes to the car." Back in the factory, McLaren uses computational fluid dynamics (CFD) analysis running on Linux on SGI Altix high performance computers to simulate and predict the car's behaviour."

Microsoft is no longer running the industry as they once did. People don't talk about a new release of MS-Office with great anticipation... they talk about the coolest new web site, blog, or web-powered app. And millions play free and open source games.

Open Source has gone from being this hippy/weird/bad thing that CIOs resist, to an accepted part of the IT ecosystem. For example, nobody questions using ISC BIND or DHCP, and when a dot-com starts up they build a cluster of Linux boxes, not Windows servers. (Not to mention that Firefox is a major force now... and has an economic model that makes it self-funding).

In fact, it may well be "game over" for old-school corporations like Microsoft; though on a historical note, Julius Caesar was assassinated in 44 BCE. Rome was besieged, overrun and sacked by Alaric in 410 AD. There were nearly five centuries of decline. It won't take quite that long for the Microsoft empire to collapse, but it certainly won't be sudden.

Microsoft repeatedly shoots itself in the foot (Vista is a great example); free and open source systems, applications and games offer so much variety, that their luxurious growth overwhelms inferior "new" products.

Moreover, despite the assertion by Christopher P. Liddell, Microsoft's chief financial officer, that "There are no Vista-related issues at all", "Sales in the segment [of Microsoft] that sells the Office productivity suite and other business applications edged down 2 per cent to $4.75 billion US from a year ago" yet "researcher IDC showed PC sales exceeded forecasts in the quarter. PC shipments rose 15 percent."
http://www.cbc.ca/technology/story/2008/04/24/microsoftprofit.html

An anonymous blogger posted this version of the next few years:

> *Microsoft gets fixated on Yahoo, and ignores the rest of their business. The battle is long and hard, but Yahoo eventually merges with some other company, foiling Microsoft's bid. Meanwhile, Microsoft rushes out "Windows 7", which ends up being little more than Vista Service Pack 2. Windows 7 is another flop. The Microsoft executive offices turn into a battle ground, with everyone blaming everyone else for the disaster. By the time things settle down, Microsoft's OS and office software have declined to 60% market share, and it's a whole new world in the computer industry.*

I'm not at all certain that I agree with all of this, but the last four items seem extremely likely.

If one looks at the automobile industry, for example, in 1985, 1,530,410 "new vehicles" were sold in Canada. In 2005, there were 1,630,310. However, in 1985, 794,965 were passenger vehicles manufactured in North America, whereas in 2005, there were 574,639. So, while sales were up by 6.5%, sales of North American manufactures were down by nearly 28%. Looked at from the viewpoint of the US Department of Commerce, GM's share, 36% in 1990, stands at 26% in 2007. Ford's 1990 share, 24%, is now 17%.
http://www.statcan.ca/english/freepub/63-007-XIE/2008002/t015_en.htm

This is the fate I imagine Microsoft has in store as the prime example of closed, proprietary software, paralleling. The 'long tail' for Microsoft will be their waning business.

At MakerFaire, the Institute for the Future distributed its "Future of Making Map." The first point runs:

> *Many of the best ideas may come from unexpected contributors, including those who are so far outside the organization's own walls that they speak a different language. To succeed in the future, organizations will need to look to both internal and external sources of innovation. Seeking an outside-in perspective on internal challenges may require long-held processes to be rethought, from the design cycle to R&D budgets to intellectual property strategies. Once open to the idea of a networked organization, it's relatively easy to identify and engage with external networks of exceptional people through community R&D platforms such as Instructables, InnoCentive, and NineSigma.*

While it is hard to translate others' words, I read "a different language" to include the language of cooperation as opposed to that of hierarchic direction.

Centralized Package Management using Stork

Justin Samuel, Jeremy Plichta, and Justin Cappos

Managing the soft ware installed on multiple systems can be one of the duller aspects of system administration. One has to deal with varied sets of packages, each replicated on numerous machines, and bring up new systems, with the complication of those that are almost like the others, but not quite. In many cases, great amounts of time could be saved and more than a few mistakes avoided by using tools specifically created to make this job easier.

A Better Way

Picture an admin sitting down to upgrade a certain package on 40 boxes, install new software on 100 others, and remove an unused package from every box on the network. Rather than undergo hours of tedium, the admin starts up her Stork management tool and five minutes later she's done. She knows that her systems are hard at work applying the changes she's specified. What about the systems that are currently offline? No problem. They'll apply the changes the next time they come online. And newly purchased systems? They'll be automatically updated to have the correct configuration.

Stork is the name for a collection of package management tools. Among the features it provides are:

- Central management of packages that should be installed on a distributed set of computers.
- The ability to organize systems into logical groups for ease of management.
- Increased speed and reliability of file transfers by utilizing a variety of efficient and redundant protocols.
- Secure architecture utilizing public-key cryptography for digital signatures.
- Low-upkeep repository not requiring a central repository administrator.
- Space, network bandwidth, and CPU savings in some virtual machine environments.
- Fast and efficient update awareness through the use of a publish/subscribe system for update notification.

In this article we'll focus on how to use the Stork tools for centralized management.

How Stork Works

There are three fundamental components of the Stork architecture:

- Client tools: These are package dependency resolution tools similar to yum and apt. In addition to dependency resolution, they also follow instructions from signed configuration files.
- Management tools: The management tools are the utilities administrators use to create the signed configuration files that act as the instructions to the client tools. The management tools are generally used by an administrator on his or her own computer.

- Repository: A repository is a Web server where configuration files created by the management tools are stored so that the client tools can access them.

Administrators can use one of the management tools to specify which packages should be installed on a node. The management tools create the necessary configuration files, digitally sign them with the user's private key, and can even upload them to a repository.

The client tools, running on each machine, determine which packages to install, upgrade, or remove based upon the contents of the configuration files the administrator created. If the actions require downloading package files from the repository, the client tools also verify that the package files to be installed are themselves trusted according to the configuration files.

Currently, the client and management tools run on UNIX-like operating systems, with active usage including Fedora, Gentoo, and Arch Linux. The package formats supported at this time are RPMs and tarballs, with deb file support in development. The client tools can wrap around any existing package management utilities that are installed on a system, so support can be added for any other package formats a user may desire.

Signatures and Trust

By having the client tools ensure that every configuration file downloaded from the repository is signed by a trusted administrator, the client tools can be certain that no configuration files have been tampered with. This allows unrelated organizations to safely share the use of a single repository: The client tools do not trust files because they are signed by a repository key, but rather because they are signed by an actual administrator's key.

As verifying a digital signature requires knowing in advance the public keys whose signatures should be trusted, this begs the question: How do the client tools know which public keys those are? There are two main approaches to making sure the client tools know which administrator keys to trust. These are not mutually exclusive.

The first approach is for an administrator to directly configure the client tools to trust specific keys. For example, trusted keys can be configured at the time the client tools are installed on each system. After that initial setup, new keys can be securely distributed via packages installed using Stork, for example.

The other approach that can be used is to integrate the client tools with an existing method an organization has to distribute or to answer queries for such keys. For example, PlanetLab [1] makes available the ability for a system to fetch the keys that belong to that system's administrators. Since a module in the client tools makes use of this secure PlanetLab API that provides access to these keys, the client tools can be used on PlanetLab without the need to manually distribute trusted keys to each system.

The client tools can be used in environments with multiple administrators. To accommodate this, the client tools will ensure that, for any given configuration file, they are using the most recent version of the file in the repository that has a valid signature of a trusted administrator. Thus, when another administrator uploads a new version of the file in which the groups are defined, the client tools will use the newer one.

Organizing Systems into Groups

An important optimization for managing systems is a group. A group is simply a logical association of multiple systems defined by the administrator of those systems. The fact that groups are purely a logical organization is very important: The systems in a group do not need to be connected in any way and a single system can be a member of multiple groups. Groups exist for the sole purpose of simplifying the job of the administrator, who may want multiple systems to have some of the same packages installed.

Stork has three types of groups: simple, composite, and query result.

Simple groups allow users to manage a collection of systems as if they were a single system. One can declare that machines 1, 2, and 3 are part of group G and can then simply say that package X should be installed on group G. This would result in each of machines 1, 2, and 3 installing package X. There is no limit to how many systems can be in a group.

Composite groups are created by performing an operation on existing groups. The supported operations for combined groups are UNION, INTERSECT, COMPLEMENT, and DIFFERENCE. Composite groups

do not have systems directly added. Instead, the systems in the group are chosen by the operation and the membership of the other groups. For example, combined group H may be defined as the UNION of group I and group J and will contain all of the systems appearing in either group I or group J. Groups I and J can be any types of groups, including composite groups.

Query result groups are computed based on some property of the network. PlanetLab is a major user base of Stork and therefore the management tools provide support for building groups from the result of CoMon [2] queries. For example, a group to refer to all nodes with more than 1 GB of free disk space could be created using the CoMon query `select='gbfree > 1'`. Queries are evaluated at the time of group creation. Administrators who wish to have query result groups that automatically update can reevaluate the query result groups periodically via a cron job.

Management Tools

There are two main management tools for administering systems that are running the client tools: a graphical tool (a.k.a. the GUI) and a command-line tool called storkutil. These tools make it easy for administrators to create the signed configuration files that need to be uploaded to the repository. Both tools are cross-platform and require that python and openssl be installed. Let's first take a look at using the GUI to manage systems.

The information that the GUI asks for when it is first started is a username and password that will be used to authenticate with the repository, as well as the location of a private key (with optional password). The private key is used to sign configuration files. The corresponding public key is what the client tools will use to verify that the signatures on downloaded configuration files are legitimate.

After the GUI verifies the repository login information and validity of the private key, the user can begin configuring the groups and desired package installation actions. The GUI retrieves the latest configuration files from the repository and displays any groups and package installation actions that were previously defined. To add a new group, click on the "add group" button and give the group a name. Next, proceed to either add nodes to form a simple group, make this a combined group by defining it as a union or intersection of other groups, or make it a dynamic group by setting a network property such as a CoMon query.

After the new group is created, it important to define package management actions for the group. That is, define packages that should be installed, upgraded, or removed for all of the nodes of the group. To install or upgrade a package, either specify the name of the package (e.g., "gnupg") and let the client tools, when they run on the systems, attempt to find a gnupg package file that is trusted or provide a gnupg package file that is stored locally. When providing a package file that is stored locally, the GUI will take care of adding trust of this specific package file to the configuration files in addition to uploading the package file to the repository so that it is available.

If any changes are made in the GUI, an icon shows that the local state is out of sync with the repository. One can then sync with the repository, which means that the new configuration files and any newly added package files will be uploaded to the repository.

The GUI is the most convenient tool for day-to-day management of systems using the client tools. However, some situations require command-line tools that can perform these tasks. All of the same functionality for configuration file modification and generation that is done by the GUI can be done using the command-line tool `storkutil.py`.

Here's an example using storkutil.py to add a new group with the name `EXAMPLE_NODES` with one system, `nodeA.example.com`, in the group:

```
storkutil.py pacgroups include EXAMPLE_NODES nodeA.example.com
```

Then, to add more systems to the same group:

```
storkutil.py pacgroups include EXAMPLE_NODES nodeB.example.com \
nodeC.example.com nodeD.example.com nodeE.example.com
```

Finally, to say that all of the systems in the new group should install gnupg:

```
storkutil.py pacpackages group EXAMPLE_NODES install gnupg
```

The command-line tool storkutil.py will have taken care of generating the configuration files as well as signing them with the private key. The one thing that the GUI does that storkutil.py doesn't is upload the files to the repository for us. Uploading the files to the repository is something that can be scripted, if desired (for example, using curl).

After the configuration files are generated and uploaded to the repository using either the GUI or the command-line tools, the client tools running on any of the nodes will retrieve these newest files and take any actions necessary based on them.

Where Stork Can Help

Stork's design makes it quite flexible in terms of the roles it can perform for an administrator. However, its primary focus, and that for which it is optimized, is centralized management of many systems.

It is common for administrators to have a base configuration for all of their systems. Various systems may then have specialized software needs depending upon the additional job requirements of the people using those systems or the services they provide. For example, everybody in the finance department may need a database program that others in the organization do not. Further, all of the secretaries and executives may require a specific calendar management program. Stork makes it easy to manage the software installed for different groups of users. You can even allow a system to be in multiple groups, thus allowing the finance department's secretary to have both the database and the calendar management software installed.

Stork's system of secure file transfer, fast and efficient propagation of changes, and ability to use tarballs as packages make it very useful not only as a package management system but also as a configuration management system. For example, suppose an administrator wants to install a set of custom scripts that cron will run periodically on all of his systems to monitor disk usage. The administrator can package these scripts as a tarball and use the management tools to install the tarball on all machines. The fact that Stork manages tarballs in a similar way to RPM packages gives the administrator considerable control and flexibility without the overhead that can be involved in using traditional package formats to meet these needs. The tarball packages can be easily checked for installation, removed cleanly, or even used to satisfy dependencies.

Incorporating package file security decisions made by other users is simple with Stork's user trust system. This trust system allows an administrator to include, in real time, another user's repository-published list of untrusted packages into his or her own list. This can be very useful in many organizations with a separate IT department that focuses on security. Although this department may publish lists of packages that should not be installed because of known security problems, traditionally it can be hard for system administrators to keep up with the latest changes to such published lists. However, by being able to automatically include the other department's package trust decisions, no further work is needed to stay current with the list of packages to avoid, which can consist of packages marked as untrustworthy by name, version, or hash. With this setup, the system administrators can know that Stork will automatically prevent these untrustworthy packages from being installed.

Other Solutions

There are other tools available for performing centralized management, but none are intended to work the same way as Stork. These other systems are generally configuration management systems that can be used for some degree of package management.

One method used for configuration management is remote command execution. The tools that provide the ability to remotely execute commands on a set of nodes all utilize multiple secure shell connections that originate from the administrator's own computer. Of these, many are oriented toward specific networks, such as PlanetLab. One such system is pssh [3], which provides parallel versions of the openssh tools, including versions of ssh and scp. Using pssh, an administrator could execute a package management command on all of their active systems simultaneously. Major disadvantages with this approach include having to manually repeat the process for new systems brought online or for systems that were previously down and have been brought back online; limited groups functionality (done by utilizing files that contain lists of hosts), but having no functionality similar to Stork's composite groups; and an administrator having to make potentially hundreds or thousands of shell connections directly from the administrator's own computer in order to install, upgrade, or remove packages.

Other solutions that provide similar functionality, all using this same approach, include:

- Plush [4], which includes a graphical interface.
- pShell [5], a command-line-only and PlanetLab-specific tool.
- PlMan [6], which includes a graphical interface and the ability to find PlanetLab nodes by CoMon queries.

Although package management on large numbers of systems would be easier using these tools compared with having to manually access each system to do so, none of them focuses on providing efficient and easy-to-use centralized package management. Using them for this purpose would be far from ideal, as none of them handles common cases such as nodes that are down at the time a command is issued or automation of setting up new nodes that come online.

A more robust way to perform configuration management is to use a powerful configuration management system such as Cfengine [7]. Configuration management systems usually involve a central server that provides configuration information to all nodes, where this configuration information is often in a format or language specific to the system. These systems provide considerable general-purpose system administration functionality, such as ensuring that certain files on all nodes are the same, configuring backups, and instructing nodes to perform basic package management actions such as installing a specific package.

Configuration management systems, however, often lack the specialized functionality for secure, centralized package management that Stork provides. For example, Stork can be used securely even when an administrator does not run his or her own repository but instead uses a shared repository. As no actions are taken by the Stork client tools unless the signatures of configuration files are verified as having been created by an administrator, the security of the nodes using the repository does not depend on the security of the repository itself.

Stork is not intended to be a configuration management system, just as tools such as Cfengine are not intended to provide the package management functionality that Stork does. In general, Stork provides a high degree of flexibility and security for centralized management, with the focus being on package management. The package management functionality of Stork, which includes very easy-to-use and lightweight packaging by means of tarball packages, allows it to function as a configuration management system in many cases.

Conclusion

For administrators responsible for multiple systems, centralized package management such as that provided by Stork can save time as well as prevent frustration and mistakes. An administrator need only define once the package management actions the systems should perform. Using Stork alleviates much of the burden of administering large numbers of systems. For more information or to download Stork, please visit the Stork Web site at

http://www.cs.arizona.edu/stork

REFERENCES

[1] A. Bavier, M. Bowman, B. Chun, D. Culler, S. Karlin, S. Muir, L. Peterson, T. Roscoe, T. Spalink, and M. Wawrzoniak, “Operating System Support for Planetary-Scale Network Services,” Proceedings of NSDI ’04 (USENIX Association, 2004).

[2] K. Park and V.S. Pai, “CoMon: A Mostly-Scalable Monitoring System for PlanetLab,” ACM Operating Systems Review, 40(1) (January 2006):
http://www.cs.princeton.edu/nsg/papers/comon_osr_06/comon.pdf

[3] pssh:
http://www.theether.org/pssh/

[4] J. Albrecht, C. Tuttle, A.C. Snoeren, and A. Vahdat, “PlanetLab Application Management Using Plush,” ACM Operating Systems Review, 40(1) (January 2006):
http://www.cs.williams.edu/~jeannie/papers/plush-osr06.pdf

[5] “pShell: An Interactive Shell for Managing Planetlab Slices”:
http://www.cs.mcgill.ca/~anrl/projects/pShell/

[6] “Planetary Scale Control Plane”:
http://www.cs.washington.edu/research/networking/cplane/

[7] M. Burgess, "Cfengine: A Site Configuration Engine," USENIX Computing Systems, 8(3), 1995.

*Originally published in **;login: The USENIX Magazine,** vol. 33, no. 1 (Berkeley, CA: USENIX Association, 2008). Reprinted by kind permission.*

Rails for PHP Developers: Refining Web Applications

Derek DeVries and Mike Naberezny

Pragmatic Bookshelf
ISBN: 978-1-934356-04-3
430pp.
***£* 22.99**
Published: 11th February 2008

reviewed by Paul Waring

As someone who relies on web development for the majority of my freelance work, I have been intending to get up to speed with this Ruby on Rails thing which everyone has been talking about for some time. Unfortunately, with several years of (possibly bad) PHP habits ingrained into my memory, picking up a new way of thinking from the existing Ruby on Rails books has been quite tricky, so a text which proclaims to be 'Rails for PHP Developers' sounds like the ideal solution to my problem.

The book begins as might be expected, with a short history of Rails and a brief nod in Ruby's direction, plus an overview of the Model View Controller (MVC) pattern. Thankfully, the authors stay clear of evangelising at this point – simply pointing out the main aspects of MVC rather than launching into a sermon – which makes a refreshing change from some other Rails books that I've read. The section on installing Rails is also kept brief, again a welcome change from some books (not just those focusing on Rails) which often pad themselves out with an entire chapter on installation, when all that is required is to run one or two lines in your package management software or download a setup file.

Moving swiftly on, I'm impressed to see my first Rails app up and running by page 11 – instant gratification if ever there was such a thing. Admittedly it doesn't do much, but by the end of chapter one I've followed the examples and got a functioning mailing list subscription form up and running, using only a tiny amount of code. Later chapters are a little bit disappointing as the pace slows drastically when new concepts from Ruby which don't exist in PHP are introduced, and you have to wait until chapter five before things start going again.

The chapter which I found the most useful however was 'Deploying the Application'. With PHP, I'm used to just copying the code to a server with Apache and `mod_perl` installed and having it 'just work', whereas Rails applications require significantly more effort. Thankfully, the book takes you through setting up and deploying a Rails application in a step by step process, in one of the most clear and concise ways I've seen.

In addition to specific pieces of content, the great thing about this book is that it really does do what it says on the tin. Each Rails example is preceeded by the corresponding PHP, which makes it easy to see how a construct in one language maps into the other, and suddenly the move from PHP to Rails doesn't seem so difficult. There are also plenty of side boxes which explain why something is different in Rails, or why some tasks are possible in one language but not the other.

My only minor criticism of the book is that in one or two places the authors have used what feels like deliberately poor PHP in order to emphasise the advantages of Rails, but this doesn't detract too much from the overall quality. Furthermore, the authors do admit that frameworks exist for PHP, so if you decide that Rails isn't your cup of tea you can still do similar things in the language you're already comfortable with.

Overall, I'd suggest that if you're a PHP programmer who wants to get up to speed with Rails without moving too far from your comfort zone, this is the book for you. Anyone else should probably look elsewhere though as this book is really only useful if you fit the exact niche which it is aiming for.

The ThoughtWorks Anthology: Essays on Software Technology and Innovation

ThoughtWorks Inc

Pragmatic Bookshelf
ISBN: 978-1-934356-14-2
248pp.
***£* 24.50**
Published: 17 Mar 2008

reviewed by Paul Waring

A collection of essays written by ThoughtWorks employees, this book initially caught my eye with its promise of 'insightful essays on modern software development practices'. Having worked on numerous pieces of custom software, I was hoping for some gems of wisdom and experience which I could apply to existing and future projects.

The essays contained within this book cover a wide variety of topics, some focusing on technical issues with lots of source code examples, others straying more towards management problems. The management essays include a chapter on 'Iteration Managers' – having read this essay several times, I'm still not sure exactly what these are (some sort of hybrid between a project manager, personnel manager and team member it would seem). Other essays cover the 'last mile' of software development, various types of testing and domain annotations.

As someone who has had to work with huge Ant `build.xml` files in the past, I found the essay 'Refactoring Ant Build Files' particularly interesting. The breakdown of over a dozen different ways to refactor Ant build files to make them either shorter or easier to read (or both), backed up with simple examples, is well thought out and definitely worth a read for anyone who has had to deal with this sort of situation. The Lush Landscape of Languages also managed to tread a careful and interesting path through the different types of programming languages, pointing out that each one is good for a specific problem, but no one language is the best choice for every situation.

However, despite some useful and interesting essays, the main problem I had with this book was that there was no central theme running through all of the contributions - the only aspect which they had in common was the fact that they were all authored by ThoughtWorks employees. Whilst this lack of a central theme does mean that you can skip over essays which don't cover your area of interest without losing out on any crucial details, it also means that the book has no overall message beyond 'you should do things the ThoughtWorks way'. Based on this, my overall feeling was that the book was a bit of a mismash and lacked any particular focus. Having essays on diverse topics means that most people may not get a great deal out of the book, as only a few essays will be relevant to their work.

In summary: the individual essays are generally good, but the book as a whole needs a stronger theme. If you want to see whether there are enough interesting essays for you, the table of contents and a sample essay ('One Lair and Twenty Ruby DSLs') are available online on the Pragmatic Programmers website.

Making Things Happen

Scott Berkun

O'Reilly Media, Inc.
ISBN: 978-0-596-51771-7
408pp.
£ 24.99
Published: 25th March 2008

reviewed by Roger Whittaker

This is a revised second edition of a book that came out a couple of years ago under a different title: the original work was called *The Art of Project Management*. I asked to be allowed to review it because I had read, reviewed and enjoyed Berkun's *The Myths of Innovation* (in the September 2007 issue). I would not have chosen to read this book (or indeed go anywhere near it) under its previous title. However, I was impressed.

This book is hard to categorise. It is a bit like a "self-help" book, except for the fact that it is actually helpful. It is a bit like one of those books by management *gurus* with words like "raincoat" and "excellence" in the title, but is actually readable and useful.

Berkun worked as a project manager at Microsoft for several years. Don't let that fact put you off. The book does not discuss GANTT charts or tell you how to use Microsoft Project: such things are not even mentioned.

Although the examples that the book uses are mostly from the world of programming, much of what he says is applicable to almost any situation where one person is responsible for getting other people to do something: "making things happen". Berkun strikes me as one of those people who has had to think hard not only about management but also about all aspects of how human relationships work. He writes:

> *. . . it took me years to understand the value of talking to people in the workplace . . .*

Here he sums up what I suspect he is: one of those rare people who goes from socially inept to personally effective through developing an intellectual understanding of how the social world works.

Berkun seems to have started out with a relatively "geeky" personality; he even mentions studying Gödel's theorem at college; he found himself in a position where he had to manage others: this book is the result of the insights he gained from that, and a lot of deep thought.

Early in the book he compares groups of people involved in various different types of endeavour: software or web development teams, kitchen staff in a busy restaurant, medical staff dealing with emergencies and people making films in Hollywood. He points out the similarites between the problems they face and the processes that they have to put in place. This is therefore a general book about management, but it is a book about management written by "one of us" for "people like us", and using examples and experience from the world of software engineering.

Not surprisingly, having worked for a large corporation like Microsoft, Berkun is very alive to the way that the organisational culture can take over and colour and affect the whole way that people work. He stresses the importance of real communication as against the usual "corporate communications" and says:

> *Face to face is the best way to tell people you appreciate their work.*

That sounds obvious, but anyone who has worked in a large corporation will be very well aware that it is not obvious to a large proportion of their managers.

I would like to quote with approval a lot more from this book, but here are a couple of nice section headings that I particularly liked: "*A catalog of lame vision statements (which should be avoided)*" and "*How to run the non-annoying meeting*".

There are exercises at the end of each chapter: some of these might at first sight appear wacky and strange, but Berkun is trying to make people look at their situation (or an analogous but quite different one) with new eyes. He explicitly mentions Zen Buddhism once or twice: the type of insight that he is trying to encourage is one that involves jumping out of the situation that you are in and seeing it from a radically different pespective.

The author's background and personality make this book a useful one for anyone who needs to manage others, but particularly for people who do not consider themselves "naturals" at management and relationships: the author is honest about the work that he had to do to break through his former naïvety and reach the point where he could write this book. Because he has gone through that process, he is in a position actually to tell us something useful.

Programming Amazon Web Services: S3, EC2, SQS, FPS, and SimpleDB

James Murty
O'Reilly Media
ISBN: 978-0-596-51581-2
600pp.
£ 30.99
Published: 25th March 2008

reviewed by Roger Whittaker

Until not so long ago, the phrase "Amazon Web Services" meant the ability to: use public APIs to communicate with Amazon's servers and get book information allowing you to run your own shop-front with Amazon sitting behind it, for example.

But Amazon has not only expanded out of bookselling into general retailing: it has also dramatically increased the scope of the public web services that it offers.

As Amazon built up its global infrastructure across multiple datacentres, it set about abstracting the services these provide for its own purposes, so that storage space, virtual server instances and systems for dealing with transactions could all be accessed and created through a set of web APIs.

It is these services which Amazon has opened up to the public, and which allow you to run your public services on Amazon's infrastructure rather than your own, at competitive rates. This is Amazon's version of "cloud computing": you pay for it by the hour or by the megabyte.

S3 is the *Simple Storage Service* which allows you to create and access unlimited data storage areas. EC2 is the *Elastic Compute Cloud* that allows you to create Xen virtual server instances on the fly: these are typically Linux systems, but the availability of OpenSolaris on EC2 was recently announced. SQS is Amazon's *Simple Queue Service* a message-passing infrastructure. FPS is the *Flexible Payments System* and SimpleDB is Amazon's *Simple Database*.

Rather than providing their own visible front-ends to access these services, Amazon have simply published a set of web APIs. REST and SOAP APIs are available: this book deals only with access over REST, and uses Ruby for all the code examples.

The book describes in detail the steps needed to access and use each service. I have not been in a position to test any of this, but the book is large, detailed and clearly written. The explanations of the principles involved are clear, and the author explains how you will need to "think like Amazon" in order to effectively make use of the cloud infrastructure that they originally developled for theor own purposes.

I find it fascinating and exciting that these services are on offer, and hope to test at least EC2 with the help of this book. I may report back at that stage on whether "I have seen the future and it works".

Contributors

Justin Cappos is a Ph.D. student at the University of Arizona. His research interests revolve around building large distributed systems.

Dave Crossland is currently doing GNU/Linux consultancy, undertaking the MA Typeface Design programme at the University of Reading and plans to pioneer software freedom for fonts when he graduates.

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

Khusro Jaleel built his first OpenBSD firewall in 2003 and has since been very happy with using OpenBSD and PF for firewalling purposes. He is an experienced Unix/Linux Systems Administrator and also holds a Masters degree in Software Engineering. His main interests are Operating Systems, High Availability and Systems Programming, particularly in C.

Richard Melville works for Cellularity Limited (`http://www.cellularity.co.uk`), a company which works on low-power, solid-state systems targeted at small companies, mainly in the "not-for-profit" sector. The aim is to provide a completely silent environment, with no local hard disks and no fans and without resorting to thin clients. An additional benefit is the minimal electrical power required to support such an environment. Cellularity is building a small, password-free, Linux-based OS with the emphasis on stability and security. Although "small" there will be sufficient packages installed, both client and server, for a business to function fully. This OS will then be installed on a solid-state micro-disk module.

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.

Jeremy Plichta is a senior majoring in computer science at the University of Arizona. He plans on pursuing a career as a software engineer after graduating.

Paul Robinson is a software developer, entrepreneur and occasional writer. With a background in BSD Unix, he has worked in R&D, systems administration, embedded systems programming and management consultancy. He lives and works in central Manchester and is a well-known member of the local technology community, having recently organised BarCamp Manchester.

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.

Justin Samuel is an undergraduate student at the University of Arizona with an interest in security. In addition to research, he teaches a secure Web application development course.

Paul Waring is a PhD student at the University of Manchester and the newest member of the UKUUG council. He is also responsible for organising the UKUUG Spring conference in 2009.

Alain Williams is UKUUG Chairman, and proprietor of Parliament Hill Computers Ltd, an independent consultancy.

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

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

Paul Waring
Council member
Manchester

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

Roger Whittaker
Newsletter Editor
London