flossUK

news@UK

The newsletter of FLOSS UK, the new name for the UK's oldest Open Systems User Group, UKUUG

Published electronically at http://www.flossuk.org/Newsletter

Volume 22, Number 4 **ISSN 0965-9412** **December 2013**

Contents

From the Secretariat

Jane Morrison

The minutes from the UKUUG Ltd. Annual General Meeting held on 20th September have once again not been circulated to members via hard copy. All members were emailed on 7th October and advised that the minutes and associated documents could be found on the web site at: **http://www.flossuk.org/Events/AGM2013**

Of course anyone who wants a printed copy should contact me here at the office.

Current Council members are: Kimball Johnson, Holger Kraus, Ian Norton, Stephen Quinney, Quentin Wright and Gavin Atkinson and Tim Fletcher (co-opted on 24th September).

We thank retired Council member, Paul Waring for his past commitment.

We are currently putting in place a full schedule of events for 2014 – to date the following have been agreed:

- 'Git for Development Teams' Full day tutorial by Lorna Jane Mitchell – Thursday 6th February 2014, London - see flyer inserted with this Newsletter
- 'Tuning PostgreSQL for a large scale web environment' – Full day tutorial by Simon Riggs – Wednesday 26th February 2014 – London – see flyer inserted with this Newsletter
- Spring 2014 Workshops & Conference – 18th, 19th & 20th March 2014 Brighton - see preliminary information and booking form inserted with this Newsletter.

We are also planning more tutorials for 2014 - if you have any particular topics that you think would be of interest please let us know.

I should like to remind you of a couple of benefits we have in place for members – we have a page on the web site – **http://www.flossuk.org/Consultants** – which lists members who can provide consultancy services. This is free to members – take a look and send me your details if you are a consultant and get some advertising for free.

We also provide other discounts for members: these are detailed later in this Newsletter.

The annual membership subscription invoices will be sent out in January, please look out for your invoice and as always prompt payment will be gratefully received!

We would like to thank the continued support of our Gold Sponsor member, SUSE and the other companies that have provided generous sponsorship of our events this year.

I should like to take this opportunity to wish you all a very Happy Christmas and a Peaceful New Year.

The Secretariat will close on Tuesday 17th December and re-open on Thursday January 2nd 2014.

Please note the copy date for the next issue of the Newsletter (March 2014) is 14th February.

We are always looking for interesting submissions from members, so if you have any news, articles etc. please send copy direct to **newsletter@ukuug.org**.

If you do prefer not to receive future issues of the Newsletter on paper (all issues can be found on our web site in pdf format) please let me know.

Chairman's report

Kimball Johnson

Council

Firstly I would like to thank Paul Waring for his service to Council over the last 6 years, It has been wonderful to work with him, and I look forward to his continued presence at the conferences as a delegate. I would also like to welcome our new members, Gavin Atkinson and Tim Fletcher. I know both have a lot to bring to Council, and I am sure they will enjoy their time working to make FLOSS UK the best it can be.

Events

Since the last newsletter we held the annual Unconference in London. It was an excelent day, with so many talks we actually ran out of time, leaving Adrian Kennard only 5 minutes for the final talk before we had to leave the venue. Many thanks to all who attended and spoke, and special thanks to Andrews and Arnold (aa.net.uk) for their generous sponsorship and to the BCS OSSG for organising the venue in London.

The next major event is the Spring Conference, this time to be held in Brighton. As I write this the ticket details are being finalised, and the full details will be on our website. There are a number of talks already confirmed, but if you have a great idea, please submit it as I am sure we can squeeze a few more in.

Before the conference we have arranged for some more courses, both Git for Developers by Lorna Jane Mitchell and PostgreSQL for Large Scale Web Environments by Simon Riggs. I would also like to encourage anyone with ideas for courses they would like to be run to email the office with their ideas.

Support for local user groups

The budget is still available for FLOSS UK to assist local user groups by helping them to obtain speakers for their events and assist with travel expenses. In addition we have a small budget for assisting with projects that would benefit the Free Software or Free Hardware communities in some way. If you have an idea and wish some support, please contact Jane on `office@flossuk.org` and it will be discussed by Council.

Get Involved

FLOSS UK exists to serve its members and we are always on the lookout for people who are keen to get involved in any capacity, whether that be through volunteering to help with organising events, writing newsletter articles, or entirely new activities which haven't been tried before. If you would like to help out in any capacity please do get in touch via `office@flossuk.org`.

Unconference: 26th October 2013

Roger Whittaker

This year's unconference was hosted by the BCS Open Source SG and held at the BCS building in London. The event was sponsored by Andrews and Arnold Limited (`http://aa.net.uk/`).

The day started with a session where various people offered talks the details of which were written on "postit" notes by Kimball and stuck to a whiteboard: the talks were voted on and put in order of popularity. The process was uncomplicated and easy, and led to a day full of

interesting talks, some of which had been prepared in advance, or for other events, and some of which were genuinely off the cuff knowledge transfer from experts in a field.

The talks started with Adrian Kennard of Andrews and Arnold, who gave a very useful (and completely off the cuff) talk that could be described as ”what everyone needs to know about IPv6”, or even perhaps ”everything you ever wanted to know about IPv6 but never dared to ask”. Adrian’s knowledge was practical and clearly expressed, coming as he does from an ISP that offers IPv6 blocks to customers as a standard offering. The talk certainly cleared up quite a few of my own confusions on the subject, and was very well received by the audience, with many questions afterwards.

Cornelia Boldyreff of BCS gave us a description of the activities of the BCS OSSG, which holds events both in London and around the country, some of which sounded very interesting. The group runs events on open source topics of general interest as well as conferences aimed at advising on government procurement. It is not necessary to be a BCS member to attend any of these events. See: **`http://ossg.bcs.org/`**.

Phil Hands then spoke about git-annex and the git-annex assistant which allow one to track large files and binary data using git without actually putting them into the git repository.

Two talks on Python followed. Cornelia Boldyreff gave a general overview of Python and why you should use it, and Andrew Richards gave a useful survey of some the available books for learning Python, as well as one or two valuable web resources. His views on all of these were based on how useful he had found them for his own extensive and practical needs when learning.

Kimball Johnson talked about solving the well known ”Inglenook Sidings” shunting puzzle (**`http://en.wikipedia.org/wiki/Inglenook_Sidings`**) programmatically, but going further and taking up the challenge of creating an Arduino controlled layout to demonstrate the solutions in practice.

An excellent buffet lunch and time for individual chat and discussion followed.

After lunch, Kay Dudman demonstrated some animations made using Scratch, both for the purpose of teaching programming via Scratch and also to illustrate algorithms graphically: one example given being ”bubble sort”.

Phil Hands spoke about the EOMA-68 CPU card from Rhombus Tech: the brainchild of Luke Leighton. This device is in a PCMCIA card sized format, and has an Allwinner A10 processor with 2G RAM, and USB3, SATA, ethernet connections and an SD card slot. Although the current number of these devices in circulation is small, Phil noted that another batch is to be produced, and that a KDE tablet based on the EOMA-68 is being planned.

Dominic Cleal of Red Hat spoke about the Foreman systems management tool, and its use together with other tools. Foreman is a lifecycle management tool that integrates with configuration management tools such as puppet for installation and configuration of physical and virtual servers, with an attractive web interface and REST API.

Jasper Wallace of London Hackspace spoke about DNSSec and the importance of its implementation. He mentioned DANE (DNS-based Authentication of Named Entities) as well as how DNSSec can be integrated with other services including SMTP and SSH.

Carles Pina of Mendeley spoke last year on ”olefactory notifications”: that was a hard act to follow, but he managed to amuse us further with ”MIDI madness” – data ”audialization”: the conversion of data to music. This led to a number of anecdotes being shared among the audience, particularly old stories about detecting what various legacy systems were doing by listening to the noises they were making.

Alain Williams described a problem with encrypted disk partitions that he had wished to solve for his customers to enable secure backups, and how he had used udev rules to solve it.

Jan Kim of the Pirbright Institute gave an interesting talk about the use of BioPython to solve problems in genetics by comparing and processing genetic sequence data.

Craig Gallen spoke briefly on OpenNMS and the work being done partly in association with Juniper Networks on improving its network visualisation capabilities.

For the final talk, Adrian Kennard spoke again - his talk on ”printer drivers” had gathered few votes when offered. But what he described was both extraordinary and interesting: how he used ghostscript and other standard open source printing technologies to create multiple virtual printer drivers under Linux to drive the Epilog laser engraving machine, which can both cut plastic and engrave on surfaces. Different virtual drivers were used to run the machine in different modes, and Adrian showed us impressive examples of the results.

Social discussions continued after the end of the day.

This was a most enjoyable event, and proof that the unconference format can work very well for our audiences. The venue was ideal, and our thanks go out to Andrews and Arnold for sponsoring the event and to BCS for providing the venue and event registration.

Debian Med Sprint announcement

Tony Travis

The meeting of Debian Med developers that has now become an annual event is a great way to learn from and collaborate with fellow coders and package maintainers with the common goal of enhancing Debian and her derivatives (Ubuntu, Bio-Linux, . . .) while learning as much as we can from each other. If you want to get involved with Debian Med or any of the other projects represented at the meeting then this is a perfect opportunity. The programme is set by the participants but will consist of a few presentations from invited speakers and informal talks plus plenty of time to hack code together in small groups. While the tasks you work on are up to you, we want to set and work towards specific achievable outcomes (see the Wiki, and please add your own suggestions).

The weekend takes place in an out-of-season seaside hotel, which gives this meeting a unique atmosphere. Coding often goes on well into the evening as new ideas are explored and tested. We also think the sea air puts us in the right frame of mind for this sort of thing

We aim to keep costs down, to encourage volunteers to attend, and thanks to sponsors Minke Informatics we have discounted the bed-and-breakfast rate to 55 a night (or just 32.50 each if you want to share a twin room). We also expect to cover lunch and refreshments through sponsorship.

Please see the Wiki page here:
`https://wiki.debian.org/DebianMed/Meeting/Aberdeen2014`

We hope that many Debian-Med hackers can make it, and we’ll also be inviting local developers and potential future contributors to come. To sign up you need to put your name down AND book in with the hotel by 31st October. Full details are on the Wiki and new info will be added as we get it.

The meeting is being organised by Tony Travis and Tim Booth along with Steffen Möller. Please contact any of us if you have questions.

Raspberry Pi – The future of our education?

Les Pounder

It's been over eighteen months since the Raspberry Pi was released, and in this short time there have been many changes made to the UK education landscape. Schools around the UK are now in a transitional period. Where for a generation they have taught basic ICT skills to students. Skills such as word processing, spreadsheets and presentations, while important for future careers, do not really encompass the enormity that is Computing.

The UK's Department for Education announced in September that Computing would be brought into the curriculum from September 2014 [1] and that the UK Government's goal is to develop a nation better equipped for future technologies and commerce. The new curriculum teaches the foundation of programming, starting at Key Stage 1 where children will create and debug simple programs with algorithms also being used to illustrate logic. Programs will need to be written in a visual language, such as Scratch, and for future Key Stages children will learn more about system simulation, boolean logic and computational thinking. When we reach Key Stage 4, which is the start of GCSEs, educators have much more freedom to create lesson plans based on their own class observations.

These changes to the curriculum are where the Raspberry Pi comes in to the grand plan. The Raspberry Pi Foundation, the creators and caretakers of the Raspberry Pi, are keen to support the goal of teaching a new generation to code, and they are working with educators and consultants to create exciting course material. OCR [2] one of the examining bodies in the UK have produced a series of lesson plans based around the Raspberry Pi and computing skills.

This is where we, the makers, hackers and tinkerers come into this process. Teachers are under prepared and scared of teaching computing and programming to our children. Teachers need our help to make computing interesting, and there are at least two ways in which we can help.

Code Club

Code Club [3] is a term time based, weekly after school club, that takes place in close to one thousand schools around the UK. The goal of Code Club is to teach computing skills to children aged from seven and upwards.

Code Club have provided a framework of lesson plans, and have split them into term based activities. For example the first term starts with a simple Scratch based game, made using the in-built libraries that Scratch has to offer. Over the rest of the terms, we move from Scratch to HTML and CSS, before finally reaching the final term, where Python is introduced to the class.

Code Club needs more volunteers to work with schools, and you are the perfect person to help. They need developers and coders to work with teachers, so that these lessons can be delivered. Once you sign up for Code Club, you must undertake a DBS criminal records check, but this is handled, along with the insurances necessary by STEM.

STEM Ambassador

STEM [4] stands for Science, Technology, Engineering and Maths, and are a national group aiming to promote those subjects to children via volunteers who become "Ambassadors" for their

chosen discipline. When you apply to become a Code Club volunteer, you also become a STEM Ambassador which is a clear double whammy for everyone involved. Being a STEM Ambassador allows you to work with schools and provide guest teaching opportunities for classes in the STEM network, we all remember those cool lessons where a guest speaker would come in and lead a class, well now you have the chance to be that person.

The Raspberry Pi, with it's unique price point, and large community of makers is the ideal device to help teach the Code Club and STEM syllabus. The chance to build a computer lab is now within the grasp of many schools, can you help make this happen?

So to summarise, if you can spare a few hours a week contact a local school and see if they would like your help. Start a Code Club, help them to understand the interesting world of coding and the marvellous inventions that can be made with the Raspberry Pi.

References

[1] http://www.education.gov.uk/schools/teachingandlearning/curriculum/nationalcurriculum2014/

[2] http://www.ocr.org.uk/search/index.aspx?keyword=raspberry%20pi

[3] https://www.codeclub.org.uk/

[4] http://www.stemnet.org.uk/

3D Printing: Building a Mendel90

Bob Clough

I have been a 3D printing enthusiast for around 6 years now, but I first started building my first printer, a RepRap Huxley, in late 2009. My first port of call when I decided to finally bite the bullet and build my own printer was to message Chris (aka nophead), whose RepRap development blog [1] I had been following since I first discovered 3D printing, and get him to print out a set of parts for me. This was the first, and easiest part of the build, and the fantastic quality of Chris' prints really stuck with me.

When Chris released the Mendel90 on Christmas day in 2011, I was struck by how elegant a design it was. With a body consisting mostly of MDF, a few fixing blocks from B&Q, and a lot of 3D printed parts, the printer was easy to assemble and align, and very accurate. Chris started selling kits in late 2012, supported by his wife Mary, and I got to talk to them first hand about the process of designing and shipping the kits at 3DPUG, Manchester's 3D Printer UserGroup.

When we decided to build a 3D printer for the Hackspace, I suggested the Mendel90. It has good pedigree as a part of the RepRap Project, being a fusion of the RepRap Mendel, Chris's own Hydraraptor, and many other open designs. and is fully open source, including the hardware designs, the onboard firmware, the software used to control it and the documentation [2].

When Kimball and I received the kit, the first thing we did was have a look at the instructions. The instructions for the Mendel90 are a thing of beauty, every subassembly is well described, and contains an exploded diagram of the part to help assemble it. This is a significant departure from a lot of 3D printer designs, where the documentation often consists of a couple of wiki pages if you're lucky!

The first section of the build is devoted to assembling the frame of the printer. All the subassemblies built later on bolt to this frame, so it has to be incredibly rigid and square. The aluminium

runners along the bottom, and the 90 degree angles help in this regard, and the 3D printed brackets enclose the nuts entirely, holding them in place and making inserting the bolts from the other side very easy. Building the frame took us around an hour, including finding the necessary parts in their ziplock bags and printing the instructions.

Once the frame was together, we started the assembly of the Z axis. This axis provides movement up and down, and is how the printer stacks layers on top of each other. The Z axis was the first encounter we had with the Mendel90 wiring. Where possible, all the wires that power and control different parts of the printer are hidden underneath or behind the frame, providing an incredibly sleek finish. The wires for the left hand motor on the Z axis run through a little hole behind the motors, up the back of the printer, over the top, and back down the other side into the electronics bay. The wires for the right hand motor go straight through a hole into the electronics bay. Having a wiring plan is a departure from regular RepRap designs, where cables are often zip tied to anything that happens to be running in the correct direction! Getting the Z axis assembled and bolted to the frame took us around an hour and a half total, with a few tea breaks along the way.

At this point, Kimball had to run to get his train, and I decided to tackle the Y axis before heading home. The Y Axis is the part that moves front to back, and on a mendel-style design contains a heated build plate, which the part is printed upon. I started by soldering the wiring to the heated bed. The wiring on the bed uses a piece of ribbon cable, similar to the type used on old floppy and PATA drives. This cable is split in half, and all the wires soldered together to allow enough power to travel down them without burning up the cable. The shape of the ribbon cable allows it to 'roll' along as the bed moves, which helps avoid wires bending or snapping The rest of the Y axis consists of a pair of parallel bars along which the bed can move, and a belt moved by a stepper motor. The Y axis took me around 2 hours, mainly due to the amount of soldering needed.

We met a week later to tackle the X axis and Extruder. The X axis itself is fairly simple, consisting of a stepper motor and belt that pulls a carriage left to right. On this carriage sits the Extruder, which is the part that makes a 3D printer actually print. It works by rotating a shaft, with a set of ridges along it that push a 3mm plastic rod through the "hotend", a brass block heated to around 200 degrees, with a tiny 0.4mm hole in it. This leaves a thin layer of plastic, and by building up these layers an object is formed. The extruder uses the same ribbon cable type wiring as the Y axis, except this time it is passing the motor control signals and limit switch for the X axis, and the stepper signals, temperature sensor and hotend heater power for the extruder. Passing all these signals over a single cable helps again with keeping the mendel90 looking clean, and making it easy to maintain. The extruder includes a fan with shroud, which is useful for printing some plastics, but removable if you dont want to use it. The X axis and extruder work took us around 3 hours total.

In the next article in this series, I will be covering connecting up all the electronics, making sure everything is calibrated, and the all-important first print.

References

[1] http://hydraraptor.blogspot.co.uk/

[2] https://github.com/nophead/Mendel90/tree/master/dibond

[3] https://github.com/nophead/Mendel90/blob/master/dibond/manual/Mendel90_Dibond.pdf

Backing up and archiving with rsync and ZFS snapshots

Tim Fletcher

Overview

I look after a server for a colleague that runs a number of KVM virtual machines on top of Ubuntu 12.04 LTS, the server has a pair of USB drives plugged in the back for backups.

I talked briefly about this in my lightning talk at LISA conference in Newcastle this week but I'm not sure I made much sense so this is a better write up.

Using USB harddrives for backups has drawbacks, namely USB transfers are slow compared to modern SATA harddisks, also I don't trust harddisks or USB to transfer and store data correctly. This solution makes use of ZFS to address both of these problems and many others.

Why ZFS for the backup target?

The ZFS features we are leveraging are:

- *End to end checksums* – ZFS provides end-to-end data integrity by computing and storing a checksum with every block on the disk. This does cost a small amount of space and CPU time, however the gains for backups are huge. You know that the data on the disk is the data you have backed up, and it's automatically verified every time you read your backups.
- *Compression* – ZFS allows for data to be compressed before it is sent to the disk, this helps because we can store more backups and because data is compressed before it travels over the slow USB link.
- *Mirroring* – The drives as setup in as a ZFS mirror so that data is stored twice, once on each disk. This helps for two reasons, firstly it means that if ZFS does detect an error on read there is a second copy of the data to repair the error. Secondly as we are mostly reading the backups when we are rsyncing to the ZFS pool we can read from both disks at once so the pool is twice as fast, thus backups happen quicker.
- *Snapshots* – We want to be able to "step back in time" to load a machine image from last week/month/year but also not to store the same data over and over again. ZFS snapshots is how we do it, as we are using both LVM and ZFS snapshots I'm covering this is more detail next.

LVM vs ZFS snapshots

The disk images for the virtual machines are stored on an LVM managed RAID5 array. The only user data on the machine is inside disk images, so we need an efficient way to backup and archive large (100GB) disk images. We can do this by leveraging the power of snapshots, both LVM and ZFS snapshots.

LVM and ZFS snapshots are very different, LVM is a block management layer and so snapshots in LVM are block based where as ZFS is a Copy on Write (CoW) filesystem and so ZFS snapshots are tree snapshots. The differences are shown in the following comparison from a Sun presentation on ZFS. The key difference is that with ZFS snapshots are a map to where the changed blocks are stored where as LVM snapshots are a copy of what has been overwritten.

Traditional snapshots versus ZFS

Per-snapshot bitmaps

- Block allocation bitmap for every snapshot

- O(N) per-snapshot overhead
- Limits number of snapshots
- O(N) create, O(N) delete, O(N) incremental
- Snapshot bitmap comparison is O(N)
- Generates unstructured block delta
- Requires some prior snapshot to exist

ZFS Birth Times

- Each block pointer contains child's birth time
- O(1) per-snapshot space overhead
- Unlimited snapshots
- O(1) create, O(Δ) delete, O(Δ) incermental
- Birth-time-pruned tree walk is O(Δ)
- Generates semantically rich object data
- Can generate delta since any point in time

You don't need to understand the exact reason why things are different, just that they are and we going to make use of them, now on to the meat of the backup process.

Implementation

First of all we need to set some variables, different people like the date in different formats here we are using `%s` or number of seconds since the UNIX epoch in 1970.

```
# These are the VMs that are sync'd before backup
vm_targets="Server-Tim MySQL"
# Get the date in UNIX time
date=$(date +%s)
# These are the LVM volume groups we are backing up
storage_targets="/dev/SSD/images /dev/raid5/VM.images"
# Where are we backing up to, needs to be ZFS
backup_target="archives/backups"
# Where we mount the LVM snapshots
mountpoint=/run/backup
```

As we are backing up virtual machines (VMs) it really helps if we can tell the machine we are backing up that it's about to have a backup taken. There are a number of approaches to this such as logging in with an ssh key or using an agent, but because QEMU doesn't have much support for agents yet and ssh logins are just another thing to setup I've settled on a different way. I have made use of the fact that virsh can send a raw key press to the VM just like someone sitting at the keyboard has pressed a key, by sending alt+sysrq+s we can trigger Linux's emergency sync mechanisms which flush all waiting disk buffers from memory to disk, perfect for a backup.

One second after requesting the disk flush we tell QEMU to suspend the machine so that nothing changes while we sync the other machines and trigger an LVM snapshot. This happens very quickly and the machine is normally suspended for less than 5 seconds.

```
for VM in $vm_targets ; do
  virsh send-key $VM KEY_LEFTALT KEY_SYSRQ KEY_S
  sleep 1 ; sync
  virsh suspend $VM
done
```

Now that we have suspended all the machines we care about and made sure all of the disk images are consistent we need to make sure that we can get a consistent backup of the disk images so we take an LVM snapshot. Later on we mount this snapshot and take an rsync backup of the disk images.

```
for storage in $storage_targets ; do
  shortname=$(basename $storage)
  /sbin/lvm lvcreate --quiet -L 10G --chunksize 512k -s -n \
         ${shortname}.${date} $storage
done
```

We have now got a snapshot on the LVM storage so we can release the machines to carry on working while we run the backups.

```
for VM in $vm_targets ; do
  virsh resume $VM
done
```

We also need to take a ZFS snapshot so we can roll back to this backup in the future.

```
/sbin/zfs snapshot $backup_target@daily.$date
```

We now have 2 snapshots frozen in time, the LVM and ZFS one, we are planning to keep the ZFS one but we need to discard the LVM one as soon as we can because it's lowering the performance of the disks. Now we need to actually make a copy of the files from the LVM snapshot to the ZFS storage.

This loop of the backup script is kind of complex, what it's doing is making a temporary mount point, checking the LVM snapshots and mounting them and using the command rsync to copy any changes from the snapshot to the ZFS backup target. The flags `--no-whole-file` and `--inplace` are worth mentioning, they force rsync to only copy over changed blocks from the LVM storage to the ZFS storage. This makes the ZFS snapshot very space efficient as well as improving the speed of the backups. Finally the loop un-mounts the LVM snapshots and removes them.

```
for storage in $storage_targets ; do
  shortname=$(basename $storage)
  mountname=$(echo $storage | sed -e s,^/dev/,,g -e s,/,.,g )
  mkdir -p $mountpoint/$date/$mountname
  /sbin/fsck -p ${storage}.${date}
  if ! mount -o ro ${storage}.${date} $mountpoint/$date/$mountname ; then
    echo failed to mount ${storage}.${date}
  else
    /usr/bin/rsync -axH --no-whole-file --inplace --delete \
              $mountpoint/$date/$mountname/ /$backup_target/$mountname/
  fi
  sleep 10 ; sync
  umount ${storage}.${date}
  sleep 10 ; sync
  /sbin/lvm lvremove --quiet --force ${storage}.${date}
  rmdir $mountpoint/$date/$mountname
done
```

Just in case grab a copy of the root filesystem as it contains the configuration of the server and VMs.

```
/usr/bin/rsync -axH --no-whole-file --inplace --delete / \
          /$backup_target/root/
```

And finally...

Yes it would be easier if the main VM data store was run on ZFS but it's not, however I think that I've designed this backup solution to get the best from both LVM and ZFS snapshots.

High Performance Browser Networking

Ilya Grigorik
O'Reilly Media
ISBN: 978-1-4493-4476-4
404pp.
£ 26.99
Published: September 2013

reviewed by Paul Waring

There are lots of simple improvements you can make to a web application to improve performance. This book attempts to cover some of the less obvious and more technical aspects which impact performance, particularly those relating to the underlying network.

Part I covers the basic elements of networking, such as the difference between latency and bandwidth, and how to measure the key indicators of performance. TCP and UDP get a chapter each, although depending on your background this will either be obvious stuff or an overwhelming amount of detail. The final chapter covers the internal workings and costs of TLS – some readers may take issue with the author's assertion that the use of TLS imposes severe performance penalties (imperialviolet.org has some excellent posts on this specific question and browser security in general).

The second part is dedicated to wireless networks. Whilst this may be a growing area as more people move towards accessing sites through mobile devices, a lot of the issues discussed are related to the network infrastructure level, which is generally outside the control of the web developer. It is interesting from a diagnostic point of view, but less helpful with regards to mitigating the problems.

Part III covers all aspects of HTTP, including its performance implications and how these can be worked around. The chapter on HTTP 2.0 is very interesting, although with over a year remaining until the submission of the proposed standard – and longer until widespread adoption – it is perhaps a bit ahead of its time. Overall though, this is by far and away the most relevant and useful part of the book for web developers, even those running sites which do not need to scale to the size of Google or Facebook.

The final part of the book covers browser APIs and protocols. The chapter on XMLHttpRequest – which is behind most pages which update without refreshing – discusses some useful ways to fine-tune this functionality, based on techniques from sites such as Facebook. Other chapters cover technologies which have yet to receive widespread adoption, including WebSockets and WebRTC, and so may only be of interest to the specialist.

Overall, this is an extremely detailed book which covers almost every aspect of networking and browser performance. However, this breadth and depth of coverage comes at a price, and the majority of the text goes far beyond what most web developers would ever need to know. If you are working on a web application where shaving a few milliseconds off each request is essential then this book is definitely worth reading – doubly so if you have a degree of control over the physical network access – but for developers working on small scale sites only part III is particularly useful.

Java Web Services: Up and Running (2nd edition)

Martin Kalin
O'Reilly Media
ISBN: 978-1-4493-6511-0
360pp.
£ 26.99
Published: September 2013

reviewed by Paul Waring

Although Java is no longer an exciting new language, its use in academia and enterprise, combined with the availability of platforms such as Amazon Web Services, make this a book which is likely to be of wide interest. There are some major changes to take account of the popularity of RESTful web services – certainly enough to warrant a new edition.

The first chapter gives a fairly lengthy overview of web services, including an explanation of the different architectures and a detailed introduction to the RESTful model which is used throughout the book. This is followed by a chapter on using REST on the server side, with examples using the 'grizzled workhorse' HttpServlet and the more recent JAX-RS.

Chapter three switches to the client side, with examples for various services, including Amazon E-Commerce. Although the focus of this book is on Java, a Perl client is also shown to demonstrate that services really are language-agnostic, and there is no need to use the same language for client and server.

Chapters four and five switch from REST to SOAP, which was the focus of the first edition. Whilst many developers – myself included – would dearly like to see this style of web service disappear, it is still widely used and therefore coverage is a necessary evil.

The next chapter is dedicated to security, split into the three categories of wire-level (i.e. the protocol, such as HTTPS), user authentication and authorisation, and WS-Security (specific to web services). Security is sometimes glossed over in programming books, but in this case the author has made it a core part of the text, going down into details such as which cipher suites are available and how a connection is actually made secure.

Rounding off the book is a chapter covering Java Application Servers – an alternative to deploying using a standalone server such as Tomcat. I wasn't entirely convinced by the arguments in favour of this method, but this chapter provides a good starting point to evaluate the different options.

Overall, this is a good all round introduction to building web service clients and servers in Java. The code samples pad out the text a little bit too much for my liking, but that is probably unavoidable given the verbosity of Java, and the author does warn readers that this is a 'code-driven tour of web services'.

Practical Programming: An Introduction to Computer Science Using Python 3

Paul Gries, Jennifer Campbell and Jason Montojo
Pragmatic Bookshelf
ISBN: 978-1-937785-45-1
350pp.
£ 28.99
Published: October 2013

reviewed by Andrew Richards

This carefully-written and nicely-paced book covers its subject well, whilst at the same time subtly introducing a number of good techniques and good practise. It succeeds very well in its stated aim of being an introduction to computer science; for those more interested in learning Python who already have the computer science basics, this book is not the best choice.

Having reviewed a Ruby book with a similar title, Computer Science Programming Basics in Ruby (newsletter, June 2013) I was very curious to see how this book would compare: It turns out the books do cover similar ground but have markedly different styles.

The care taken in writing this book is apparent when reading the text and the exercises (solutions online); it has a meticulous attention to detail – as well as having almost no obvious mistakes or typos. The reader is carefully led through the subjects being discussed, and the examples always obey good practise for things like naming conventions and docstrings (docstrings are Python's recommended summary comments for each function/method that can do double-duty for testing). Suspicion that inclusion of docstrings might lead to a discussion of testing and test-driven development is well-founded, but this is introduced a good way through the book after the programming fundamentals have been covered; an appropriate place.

Besides test-driven development, various other important topics are addressed, such as top-down design, sorting and selection, algorithms, or measuring the speed/efficiency of algorithms including looking at how execution time can rise linearly or exponentially. The authors take pains to explain how data is laid out in memory in Python – worthwhile knowledge when using Python – including 'memory model' diagrams; this is a good example of the text being very thorough in delving below the surface to give its readers well-rounded knowledge.

As well as programming staples (functions, conditionals, types, modules, loops, file I/O, data structures, methods, objects, testing, debugging), it was great to also see coverage of GUI and database programming (with SQLite) – meaning that the student can create programs that look good/modern and do real work.

The difference in styles between this and the Ruby title is illustrated by the latter being less than half the size (or number of pages) of this book; the Ruby book is particularly concise and spartan, whereas this book has a more wordy approach. This is not a bad thing: Students are likely to find that this gives them more of a safety net, but the style of each book is strikingly different, and each book has a resulting Marmite factor: some will love or hate each title for its sparseness or thoroughness respectively.

Unlike the Ruby title this book is less suited to just learning the language; I think there are better books for that. The Ruby book also provides a set of slides to accompany it for teaching which this title lacks; both have good sets of exercises; this one includes answers to the exercises online. This book also provides a detailed index, which was sadly lacking in the Ruby book.

I was pleased to see that the book farms out the details of installing Python to a separate web page, rather than wasting space in the book itself for something that the reader may only need to do once if at all.

I particularly liked the quality of the exercises: They are clearly carefully designed to make the reader think about what they have just learned and reinforce it. Somewhat less compelling were the examples through the main text: Although there were some interesting examples such as analysing real-world bird migration data, a number of the (especially earlier) examples were pretty dull, and later on there seemed to be a preponderance of chemistry-based examples that might be off-putting to the non-science-oriented student. I think devising interesting examples is a particularly difficult task, nevertheless I think this book could have done better, since it does so in most other areas. In their favour however, the examples given are largely of a form that can be typed at Python's command prompt, making them very suitable for interactive use in a 'lab' style of teaching.

Barring one exception misrepresenting Python's `with` statement, the book is very careful to accurately describe the parts of Python it discusses, making it a reliable text to refer back to. Some topics are absent; recursion, Python exceptions, character encoding and design patterns being some I can think of here; even if they weren't covered in the book itself, pointers to topics such as these would have been welcome. There is a good list of further reading on Python and other programming languages given in the preface though.

Overall, the book achieves its aims, providing a well-rounded and thoughtful introduction to computer science. I would have been very pleased to have been a student or lecturer using this text. Whilst it is designed as a text book to accompany a taught course, the thoroughness of the text and the thoughtfulness of the examples means that it could be used very successfully by a self-motivated learner without formal teaching. Despite its good coverage of subjects, I nevertheless wouldn't recommend it as a book for learning Python itself, but then that isn't its aim.

Git Pocket Guide

Richard E Silverman
O'Reilly Media
ISBN: 978-1-4493-2586-2
234pp.
£ 11.50
Published: July 2013

reviewed by Andrew Richards

I'm a huge fan of compact books that provide thorough coverage of a subject. This book delivers just that. It works well as a read-it-through textbook and as a handy reference, not least since its small size and provision of useful headings make it easy to flick through to locate the information needed.

I've been using Git myself for about a year but still regard myself as a relative novice. I've found the larger O'Reilly Version Control with Git book helpful but sometimes confusing; I've also benefitted from the various Git 'Quick Reference' and cheat sheets available free online. Despite all of those resources there was plenty for me to learn from this very readable pocket guide.

The author (rightly, I feel) makes the assumption that anyone engaging with this book will already have certain technical skills, such as how to install a package (git) or read a man page; he also unashamedly focuses on Git in a solely Unix/Linux context. This means that the text is carefully focused on its subject, and there's almost no space amongst its 240 pages wasted on peripheral subjects.

It is written in a very pleasant flowing style, easy to follow and understand. Each chapter is fairly short yet manages to cover all the key points needed, illustrated with short pithy examples.

Although it describes itself as an introduction (and reference) to Git, the depth of coverage is likely to be fully adequate for many relatively demanding users of Git; the author provides pointers to further information – man pages, external packages etc.

It starts by explaining some of the underpinnings of Git in the first chapter 'Understanding Git'; the reader is encouraged to digest this so as to better understand the rest of the book. I did so and did indeed find the rest of the book made more sense as a result.

The 'Getting Started' chapter has many useful tips including on setting up your environment. Thus my Git now gives me colour highlighting in its output, something I didn't realise was an option. The book recognises how Git is actually used, so includes for example details of the conventions used for commit messages – very handy to ensure that Git commit messages integrate well with Git-related tools – and there are also some power-user tips, such as enabling Git tab completion. The bulk of the book is given over to chapters for different aspects of Git: Commits, branches, repositories, merging, history/logs, patches, remote access, covering each area in sufficient detail to make good sense of it and highlight any nuances and gotchas.

The book has a number of well-placed 'tips' on what to do when the unexpected occurs (e.g. a merge fails): Just enough information to get you out of trouble and where appropriate, pointers for more information.

There were relatively few typos in the book, and the fact that the text flowed so smoothly is a credit to the author and his proof reader(s). The minor niggles I had were: in the 'Understanding Git' introductory chapter, the concept of a 'detached HEAD' is explored, which is too much detail at this stage – it would have been far better to leave this particular issue until the 'Branching' chapter (where it is explained well); and the scanty coverage of git fetch – the author's skill in clarifying the relationship between and typical usage of git pull and git fetch would have been most welcome. The penultimate chapter on Remote Access is mostly dedicated to quite detailed coverage of setting up and using an SSH keypair for ssh access to Git: This seems to go well beyond the core subject. The author's response is that this was deliberate, "it is dangerous to give short shrift to security issues". He did write the O'Reilly book on ssh, so he may have a point, but I'd have preferred a couple of pertinent web links on keypair setup and a bit more coverage of all of the various remote access methods for Git.

After the skilful prose in the rest of the book, the index is slightly disappointing: it's good, just not of as high a quality as the rest of the book, with some topics and commands not appearing in the index, or the index just giving some but not all page references for a given topic.

Comparison with the larger O'Reilly book Version Control with Git (VCG) is inevitable: With both covering the same subject, which one is 'best'? The books actually overlap in their coverage rather like a Venn diagram: Whilst the Git Pocket Guide (GPG) is very good on using Git and getting the most out of it (e.g. setting up your environment), VCG covers some areas that are not touched atall by GPG, like Git's features to work with a Subversion repository or the git blame command. There is a major difference in writing style however. Whereas GPG is universally succinct in its explanations, VCG tends towards over-describing topics. Picking an example at

random, I understood the git revert command better after reading the GPG coverage of it (107 words) than when reading the VCG coverage (196 words + 2 diagrams). GPG is about half the price of VCG, so I think the answer is to buy GPG first, and VCG for additional coverage if desired. Note that I've been referring to the 1st edition of VCG; the current 2nd edition may compare differently.

In summary then this is a superb and well-written book on Git; being compact it's very portable and accessible. It should work well in eReader format if that's your preference, with the content being mostly textual and the diagrams being uncomplicated so suitable for a small screen. It makes for very enjoyable reading as well as providing a good reference for day-to-day Git usage.

RESTful Web APIs

Leonard Richardson, Mike Amundsen and Sam Ruby
O'Reilly Media
ISBN: 978-1-4493-5806-8
408pp.
£ 26.99
Published: September 2013

reviewed by Lindsay Marshall

There are a few books out there on using REST (some written by friends of mine), but there are not nearly as many as you might expect given what a sensible and useful idea it is.

This book provides a clear and sensible introduction to the topic, which is, most importantly, firmly rooted in reality – yes, quite often you are pretty much stuck with GET and POST operations and you have to live with it. Which is not to say that the authors ignore the "proper" (let's not argue) ways of doing things.

And once over the HTTP hump they dive into Hypermedia and XML and the lands beyond. It's all good and useful, even it if it is perhaps a little bit in the future before the world starts working like this, and there is always a chance that it won't (boo). I've learned a lot from this book and I need to go back and learn more from it. If you are in the business of connecting systems together with data then you probably need to read it too. Definitely recommended.

CSS Fonts

Eric A Meyer
O'Reilly Media
ISBN: 978-1-4493-7149-4
68pp.
£ 5.99
Published: August 2013

reviewed by Lindsay Marshall

See the combined review below.

CSS Text

Eric A Meyer
O'Reilly Media
ISBN: 978-1-4493-7374-0
56pp.
£ 3.99
Published: September 2013

reviewed by Lindsay Marshall

See the combined review below.

HTML5 Pocket Reference

Jennifer Niederst Robbins
O'Reilly Media
ISBN: 978-1-4493-6335-2
184pp.
£ 9.99
Published: August 2013

reviewed by Lindsay Marshall

2 slim volumes and a shorter, slightly plump one. The two CSS books are interesting in that they are fairly reasonably priced – £3.99 for 50 pages and £5.99 for 60 pages, which suggests that O'Reilly see a market in selling tightly focused extracts of larger, more expensive books. Because that is what these seem to be (at the moment I can't find my full CSS book to check) – the book on text describes the directives for laying out text, and the font book those for font handling. And nothing else.

Of course, these are books by Eric Meyer so the quality of the information is high and the examples are clear, understandable and useful, so, if you need only information about text layout or font handling, these are a pretty good way of getting a cheap, reliable printed source. But I can't help but feel that you might just be better off buying the whole manual since you never know what else you might need and in the long run it would probably be cheaper than buying extracts.

The HTML5 Pocket Reference nearly does what it says in the name – it's a reference and it fits in your pocket and it certainly covers HTML5, and in the usual comprehensive and helpful way this series of books has always done. But all it covers is the HTML5 markup (and I have to confess there are things in it that I hadn't realised existed), and I think that many users consider all the various APIs such as location and local storage to be an integral part of HTML5. Certainly when people talk about it on the net they do seem to implicitly be including all the extra goodies, so I think that some people might be disappointed when they find nothing about the APIs – I know was. Caveat emptor, but I'd still recommend it.

Book discounts

FLOSS UK members are entitled to various discounts on books from different publishers. Principally, and most popularly, is a 40% discount on all O'Reilly books (includes forthcoming books).

The full set of discounts is:

- 40% on all O'Reilly titles
- Discount on the Exim book
- 25% off books from Pearson Education – includes Addison Wesley and Prentice Hall
- 30% off books from Wiley.

The 40% O'Reilly discount is through online orders only, however FLOSS UK members can obtain a 37.5% discount when ordering through the office, and the office will accept a purchase order from members.

Online orders go via the US shopping cart - members can obtain the discount code from the Secretariat.

To obtain the 37.5% discount, contact the FLOSS UK Secretariat to place your order, giving ISBN, book title, FLOSS UK price, your postal address and phone number.

FLOSS UK members are also offered the chance to review books from these publishers. Reviews are printed in the FLOSS UK newsletter and are available online.

If you're a publisher and wish to see your books reviewed by FLOSS UK, please contact the office at **office@flossuk.org**.

Other Member Discounts

Discounts for FLOSS UK members

In addition to the significant book discounts, and other benefits available to our members, we have negotiated the following generous discounts.

2ndQuadrant

Specialist Database consultancy 2ndQuadrant is pleased to offer 10% discount to UKUUG members on any/all scheduled PostgreSQL courses, which run regularly throughout the year.
http://www.2ndquadrant.com/

GBdirect

Well-known Open Source training company GBDirect is pleased to offer the following discounts to UKUUG members:

- 10% off all scheduled courses, including Linux, Unix, PHP, Perl, Web standards, Web Accessibility, MySQL, Apache and many others
- GBdirect also offers a wide range of on-site courses, Linux and Open Source support and consultancy services.

http://www.gbdirect.co.uk/

Bytemark Hosting

Bytemark Hosting is pleased to announce that UKUUG members will now receive a 10% discount on all of our virtual machines. Please visit **http://www.bytemark.co.uk/r/ukuug**

Bytemark provide a range of Linux hosting products providing root access from only Â£15 per month.

Opengear

Opengear, manufacturer of open-source remote management appliances, is pleased to offer members 10% discount across its full range of products including:

- Console servers
- Cellular routers
- Industrial device servers.

Please visit: **http://opengear.com/uk/products.html** for product details.

Contributors

Bob Clough is a Founder Member of Hackspace Manchester (**hacman.org.uk**) and Manchester's 3D Printer User Group (**3dpug.co.uk**).

Tim Fletcher is a long time Linux sysadmin who enjoys playing with learning about technology. He previously worked for a long time in large school in South Manchester and is now working for an IT Practise in Huddersfield. Tim has been involved with Open Source for many years and recently joined the FLOSSUK council.

Kimball Johnson is a Systems Developer for Lancashire County Council He has been programming since a very early age, starting with BBC Micros, then MS DOS and Windows Systems, however was enlightened with a copy of Debian GNU/Linux Woody at university. He is always wanting to learn and has recently started to take on embedded programming on a variety of devices, on everything from a Arduino to a Nintendo DS.

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

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 (Fibre-optic Industry Association) that she also runs from the Manor House office.

Les Pounder works closely with North Western Linux and Free Software groups to promote the use of Open Source software as opposed to proprietary software. He is also the organiser of UCubed, a free Linux and open source event in Manchester, and an organiser of Barcamp Blackpool and Blackpool Geekup, and has been head of crew at Oggcamp. He writes for Linux Format magazine and contributes to Linux podcasts including Fullcircle, Ubuntu UK Podcast and Linux Outlaws.

Andrew Richards is an IT consultant specialising in Linux and Unix based email servers, general sysadmin and networking. He dabbles in C and Python and has written a number of open source add-ons and patches to qmail. In his free time he enjoys arty foreign films, badminton and cycling. Find him at `www.acrconsulting.co.uk`

Tony Travis is a research fellow in Bioinformatics at the University of Aberdeen, and runs Minke Informatics.

Paul Waring currently works in teaching and software support for a large UK university. Outside of work he can usually be found filing documentation bugs against various open source and free software projects.

Roger Whittaker works for SUSE supporting SUSE Linux Enterprise Server for major customers in the UK. He is also the UKUUG Newsletter Editor, and co-author of three successive versions of a SUSE book published by Wiley.

Contacts

Kimball Johnson
UKUUG Chairman
Preston

Gavin Atkinson
Council member
York

Tim Fletcher
Council member
Manchester

Holger Kraus
Council member
Leicester

Ian Norton
Council member
Rochdale

Stephen Quinney
Council member
Edinburgh

Quentin Wright
UKUUG Treasurer
Warwick

Roger Whittaker
Newsletter Editor
London

Alain Williams
UKUUG System Administrator
Watford

Sam Smith
Events and Website
Cambridge

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