floss UK

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 25, Number 1 **ISSN 0965-9412** **May 2016**

Contents

From the Secretariat

Thank you to all the members who have paid their subscription invoice promptly this year. We have recently chased the outstanding subscriptions and hope the remaining payments will come in promptly.

The Perl tutorials, held the week beginning 8th February, again got very high marks from all who attended. The feedback on the questionnaires show that the Perl tutorials are always very well received by our membership.

The recent Spring Workshops and Conference, held in London from 15th - 17th March, although having a reduced number of delegates, was very successful and there are write ups on the event and photographs later in this Newsletter.

The event was kindly sponsored by Google, Chef, Eligo Recruitment and our Sponsor members SUSE and 2nd Quadrant Ltd.

Our next event is the BarCamp. This annual event is again being held in Birmingham.

Please note our Annual General Meeting will be held in London on Wednesday 21st September.

We shall be hoping to fill some vacant places on Council at the AGM but if you are interested in joining Council now please let us know as we have the option to co-opt members now.

As previously advised this is the only hard copy Newsletter being printed in 2016. Future announcements, book reviews etc. will all appear on our new look web site. Have you taken a look yet? - www.flossuk.org

---

From the Editor

As reported in the September 2015 newsletter, FLOSS UK have moved to an annual printed newsletter, supplemented by book reviews and event reports which will be published on the website throughout the year. This is the first newsletter to be published under the new format, which allows us to include photographs and other images.

For those of you who are interested in such things, the newsletter is now composed in Markdown and built using a fairly simple Bash script which pulls all the content into a single file. Pandoc is then used to produce a PDF version for online distribution and printing.

The big advantage of Markdown is that it removes the need to mark-up every paragraph, list item etc. in SGML, which saves a huge amount of time. The new process also uses filenames for sorting, so adding a new contributor biography involves dropping a file with the relevant name (e.g. `waring-paul.md`) into a directory and the build script will automatically pick it up. The same applies to articles, event reports and other content.

All of the code and content is available on GitHub:

https://github.com/pwaring/flossuk-newsletter

If you have any comments on the new format, please do get in touch via email: newsletter@flossuk.org.

BarCamp Birmingham: Saturday 11th June 2016

Free-of-charge to attend - full day - 10:00 - 17:30

Registration now open!

What is a bar camp? A bar camp is a conference where what happens is organised by the delegates on the day. The event organisers have to arrange something, the main one being the venue, but the rest is down to the delegates. So all the hassle of talk submissions, review and scheduling is taken away.

Typically at the start of the day everyone gets up in turn and says who they are, what their interests are and what they'd like to do. Based on this people write proposals on PostIt notes and stick these on a board.

A moderator may read out the proposals in turn to gauge interest, and if sufficient the proposal will be put on a scheduling board. Delegates may adjust this to avoid clashes etc.

Experience shows that this format results in high quality sessions focussed on what delegates want.

There is no charge to attend. Everyone should be able to afford to attend and as it is a one-day event you will not need to incur accommodation costs, though there are several hotels nearby for anyone who wishes to stay over.

Refreshments, tea, coffee & biscuits will be provided morning and afternoon, and lunch will also be provided.

Why attend?

There are lots of reasons to attend the FLOSS UK Bar Camp 2016, including:

- Keep abreast with new/emerging technologies.
- Network with some of the people who are responsible for developing critical applications.
- Become part of the UK Open Source community – build up informal relationships that can be invaluable in problem solving.
- Benefit from the experience of delegates with similar interests.

See: http://www.flossuk.org/events/barcamp-birmingham/

Spring 2016 conference review I

Mark Keating

This is a general impression of the recent FLOSS UK event that I attended. I should point out that I do act as a member of the organising team and the FLOSS UK Council but this is a personal review of the experience not an official account.

The 2016 FLOSS UK Spring DevOps Conference was held in London at the historic Mary Ward House between 15th-17th March. This year's conference could have held the subtitle of 'configuration management and communities' as there was a strong undercurrent of both. The spring conference is special to the members of FLOSS UK as it is the oldest DevOps event of this type in the UK, by which we mean dedicated more to systems than programming alone – though that is not a strongly defined prerequisite.

Changes

In the build-up to the event the FLOSS UK Council undertook a couple of new moves as they continue a transition from supporting just Open Source Software to supporting openness across a number of disciplines such as data, hardware, rights and standards. This is reflected in their new tagline Open Technology and the notion that FLOSS has evolved to mean Free and Libre Open Source Systems to reflect the changing world.

The FLOSS UK Council also took the decision this year to have an 'anonymous' submissions system. This wasn't advertised as the Council wanted to test the process without being rigidly defined. It would also allow for the methodology to be tested for issues. Submissions were anonymised by a non-voting member and presented to the Council as simple abstracts. The Council weighted each submission with a score which was tallied by the non-voting member. They then ranked the submissions by order of score which gave them a selection group, from that there was a consensus as to which talks would make good plenary/keynote speeches.

There is a feeling that this has resulted in a broader talk base, especially in the plenary talks, in comparison to previous years. There is some discussion as to how this is evolved for next year. One change will be that the anonymous submissions will be used again and will be advertised as such and the Council hopes this will further encourage a variety of submissions.

Mary Ward House

The historic Mary Ward House in London's central Bloomsbury district was financed by the wealthy philanthropist Passmore Edwards. It is named for Mary Ward, the novelist and social reformer, who was the inspiration behind the endeavour to provide a centre of training, care and entertainment for the less fortunate in society. One of the most stunning features of the building is the Dickens' Library which is where the sponsors had tables and breaks took place.

Mary Ward House is one of the more unusual, and dramatic, conference venues in the country and as such has a small wealth of peculiarities. The historic nature of the building gave a wonderful taste to the atmosphere of the conference but I feel it also made for a number of issues that in the end overwhelmed the positive.

There are a number of elements that I think the event owners need to address if they wish to hold conferences of any size here in the future. If we were a limited event, say one day, in a single room then the venue is a pleasant meeting space, but for a three day conference there were a number of small issues that marred the overall event slightly. One of the largest issues was the echoing rooms with high noise transportation from room to room.

My general impression is that the venue mostly caters to groups seeking a small space for a half or single day event. There seems to be no conception of standard needed for a longer conference. This may be down to the event having restructure and refurbishment work, or just inexperience. The staff themselves were generally very pleasant if a little uninformed of all that was needed to make a conference work.

The Presentations

FLOSS UK Spring is a three day event. The main part of the conference takes place on two days but there is a workshops day which precedes these. This year we had two primary workshops, selected from five available by the attendees. Kimball Johnson did a whole day of Chef training and Paul Waring did a half day on Ansible with Git.

The workshops seemed to be well received by the participants with a good mix of discussion on the two different systems. Paul had a larger attendance list, which is generally the case as half day events are always more popular with an audience used to late starts.

The main part of the conference was two days with two tracks of talks. The talks themselves were mostly 50 minutes long with a few 30 minute talks to allow lunchtime to be slotted into the schedule.

A new element for this year was the inclusion of BoF (Birds of Feather) sessions into the main schedule. These are aimed at structured discussion on particular topics, a formalised version of the hallway and social tracks. They are run at a number of conferences as they allow casual conversation but in a structured environment on a dedicated subject. They went well and plans are in place to run them again next year.

As always some of the best talks were the lightning talks. These are always fun and frivolous. I chaired the session this year and I kept everyone on their toes with a strict 5-minute time slot to present. My particular favourites were Paul Waring and Matt S. Trout. Though I agree with the audience who selected the talk on 'River, Home and Mother' automation and observation.

The plenary, or keynote, talks at the conference this year were given by Mandi Walls and Lameck Amugongo. Mandi gave the opening talk called 'Always be Learning' aimed as a lesson about being in an open source life. Lameck Amugongo closed the conference with 'Data Revolution Catalyst' when he spoke about the empowerment of using open data in emerging economies, in particular how in Namibia they are using 'open data' to help communities gain access and enablement. I will cover the talks that I attended in a little more detail below

Sponsors

It is always nice to see sponsors at any event and particularly when they bring along such great swag for us to play with. This year was a bonus with awesome notebooks, toys, pens, t-shirts, flash drives and power packs from the wonderful people at Chef, Suse, Google and Eligo.

A special thanks to Roger and SUSE who have been with FLOSS UK for a long time and constantly give strong support and present talks. Also to Rick at Eligo who sponsored and gave a talk on 'Developing Your Brand'.

The Presentations

Disclaimer: The following are a description of a presentation with some dissection of what was discussed. It is not a verbatim account and will contain personal impressions and interpretation. The content therefore does not reflect the quality of the original presentation and should be considered a review and personal opinion.

Always Be Learning: Mandi Walls

Mandi is a software developer currently working for Chef in London, the talk wasn't about Chef though, it was about the Open Source world and our part within. Mandi started with a discussion of an early large HP conference that she attended. The event had a number of communities gathered together to discuss a certain topic, however they were still segregated, this is an attitude that still persists today. We might have shared features but we remain tribal about our software choices.

The world itself has changed, when you meet people today there is a sea change from the days of massive, expensive, and singular control systems to smaller, cheaper, multiple systems that are deployed and made to interact with each other via open APIs.

Part of this change has been at the behest of the adoption to Open Source, OS software is now a dominant player in many larger systems, corporations and projects. It has taken over from proprietary systems. This gives rise to the idea that:

> technology becomes more varied when not constrained by controls

There are a number of issues. One of the ones that aggrieves Mandi is the larger enterprise customers who use open source software in 'largely an opportunistic fashion'. This is where they will build large systems upon open source but fail to engage with contribution to code or communities that created the software.

Figure 1: Mandi starts the conference with the opening keynote (photo: Mark Keating)

Open source sometimes seems to sneak into corporate environments, this might be because of: advocacy by a particular programmer; by being the only acceptable solution without creating an in-house solution; or having features and abilities not found in proprietary systems.

> Organisational change is hard, some of them seem to be doom-laden, heading inexorably, Titanic-like, towards a disaster. They are risk-averse and therefore change-averse.

It is often too hard to change the mindset of an entire organisation, even with a great deal of advocacy or solution solving, therefore the goal should be shifted towards the individual.

> Changing orgs is hard but people-change is easier.

This sea of change has started to enter the corporate world. It is now impossible to find careers that last a lifetime in a single organisation, or organisations who look further than a decade in terms of their existence. Organisations, like the software platforms they depend upon, seem to have a shorter and more evolutionary lifespan. Corporations, systems and programming languages seem to fall in and out of fashion without and real reasons or need.

There is a growing pace in this state of change so that it is increasingly hard to keep abreast of changes to software, to innovation and to the new.

This has reached a crunch with developers, innovators and technical employees where Enterprise has exported the risk of learning and acquisition of new knowledge onto the individual, removing it from the corporate structure. This pressure unfairly benefits those who are younger or unattached and places greater responsibility on individuals to:

- Gain experience in multiple existing technologies.
- Be pliable and flexible to new learning.

Part of an individual's response to this, in order to keep some sense of progression, must be to build 'aspiration models' into their daily life. There is a need to 'be naturally curious'. Programming attracts people who are naturally curious, who want to see how something works, and this behaviour should be encouraged. You may need to 'make yourself curious' about a topic by using 'curious language' in order to relate or describe it.

You must learn to embrace your vulnerabilities, to see them not as a weakness but as an area to be aspirational about. You must always approach anything as a learning experience, this is especially true if you think you know it, try to approach it as if you do not understand, use the language associated with learning.

> New is scary (and exciting) and that is a great challenge

In this learning environment where you may have greater knowledge or experience it is important to be wary of creating a negative environment. This is especially helpful to those who do not yet know but are trying to learn. So many people have turned away from a particular solution or language because of the toxicity encountered with their first innocent questions.

Learn to be self-aware. Be wary of how attitudes can force behaviours. Anyone who has knowledge and experience has a position of power and therefore a responsibility. Your opinion, your choices, will create an environment that may be opposite to what you may want to achieve.

Mandi finished with some basic principles:

- Learn cool stuff.
- Hang out with cool people who build stuff.
- Build your own skills.
- Come to know about the cool things.
- Always look out for an opportunity and ways to learn.

When Applications Make Promises: Thom May

Thom is an applications developer with Chef, his talk was focussed on how Chef works in regards to Promise Theory.

> Promise Theory, in the context of information science, is a model of voluntary cooperation between individual, autonomous actors or agents who publish their intentions to one another in the form of promises. A promise is a declaration of intent whose purpose is to increase the recipient's certainty about a claim of past, present or future behaviour. For a promise to increase certainty, the recipient needs to trust the promiser, but trust can also be built on the verification that previous promises have been kept, thus trust plays a symbiotic relationship with promises. Each agent assesses its belief in the promise's outcome or intent. Thus Promise Theory is about the relativity of autonomous agents.[1]

The way that Promise Theory is implemented in Chef underpins much of how it works as a configuration management system, the belief in Chef is that systems work well with Promise Theory.

In a configuration management promise we have the Promise, the Changes and the Outcome. In Chef this turns into:

- Node is a promise
- Node declares what it will be
- Node reports if it was successful and why (changes) that led to that success

In this way the whole of the configuration management promise is 'data driven'

> On millions of nodes is it implausible to directly control each one of them. Their has to be a level of stratification, but in massive systems there has to be the expectation of automation.

Chef uses Orchestration is synchronous, it uses a direct connection not APIs, this is an overhead and limits the number of simultaneous connections, so this needs scheduling. This is a complex system and relies on the use of central logic to control, the install code for everything is inside the orchestrator.

[1] https://en.wikipedia.org/wiki/Promise_theory

Chef also uses Actors which are the instances and are synonymous in Chef with applications. An actor can receive a message, send a message, spawn new actors, change itself - which is to change the system and state by sending a message.

Orchestration allows the actors to be manipulated which is considered to be Choreography. In choreography there are a number of elements, a promise is made, agents are the ones to make a promise, they can perform actions, this delivers dynamic relationships between nodes.

The use of Promise Theory to make a contract that the system performs using applications as actors that perform a specific routine (choreography) that is centrally directed (orchestration) and controlled makes for a very powerful system. It occurs to me that by using source control via repository and management it would serve as a change control system for the systems as well as a configuration management. The fact that Thom was able to convey all of this understanding and information in such a clear manner is mark of how great his presentation was. It certainly makes for a good introduction to the conceptual working of Chef.

Configuration Management with Ansible and Git: Paul Waring

The conference had a strong feel of configuration management to it and the second talk on the subject I attended was from Paul Waring about using Ansible. A section of Paul's talk was a little confusing to me as I lost the thread in the second half of his presentation. This was less to do with Paul's presentation style and more my inexperience with the subject and sitting too far to the back of the room so that I couldn't follow the slides (again this wasn't Paul's fault the room had very large windows and no blinds).

To begin Paul discussed why there was a need for Configuration Management, the situation was that packaging and installation was a manual process. This was boring for the developer to repeatedly perform; time consuming, and so therefore costly; prone to mistakes, mostly due to the first two reasons. Ansible seeks to solve this issue, in much the same was as Chef, a major difference is that Ansible calls orchestration a 'playbook'.

Ansible is fully open source software and protected under the GPL3, it is available on Github. Ansible has been incorporated into the RedHat Linux distributions when they acquired the parent company that supports its development.

Ansible has a minimal set of dependencies and these are found in most Linux distributions. It also works cross environment to the other popular operating systems. It has become known as working well in both smaller environments as well as large server farms. It goes without saying that Ansible has probably the best integration with RedHat which is a favoured distribution in corporate environments. If you have a larger configuration management, change management or process management plan then Ansible can be a player.

Ansible is written in an INI format which is based on the principle that it is easy to configure and to write. In this manner if the requirement is for a small rollout of a staging and deployment server then Ansible works well for easy replication. The requirement is that the person configuring Ansible must be in a User Trust Status or logged in as Root. You can specify keys and host names if required.

Ansible uses a distributed environment for adding extra features which are created as modules. To configure Ansible you write a set of instructions in the playbook.

Ansible works well with Git for a central repository, you can use other management systems like CVS and SVN. The advantage of using Git is the powerful way in which it uses rollback, undo and rebase, the good view history and the cheap method for utilising branches. You can also use Git hooks to perform actions in the workflow both as pre- or post-commit. This would allow you to perform tasks such as running unit tests, checking syntax and checking dependency structure.

Ansible seems like a lighter configuration system than Chef or Puppet and using Git as an extra layer for working with this adds to its value. Much like Thom's talk Paul was able to convey how Ansible works it was just unfortunate to me that I couldn't read the examples he gave on his slides.

Icinga - Middle of your Toolchain: Bernd Erk

Bernd is a core developer on the Icinga project and has returned to the Spring Conference on a yearly basis to tell us about the latest developments. In the last few years the project has gone from strength to strength and the Icinga Version 2 carries that torch far forward. Icinga was originally a fork of Nagios with the intention of making it more accessible.

Icinga 2 takes this to a new level by re-writing the underlying system, it now has no share with its original parent and is only forwards compatible (you must migrate) from Icinga 1 to 2. Icinga 2 does however still allow you to use anything from the Nagios plugins. Bernd made a passionate plea, well passionate for Bernd, at the event that if you are using Icinga you make it known to the development team, at this moment they have no way of tracking who is using the software and which version and they would like to know.

The focus on Icinga 2 has been the need to make it scalable and extensible. At the same time they have launched a web-based Icinga Web 2 that needs less configuration. Icinga 2 has a new way to configure which is not compatible with either Nagios or Icinga 1, it uses an enhanced configuration language. Icinga 2 has no enterprise edition, they have one consolidated edition whose entire source is open.

Icinga 2 also uses Icinga Director which is a configuration management for Icinga and user commit management. As they roll out Icinga 2.4 they will also roll out new modules to support it including the Graphite, Grafana and Elastic Search.

Bernd spent a good 15 minutes giving us a live demonstration of the Icinga Web 2 system and it looked extremely promising. The use of more modern graphing systems, of exporting the metrics to other systems and allowing configuration management are all mature decisions of a project that gets stronger with each new release.

Developing Your Brand: Rick Deller

Rick Deller is an Open Source recruitment specialist at Eligo. Before you start to instantly make assumptions he is probably one of the few really good recruiters as he does care about the industry and the communities. Rick, and Eligo, are sponsors of a number of community events and Rick is always on hand to help out. One of his ways of contributing is in talks. Last year Rick spoke about how to write a good CV for your next job and this year he returned with the ideas of how to develop yourself as a brand to make your career stronger.

Rick wanted to make sure that we all understood that in a world that is quickly becoming dominated by your online presence the notion of public and private are becoming blurred. So part of this is to control your image and project yourself correctly, that is making your brand. When someone looks at you online you have a short period of time in which to make the best impression. There are a number of things you should do:

- Only show core skills
- Show projects
- Core achievements
- Blogs
- Social profile
- Make sure your Linked-In page is up to date
- Peer recommendations
- Be careful of your online attitude
- Make sure you are visible in your community.

There has to be another side of this equation. If you are a company and you are seeking to employ the right people for your positions you can also do a little more to make yourself more attractive to the best applicants. So you should be aware of:

- How you connect, speak to people in their language
- Representation of your business
- Your reputation in the marketplace
- What technologies you use
- The management structure
- What do your ex-employees say?
- What is the company progression?
- What are the training and skills?

As always Rick spoke with a quiet authority on this subject. I have since discussed with Rick his two talks and know that he is keen to return in 2017 with the next stage. He is also developing a workshop to help people practice skills of choosing the right company, presenting themselves and getting a good career path.

Is Configuration Management Still Relevant in a Containerised World: Stephen Grier

Stephen Grier works at the University of Central London where he mostly focuses on what he terms as 'web technologies' and utilises Puppet for configuration management. At the start of the talk he posed the following question:

> What does Configuration Management actually give us?

To Stephen it provides:

- Automation
- Reproducibility
- Reliability
- Infrastructure as code
- Convergence towards a desired state

We should look at containerisation. To Stephen the idea is gaining popularity now but it is not a new idea. In the 1970s we were introduced to a concept of it in the CHROOT daemon, spin forward a couple of decades and FreeBSD has the first container project with the introduction of Jails. This is probably the first proper usage of containers as we have come to see them today.

Since the field was only occupied by Jails for a decade and then has grown quickly there is a sense of a new land. As such there are few similarities to existing ideas and virtually no standards for containers. However due to popularity Docker has now emerged as a de facto, or proto-, standard.

Steven gave a list of what he sees as the reasons for using containers:

- Process isolation
- Security - processes can't interfere with each other
- Resource limitation
- Portability
- Self-contained with all their own dependencies
- Fast to deploy
- It is a unit of deployment
- Non-repudiation

There is a sense that containers can allow you to forget about versions, dependencies as when the container is created it is properly baked. Then it is just a matter of controlling it. You can even add levels of security as to who can bake a container by using a notary framework. It is also wise to use just one process per container, containers do allow you to use more than one process (after a fashion) but this is an ill-advised behaviour.

Since the containers are baked and a known unit we suddenly have a measurement for our deployment systems, there is a 'unit of deployment' that is a container and its contents. The nature of containers means that we do not usually need to config manage them, but then we will have to build the container images manually. This seems like a backwards step in automation. With Docker there are Docker-files to build the Docker images to make the manual process creation easier but it still feels retrogressive in regards to configuration management.

Then we come to the issue of management and control of an underlying system. Docker may give application, or more correctly process, containerisation but you may still need to manage a number of underlying kernels across many servers or VMs.

Stephen recommends the use of a system such as CareOS. It would seem that although the top level system can be containerised there is still a place for configuration management of the underlying systems. This is where Puppet has a place as management of Docker using Puppet is made easy with a gathr-docker module.

The advantage of containerisation is that you can define applications with code. You can now move from just having the infrastructure as code to having the application layer as code.

> To bake an image you need a recipe

Stephen used the above (Chef-aimed) pun to indicate that although containers are a step forward you still need the power of configuration management but it is in a much reduced manner. Configuration Management tools make a powerful way to declare the state of an image with values, resources and packages. So we should run configuration management tools to make local file baking automated and share this across our systems.

> Convergence is less important, declaring desired state is more important.

So configuration management is still relevant, but in a greatly reduced capacity.

Architecture Automation: Matt S. Trout

The talk from Matt was first presented two years ago but Matt recently updated it and delivered it for us at FLOSS Spring. The talk is still relevant as it is a series of steps for good automation practices when you come to automate a system not a discussion of a recent piece of software.

> When you're up to your ass in alligators it's hard to remember you were supposed to be draining the swamp

Most of the talk was delivered as an homage, and dedicated, to a man called Wild Bill Walton, originator of the above quote, who was a man who in Matt's words:

> Managed to deliver a metaphor that both managers and technical people could understand... [and was a]... master of the folksy metaphor

The premise of this talk is that you may have to automate a server in your existing company, or a new company, or an acquisition, where there has been no accurate record of what has been placed on it. Even if you have a good development environment, use virtual machines and containerisation for segregation of services, it is still possible that you may have to do forensic analysis of someone else's work.

It is also possible that you are working as a start-up and don't have the option to do things 'the right way' in the first instance:

- Build customers first
- Build technical debt - this is inevitable and is not 'bad'
- Refactor when you have the time and funds

- Don't 2nd System yourself up-front

The ideal situation is to be in profit and at a period of relative stability. You may have rotated through some developers and iterations of codebase and have systems that now are starting to be the blockage in your evolution. Now we:

- Figure out what systems you have
- Figure out what services are where
- What's installed
- What's currently running

To do this we start by asking the Operating System and then work out the custom code. One thing to be wary of is if someone did development on the production machine that isn't replicated anywhere else.

> Grep everything for IP addresses... Firewalls are not just for security, they control what connects and force transparency

It is essential that you use configuration management even on small systems, some automation is better than no automation. The recommendation that Matt gave is that Pull-Based systems are preferred.

> Make sure it is Predictable, Repeatable, Stupid, Understandable - don't try to be clever, systems should not be smarter than you. Don't trust DNS TTL to be honoured by other systems: keep it simple, keep it stupid, keep it one Alligator at a time

Open Source: A Job and an Adventure: Dawn Foster

Dawn has spent a number of years working across the broad spectrum of software development organisations in the United States and beyond, she gave a very personal account of how to have an open source career and life. Her earlier career was working with manufacturers, since they do not see software as anything but a sunk cost, it isn't their income they are averse to spending large sums. It is for this reason that there is a generous uptake of open source in the manufacturing industries.

This is where Dawn first encountered open source and she quickly became fascinated by how the communities worked and in particular how they came to evolve such complex software projects.

Dawn covered three main areas in her talk as well as highlighting many of the people she had worked with across various communities. She spoke on:

- Why have a career in Open Source?
- How to make it happen.
- Time Management - and how to avoid burnout.

This very personal lists of reasons has echoes for me personally as it could have been a list I wrote myself. Dawn managed to summarise quite distinctly the reasons to be involved with communities. Splitting them into the three sections made them easier to absorb and relate to.

Why?

- Meet people from all over the world.
- Meet them in different groups - the same people often belong to more than one community.
- Travel - attend conferences and work with different companies of people.
- Open Source feels like you can talk more freely and companies seem to enjoy being known for their use of free software - a badge of pride.
- Open Source gives you more personal visibility.
- Open Source shows your talents and work.
- You get to have fun even with those people who are your competitors.

How:

- Start a new project - it should always be something that you need, not what you think others will want.
- Open Source projects need people with lots of skills.
- Work for an open source company.
- Bring open source software into your current job.
- Write and speak about open source.
- Try consulting on the open source skills you have.
- Documentation - the best route into any new project and the most common forst step.
- Be Nice.

Time Management:

- Prioritise.
- Document processes and procedures.
- Take time off for hobbies or a vacation.
- Do something fun each day.

Dawn's talk was sensible, empathetic and thoughtful. It was clear that she had found a good balance in both her life, her career and her open source participation which was healthy for her.

Data Revolution Catalyst: Lameck Amugongo

Lameck, who currently works in Germany, is originally from Namibia and still participates heavily in the open source and open data movements in his home country. Namibia is on the cusp of a new era in how it deals with open access with strong movements from their government towards empowering people using open standards.

Figure 2: Lameck prepares to deliver the closing talk (photo: Mark Keating)

We need data for everything

The idea of using open data, big data (government and large organisation records and readings) has been enshrined as a right by the United Nations who have called for 17 data codes to protect and distribute data to people.

Open data will help us to build sustainable tools and a sustainable future

In 2013 McKinsey and Company estimated the value of big data to be in excess of three trillion dollars, in 2015 that had raised to five trillion. This estimate is considered to be a conservative rationale. We are still in a fledgling stage with the use of big/open data, there are a number of needs that must be addressed as we move towards a more accessible future:

- Research into the current levels of availability and usage.
- Measuring via access, applications or solutions that use the data.
- Raw data available to all via on- and off-line access.

In emerging economies the best usage of this data is via the mobile networks. New dominant countries have stronger telecommunications infrastructure via mobile and satellite than traditional wire or fibre networks.

Why Open Data?

- Social functions.
- Economic freedom.
- Cultural need and change.
- Civic engagement.
- Decision making, and electoral services.
- Transparency of activity and accountability of usage.

This data revolution is not free but the people have already been paying their governments and organisations to collect it for decades via the existing mechanisms of taxation and civil records. There has also been massive government investment into environmental monitoring and scientific research producing petabytes of data for analysis and statistical modelling.

This data is closely tied to the phenomena often described as ‘the internet of things’ (IoT) - certainly the availability of cheap hardware and sensors, coupled with connected systems will drive a large wave of data collection. This could be a way of resolving the major challenges the world is facing, if it is available and understandable.

The Issues

There are some major problems in this data-driven world. There is a huge discrepancy caused by digital divides, this isn’t just the access to digital material but the competence in using or understanding it. Open data can give us a greater wealth of understanding and it can enrich the lives of everyone who has access to it. As long as they have access and can use it.

Open Data Innovation Hackathon

The Open Data Innovation Hackathon (ODIH) is an initiative created in Namibia and backed by the government to create applications that make use of the ideas behind Open Data. It has been held for the last 2 years and has created a number of innovative applications that are being developed for free distribution.

- GoToVote - Kenya - Helping people to register and to vote in elections.
- High Fives - Tracking the development of the African Continent.
- Fix My City - Reporting on local infrastructure issues - allows citizens to report directly to local authorities.
- Smart Meters - allows you to monitor and preserve water usage.
- eMERGE - help with emergency services notifications, report an issue and a location.
- Food Bank - trying to eradicate poverty by reporting food levels, access and need.
- eHealth - Pay for health services.
- CoW Bus Service - increase the mobility of citizens and access to transport.

Lameck proposes that the way forwards is the creation of more APIs to access and share the data.

> Big data is velocity, volume and variety, it is about dynamic scaling and performance. To use it we need flexible schemas, semi-structured data sources and personalisation of data to make it relevant to each individual so that they are engaged. It should enable us to process, understand, help and serve us in our environment. We need real time data and that includes operational data for access and consumption in real time.

The conclusion is that data has become an essential component of a modern writable society. To support this we need to create flexible bureaucracy that better suits the pace of technological change and digital freedoms.

> It is our data, let's liberate it.

One Step Beyond

Next Year's Spring Conference was announced at the end of the event. As usual the conference will move to a new city, this time it will be in Manchester, once again in March. The venue has yet to be announced though it is in negotiation/discussion, there will be workshops.

On the way to DevOps 2017 there are a number of FLOSS events this year. There will be Barcamps, in London, Birmingham and possibly Manchester.

DevOps is now the buzzword for what would have been programmatic system administration. The field seems new but is in fact mature and this is reflected in the quality, breadth and depth of the presentations and workshops at the FLOSS conference. The videos of the talks will be available on the FLOSS YouTube channel[2] in the coming weeks.

Spring 2016 conference review II

Kenneth MacDonald

Mandy Walls (Chef) opened the conference with a stirring keynote to reaffirm our enthusiasm of the sysadmin/DevOps field of endeavour. She suggested many of us are driven by curiosity and we should rely on that when “work stuff” doesn't appear cool - ask ourselves questions about it and it can suddenly become cool. As community members progress through life, we hope their coporate employers shoulder the risks of continuous learning as individuals' time is filled with family demands.

I chaired the Lethaby room track over the two days and am grateful to the speakers who all kept to time and to the attendees who engaged in the question and answer sessions at the end of each talk. The chair should always have a good question ready as a standby and fortunately none of mine were required.

Julien Pivotto (Inuits) supplied numerous hints and tips for getting the most out of systemd, many of which I was unaware of: “patching” vendor supplied units via additional configuration files, isolation features, making journald's log persistent and the “systemd-run” command.

Petr Jelinek (2ndQuadrant) reviewed the history of replication in PostgreSQL and introduced the new, powerful and extensible logical replication features currently under development.

Craig Gallen (OpenNMS developer) highlighted the problems of using RRD files for large scale network monitoring systems - its primary function was to be efficient when reading, not writing data. He described the changes made in OpenNMS to enable the operator to choose between RRD files or a Cassandra database without loosing any of the reporting and graphing

[2]http://www.flossuk.org/youtube

capabilities. This work has also allowed Grafana to pull data straight from OpenNMS core and measurement filters to be written in R.

Colin Charles (MariaDB Corporation) gave a tour de force of best practice for running your databases on your Linux infrastructure, whether physical or hosted by a cloud provider.

Richard Melville (Cellularity Limited) introduced us to the next generation packet filtering tools being developed on top of the Linux netfilter framework. His examples demonstrated its key differences from our familiar iptables rules.

Dave Cross (Perl consultant) closed the track with an entertaining report of his journey to publish his own book. Although it was never written, he did settle on an Open Source toolchain that started with editing Markdown and ended with an e-book on a well known Internet based shop, or a PDF.

A selection of excellent lightning talks rounded off the day. Monitoring your own elderly mother was perhaps one of the most unusual topics I've seen discussed at these conferences. A five minute trashing of almost every programming language you've heard of in favour of PHP raised the most laughs of the day!

The second day started with Andrew Beverley introducing his simple home-brew monitoring system - Admonitor, including a novel use of data collection. He appealed to the audience for ideas on how to progress the tool, and the discussion afterwards was full of valuable and friendly advice.

Jan-Piet Mens won the best talk of the conference with his entertaining, insightful and comprehensive review of currently available Open Source DNS servers. He enlivened the talk with anecdotes from his consulting work and some forthright personal opinions.

Colin Charles returned to the floor for his second talk which was a wide ranging review of options securing your databases, from the historical poor defaults through to the latest developments of encryption and authentication plugins. Although he works for MariaDB, he kept to the spirit of the conference by giving unbiased advice on several competing products.

Dawn Foster (Community/Open Source Consultant) closed the track with numerous short stories showing the different ways people became involved in the Open Source communities they now lead or are well known in. The lowest barrier to entry - that of helping with documentation was a pleasingly common way in.

The conference broke for an extended lunch which led into several "birds of a feather" discussions. I thoroughly enjoyed leading one on Configuration Management where we shared frustrations, ideas, practices, warnings, hopes and stories of our experiences of almost all of the configuration management frameworks out there.

Lameck Amugongo closed the conference with an inspiring presentation of Namibia's information and data access strategy to support its aspiration to become a knowledge based economy. This provoked much discussion on data protection, the difference between "old" and "new" legal systems and societies.

Chairing one track meant I only saw half of the talks, but if the others were as good as those I sat through, then all the delegates were treated to a wide ranging, entertaining and informative diet.

I am looking forward to meeting you all again in Manchester next year!

Spring 2016 conference review III

Quentin Wright

Everyone's conference is different. With multiple streams going on and time taken out to re-make old acquaintances or chat at one or other of the stands one can at best present a partial view of the goings-on. The venue was a great success and worked very well not just for the talks but for the dinner as well. Here follows a brief summary of all the talks that I attended. Some are proper write-ups, others are in the form of key-points. Enough I hope to give a flavour of the conference.

Keynote: Mandi Walls

Mandi described the changes that have taken place in systems particularly in the role of operations in larger organisations. Cheap and powerful machines on the desktop had debased the operator's role. Now with the emergence of the cloud and sophisticated deployment tools we have the new field of DevOps which is quite different in its importance to large organisations. Our work isn't confined to responding to the 'Call Tech Support - my mouse is dead' events.

Mandi had entertaining cat and mouse graphics to illustrate her points!

Other changes have occurred outside organisations. The resources to support the use of opensource have blossomed, so for example the latest OScon has about 20 parallel streams.

There are more radical changes coming in the future, particularly with the Internet of Things. Now the idea that the smart toaster can talk to the weather forecast and display information or adjust its settings is becoming reality.

An aversion to risk and blame culture exists in large organisations and it's hard to change them. It's a bit like stopping the Titanic. This factor accompanied by the ever decrease in job security has meant that DevOps favours the young and childless.

These changes can be daunting but you can change yourself. You have to accept that continuous learning is required.

To succeed at learning, you should cultivate aspiration and curiosity, acknowledge your vulnerabilities and become self aware.

Getting the Maximum out of systemd: Julien Pivetto

Julien has been using systemd since 2010. One of the advantages of systemd is that it starts more processes in parallel and makes better decisions during the process. It uses inotify, cgroups and other Linux features and is in most distributions.

There are alternatives to systemd: System V, Upstart in Ubuntu and OpenRC - which is mainly found in Gentoo.

Everything in systemd is a 'unit'. These are the base bricks of systemd. Some examples are: `network.target`, `mariadb.service`, `home.mount` and `session-1.scope`.

Units can be configured through INI-style text files. List: `systemctl list-units --all`. Read: `systemctl cat`. Units can be found in `/etc/systemd/system/*`, `/run/systemd/system/*` and `/usr/lib/systemd/system/*`. Packaged files go in: `/usr/lib`.

Units can be over-ridden. You can add, remove or change parameters and adapt them to your needs. It was hard to change services before systemd. You can use partial override, so just have commands in a file.

When implementing an over-ride use: `systemctl deamon-reload` and then verify that the unit has been loaded: `systemctl cat <service_name>`. To control units use: `start`, `status`, `stop`, `kill`, `help`, `enable` and `disable`.

Has a feature called 'masking' rather than disabling it will prevent units being launched by hand or systemd.

Systemd Services uses cgroup to track processes (see the [Service] section in units). The service is defined by the commands to run that service.. Have a 'simple' type, a 'forking' type and a 'oneshot' type - it is a reliable way to run commands as it will be run in a cgroup and is unrelated to the current shell

Problems solved by systemd:

- Run service as a different user
- Java service wrapper
- Go service wrapper
- You can still use custom scripts

Can use systemd to manage temporary files, to manage mount points and manage journals.

Can replace cron jobs using the concept of timers.

Can use systemd for socket activation.

All-in-all Julien gave an excellent overview of systemd, something that's hard to find. Other references that I've come across in endeavouring to get to grips with it are:

- Overview of systemd for RHEL 7 - Red Hat Customer Portal[3]
- Understanding and Using Systemd[4]

Configuration Management with Ansible and Git: Paul Waring

In the 'Olden Days' we used to edit files on each server that we were maintaining. This was a manual task and boring, repetitive and error prone. Nowadays we write a playbook or manifest and let the software do the rest.

Ansible needs Python 2 and ssh and runs on many OS's and scales up and down.

Paul ran through the basics of Ansible:

- Configuration contains global options, that's to say options that apply to all nodes. The file has a format similar to that of a Windows INI file.
- There is an inventory file. This lists individual managed nodes and allows global options to be overridden. Similar nodes can be grouped.
- Ansible modules. These are an abstraction of functionality, and as an example can be used to create accounts. Paul described the different types of modules. There are core modules, extras and third party modules.

Playbooks are list of tasks to run on nodes. They are Ansible's configuration, deployment, and orchestration language. They can describe a policy you want your remote systems to enforce, or a set of steps in a general IT process. If Ansible modules are the tools in your workshop, playbooks are your design plans.

Paul explained the differences between imperative vs declarative code and the virtues of using list comprehensions rather than loops. For more information on the use of playbooks see the project documentation.[5]

[3] https://access.redhat.com/articles/754933
[4] http://www.linux.com/learn/tutorials/788613-understanding-and-using-systemd/
[5] http://docs.ansible.com/ansible/playbooks.html

Ansible can be used very effectively with git hooks. A hook script can be used to check playbook script syntax before running the script. If the syntax checking fails a nonzero exit status can be set and Git doesn't proceed.

#Monitoringsucks To #Monitoringlove: Kris Buytaert

Monitoring is usually an afterthought. Often it used to be that monitoring is when the customer calls with a problem!

There was definitely a need to automate monitoring and in 2011 something better was needed.

There was famous debate in #monitoringsucks:

- manual config with gui
- not in sync with reality
- hosts only
- services sometimes
- application never

This was followed by #monitoringlove:

- a new era of tooling
- had hacksessions

Wanted:

- small components
- collect
- transport/mangle
- store
- analyse
- act/alert
- visualise

Sensu came along:

- It was awesome for non-static environments
- But this is Europe, U no do cloud..

There is a whole ecosystem around Graphite.

Graph-Explorer:

- Vimeo produced graph-explorer
- seems to be dead

. . . but now we have Grafana:

- which has a better display
- the ability to quickly make dashboards

Riemann:

- I still don't get it?
- Distributed Top
- Do you like Clojure?
- Riemann Health plugin?
- s/riemann-health/collectd/g
- output to graphite

Now Prometheus:

- Started 2012
- SoundCloud

- Metrics

But I have log files:

- Collect from anywhere
- Filter
- Send from anywhere
- logstash output
- Keep patterns around
- Selectively purge data

Checking for failure:

- Icinga
- Sensu
- Prometheus

Waking you up at night:

- Flapjack: notification routing
- Openduty: never managed to work

Aggregating:

- Thruk
- Grafana
- Dashing: use to build dashboards

nftables: Richard Melville

> The nascent Linux packet filtering framework

The benefit of nftables is that it replaces and combines `{ip, ip6, arp eb}tables`

Take a look at netfilter.org to see the differences. nftables looks very different from its predecessor. It has a different syntax and has done away with predefined tables and chains. However there are now families: `ip`, `arp`, `ip6`, `bridge`, `inet`, `netdev`. If you link to the inet family this will do ip and ip6.

netdev allows packet inspection before prerouting.

In one command multiple actions can be expressed. Note that packet and bytes counters are turned off be default and have to be explicitly turned on.

How is nftables better?

- Small code base
- Better syntax
- Simplified ipv4/ipv6 stack administration

See the source at: git://git.netfilter.org

There are two new libraries: libmnl and libnftnl

Getting started:

```
$ nft
$ modprobe nftables
$ lsmod | grep nftables
```

```
$ modprobe nf_tables_ipv4
```

New features: sets, dictionaries, intervals, maps, concatenations.

```
$ nft add table ip myfilter
$ nft add table arp myfilter
$ nft add chain ip myfilter input {type filter hook input priority 0 ;}
```

Can have negative priorities.

```
$ nft add rule ip myfilter input tcp dport 80 drop
$ nft list table ip myfilter
$ nft -a list table ip filter
$ nft delete rule ...
```

Backing up rules:

```
$ echo "nft flush ruleset" > backup.nft
```

Restoring:

```
$ nft -f backup.nft
```

Also see: Why you will love nftables - To Linux and beyond[6]

Writing Books - the Easy Stuff: Dave Cross

We're more familiar with Dave as the giver of courses on all things Perl from introduction through to advanced. As a departure from this area he took time out to tell us about his experiences in self-publishing a book and getting it to market, or rather as he said 'NOT publishing' his next book.

Dave identifed three stages in the process: Writing the book, publishing it and then doing the marketing.

One of the useful sources he found was 'APE - How to Publish a Book' by Kawasaki and Webb. The first observation that Dave made is that publishing is the easy bit. Once you've produced your book in the right format the rest is easy.

Various sources say 'Write a book in Word'. So before you even contemplate this: No! No! No! Don't do it! As a geek you should use text formats. Then you have access to the full Unix toolkit - grep, wc, version control and the rest.

So having got that out of the way which text format should you use?

- POD? He'd used it for the O'Reilly book.
- HTML? That's really a display format.

In the end he decided to use Markdown. Perhaps this is new to some, but is an excellent tool.

Having decided on the text format which output format should you use? There are three options that immediately come to mind:

- `.mobi` - format for Amazon eBooks
- `.epub` - most of the other places use it - it's an open standard
- `.pdf`

Well, yes maybe all of these! The Pandoc program will do the conversion from Markdown to ePub (and PDF if desired).

Amazon has the Kindlegen tool which will do conversion to, obviously enough, Kindle format.

Calibre, although it has the worst UI in the Universe has excellent command line tools and will do the conversion from ePub to PDF.

Other useful tools are `ebook-convert` and `epub.css` which changes the look and feel of an epub book from Amazon's standard layout.

[6] https://home.regit.org/2014/01/why-you-will-love-nftables/

Dave has documented the whole process on GitHub.[7]

So finally, does it work? Dave took a look at some blogs and decided to sign up to Kindle Direct Publishing and posted it on Amazon.

In addition Puppet have a document about the process[8] and also see the Pandoc web site.[9]

This was an excellent and interesting overview from Dave Cross of the process of preparing and publishing your own ebook. So now there can be no excuses. They say everyone has a book in them, so maybe now's the time to write it!

Is Configuration Management Still Relevant in a Containerised World: Stephen Grier

Stephen is in the IS Division of UCL. Although his presentation was about Puppet everything is relevant to other CM Tools.

Why CF? What does it give us:

- Automation of system configuration.
- Reproducibility
- Reliability
- Infrastructure as code
- Convergence towards a desired state. (Steve Burgess)
- A good CF tool is declarative.
- Puppet has a good sub-system of providers.
- A good system should be idempotent.

Containerisation - a brave new era. There has been steady progress with discrete stepts towards containerisation:

- chroot Unix v7 - 1979
- FreeBSD Jails - 2000
- Control groups - 2.6.24 - 2008 (Google - process containers)
- Linux namespaces - 2.6.24 - 2008
- LXC - 2008
- Docker - 2013 *Rocket - CoreOS - 2014

Docker adds tooling, so extends the capabilities of LXC.

We don't have a good specification around containerisation.

It's all about process isolation. Containers provide a new unit of deployment. You don't have to know about the libraries on the host and the application developers can use whatever language they want.

Why Use Containers?

- Process isolation
- Security
- Resource Limits
- Portable
- Self contained with all dependencies
- Very fast to deploy
- Unit of deployment
- Non repudiation - Docker 1.8

Should you bake or should you fry?

[7] https://github.com/davorg/make-ebook
[8] https://puppet.com/blog/how-we-automated-our-ebook-builds-pandoc-and-kindlegen
[9] http://pandoc.org/epub.html

- Docker images (or AMI or Vagrant boxes) are baked.
- Typically we don't config manage a container.
- Ideally you should have a single process running in a single container.
- Images might be built using Jenkins..
- You're interested in the final state of the image rather than using a script.

Docker has dockerfiles which allow you to define your image. Note that dockerfiles can be version controlled. This means that there is a repeatable way of building images.

With `docker::compose applications` can be defined as code and puppet apply will run the images from a dockerfile.

Architecture Automation: Matt Trout

I felt like taking Matt to task because at the start of his talk he maintained that he had to use a Windows laptop because it was the only machine that would work first time with a projector! Oh well!

Matt described his accumulated experience as a consultant carrying out migration or consolidation activities in new organisations as they moved from theinitial start up to something more mature.

With early stage startups technical debt is everywhere! Priority has gone into making a product fit for the market rather than striving for elegance. Maybe the outfits that seek elegant solutions never make it anyway!

There are a number of steps in the process:

Step 0:

- (Geeks always start at square zero instead of square one!)
- What servers do we have?
- Look for that key machine that has been forgotten!

Step 1:

- You get bonus points if the guy who knew it all left - it's often the case!
- Don't trust the documentation
- what's installed?:
- use dpkg
  - other tools provided by the os
- what about custom code?
  - seek it out, maybe destroy
- repositories - locate .git
- is it the real thing?
  - has development been done on the development machine?
  - often development is done everywhere and production is for some unknown reason running on the development machine.
  - use `Dist::Surveyor` in Perl environment. It does checksums for files.
  - what talks to what?

Step 2:

- enumerate running services
- use daemontools
  - `ps aux` every 5 minutes for 24 hours
  - `lsof`
  - `netstat`
- do cross-referencing
  - put into a wiki format
  - dump the output into a git repo, maybe using JSON

  - grep out things you recognise
- grep everything for hard-code ip addresses
  - there WILL be one somewhere

Step 3: Go find a beer..

If possible use fresh machines. Your existing machines will be missing security fixes. Always assume the worst.

Automation approaches:

- pick something pull-based
- Ansible is a choreography tool - avoid
- pull-based systems converge
- config generation
  - if you already know TT just use TT
  - don't try to be clever

DNS is a mess? Well dns is usually a mess. `rsync /etc/hosts`

Backup everything

Build new machines and restore backup data onto them (now you've tested your backups).

Change something and check the slaves.

Think about migration strategies. Don't trust DNS timeouts. use rsync, change dns, rsync again.

Step 4:

- Decide which service to tackle next.
- Repeat

As ever Matt gave an entertaining and stimulating talk about how things really are at the coal-face rather than the way they should be.

Open Source: A Job & An Adventure: Dawn Foster

Dawn described herself as a 'Geek, traveller and reader', with a 20 year tech career. She's currently a PhD student at the University of Greenwich researching communication amongst the Linux kernel developers. Previously she'd worked at Intel on a team looking at developer tools. She also spent time looking at the communities for these tools and discovered that there was lots of structure amongst the communities.

Why have a career in open source?

- Meet amazing friends from around the world.
- Run across the same people on different project.
- Attend conferences and work with other companies
- Because it's open source it's possible to be open about the technical content
- Visible work and connections to people

Working in open source is awesome. You or free as in freedom, innovative, and the work is interesting and collaborative.

How do you get started in open source? One way is to start a new project that you need. However this is probably one of the hardest ways to get into open source. It's maybe one of the most demanding, but then is perhaps the most rewarding.

Another way is to participate. This way you learn new skills, and become known within the project. This pathway does not just apply to developers. There are other ways to get involved for example by making blog posts or getting involved in documentation.

Dawn's blog is at fastwonderblog.com.

Micro:bit finally reaches UK children

Les Pounder

The BBC's micro:bit is finally making its way into the hands of eager children across the UK and I have seen reports of teachers getting to grips with this new single board microcontroller ahead of their classes. One million children in Year 7, the first year of secondary education in the United Kingdom, will receive a free micro:bit board with which they can learn new skills. Schools will also be able to purchase more boards.

The micro:bit is a powerful platform for experimentation and offers a 32-Bit ARM Cortex M0 CPU, which includes Bluetooth 4 and Bluetooth Low Energy (LE) wireless connectivity. Micro:bit also comes with an accelerometer which can be used to detect movement and gestures, a compass to detect direction, input comes in the form of two programmable buttons, lastly there is an LED matrix of 25 LEDs which can be used as a basic form of output.

The micro:bit is designed to be used with crocodile clips, attached to five GPIO (General Purpose Input Output) pins present on the board. However there are more GPIO pins available thanks to the 20 pin edge connector. Using these pins requires a breakout board and the leading provider for these are Kitronik[10] who have been working with the BBC to create a range of products to support the micro:bit. The micro:bit Prototyping System is a great starting point. I recently picked one of these up and it has helped my prototyping immensely. The Prototyping System provides a means to break out all of the GPIO pins and use them on a breadboard. This means that I can use the micro:bit with existing cheap components, that I have plenty of thanks to other boards such as the Arduino and Raspberry Pi.

Programming the micro:bit is possible in multiple languages,[11] with Microsoft offering their own version of a Block based editor. This editor is very similar to Blockly and offers an easy to use introduction to programming the board. The next language has a direct link to the Block Editor. Touch Develop is a typed language that has an unusual, almost tablet like interface that enables children to develop their code from the Block Editor and turn it into a "real" language. Code created in the Block Editor can be converted to Touch Develop to illustrate the link between the languages. It is also possible to program the micro:bit using JavaScript, specifically with Code Kingdoms and this would be a great way to introduce JavaScript as a general purpose language, and not just as a web development tool.

But the most exciting language on the micro:bit is something that has been grown and supported by a community of makers, hackers, coders and thinkers, and that language is Python.

Python has been chosen as the language to teach in schools across the UK, largely due to the Raspberry Pi but also thanks to a community who have lowered the barrier for entry and enabled anyone to write Python. The Python Software Foundation (PSF) were approached via Nicholas Tollervey to create a Python framework for the micro:bit. Thanks in part to the micro python project[12] headed by Damien George, which offers an implementation of Python 3 for micro-controllers. The result of this consultation with the PSF community are two fold. Firstly we have a web implementation on the micro:bit website, here we can write Python code in the browser and download it to the micro:bit. Secondly we have mu,[13] a Python editor for the micro:bit that is designed for children to get coding quickly. Mu has a standard editor

[10] https://www.kitronik.co.uk/bbc-micro-bit-accessories.html
[11] https://www.microbit.co.uk/create-code
[12] https://micropython.org/
[13] https://github.com/ntoll/mu

window into which we can type our code. By inserting our micro:bit into a computer we see the device appear as a USB flash storage device. Our code can be uploaded to the micro:bit and it will auto reset to start the new code. Also present with mu is REPL (Read Evaluate Print Loop) a method of connecting to the micro:bit via a USB serial interface, using this we can easily control the micro:bit in an interactive session. This is very handy for quick tests and hacks.

The micro:bit is an excellent piece of kit and will prove to be an interesting introduction for children and adults looking to get started with physical computing.I have it on good authority that the micro:bit will be on sale to the public this year, but no firm dates as yet.

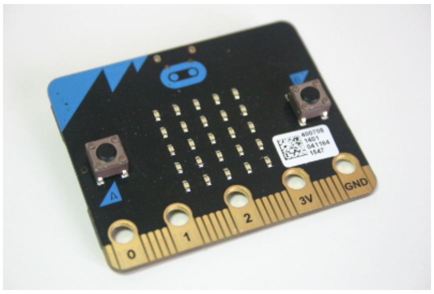

Figure 3: Microbit

Cooking for Geeks, 2nd Edition

Author: Jeff Potter
Publisher: O'Reilly
IBSN: 978-1-4919-2805-9
Pages: 488
Price: £22.50
Published: October 2015
Reviewed by: Nigel Barker

Do cookery and geekery go hand in hand? O'Reilly think so and indeed the first edition of this book sold more than 100,000 copies. I fit in that target demographic, but although the book was on my wants list, I never actually picked one up until this second edition which comes with 150 pages of new material.

As I expected there is a lot to like about this book. It's added to my wealth of trivia, e.g. the origin of the word restaurant, and it's made me curious about cooking a number of things I hadn't considered before, like endive, edamame, kale chips and zabaglione.

Chapters are based on science ideas, so recipes don't follow in the usual cookbook order of breakfasts, soups, starters, main dishes etc. Instead the chapters cover taste, time and temperature, air and water, chemicals and, the one I thought was most fun, hardware (pressure cooking, sous vide, fast freezing etc.). Each chapter has theory, experiments, interviews and recipes.

Some of the recipes seem a bit simple compared to other cookbooks, I guess this is because they are intended as starting points to illustrate a particular idea rather than fully finished recipes for successful entertaining.

There are tips that are new to me, like cutting tomatoes with a bread knife, and science which I hadn't read before, like the actual definition of the Scoville scale. There is a handy cooking personality quiz in the introduction – I'm a methodical cook whereas my wife is a giving cook.

My family are vegetarian so lots of stuff about meat temperatures wasn't really relevant to my cooking but I was still interested in the information about safety and bacteria.

My only gripe is that this is an American book and the cultural differences show. Quantities are given in grams as well as cups, but us Brits have to remember a barbecue is called a grill while a grill is a broiler. And what the heck are Snickerdoodles?

You don't know JS: ES6 and Beyond

Author: Kyle Simpson
Publisher: O'Reilly
IBSN: 978-1-4919-0424-4
Pages: 278
Price: £11.50
Published: December 2015
Reviewed by: Charlie Harvey

Now that JavaScript has become the defacto virtual machine of the web and modern browsers make up a significant part of the market, it is both desirable and possible for the language to develop further. Hence the finalization of the ECMAScript 6[14] in June 2015. And hence also a crop of books, this one included, covering the new language features.

The "You don't know JS" series is intended to cover JavaScript comprehensively and deeply – a sort of inverse of Douglas Crockford's "JavaScript: the good parts" it sets out to cover all the parts, good, bad or otherwise. In the words of the series preface:[15]

> JavaScript is awesome. It's easy to learn partially, and much harder to learn completely (or even sufficiently). When developers encounter confusion, they usually blame the language instead of their lack of understanding. These books aim to fix that, inspiring a strong appreciation for the language you can now, and should, deeply know.

The entire series was crowdfunded on Kickstarter[16] and the draft manuscripts[17] are available to read free (as in beer) on GitHub. This more open, though not quite free/open source, approach to publishing is a good model for how publishing might work in the future. You gauge that an audience exists and cover some costs with a crowdfunder; you build an audience, spot the errors and act fairly by letting everyone see the content at no cost online; and then you make some profit on sales of a printed book.

The book covers a lot of material and is pitched at a reasonably high level. Perhaps the latter fact can be attributed to this being the last book in the series. It is certainly the case that the text refers frequently to earlier books where more in depth explanations of language features can be found.

The content is arranged into thematic chapters covering the current and future situation of ES6 (browser support, transpilers and so on); new syntax (block scoped declarations at last), organization of code (classes, modules, iterators and generators), async flow control and promises, collections, new API features, meta-programming, and some future-gazing in a chapter called "Beyond ES6".

Simpson is not afraid to be opinionated or to say when he feels that the design might be better. His code examples are short and to the point, though on occasions I might have appreciated

[14]http://www.ecma-international.org/ecma-262/6.0/ECMA-262.pdf
[15]https://github.com/getify/You-Dont-Know-JS/blob/master/preface.md
[16]https://www.kickstarter.com/projects/getify/you-dont-know-js-book-series
[17]https://github.com/getify/You-Dont-Know-JS/blob/master/README.md#you-dont-know-js-book-series

more of a walkthrough in the text. It's tricky to evaluate JavaScript in one's head, especially when new language features were being demonstrated.

Clearly this is not the book for you if you want to "Learn JavaScript in N hours/days"; it is aimed at those who want to understand the whole language deeply and it is certainly comprehensive. Indeed, this volume could function as much as a reference as a tutorial. If your job involves more than a few hours a day of JavaScript wrangling or if you are deeply unsatisfied that most of your JavaScript programming consists of cut and pasting jQuery that you found on StackOverflow then the "You Don't Know" series should be on your reading list.

Learning MySQL and MariaDB

Author: Russell J.T. Dyer
Publisher: O'Reilly
IBSN: 978-1-4493-6290-4
Pages: 408
Price: £23.00
Published: March 2015
Reviewed by: Charlie Harvey

One of O'Reilly's excellent "red" series of database books, this is a comprehensive primer on using both MySQL (now owned by Oracle) and MariaDB. MariaDB is a community effort started by MySQL's original creator Monty Widenius, who also supplies a foreword for the book.

The book is divided into 5 main sections, covering:

- Basic installation and use of the software
- Creating and altering tables
- Inserts, selects, updates and joins
- Built in functions and administering MySQL. This final section also very briefly covers the C, Perl, Python, PHP and Ruby APIs.

I can imagine this being a useful book for a junior dev to pick up and work through near the start of their career. It is aimed at folks with no previous experience of MySQL, but the coverage is detailed enough that it would probably also suit people who have worked on another RDBMS and wanted a quick introduction to how things are done in the MySQL world.

At 370 pages, this book is clearly less comprehensive than Paul DuBois's MySQL Developer's Library (aka "the purple MySQL book"), which is one of the best MySQL references I have encountered. However, that may be no bad thing if what you are after is "just" a primer rather than something more comprehensive. Shorter books have the advantage of what some media people call "finishability". Being able to get to the end has a powerful motivational effect on many readers.[18]

As well as being concise, Dyer writes accurately and accessibly. Importantly, he covers pretty much all of the basics. Also, he takes time to explain the motivation behind the examples in the text, of which there are plenty.

Each of the 16 chapters ends with a series of exercises, giving you a chance to practise some of the material covered in that chapter, an approach which should help cement your learning. The exercises seemed straightforward enough to me, mostly taking the form of practising using syntax covered in the text rather than requiring any great leaps of insight.

There are decent links to further resources at various places during the book – notably in the very brief chapters covering the various APIs whih you can use to talk to MySQL.

[18] http://www.gyford.com/phil/writing/2009/12/18/finishability.php

There are a small number of areas which may have been useful to include - the different types of tables (InnoDB, MyISAM, etc.), and sorting out collation and character encoding, for example. But overall I felt that there was enough material in enough detail to make this a useful read. If you haven't worked with MySQL or if you want a solid tutorial for a new developer, then Learning MySQL and MariaDB would be a good choice of book to start off with.

Information Architecture, 4th Edition

Authors: Louis Rosenfeld, Peter Morville, Jorge Arango
Publisher: O'Reilly
IBSN: 978-1-4919-1168-6
Pages: 486
Price: £25.50
Published: October 2015
Reviewed by: Lindsay Marshall

I'm pretty sure I reviewed the first edition of this book back in 1998, but the architecture of my information archive is such that I can't easily locate that text - clearly, I should have read it more thoroughly! However, I do recall that I thought the book was excellent then, and that is still true now. The material is up to date and shows the experience of nearly twenty years of development and expansion.

The books is in three parts, with the first providing an overview of the fundamental concepts that underly the field, focussing on the importance of finding and understanding information. With a dig at the much discussed information architecture problems of iTunes along the way. This section sets things up well and goes through things in a comprehensible way without ever getting too technical - even managers can read this. And they probably should.

Part 2 looks at the principles you need to understand: organising, labelling, navigation and searching. Each chapter takes you through a topic, introducing it and then looking at solutions. I found the sections on navigation and searching to be particularly good, helped immensely by the full colour screenshots of web pages (many of them already familiar and widely used) that appear throughout the book. The final chapter in this part looks deeper at semantic issues and touches on metadata, a topic that I think that there is much more to be said about than is included here. One of the worst aspects of most websites is their lack of support for machine readable metadata schemes (like Dublin Core or Open Graph) and there is nothing here to promote them which is unfortunate.

The final part looks at pulling all this together and how you design and build an Information Architecture for a business. This covers research (with lots of emphasis on looking at real users), getting buy in, making a strategy and then how to document it all effectively. All of this is at a high level - the book is not recommending specific tools or techniques, but that is not significant as it is easy to apply the ideas to particular work environments and practices.

This is still the best book available on the subject and if I were teaching a course on Information Architecture I would certainly recommend this as the supporting textbook, and would structure my course in a similar fashion to it.

UX for Beginners (A Crash Course in 100 Short Lessons)

Author: Joel Marsh
Publisher: O'Reilly

IBSN: 978-1-4919-1268-3
Pages: 256
Price: £20.00
Published: December 2015
Reviewed by: Lindsay Marshall

There's a lot to like about this book, but also things you might dislike. The genesis of the book was an internal mailing list, and then a blog, and then an email-based short course, and these lead to the 100 short lessons of the title. Mostly these are one or two pages of text with a small number running over more, though these usually contain several illustrations rather than more words. The aim is to cover every aspect of UX at some level and the book definitely does that, giving sound information and with illustrative stories from the author's own experiences. There's lots of good background information drawn from psychology and HCI research, all of it presented in digestible chunks that don't demand previous technical knowledge. The more design-oriented sections work in the same way, presenting the bits of standard graphic design knowledge that are relevant. There's useful material on content creation, measurement and business aspects as well.

So, overall it looks good and does what it says on the cover, but as I said above, there are things about it that you just might not like (as always your mileage may vary!). First off, the book is landscape format rather than portrait so if you have narrow shelves it will stick out awkwardly: a minor issue I know, but potentially annoying. The book is profusely illustrated, and if you don't mind the style used then you'll have no problem, but, personally, I didn't like it at all and found it put me off. The writing style is light and tries to be slightly jokey - I wasn't amused though and found it a bit wearing after a while: what may work well in email messages that arrive over a period of time doesn't feel right for the flow of a book. Some other reviewers did not like the language used and thought the book inappropriate for a business environment. Flow is also a problem with the 100 short lessons. There are lots of books that are split up into short sections - one of the best (that everyone should own) is Universal Principles of Design - but where this works best is where the sections do not directly relate. In UX for Beginners there is necessarily a flow through the sections and some are so short that it just starts to feel bitty. You would think you could dip in and out, but you can't. From a technical standpoint though the coverage and ideas are fine, but I would have liked to see more coverage of accessibility - the section on it is very weak indeed - and I would also have liked to see many more pointers to places that I could read about a topic in depth. You don't stay a beginner forever and you want ways to move on.

All that being said, I would probably recommend this book to my students as an all- round introduction, but at at nearly £20, very few of them would take it up. Pricing is part of UX too.

Learning Unix for OS X, 2nd Edition

Author: Dave Taylor
Publisher: O'Reilly
IBSN: 978-1-4919-3998-7
Pages: 238
Price: £15.50
Published: February 2016
Reviewed by: Chris Stoddart

The Unix command line is where I spend my working day; my desk is dominated by two large Linux monitors with a MacBook alongside for watching email and keeping me awake when I take it to meetings. However, in common with many other Mac users, I only occasionally venture into the Mac's terminal and even more rarely into its command line. I was therefore looking forward to reading this book as a spur towards using my MacBook in a more creative way.

Dave Taylor's book is quite slim for an O'Reilly publication at just over 200 pages. This latest version also covers the changes introduced with El Capitan (OS X 10.11). It is written in an accessible, readable style that begs to be picked up and dipped into at random. I kept it beside my bed and read a few more pages and learned a couple of new things, or more accurately, a couple of Mac ways of doing previously familiar things almost every night. The book largely covers commands that most of us will have been using for a long time. I personally do not mind having my memory jogged about commands I have not really thought about for a while, plus every so often there would be a particularly Apple quirk that made me go 'aha'. For example there are a few references to Applescript commands that I was not previously aware of at all.

It has to be said that this book is not aimed at the seasoned Unix user and I would be happy giving it to someone who had never been exposed to the Unix command line interface before. It starts off from very simple beginnings - what a command line is, where the terminal program can be found and so on. However, even for an experienced user it can become useful quite quickly. Early on there is a section on customising the terminal, which for those of us who don't particularly enjoy experimenting with click boxes and radio buttons, is a handy guide to making our terminals behave in a more personal way. From there onwards we progress through a couple of large sections on dealing with the filesystem and files, some basic I/O redirects, some simple online commands, a chapter on X11, and finally a frustratingly short teaser on further learning.

Ultimately I would have liked to have seen more advanced commands and options discussed later in the book - surely some scope there for a larger edition? To that end, I suggest that the 'Going Deep' part of the book's subtitle is something of a misnomer. Despite that, I still enjoyed reading it and found the book useful. On many occasions I caught myself wondering why I had not bothered to fire up the Mac's terminal and type a few commands before instead of, for example, battling on with the dumbness of the Finder. As a result of this book, I shall certainly be using my MacBook a little differently from now on.

C in a Nutshell

Authors: Peter Prinz, Tony Crawford
Publisher: O'Reilly
IBSN: 978-1-4919-0475-6
Pages: 824
Price: £28.00
Published: December 2015
Reviewed by: Paul Waring

The O'Reilly 'in a Nutshell' series are reference works, describing the details of a particular topic with examples. As it made clear up front, this is not a book for those new to C or something which you will read cover to cover, instead you should view it as a reference to keep beside you when programming.

It's good to see that this book is bang up to date, covering the latest C11 standard. Whilst this hasn't been fully implemented by all compilers – and those using embedded platforms may have to wait a long time for full support – the book has at least got some future-proofing built in. The authors have also clearly labelled C11 and C99 features so you can skip over these if you need to use ANSI C.

It's always difficult to review a reference book as this is something which you dip into over time, but some of the things I learnt from an initial pass include:

- How wide and multi-byte characters work in C11, and how to define UTF-8 strings.
- There is a pre-defined constant called `__func__` which contains the name of the current function (useful for printing debug messages).
- What you can and can't assume about types (e.g. `char` may be `signed` or `unsigned` by default).
- How to use member designators to initialise specific parts of a struct (used frequently in the Linux kernel).

Arrays, pointers and dynamic memory management get a chapter each, as befits their importance and complexity. The standard library is also covered, and the final part of the book looks at the various components of the C toolchain (the authors assume GCC is the compiler, but clang supports many of the same command line options).

Overall, this is a solid reference to C and especially useful if you need to write code which works on a wide variety of platforms, as the authors spend a lot of time exploring undefined behaviour. If your job or hobby involves writing C, this is the book which should be on your desktop (physical or virtual).

Electronic Dreams

Authors: Tom Lean
Publisher: Bloomsbury Sigma
IBSN: 978-1472918338
Pages: 288
Price: £15.00
Published: February 2016
Reviewed by: Paul Waring

Disclosure: I was at university with the author of Electronic Dreams - though I still had to pay full price for my copy of the book!

Most of the books reviewed for the newsletter are hot off the press and relate to the latest tools and languages, such as C11, ES6, Swift etc. Occasionally though it's fun to take a trip down memory lane and reminisce about the 'good old days' – in this case 1980s computing.

The first thing I noticed about this book is that Tom has done his research, no doubt assisted by his current project involving conducting interviews with figures from science and techology, including Steve Furber and Sophie Wilson. Many of the interviewees are refreshingly frank about topics such as the (un)reliability of the computers and the 'flying by the seat of our pants' method of running a computer-related business in the 1980s. My favourite anecdote though is the one where Furber and Wilson are separately told that the other has said it would be possible to have a prototype ready by the end of the week, as a way of spurring them on.

Even though I have read widely about the period in question, there were still some new facts and titbits to enjoy. For example, I'd somehow managed to miss Prestel, the Post Office system which could be seen as an early version of the World Wide Web. Also, whilst the information on the Acorn, Spectrum and Commodore machines didn't cover anytthing new to me, the Oric computer was something which I hadn't heard of before.

After reading this book I had mixed feelings about the subject matter – pride that Britain was a pioneer in the field of computing, and disappointment that we failed to follow up on early successes and instead left it to the United States to lead the world. Even where we have succeeded, it's often 'behind the scenes' and uncredited (e.g. both the iPhone and iPad use hardware designed by UK companies ARM and Imagination Technologies). Hopefully this book will go some way to spreading the word about historical British contribution to computing, and perhaps even spur on the next generation.

If you have any interest in the history of computing, this book is well worth reading. Tom has managed to tell an interesting story without resorting to hundreds of footnotes – something which academics writing books aimed at a wider audience sometimes fall down on.

Contributors

Nigel Barker lives in East Lothian and works at Dell SecureWorks. His first exposure to Unix was when the comic distributor he worked for bought a Texas Instruments TI-1500 mini computer in 1989.

Charlie Harvey is IT Director of New Internationalist workers co-op, who produce an award-winning magazine, books about global justice and run ethical online shops. He has been working with free software since the late 90s. He enjoys cider and blogs at charlieharvey.org.uk.

Mark Keating is Managing Director at Shadowcat Systems, a member of FLOSS UK, prominent in the Perl community and organiser of events such as London Perl Workshop. Follow him on Twitter: `@shadowcat_mdk`.

Kenneth MacDonald is a senior computing officer at the University of Edinburgh and FLOSS UK Council member. He discovered Linux at a time when we had to swap the kernel and root floppy discs at boot time while coding for his PhD in Geophysics and has been managing systems at the university ever since, with a particular interest in configuration management.

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 FLOSS UK, and manages the FLOSS UK office at the Manor House in Buntingford. She has been involved with FLOSS UK administration since 1987. In addition to FLOSS UK, Jane is Company Secretary for a trade association (Fibreoptic 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.

Chris Stoddart started programming seriously during his physics degree when he re-wrote the software that interfaced an X-ray diffractometer to a BBC micro. He was introduced to UNIX sysadminning during his PhD when he was somehow left in charge of the Sun workstations. Since then he's looked after Vaxen at Bradford University, Alphas at The Open University and currently takes care of the server room for the Department of Computer Science at Sheffield University.

Paul Waring works as a freelance IT consultant. Outside of work he can usually be found filing documentation bugs against various open source and free software projects. He also edits the FLOSS UK newsletter. Follow him on Twitter: `@pwaring`

Quentin Wright is a Director of Sunfield Technology working with Linux and Ruby and system integration projects. In his spare time in between playing with reluctant motor vehicles he is pre-occupied with Web and Javascript programming.