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 23, Number 4 **ISSN 0965-9412** **December 2014**

Contents

From the Secretariat

Jane Morrison

The minutes from the UKUUG Ltd. Annual General Meeting held on 25th September have once again not been circulated to members via hard copy. All members were emailed on 9th October and advised that the minutes and associated documents could be found on the web site at: `http://www.flossuk.org/Events/AGM2014`

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

Current Council members are: Kimball Johnson, Ian Norton, Stephen Quinney, Quentin Wright, Gavin Atkinson and Tim Fletcher. We thank retired Council member, Holger Kraus for his past commitment.

We are currently putting in place a full schedule of events for 2015 - to date the following have been agreed:

Our 5th un-conference - London - Saturday 7th February 2015. London Hackspace, 447 Hackney Road, London E2 9DY see page 7 for more information.

DEVOPS Spring 2015 - 24th, 25th & 26th March 2015.

The Hilton York, 1 Towers Street, York YO1 9WD

Tuesday 24th March - Workshops

Wednesday 25th & Thursday 26th - Conference

This event is the UK's only conference aimed specifically at systems and network administrators. It always attracts a large number of professionals from sites of all shapes and sizes. As well as technical talks, the conference provides a friendly environment for delegates to meet, learn, and enjoy lively debate on a host of talks.

OpenTech 2015 - Saturday 13th June

Student Central (formerly University of London Union)

OpenTech will be 10 years old in 2015! for more information see page 7.

Dynamic Languages Conference - Saturday 20th June 2015 Manchester. For more information see page 8.

We are also planning more tutorials for 2015 - if you have any particular topics that you think would be of interest please let us know.

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

We also provide other discounts for members see: `http://www.flossuk.org/OtherDiscounts`

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

We would like to thank the continued support of our Silver Sponsor member, SUSE. Their involvement does help us to keep event fees down.

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

The Secretariat will close on Thursday 18th December and re-open on Monday 5th January 2015.

Please note the copy date for the next issue of the Newsletter (March 2015) is 13th February. We are always looking for interesting submissions from members, so if you have any news, articles etc. please send copy direct to: **`newsletter@ukuug.org`**

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

Chairman's Report

Kimball Johnson

Resignation of Chairman

I announced in the last newsletter that I was tendering my resignation as Chairman. It was however not accepted by Council, and with the pledge to share some of the burden I have agreed to continue for now, at least until after the Spring Conference in York. I would like to mention that we have spaces available on Council, and that help in running the organisation and the events would be greatly appreciated.

Events

The Birmingham Barcamp was once again a success, with a number of people in attendance, I want to give my thanks to Quentin for running it, and for all those that attended. I look forward to it happening again next year.

The ever popular Perl Tutorials by Dave Cross were a success once again, thank you to Dave, and all those that attended, I hope they were greatly beneficial to you. I suspect they will run again next year if you missed them this time.

Coming up we have a number of events planned, firstly in February in London we have the Annual Unconference. This year it is moving to be held at the London Hackspace, on Hackney Road. A number of the members have been to the previous London Unconferences, and I thank the LHS for welcoming us. The date is Saturday 7th Feburary, this is the Saturday following the popular FOSDEM event in Belgium, so if you are traveling through London for that, consider staying around in London for the Unconference if you can.

In March of course is our Annual Spring Conference, this year to be held in York from the 24th to the 26th of March. The call for papers is still open, please visit our website to submit a talk. Finally planned in June, on the 20th, is the Dynamic Languages Conference, this is not the first time this event has run, but it is the first time FLOSS UK has assisted. Further deatails are available at **`www.dynamiclanguages.co.uk`**. I hope many will attend, if you wish to submit a talk please visit the website.

I would also like to encourage anyone with ideas for courses they would like to be run to email the office with their ideas.

Support for local user groups

The budget is still available for FLOSS UK to assist local user groups by helping them to obtain speakers for their events and assist with travel expenses. In addition we have a small budget for assisting with projects that would benefit the Free Software or Free Hardware communities in some way.

If you have an idea and wish some support, please contact Jane on **office@flossuk.org** and it will be discussed by Council.

Get Involved

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

From the Editor

Paul Waring

It is with great sadness that I include an obituary of long-standing member Mick Farmer, who passed away on Sunday 16th November 2014. Jim Reid has collected contributions from a number of members who have known Mick over his long period of involvement with the organisation, which can be found on page 5.

On a happier note, we have three events coming up which are advertised in the newsletter: OpenTech, the annual Unconference and the Dynamic Languages Conference. We also have an article from Les Pounder on Creating Coding, and six reviews of books from O'Reilly, No Starch Press and Pragmatic Bookshelf.

We are always looking for newsletter content across the following categories:

- Articles: 500-1000 words on a technical subject. Longer articles can be split over multiple issues if required. Past issues have included topics as diverse as Arduinos, ZFS and software patents.
- Event reports: Both FLOSS UK events and those which are of interest to our membership in general, e.g. Oggcamp.
- Book reviews: 250-500 words on a technical book. We can now provide review copies of O'Reilly books in electronic (PDF, ePub, Mobi) and print formats. Subscribe to the mailing list at **http://lists.ukuug.org/mailman/listinfo/books** to find out which books are available for review.

If you have any ideas for newsletter items, please do get in touch via email: **newsletter@flossuk.org**.

Mick Farmer: 1945-2014

Jim Reid

Mick Farmer died suddenly at home on Sunday 16th November 2014. He was 69.

Mick was involved with UNIX systems since the early days of his career in the late 1970s and early 80s. He was one of the first members of what was then called the European UNIX Users Group, EUUG, and joined the board of the UK Unix Users Group (UKUUG) shortly after the organisation was formed. When UKUUG became a limited company, Mick served on the Council for over a decade, first as Secretary (1986-1992) and then as Chairman between 1992 and 1998. When he stood down in 1998, Mick was awarded Honorary Membership of UKUUG.

Mick was a lecturer in the Department of Computer Science and Information Systems at Birkbeck College from 1968 until 2011. He taught courses at both undergraduate and postgraduate level on operating systems, compiler design, computer networking, programming (C, Pascal, Java, C++, etc), Internet technology and e-business. He was one of the first enthusiasts for the Amsterdam Compiler Kit developed by Andy Tanenbaum. Mick wrote successful text books on compilers and C and Pascal programming. Mick also gave seminars and delivered academic papers on numerical analysis, his PhD topic. In his spare time he did training and consultancy projects on HTML, PHP, Perl and web design, usually on Linux systems.

Mick's involvement with UNIX began in 1979. He heard that UNIX could run on the department's PDP11/05 computer and wrote to Dennis Ritchie, enclosing a 2.4 MB RK05 disk. Dennis returned the disk with the complete operating system and source code installed on it!

His contributions to the UKUUG were very significant, particularly in the early days of the organisation. Much of that went unnoticed by the membership: Mick chose to keep out of the limelight and just get the job done. He oversaw the growth of the organisation in the 1980s and 90s and was involved in the appointment of a professional secretariat, essentially the same secretariat which currently administers UKUUG's successor. He arranged for a paid editor to take charge of the newsletter which until then had been produced on a volunteer, best-efforts basis. Mick set up the relationship with O'Reilly which led to the company's long-running support for UKUUG as well as member discounts on O'Reilly publications which still apply today.

He had a pivotal role in the planning and co-ordination of the UKUUG's most successful conference, the 1990 'UNIX: The Legend Evolves' event. Mick was Conference Director. This was where Bell Labs first announced Plan 9. There were presentations at the conference from almost all the leading figures in the UNIX world at that time: Dennis Ritchie, Ken Thompson, Rob Pike, Dave Presotto, Brian Kernighan, Kirk McKusick, Chris Torek, Tom Duff, Jon Bentley and Stuart Feldman.

Mick served with distinction as a Council member for 12 years, 6 of them as Chairman. Mick's warmth, good nature and pragmatism were greatly appreciated by his fellow Council members and by his peers at other national UNIX Users groups. He also helped to organise various meetings and workshops for UKUUG and EUUG, often chairing sessions and producing video recordings of the events. Mick's ability to see both sides of an argument and his common sense approach quickly resolved matters whenever the odd disagreement or personality clash popped up. His advice and experience was invaluable and particularly welcomed by new Council members.

Mick was a strong, enthusiastic supporter of UNIX and open source software for 35 years. He maintained a keen interest in IT after his retirement, doing some consulting projects and retaining his memberships of Usenix and BCS. Mick had just completed a home IT project to build and set up a media streaming system with a Linux-based NAS which held his CD and photo collection.

After retiring from Birkbeck College, Mick completed his PhD in numerical analysis, 'Computing the Zeros of Polynomials'. His graduation ceremony was to be held the day after his death.

Outside work, Mick's interests included holidays, fine dining, wine, photography, relaxing with friends/family and bird-watching. He particularly enjoyed advising and guiding Mel, his daughter, as she began her career in web design.

Mick leaves a daughter from his first marriage, Mel, and his wife Sue Small. Sue was also active in the UKUUG for many years and edited its newsletter, news@UK.

Unconference 2015

When: Saturday 7th February 2015

Where: London Hackspace, 447 Hackney Road, E2 9DY

This is our fifth unconference and has moved to the London Hackspace this year.

What is an unconference?

An unconference 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 a 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 the schedule to avoid clashes, etc.

The unconference starts...

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

Why attend?

There are lots of reasons to attend the FLOSS UK Unconference 2015, 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

---

OpenTech 2015

When: Saturday 13th June 2015

Where: Student Central (formerly ULU), London

Cost: £5 on the door donation

OpenTech will be 10 years old in 2015.

It will be back on June 13th 2015, for the usual mix of technology, experience and everything else.

For now, there's a date for your diary, and the call for talks will open in a couple of weeks. If there's something you'd like to hear about at OpenTech next year, we'd love to hear from you and we'll see what we can do.

OpenTech is only possible because of wonderful and generous sponsors in the past. If you or your company is interested in sponsoring please get in touch - `opentech@opentech.org.uk`

The event's predecessors were low cost, one-day conferences about technologies that anyone can have a go at, from 'Open Source' style ways of working to repurposing everyday electronics hardware.

`http://www.opentech.org.uk`

Dynamic Languages Conference 2015

When: Saturday 20th June 2015

Where: The Studio, Manchester

The Dynamic Languages Conference (DLC) is a cross-language event aimed at Open Source Dynamic Languages, we hope to bring together all the languages in an open forum to discuss and present the manner in which these languages approach and solve tasks at the cutting edge of development technology. It is a joint initiative organised by FLOSS UK and Shadowcat Systems. We hope to encourage a wide uptake from business, entrepreneurs, freelancers and interested individuals.

This year we will be focussing on Web Applications and System Administration in which we will discuss how the Dynamic Languages are utilised for these areas. We will accept talks or submissions on these subjects.

If you would like to sponsor the Dynamic Languages conference then please contact Jane who will be able to help and guide you further: `office@flossuk.org`

To submit a talk please login to the website: `http://www.dynamiclanguages.co.uk/`. The closing date for talks is Friday 3rd April and the Schedule will be released on Monday 27th April.

The location of the conference is in Manchester's trendy Northern Quarter, less than 1km from either central train station and close to the city's bus and tram links.

Creating Coding

Les Pounder

Programming is an exciting and interactive activity which rewards the user with a rich knowledge of ideas and logic. The issue with learning programming, especially for children, is that learners require something to capture their imagination. For example Python is the preferred language for children to learn coding but it is commonly used in classes to demonstrate coding concepts, such as data types loops and logic via a traditional approach where text is printed to the shell. But what if we could enable children to learn via creating their own projects? The Raspberry Pi is still the go to platform for schools around the world, and in the UK they have made an impact on the lesson plans of many teachers with lots of cross curricular activities centred on the Pi, such as time-lapse photography to record the life cycle of a plant in Science.

Let's take a look at a couple of cross curricular projects for the Raspberry Pi, see what they deliver to the class and their cost.

Project 1: Robotics

Robots are an excellent way to introduce key programming concepts such as:

- Sequencing - Enabling our robot to follow a sequence of commands.
- Looping - We can repeat the sequence either for a set number of iterations using a for loop or use an infinite loop via while True.
- Data types - Using integers to control the iterations of our for loop and using floats to control the duration that the motors are used for, enabling precise control.
- Debugging - If our code does not perform as expected, for example our robot does not stop moving forward, then we need to debug the code and fix the issue.
- Conditionals - We can add sensors to our robot for around £5 (HC-SR04 ultrasonic is a cheap solution) that can detect our distance from an object. If we are too close then stop and turn around.
- Data Storage - Using variables to contain data such as motor timings or sensor values we can teach the methods of handling data for re-use in our code.

Robotics projects can be as expensive as you like, but I really like the Ryanteck Budget Robotics Kit for the Raspberry Pi [1]. It packs all of the components that you will need into one affordable package retailing for around £35. The kit includes a motor control board that fits over the Raspberry Pi GPIO (General Purpose Input Output) pins. It has an input for an external battery pack (included) to power the motors and it has two outputs that enable control over two DC motors. The motor control board also comes with an H bridge which enables our robot to spin on the spot and go in reverse, not something that all motor control boards can do.

The kit comes with a battery pack for the motors but your Raspberry Pi will also need a portable power supply and these can be readily found on Amazon for around £10, I use an Anker 3200mAh phone charger [2] which provides around 2-3 hours of power for my Raspberry Pi.

Cost:

- Raspberry Pi: £30
- Ryanteck Robot kit: £35

- Anker charger: £14

Total: £79

Project 2: Music

Music is a wonderful way to teach coding in class. Music has its own structure and syntax but tackling this project from a programmatic point of view we can create compositions rather quickly. The best tool for this project is Sonic Pi [3], developed by Sam Aaron and supported by the Raspberry Pi Foundation. Sonic Pi is currently at version 2 and enables music to be created using the Ruby programming language, an exceptionally easy language to learn.

Creating music via Sonic Pi 2 uses the following programming concepts.

- Sequencing - Structuring our code so that the right note or chord is played at the correct time. If it sounds wrong…
- Debugging - Does a note play out of sequence? Does your music play too fast? You may have a bug - can you find it?
- Looping - We can repeat a loop five times using a 5.times structure. Anything inside of the do and done section is repeated five times. An infinite loop is handled in a similar manner to a Python while True structure.
- Concurrency - In music there are many elements that work together. For example there will be drums, bass and guitar tracks. In Sonic Pi we can layer these sounds using in_thread, which creates a separate thread in which another sequence of code can be run.
- Data Storage - We can play notes one by one, line by line. But by using the play_pattern structure we can create an array of notes that can be played one by one.

Cost: Raspberry Pi £30

Sonic Pi 2 comes pre-installed on the latest Raspbian image [4], released in September 2014.

Conclusion

Teaching programming is a great skill and using interesting and inventive projects such as those in this article we can help children to enjoy learning these skills that will last a lifetime.

[1] http://store.ryanteck.uk/collections/robots/products/ryanteck-budget-robotics-kit-for-raspberry-pi?variant=742663987

[2] http://www.amazon.co.uk/dp/B005QI1A8C/

[3] http://sonic-pi.net/

[4] http://www.raspberrypi.org/downloads/

The Book of PF, 3rd Edition

Peter N.M. Hansteen
No Starch Press
ISBN: 978-1-59327-589-1
248pp.
£ £22.50
Published: October 2014

reviewed by Andy Thomas

In the short space of just three years since I reviewed the second edition of this book in the September 2011 UKUUG newsletter, much has happened in the BSD world and enough new features have found their way into Packet Filter (PF) to warrant a third edition of The Book of PF. Rather than regurgitating the previous edition's review here, I will concentrate mainly on the new material that has been included in the latest book covering the evolutionary changes that been made to PF in the meantime; readers interested in this book are encouraged to read the earlier review as well since most of this still holds good for the 3rd edition.

With a cover almost identical to the second edition (but with green bands replacing the red ones on the older version), the layout is essentially the same. The first 6 chapters cover in turn the same topics as before - the history of PF, basic PF configuration, implementing a 'real world' gateway, wireless networks, larger networks including NAT, DMZ and Ethernet bridging, and dealing with deliberate network nuisances such as attacks and spam.

But with OpenBSD 5.5 came a major change to PF - in came the new Priorities and Queues scheme, bringing in its wake a whole slew of new traffic shaping tools, and out went the aging ALTQ system of the earlier BSD releases. So the old chapter 7 covering queues, traffic shaping and redundancy has been rewritten to cover first the new priorities and queues scheme and then the legacy ALTQ system while redundancy topics have been moved to an entirely new chapter 8 entitled Redundancy and Resource Availability.

This change is very welcome because CARP (Common Address Redundancy Protocol) and pfsync are now covered much greater detail than ever before; firewall and router redundancy is increasingly becoming a 'must have' essential rather than a luxury optional feature but in my experience, I have found CARP is not always easy to set up or get working so this expanded coverage should go some considerable way to making a CARP implementation a resounding success rather than the embarrassing failure it sometimes is (on-site at a client's, sheepishly looking at the floor with red faces all round after having spent a whole day there trying to get CARP to actually function as expected - I'm sure many readers will find this kind of scenario rather familiar!). The remaining two chapters cover monitoring & logging and tuning as before.

Although the coverage of the new priorities and queues scheme and the spinning off of redundancy and high availability issues into a completely new, dedicated chapter are the most obvious changes for the third edition, the book has been overhauled, enhanced and tweaked throughout, with lots of minor changes and better clarification of certain points and concepts, gaining over 30 more pages in the process. But the memorable phrases of the earlier edition are still there - an Ethernet bridge defined as 'an intelligent bulge on the network cable' and 'that sad old FTP thing' are just two that still stick in my mind.

Written in a warm and conversational style, even an absolute firewall/router beginner will soon feel at home with this book which, put simply, is a good read. It is not a cookbook of PF rulesets nor is it crammed with ready-to-deploy examples since the author wisely points out that understanding PF is the key to getting the most from it, not blindly cutting and pasting other people's (possibly inappropriate, out of date or plain broken) rulesets into your own /etc/pf.conf, crossing your fingers and hoping for the best. Rather, it is a book written by an experienced lecturer and trainer who has prepared and given courses on PF for UKUUG and FLOSS UK, among others.

Naturally the emphasis is on PF as implemented in OpenBSD, because that is where all the PF development is done, but users of other BSDs such as FreeBSD, NetBSD and Dragonfly BSD are not left out in the cold; this latest edition now covers Packet Filter in FreeBSD 10.x and NetBSD 6.x, as well as OpenBSD 5.6. Linux users are given some helpful pointers too although recent Linux distributions are increasingly abandoning the ubiquitous 'eth0' style network devices for more BSD-like devices names such as bge0, em0 and so on so this will not be the shock it once was.

With the same two appendices containing useful pointers to additional PF resources and a brief discussion of finding suitable hardware for basing a OpenBSD PF system on, the book concludes with a comprehensive index. As is the norm for books from No Starch Press, the book is well-packed with text as well as some diagrams without the acres of wasted white space books from some other publishers have.

If you need to do firewalling, routing, network traffic control, NAT, wireless networking with an intuitive configuration syntax at zero software cost, choose PF. And buy this book.

Seven Concurrency Models in Seven Weeks

Paul Butcher
Pragmatic Bookshelf
ISBN: 978-1-937785-65-9
300pp.
£ 24.50
Published: June 2014

reviewed by Paul Waring

With clocks speeds stalled for several years, the main way of making faster processors today is to add more cores - even smartphones now come with dual or quad core processors. In order to fully utilise this extra power, software developers need to write programs in such a way as to take advantage of concurrency and parallelism, and this book examines seven models for doing so.

The first chapter outlines the 'multicore crisis', there are no more 'free' speed increases for software without a fundamental change in how software is developed. An explanation of the differences between concurrency and parallelism is given, as these are two different concepts, despite often being used interchangeably. Some types of parallelism are introduced here, such as bit-level and data parallelism.

The first model to be examined is threads and locks. Despite being a primitive mechanism for concurrency, they are still the default choice for many developers. The author points out that threads and locks are effectively a formalisation of how the underlying hardware achieves concurrency, and so they can be seen as a relatively low level of abstraction. Basic examples are given in Java to illustrate key problems such as race conditions (where behaviour depends on relative timing) and the re-ordering of instructions by the compiler, JVM and the hardware. Multiple locks and deadlocks are explained with a simple scenario of multiple diners attempting to eat when their chopsticks are shared.

The second model is functional programming, which models computation as the evaluation of expressions. The immutable state makes functional programming ideal for concurrency, since multiple threads can access the same data without requiring any locking mechanisms. Clojure is used to illustrate all the examples, but unfortunately the author's 'whirlwind tour' was so brief that readers may struggle to follow the code samples. I certainly found it hard work, but at the same time I was pleased to see how to take an imperative function to sum a list of numbers and convert it into a functional equivalent which was simpler, clearer and easy to parallelise.

Chapter 4 goes into more detail with Clojure, which as an impure functional language can have mutable variables. The majority of the examples take the form of a multithreaded web service with mutable state - a fairly common use case. As with the previous chapter, those will little experience of functional programming or Clojure may struggle to absorb the information from this chapter during the first reading.

Chapter 5 introduces us to Actors, a general purpose model for concurrent programming. In contrast to functional programming, which avoids mutable state, the use of Actors allows state to be mutable but avoids the multiple-write problem by not sharing state. Every concept is clearly illustrated by one or more code samples, although another programming language (Elixir) is introduced for this.

Chapter 6 covers communication between sequential processes, which initially sounds like the same thing as Actors but turns out to have a different focus. By this point I was struggling to keep all the previous concepts in my head, but the author's judicious use of code snippets and diagrams kept me going. This was followed up by a chapter on data parallelism, which again introduces another language (OpenCL) and made me wish I'd paid more attention in the computer engineering unit of my degree.

The penultimate chapter covers the Lambda Architecture, which I think I can sum up as 'MapReduce is awesome and I must learn more about it'. Finally, the author wraps up with some thoughts on the future direction of computing, all of which naturally involve concurrency and parallelism - just in case you weren't completely convinced by this point.

Overall, this is a thorough overview of the various methods of achieving concurrency and parallelism in software. This isn't a book which you can read once and understand everything - patience and persistence will be required - but that is more down to the subject matter (writing concurrent software is a big change for most programmers) rather than any fault on the author's part. The only critique I would make is that the use of different and relatively new programming languages to illustrate key concepts makes it harder to follow the examples, but there is always going to be a tradeoff between using the same language throughout and using the language in which a given model can be most clearly expressed (unless this is a cunning plan to drive up sales for one of the Seven (More) Languages in Seven Weeks titles, which happen to cover Elixir and Clojure!).

Understanding and Using C Pointers

Richard M. Reese
O'Reilly Media
ISBN: 978-1-4493-4418-4
226pp.
£ 25.99
Published: May 2013

reviewed by Paul Waring

Pointers: the cause of much frustration to anyone who programs in C, particularly beginners. Useful in the right hands and potentially destructive if used incorrectly, there is definitely a market for a book dedicated solely to this subject, and the only surprising thing about it is that it has taken so long for someone to notice and satisfy this need.

Chapter 1 presents an overall introduction to pointers and their interactions with memory. Most of this will be useful only to beginners, but even experienced programmers may find topics such as 'how to read a declaration' (e.g. const int *pci) a useful refresher. The six types of 'null' are also covered, as are the basics of pointer operators and arithmetic.

Chapter 2 covers dynamic memory management, which is probably the most useful aspect of pointers and also the cause of so many bugs - especially leaks from memory which is allocated but never freed. Common problems with memory management, such as dangling pointers, are discussed in detail, along with techniques for avoiding them.

Chapter 3 covers the use of pointers with functions, including the passing of parameters and declaring pointers to functions. Useful syntax, such as passing a large object by reference but prohibiting editing (similar to copying but without the overhead) is included, as well as the dangers of returning a pointer to local values. This is followed by a chapter on pointers and arrays, which carefully outlines the similarities and differences - a common misconception being that the two concepts are completely interchangeable.

Chapter 5 brings us into the world of strings and how they relate to pointers. All the standard string operations are covered, such as comparison and copying, as well as warnings about unsafe operations. The next chapter covers perhaps the most common use of pointers, building structures such as linked lists, including the process of ensuring that memory is deallocated correctly for nested or recursive structures.

The penultimate chapter of the book covers my favourite topic: security (and improper use of pointers, which often leads to security bugs). Most of the security issues raised are due to sloppy programming errors, such as failing to initialise pointers, many of which can be caught by compiler options and static analysis tools, and indeed the author recommends using both. The final chapter deals with 'odds and ends', pulling together lots of little hints and warnings which don't easily fit into other chapter headings.

Overall, this book covers pointers in great detail, and is a useful guide to all aspects of how they can be used. If you write C on a regular basis, you should definitely read it through at least once, as you're sure to learn something new.

Becoming Functional

Joshua Backfield
O'Reilly Media
ISBN: 978-1-449-36817-3
134pp.
£ £19.50
Published: July 2014

reviewed by Fred Youhanaie

The subtitle of the book is 'steps for transforming into a functional programmer', however, from the start one realises that the object of this transformation is 'a Java programmer'.

The entire book is based on an example which is the legacy Java code of a fictional company named XXY, and the reader is gradually introduced to the elements of functional programming by way of transforming the Java code into the Scala and/or Groovy equivalents.

Chapter 1 gives an overview of functional programming in terms of the seven concepts: First-Class Functions; Pure Functions; Recursion; Immutable Variables; Strict and Nonstrict Evaluation; Statements; Pattern Matching. These concepts are then described, using Java examples, in the seven chapters that follow the overview.

In chapters 2 and 3, first-class and pure functions are introduced by means of transforming Java examples into more functional looking variants. In chapter 2 we learn about lambdas and closures, while chapter 3 demonstrates how to avoid side effects in order to create Pure Functions. Here, we encounter some Groovy (the language) examples.

Chapter 4 demonstrates how immutable variables can be implemented using a series of transformations of the standard example Java codes. This is followed by a chapter on recursion. Tail recursion, an important concept in functional programming, is also covered here. This is the chapter where we start learning about Scala.

Chapter 6 is about strict and non-strict (lazy) evaluation. The concepts are explained using examples in Groovy and Scala. The advantages and disadvantages of each of the methods is also covered.

In chapters 7 (Statements) and 8 (Pattern Matching) we are treated to more Scala code. Chapter 7 shows how one can obtain a more concise code since every statement in a functional language returns some value. This is demonstrated by means of transforming the example Java code to its Scala equivalent. This leads nicely to the chapter on pattern matching, which contains plenty of example codes in Scala (and no Java!)

In chapter 9, Functional OOP, we are treated to more examples in Scala and shown how we can got about taking advantage of both worlds.

The book is short and concise. The text is well written and the structure makes it easy to get the gist of the contents. Most of the chapters have a conclusion section that sums up what has been covered in the chapter, including the 'Conclusion' chapter, but the recursion stops there!

If you are a Java programmer and have not had any exposure to functional programming, then you can benefit from this book. On the other hand, if you have had some exposure to functional programming, but programming in Java is not second nature to you, then you may find the Java examples rather distracting.

Functional Thinking

Neal Ford
O'Reilly Media
ISBN: 978-1-449-36551-6
164pp.
£ 25.99
Published: July 2014

reviewed by Fred Youhanaie

The preface starts with a history of the book. The author tells us how his learning process culminated into a series of talks and articles, which now form the basis of the chapters of the book. Ford also tells us that his '. . . goal in the talk, the article series, and this book is to present the core ideas of functional programming in a way that is accessible to developers steeped in imperative, object-oriented languages.'

Throughout the book we are treated to many examples of functional code snippets, written in the three JVM (Java Virtual Machine) based languages, Clojure, Groovy and Scala.

Chapter 1, Why, gently sets the scene for the functional programming concepts that are to follow. It introduces the virtues of functional programming by showing some long Java code snippets for solving simple problems, these will suit the developers steeped in imperative, object-oriented languages.

Chapter 2, Shift, shows us how to shift our thinking from imperative programming to functional programming. Ford demonstrates this by means of a simple case study, number classification, deciding if a given integer is a perfect number not. We are first introduced to a solution in Java (not Java 8), followed by a much shorter one written in Java 8. We are then introduced to a few common building blocks that are 'ubiquitous' in functional programming. These 'Useful things' are filter, map and fold/reduce, and they are demonstrated using numerous examples in Clojure, Groovy and Scala.

In chapter 3, Cede, we are shown how to cede control of the low level details (such as the author's least favourite, garbage-collection) to the underlying run-time system. We are introduced to five ways of ceding control so that we can spend more time thinking about the main problems. These methods are higher order functions; closures; currying and partial applications; recursion and streams. For each of these we are treated to a number of concise examples using the familiar trio of languages.

As we move through the chapters, Ford delves deeper into the functional programming paradigm. In chapter 4, Smarter, Not Harder, we learn about memorization (caching) and lazy evaluation. As usual, there are plenty of examples to help us work smarter.

Chapter 5, Evolve, demonstrates how one can bend the language so that it fits the problem being solved, rather than the converse. The main topics covered are operator overloading and functional data structure. The latter includes pattern matching and Either/Options classes.

The main theme of chapter 6, Advance, is design patterns. Here we learn more advanced techniques when programming in functional languages.

In chapter 7, Practical Thinking, Ford covers Java 8, the latest version of Java that understands concepts of functional programming, as well as some real world examples such as web frameworks and databases, although not in detail.

The book is concluded with chapter 8, Polyglot and Polyparadigm, where the author discusses functional programming within the wider world of languages and programming paradigms.

Overall the book is well written and worth a read for anyone interested in learning the underlying concepts of functional programming. The presence of Java examples does suggest that the book is aimed at Java programmers, however, the non-Java developer should still be able to understand the main message of the book - 'Paradigm over Syntax'. Additionally, one also gets a fair exposure to the trio of the JVM functional languages used in the text.

Learning MCollective

Jo Rhett
O'Reilly Media
ISBN: 978-1-491-94567-4
264pp.
£ 25.99
Published: August 2014

reviewed by Fred Youhanaie

MCollective (M for Marionette, as in puppet!) is a Ruby based software framework from Puppet Labs for orchestrating configuration changes across Puppet or Chef based groups of servers.

MCollective communicates with the servers under its control through a message broker, currently only ActiveMQ and RabbitMQ are supported. The main text along with the examples only covers ActiveMQ, however, those who prefer RabbitMQ are provided with guidance in the appendix.

The book is divided into four parts that take the reader from initial installation of the base software to writing custom plugins.

Part I, Getting Started, takes the reader through the basic installation and configuration of the software, MCollective and ActiveMQ. It then proceeds to introduce, with examples, the main command line interface, mco. Although, there are web clients for the software, we are urged to stay with the command line.

Moving along, we are introduced to the basic workings of MCollective plugins and agents, as well as brief tips on system level maintenance, such as time sync, log files, monitoring etc. We are then shown how to configure MCollective with Puppet and Chef, yes it does sound like a chicken and egg situation, but the modules are there for us to use.

Those with a large enterprise installation are catered for in part II, Complex Installations. The message broker plays a very critical role in an MCollective based environment. In the next five chapters the reader is given guidance on how to create broker networks to aid scalability and resilience, as well as using certificates to make the message broker(s) more secure. Admittedly, I only skimmed over this part, but I am planning on a more through read at a later date.

As one can expect with any framework these days, MCollective comes equipped with the ability to create and use plugins. Part III, Custom Plugins, provides various examples of plugins and walks the reader through creating and deploying them.

Finally part IV, Putting It All Together, sums up the preceding chapters, provides a set of best practices and gives guidance on expanding the MCollective deployment.

I had a play with the command line examples on a Vagrant/VirtualBox based demo environment that is provided by Puppet Labs, all good fun, and no surprises. The example codes and related data files used in the book have been made available on Rhett's github repository.

Throughout the book, there are numerous hyperlinks for additional material, however, they are all Bit.ly shortcuts, which means the reader of the paper copy of the book will need to type in the links manually. It would be nice if the author could include these on a web page, or somewhere on the book's GitHub repository.

Overall, this book gives a covers a lot of ground, going beyond a mere introduction to the basics of the framework. Although, its main focus is on Puppet and ActiveMQ, those who prefer Chef and/or RabbitMQ can still benefit from the text.

Contributors

Kimball Johnson has been programming since a very early age, starting with BBC Micros, then MS DOS and Windows Systems, however was enlightened with a copy of Debian GNU/Linux Woody at university. From this developed an affinity to Systems Administration, but like all good admins he is lazy and so tries to automate as much of his job as possible, and now follows the way of the Infrastructure as Code.

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.

Andy Thomas is a UNIX/Linux systems administrator working for Dijit New Media, Imperial College London and a number of small companies. Having started with Linux when it first appeared in the early 1990?s, he now enjoys working with a variety of UNIX and Linux distributions and has a particular interest in high availability systems and parallel compute clusters.

Paul Waring currently works in teaching and software support at a large UK university. 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.

Fred Youhanaie is a freelance hands-on technical consultant specialising in Unix/Linux infrastructure systems. Over the last 30 years he has engaged in programming as well as systems administration and engineering projects. More recently he enjoys dabbling in Erlang based projects.

Contacts

Kimball Johnson
Chairman
Preston

Gavin Atkinson
Council member
York

Tim Fletcher
Council member
Manchester

Ian Norton
Council member
Morecambe

Stephen Quinney
Council member
Edinburgh

Quentin Wright
Treasurer
Warwick

Paul Waring
Newsletter Editor
Manchester

Alain Williams
System Administrator
Watford

Holger Kraus
Website
Leicester

Sam Smith
Events and Website
Cambridge

Jane Morrison
FLOSS UK Secretariat
The Manor House
Buntingford
Herts
SG9 9AB
Tel: 01763 273475
Fax: 01763 273255
`office@flossuk.org`