Dr Damian Conway's PERLS of WISDOM

We recently caught up with DAMIAN CONWAY and asked him about life, the universe and Perl 6. Well, mostly Perl 6...

Tell us a little bit about what kind of things you were working on before you became a full member of the Perl developer community...

DAMIAN CONWAY: Before Perl I was an academic at an Australian university – in fact, at the largest Australian university. I'd been teaching since my undergraduate days. I had been brought in fairly early to do classes and things, which started around 1986. I quickly became a lecturer, which is kind of the first level of professional academics, and I was working my way up the academic food chain doing the same kinds of unusual research, the same thinking about the interface of programming languages, but I was doing it in C++ because I wasn't aware of Perl. That was stunningly difficult to do.

LXF: **How long ago was that?**
DC: We're talking 1989 and after.

LXF: **So this was before Perl was really out there in the wild?**
DC: It wasn't big and it wasn't on anyone's radar really. At that time, C++ was just taking off and I was involved very peripherally in the ANSI standards process for that, and quickly became very dispirited by the difficulties of the standardisation process. I was basically just a working academic: I loved to teach and I was pretty good at it. Ever since I knew how to program, I wanted to customise and recustomise my own environment: my shell environment, my window environment, the whole works.

I had literally dozens of little shell scripts that were doing the most arcane things imaginable, and it was just getting to be too difficult to do. Then someone said to me, "Why don't you have a look at Perl? It's kind of like a souped-up shell." So I had a look at it, and I kind of fell in love with it, and I realised that it would actually work pretty well for my linguistic work, and it would work well for the other kinds of research that I was doing.

LXF: **What kind of research was it that you were working on?**
DC: Well, I've always been a dabbler in things. Most academics will get themselves in some area and stay there for five or ten years and evolve over that time, and get a long publication record in it. In the ten or 12 years I was an academic, I never did anything for more than six months. I've published in dozens of different areas and different types of areas of computer science, everything from the psychophysics of perception to the user interface design, to bio-informatics. There were bits everywhere, and I guess, at the time, I was kind of the academic that got sent the problem postgraduate students, the postgrads that didn't quite fit into anyone's research group. When we normalised our research groups at the university a few years ago, we basically clumped everyone together into six groups, and five out of those six put me in their group, which kind of indicates how it was.

I think that kind of thing is important, because in a group like that, where you have very clustered and cliqued research areas, what you really need to make it work properly are those long connections. A lot of the mathematical research indicates that the way you dramatically cut down the cliquing problem of, you know, how many steps is it to Kevin Bacon, is you have a few people that are reconnected in very wide, very disparate kinds of groups. I guess I filled that role at the university.

Then I went to my first Perl conference, which was in fact the second Perl conference at San Jose, back in 2000 maybe. Anyway, I went to this conference and I gave two papers. One was on the inflection stuff (see www.csse.monash.edu.au/~damian/papers/extabs/Plurals.html for more information), and the other one was on a module that enabled you to do command line argument processing much more easily. It was basically about writing yourself a usage statement and it would automate it, so it was the same trick I used for smart comments (see http://search.cpan.org/~autrijus/Smart-Comments-0.01/lib/Smart/Comments.pm).

LXF: **That's a pretty dramatic entry into the community!**
DC: I think it was kind of indicative of the ways in which my brain goes all over the map, and the two of them together won the inaugural Larry Wall award. Apparently people were arguing over which of the two they should give it to, and then someone pointed out they were both by the same person, which simplified things a bit!

From that point on, my movement into the Perl community was inevitable because I found a group of people that were interested in the same kind of issues that I was interested in, and in terms of programming, who were receptive to my way of thinking about the world and enjoyed what I had to say and wanted to hear me speak. It soon became apparent that they would actually pay me money to have me teach them stuff.

After a decade of teaching undergraduates how to program from scratch in C, you can teach pretty much anything in programming, so that was basically my entry into Perl.

LXF: **Was it quite hard for you to jump from being an academic into working with Perl full-time?**
DC: Well, it wasn't so much hard because it was very gradual. For three or four years I was doing both – I was a full-time academic, and then in my four or so weeks of conference leave, I would go to conferences and I would teach people. My university was very

supportive of its academics also engaging with industry, and it didn't have to be industry in Australia – it could be industry anywhere. They said they would give me one day a week to do this kind of thing. Given what they pay academics, it's a good way of keeping people that could go out and get five times the salary, and certainly in the mid- to late-90s, I could easily have gone out and got five times the salary if I wanted to. This way they were keeping their staff there and keeping their abilities, but also engaging them with industry and with the commercial world so they could bring back relevant understanding.

The really nice thing they said was, "Yes, we'll give you a day a week to do this and yes, we'll let you take it in big chunks, rather than actually taking one individual day a week." There was a point where I said I needed three weeks and they said that was fine. I then told them it had to be in the middle of the semester, but we worked it all out. I was team teaching with another person and we just chunked it so that I wasn't on deck for those three weeks. They were incredibly supportive of that.

***LXF:* And then you got the grant from The Perl Foundation (TPF)...**
DC: What that actually did was it channelled through TPF on this side so all the donors got their tax relief for doing it, then TPF channelled it to the university to basically buy out my contract for those 12 months. The exchange rates at that time were such that just on the donations of a few big companies, but mainly the individuals in the Perl user community, they were able to raise enough money that would only have been a very moderate kind of wage for a programmer, but it was enough to cover the costs of an Australian academic for 12 months.

So that's what I did. I still had my office in the university, so I had access to all that support structure, which was a very generous donation on their behalf because they weren't actually getting paid for that. I also had a travel budget, I had my income, which was just my normal salary for that period of time, and I was able to travel around the world. In the end it was 20 months that TPF covered me for, and I was travelling for six of those months. I was literally on the road for six of those months, and I guess I visited 50 or so venues where I gave talks to people about Perl, and at the same time I wrote maybe 15 or 20 new modules, four or five of which ended up in the core distribution, and many others are widely used and very popular. These include things like Parse::RecDescent (see **http://search.cpan.org/dist/Parse-RecDescent/lib/Parse/RecDescent.pod**), which I developed quite a lot during that period of time.

That gave me the opportunity to see what it would be like to be living and working for the Perl community, and not actually for academic stuff. I found that irresistible, because wherever I would go, people were keen to have me there, either having paid or having heard about what I do, and they were always very receptive to what I was talking about. They understood a great deal more than my undergraduates. I love teaching undergraduates, there's no question about that – I really enjoyed helping them discover the world that I love.

However, there's also tremendous pleasure in dealing with people who are experienced and knowledgeable, who are very bright, and seeing something light up in their faces.

I just saw that this morning: people saying, "Wow, that is cool." And even more than that, saying, "Wow, I could actually use that – that would make my life better".

***LXF:* What effect did those 20 months of working for the TPF have on you?**
DC: Well, after 20 months, when we started to go into the downturn of the economy and TPF couldn't afford to support a full-time servant, at that point I had to make a decision. Was I going to go back and be a full-time academic again, or was I going to chance my arm and see if I could create a business for myself?

I had made some very good business contacts because TPF had some very large corporations that were very generous in their donations to support my work and Larry's work [Larry Wall] and other people's work, and so I had those contacts now.

I took them and said, "Look, I'm effectively unemployed and I'm looking for some training work. You've heard me speak – I can do that for a full day for five straight days, and I can actually teach you serious stuff." And so, very gradually, I built up a clientele of people who were willing to give it a go, and things have pretty much gone on from there.

Things have gone up and down a great deal. The last couple of years have been very, very hard, because with the downturn, the two things that people cut are buying books and taking training. Oh, three things: going to conferences, which is basically where I make all my money. It's been tough over the last few years, and I've had various small grants from TPF that have certainly tided us over and enabled me to keep working on Perl 6. However, it looks like it might be turning around this year. A lot of my friends who are also trainers are saying the same thing.

I've actually been able to do a bit more this year, and it looks like we're actually going to be able to eat for another year.

***LXF:* How does working as a Perl trainer suit you? You must spend a lot of time travelling.**
DC: For me, of course, it's much more difficult than for many of the other Perl trainers because most of them are based in the US or in Europe and can basically go somewhere at a week's notice, like fly across the States. For starters I'm not a US citizen so I'm not allowed to work in the US. In addition to that, I live 14 hours from LA, I live 23-24 hours from the east coast, and I live 24 hours from Europe. I really have to aggregate all my work together and do one big trip.

That's generally what I tend to do. Every year I tend to go to the Yet Another Perl Conference (YAPC) at the start of summer, OSCon towards the end of summer, and in between I go wherever there's employment to be had, and I deliver training for the company I work for in Australia.

***LXF:* For those conferences, how much of it is teaching new things and setting really high goals for people to reach, and how much is evangelising Perl 6?**
DC: It's actually pretty easy to work that out if we look at this conference (OSCon). At this conference, out of the four-and-a-half days, I spent one-and-a-half days doing nothing but teaching existing Perl techniques. Now this was fairly advanced stuff, but still stuff I would be teaching out in industry as well. So, let's say that a third of my work here is doing that kind of thing. That probably generalises pretty well to my life, in terms of the total amount of time I have to spend preparing materials, because that takes a long time, and delivering materials to employers around the world. I probably spend about a third of my time doing that.

Of the setting the bar, of inspiring people to go beyond what they're doing right now, I like to think that everything I do is aimed at that. I'm always thinking how I can engender in these people a sense of excitement, a sense of wonder; the sense they had when they first discovered programming. Can I take them back to that feeling of energy? Wanting to know more, wanting to do more? Can I send them away from this conference wanting to do nothing more than get back in front of their keyboard and do something new?

Whether I'm doing straight training or talking about Perl 6, or talking about this kind of out-there, edge stuff – you know, pushing envelope things – always in the back of my mind is that people need to be inspired. Because for most people, the work they're doing, even if they are working in Perl and love what they are doing, is a grind. And it's a grind every day, and it's always pressure, and it's always, "We don't have time to do this properly, we have to do it quickly." It becomes a rut, because you're doing the same sorts of things all the time, and it's the same business area. Even if you move, you're frantically trying to get up to speed on something.

"CAN I GIVE THESE PEOPLE A SENSE OF EXCITEMENT, A SENSE OF WONDER?"

»

PERLS of WISDOM

LXF: **So you're showing them that there's more to programming than pressure and deadlines?**
DC: What they need to see is that there's a level beyond that – there's a place that you can put yourself where the sheer joy you experience in coding comes back into your life. That you can do it as a professional, but you can do it as a professional at a very high level, and you can shake yourself out of that mental rut that you feel. Like you have to be in to get the job done, but you don't have to be in that rut, at least not all the time. Yeah, you've got to slog and you've got to cut code, and then there's the hard work and the design, but you also have to make time, space and calm so you can play again.

The only way you can be innovative, the only way you can be creative, the only way you can be imaginative is if you have at least a little bit of play in your life. And that's what I do: I get up on the stage and show them things I've been playing with. I show them me playing. I try to do it in a way that engages them and entertains them, and makes them go "wow!". When they do that, it inspires them to do it themselves. They say, "Wow, I didn't realise that coding could be this much more than what I'm doing, but I can see how I could use this. I can take this and go beyond it."

I get so many patches from people who say, "I love this thing that you did, but I just thought if it would only do *this* as well." And I just think to myself, "Man, I should have thought of that! That's just so clever – thank you for giving that back to me!" Sometimes I get emails back from people saying, "I went away and took on board the things that you said and thought about it, and had these really weird dreams after I went on your course. The next day I got up and refactored this piece of code and I tried this thing that you said, and it saved up to 20% on performance." And I just think yeah, that's what I'm here for.

LXF: **So would you say it's almost your reason for being?**
DC: Sure. I would do this stuff if I wasn't paid. Don't tell anyone that… well, too late! But for me, it doesn't feel like I'm working because when I'm up on there on that stage, I'm just enjoying myself – I'm having a great time. I'm being fed as well as feeding.

When I'm at home and I'm not frantically getting ready for my actual business, I'm working on things like Perl 6. I get to work with Larry Wall every day, and that's an extraordinary opportunity. I've learnt so much from him, and it's been so much fun working with people at that level. And not just Larry – the rest of the design team are incredibly smart people too.

I know a lot of people get up in the morning and think, "Oh, damn, it's Tuesday – I've got to go to work for another four days." You get in the car and you commute. I work from home most of the time – I have an eight-second commute down the hallway. And you know what? Those eight seconds seem too long to me because I can't wait to get in there and do stuff. That's what I want everyone to have: I want everyone to find something that works for them, that does that. Part of that inspiration is to say, "Look, there are things out there that you could be doing that would engage more levels of you, therefore enriching you, your employer and ultimately, if you can feed it back, the whole community".

So, I have a whole lot of respect for everyone in the community that can do that, and there are so many people. Just the number of modules on CPAN – every one of those has been made by someone excited by and fanatical about something, and wanting to make it better, and then wanting to share it with everyone. Giving of themselves. When I see the whole community doing that, and when I see the whole community donating to TPF so that Larry can do what he does, so that I have been able to work for the community… when I see that kind of generosity, I want to do it too – I want to give back. I'm very lucky to be in a position where I can do that.

LXF: **Guido van Rossum has recently been giving a talk where one of the opening points is that he's not really involved much in Python any more because he's doing other things. However, people always have this perception that he *is* Python. Equally, a lot of people must think that you and Larry get paid lots of money for all the work you do. How do you combat that?**
DC: Well, you saw some of that today. Me not doing it for me, but me doing it for Larry. I'm always astonished by the number of people who don't know that Larry has been out of work for coming on to three years now. The assumption is that he's still being paid by O'Reilly, and if you sat down and thought about it, it would seem nonsensical to even think that because we've all been through a recession, so many of us have been thrown out of work or made to work longer hours, and seen cutbacks in conferences and in the ability to buy textbooks.

Those are the two things that O'Reilly does, so O'Reilly – not because it wanted to, and I know that it hurt Tim every single time it had to let anyone go – had to let people go. Like every company has had to do: to downsize. And Larry was one they had to let go, because when you do that you have to keep the people who generate the money in the company, and as rich as Larry's contribution was, he wasn't bringing money into O'Reilly, except for the few books that he was writing, and he was getting paid for that in another channel.

LXF: **And his book sales must have been substantial…**
DC: When you talk to Larry, he points out that O'Reilly basically gave him half a million dollars over a number of years, and he doesn't feel at all bad because he was actually telling Tim, "Come on, you've got to let me go. You've got to say you're cutting the line, and I'm one of the ones you've got to let go off the life raft." However, people don't realise that. They don't think about that. They think that Larry is okay, and I guess they think the same thing about me.

I was looking at one of the regular Python or Ruby newsgroups just the other week, and someone wrote a message that said we should get together and set up a foundation, and we should support developers like TPF supports Damian Conway. There was a big part of me that wanted to butt into that conversation and say, "Well, that's not actually happening – that hasn't happened for two years now." You get the fanfare when something happens,

"THE ONLY WAY YOU CAN BE IMAGINATIVE IS IF YOU HAVE AT LEAST A LITTLE BIT OF PLAY IN YOUR LIFE"

but when something stops happening, of course, there's no fanfare because it doesn't make sense for there to be a fanfare when it's not good news.

I was tremendously generously supported by them, and I have lots of friends that don't have the kind of money they had two years ago. *I* don't have the kind of money I had two years ago, so I understand that. How do we combat that impression? I'm not sure I want to, personally at least. I definitely want to do it for Larry, because without Larry you can forget it – we don't have Perl any more.

You know, the community swirls around this man. He gives us so much on so many levels and not just on the technical level and the brilliant job he's doing on Perl 6 development. The kind of the man he is, the way that he keeps the community together, the way he defuses situations. He's a focal point for our community and something to aspire to, a genuine hero. And not just in our community, as we saw when he won the Nobel Prize for Open Source.

Everyone feels that way about the man. He's an extraordinary man, and yet here is, struggling not just with health problems and the incredible costs involved in that, but also with the fact that he has a mortgage, he has four kids, two of which are in college and two of which are about to go into college. He hasn't had a full-time job for that time. For myself, I don't care. I have been blessed with enough gifts and with enough brains that I can make a living. If necessary, I can just say I have to leave the Perl community and get an academic position again. I'm pretty confident I could probably land a position. Or I could go and cut code for somebody, and I could probably find somebody who would pay me to do that.

LXF: So you're doing this by your own choice...
DC: At the moment I'm able to do it. We don't live like kings, but we live. I'm able to come to these conferences by the generous support of O'Reilly – they're covering my costs to come here. I'm able to do my work, and my real work is working for the Perl community, so I'm able to do that and we get by.

So, I don't try to combat the idea, because at some level, whatever community it was that was saying we should do the same thing that Perl does for Damian, well, if I'd said that Perl doesn't actually do that, then that just kills them. That stops their enthusiasm for it, and I don't want to do that. I want *them* to be doing the same sort of thing. So, I guess what I do is that I go around and when people ask why Perl 6 is taking so long, I list the people who are working on Perl 6, and when I do that list it's "X, unemployed; Y, unemployed; Z, part-time employed cutting code for blah." And we have a lot of people who, at some stage during the development process, haven't had any work at all, and have just worked on that instead of finding a job.

Each of us has gone in and out of jobs, and we don't expect special treatment, we really don't. We know how tough the whole community is, but people don't realise that. So when I say X, Y, and Z don't have work, they're shocked. It doesn't really seem to make a huge amount of difference in terms of what happens, but at least people are aware of the situation.

I think that making people aware of this, and making them aware of Larry's situation, of the fact that TPF was run, until recently, by just one person slogging away out of working hours, we see others volunteering.

All the community have their own lives to lead, their own families to take care of. They have their jobs that are no longer nearly as secure as they used to be. They have their social lives. Perl may be a large part of their lives, but it's not their life. And so for those of us where Perl is a large part of our life – apart from my wife who is, okay, all of my life, so apart from her, then the Perl community is it for me. It's where all my friends are, it's where all my income comes from, and, quite separately, it's where all my work is – where my true vocational work is. So for me, it's a privilege to do what I do, and I'm just thrilled I can do it.

LXF: If you had more monetary resources at your disposal, how would you spend them on Perl?
DC: If I had my way, if I had unlimited amounts of money, then I would be paying the rest of the team to just not have to worry about having a day job building websites, not have to worry about looking for a job – that takes more time than actually having a job, and just say, "Look, work for us – do what you want to do. Here's a grant for each of you to do what you want to do."

I don't have that, so I do what little I can – to go around, talk to everybody that I can possibly talk to, anywhere I go in the world, if I'm there for more than one night, I will offer to come out and talk to the local Perl people, and I've done that ever since I was invited anywhere. That's part of my job. It's really part of my job as a member of the Perl community – and as a senior member of the Perl community, a member that's well-known – is to say, "Let me bring part of the Perl community into your part of the world, let me connect you with that for a period of time. Let me bring some of the wonder of Perl back into the routine of your daily work".

I'm sure their lives aren't routine and I'm sure they have many wonderful things going on in their lives, but work feels like routine. So I go in there, I talk to them, and I've been saying to them, "Larry has been out of work for some time. You wonder why Perl 6 is taking so long? Because he has to feed his family as well." So I guess that's how I address it.

For myself, I don't care – I've always felt that way. I'm so grateful for the support I get from aspects of the Perl community, and that I have had, over historical periods, that I don't want to ask for any more. I've had my bowl of porridge. There are other people that need bowls of porridge.

As long as the corporate sections of the Perl community continue to want what I have to give, I can afford to do that. I can afford to take a quarter or a third of my year and pause my other activities, although going around, I can also do my outreach stuff and get enough money to live, at least out in rural Australia, quite okay.

LXF: Couldn't Larry take on some of that same work?
DC: No, Larry can't do that. He's got a skill set that's quite different from that. It would be an incredible waste of his time travelling around teaching people about regular expressions. So I guess that's what I try to do. Is that a third of my time? I guess time-wise it's a third of my time, but in terms of what I think is important, I think supporting Larry's work is a big part of my work – that's what I'm supposed to be doing.

I do that in different ways. I do that in terms of evangelising his work, and that's another main role in terms of the Perl 6 project. I guess one of my

»

"O'REILLY BASICALLY GAVE [LARRY WALL] HALF A MILLION DOLLARS"

PERLS of WISDOM

« roles is to be one of the major public faces of it. To be the guy that takes the message to the people. The other role is to be a foil for Larry. In a keynote he gave, he made the point very well. Larry's brain and my brain work unbelievably differently, and I'm sure we're incomprehensible to each other sometimes. I know he's incomprehensible to me sometimes! I wish I knew how he got from here to this brilliant solution, and I don't always know that.

***LXF:* And you must have things that fill in gaps for him too...**
DC: We complement each other nicely. Then you add in the other members of the Perl 6 community and the rest of the Perl 6 development team – the inner cabal as we like to refer to ourselves, although it's no cabal – you look at *their* skills. You look at Allison Randal's amazing skills, not just in terms of her deep linguistic understanding and her real connection with what actually works in practice for people, but also her organisational skills – we'd have been totally lost without them! Larry and I together couldn't amass enough organisational skills to get ourselves out of a paper bag, but she has those skills and she brings them to us.

We have people like Dan Sugalski, who has incredible skills in the implementation side, and has a deep understanding of what's actually practical. When Larry and I get going, we start hitting the stratosphere and beyond, and Dan has on many occasions said, "You know, that would be great, and if we just solve this halting problem first it'll happen." So we have people like that.

We have Hugo van der Sanden, who is the Perl 5.10 pumpking right now [*"pumpking" is a Perl community term used to describe the leader of project development – Ed*], giving us tremendous insights into how things currently work, asking the questions that, once he asks us, are incredibly obvious. You know, why didn't we ask that question ourselves? Because of our focus.

I could go on and on and on; there are so many people involved. Luke Palmer, who basically works on the next layer of the onion of Perl 6 development, which is the Perl 6 mailing list. From time to time, Larry or I or Dan or Allison will try to answer their questions and try to give them feedback about how their suggestions are feeding into our process. However, as our lives have become more and more complex and busy, we've been able to do that less and less, and Luke has stepped up to the plate in an extraordinary way.

Just to understand the language at an incredibly deep level, and have the patience to explain it to the people that are coming on this mailing list for the first time. Often they're asking a question that's been answered 20 times before, just asking it slightly differently and giving us something extra, but needing someone to answer so they don't feel like they asked and got no response. We need people like that, and he's another very public face of our outreach.

This is why it's such a collaborative effort, and I think you can tell how stoked I am to be doing this.

***LXF:* Something that's confused us is that you say there are lots of important Perl people out of work, but here at *Linux Format* we'd easily pay someone like you $1,000 for four pages of Perl, as I'm sure other magazines would. How come you guys aren't doing this sort of thing?**
DC: Well, I do actually do some magazine writing. I think, in part, that writing for magazines – and in a sense I do that when I do the exegesis documents, that we are writing for an online magazine essentially – I'm not sure that it plays to our strengths. I certainly know that when I write an article, I don't want to write four pages. I want to write 20 pages, understandably, and I want it to be perfect. I want it to cover the ground completely. , so that's hard to move away from.

***LXF:* Surely you could split it up across five magazines?**
DC: I could, yes, but I find that if I'm writing an article for a magazine, it takes me two weeks to do it because it has to be perfect. Not only does it have to be perfect from a code point of view, but it has to be perfect from a pedagogical point of view – it has to be lucid and eloquent, it has to be well written, and it has to be amusing. And I know that magazines don't care about a lot of those things to that level, but *I* care about a lot of those things to that level. I would rather starve to death than write an article I didn't think people would be glad to have read and felt entertained by.

If we're being honest here, I'm an entertainer before I'm anything else. I want people to have that experience. I will do it if I write an article. I hope the article would be funny, it would be irreverent, it would be from a skewed angle. You can't do that in two days.

"TO BE HONEST, IF I SIMPLY WANTED TO MAKE $3,000, I'D DO A DAY OF TRAINING"

***LXF:* I see what you're saying, but for those two weeks spent writing your feature, you'd get paid $3,000 or $4,000 very easily.**
DC: Yeah, and in those two weeks I won't be able to design Perl 6, I won't be able to answer anyone's emails, I won't be able to write any more modules and I won't be able to travel anywhere. It's deciding.

To be honest, if I wanted to make $3,000, I'd do a day of training in the corporate world. I'm top-shelf there and I get paid that way. So, frankly, as much as I like doing magazine articles, and I do a couple of them a year, economically there are better ways of earning money. Even non-economically, I think that there are probably better uses of my time.

***LXF:* Let's get back to Perl. What do you think of Perl 5 as it stands right now?**
DC: I love it! I think it's wonderful. It's the only actual useable programming language that I actively love. I admire other languages a great deal – I think Python has an enormous amount to recommend it. There are other languages that I think are simply brilliant, like ML, Icon and some of the object-oriented languages I quite like, just for their elegance and the ways they bring things together. Eiffel, for example, strikes me as being a lovely language, while Self is just such an interesting take on object orientation.

However, it's only Perl that actually sings to me. When I very rarely have time off and my wife is away or I'm away from her and there's nothing else I'm doing, I sit down and recreationally code. At that point I'm just playing in Perl, so it fulfils my needs on so many different levels: on a practical level of getting work done, it fulfils my needs and on an aesthetic level of getting work done in interesting and imaginative ways, it fulfils my needs. In terms of performance, it almost always satisfies my needs, so I think it's an extraordinary language.

***LXF:* But surely even Perl has some flaws too...**
DC: Yeah, I'm not blind to them – it has very many, very serious flaws to it. Things that it doesn't do very well. Things that feel like rough edges that no one has got around to smoothing off yet. Things that it makes more difficult than they ought to be. Things that it does incredibly well that were important ten or 15 years ago that aren't any more. Things that are in the core that no one uses any more.

I mean, how many people work with sockets in Perl? There's a small core of people that do, and they love the fact that sockets are core in the language, but I'd venture to say that 95% of Perl programmers never use them in their code.

It just kind of feels like it's nicely optimised for the late 80s, early 90s, but some of the things aren't really optimised, and some of the things that we do all the time aren't pleasant to do – dealing with Unicode still isn't a joy. It's getting a lot better, but for a long time it hasn't been a joy.

Dealing with markup languages, whether that's HTML, XML or whatever, is still not pleasant. There are now tools you can do it with, but it's not nearly as easy as it ought to be. There are bits that are missing that are fine to be missing in a language that was a scripting language, whatever that means, which was basically a souped-up shell.

LXF: **Could you give us some specific examples?**

DC: Well, it was fine that subroutines took all their arguments in @_ and didn't typecheck them or do these other things. However, there wasn't really a strong type system underlying the language that you could extend. I mean, it has a type system but it's not really one that you can hook into and do a lot with. Those things were fine back when most people used Perl to do things that type systems just get in the way of doing. That's changed though, and it's changed for me too.

There are things that I'd love to be able to do, that I know are easy to do in these other languages. They have facilities that enable me to do continuations or multiple dispatch or co-routines, or very complicated object-oriented structures and component-based software development, but I don't want to do them in the other languages because the other languages suck compared to Perl. I want to do them in Perl, and I guess we're moving into the whole rationale of Perl 6.

LXF: **So with Perl 6 coming over a horizon far away, Perl 4 is still being used. There's stuff in there that maybe can't, or won't, migrate. Do you think another big jump will leave these people so far behind they just can't catch up?**

DC: It will certainly do that for some people. And it will do that not only for Perl 4 programmers, but for Perl 5 programmers as well – some of them won't make the jump. We know that happened when folks went to Perl 5, otherwise we wouldn't have those Perl 4 users still. I don't think it will be quite as big a problem for Perl 5 to Perl 6, because of the backwards-compatibility modes that we're leaving in place. That will allow people to do it on a gradual basis, as they wish to do it, but yes, it is an issue that we looked at very carefully.

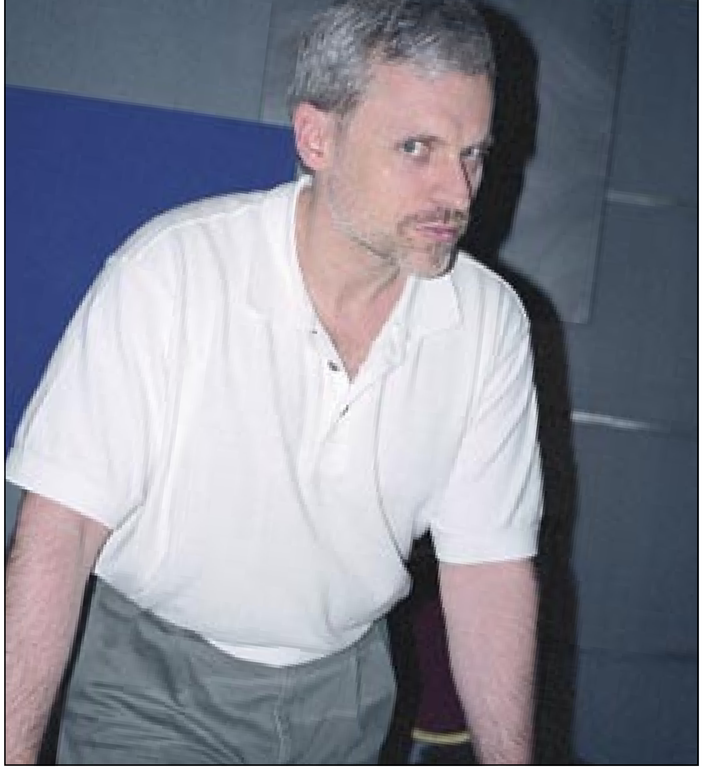
The thing we looked at that we think will be the saving grace is that there is no sign that Perl 5 is going to give up – that people are going to say, "Right, let's just drop Perl 5 and all go over to this side of the boat." For a start, we actually have more of Perl 5 running on top of Parrot than we have Perl 6 running on top of Parrot, so it's pretty clear to me that Perl 5 will run on Parrot, so people will be able to continue to use it if they want, and they will get the performance benefits that will come from that migration.

People won't move, but new people will come into the community. A lot of new people will come into the community when a lot of the Perl 6 code doesn't have to be as scary any more. You can do object-oriented class development that looks just like Java. Your boss might not even be able to tell that it isn't Java, so you can just say, "Yes, we're doing it in Java. All those dollar signs? Yes, they're new." And that will be it.

People will want to try it out. When they see the kind of functionality we're putting in there, that will happen. We're not under the illusion that people will suddenly say think that this is going to be the messiah. We know that this won't happen. We know that people will be slow. We know that, because we know people who still code in COBOL, in Fortran and in APL. That is the nature of the beast.

There are always more people coming into the industry than there are currently in the industry because computers continue to become closer to ubiquitous, and everything's going to have to run on something. Some people will migrate, some people won't migrate, and all of the new people coming in won't even know what Perl 5 is, so we're not really worried.

"WE KNOW PEOPLE WHO STILL CODE IN COBOL, IN FORTRAN AND IN APL"



LXF: **If you had to limit yourself to just four, what would you say the key features in Perl 6 are for you?**

DC: The four things that I most feel are important in Perl 6... Well, one is the underlying type system. The fact that we have a real type system on which we can build all types of other features like multiple dispatch and subtyping. Also, most importantly from that point of view, compile-time checking on a lot of things that currently can only be checked at run-time, if at all.

The new object-oriented system, which kind of comes from the type system, is also critically important. There's now a canonical way of creating classes that is declarative, largely compile-time checkable and safe under multiple inheritance. All of those things are critical improvements that will allow Perl 6 to move into production environments in a way that Perl 5 often hasn't been able to do, so that's number two.

If I can be allowed an indulgence, number three would be something that I invented and put into the language – the concept of junctions. A junction is a single scalar value that effectively acts like a set of scalar values, but a set that also has a logical predicate with it, so you can have a set that represents all of the values, a set that represents any of the values, and a set that represents one of the values. The keyword here is quantum superpositions, but we don't talk about that any more because it scares people. We call them junctions now.

The thing about this is that they open up an enormous number of idioms, new algorithms and new ways of structuring code with very much less code, and yet that's very much more readable. So, for example, you can say, "If any of my numbers is greater than 1," and you can literally write "if any(@numbers) > 1)", and you can immediately see what that means. You don't need to put a comment there because what comment could you put there that could tell you any more than the code itself? But the really important thing about that is that you've not only made your code a lot cleaner, but you've also written it in such a way that the compiler can look at it and say, "Hey, that's something I can do in parallel. When I see an operation on a junction, because of the set-like nature of it, that operation I can do in parallel, and I can short-circuit if any thread yields a value that decides what the overall result answer will be."

It may not be that the initial versions of Perl 6 actually do that, but what I wanted to open up was the possibility of writing code that mere mortals could write, that could be inherently parallelisable. There are other constructs in there too, like hyper-operators, that will allow me to do the same thing for operations. So, for a hyper-operator, you can say, "This is normally an operator between two scalars, like a multiply operation, but if I put these symbols around this multiplication operator, it becomes a vector operation I can do between two arrays of values, and it will do it element-by-element across the array". »

PERLS of WISDOM

LXF: **So, operator overloading?**
DC: Not really. It's the same operator, but now instead of operating on just one pair of scalars, it's operating on multiple pairs of scalars drawn from two arrays. The tremendous advantage of that is that it's another way the compiler can be told to thread this out, to process this out, to *processor* this out. That kind of general support for parallelisation and for cleaner ways of describing that process is what I think is the third really important feature of Perl 6.

For number four, as much as I would like to say Perl 5 compatibility, I won't. What I'll say is that living on top of Parrot is the other critical thing. What that gives us is not just tremendously better performance on a wide range of platforms, but it also provides us with a much easier way to build the compiler, because all we have to build now is a translator to Parrot representation.

In addition, this provides a much, much, *much* cleaner operability with all of the other languages that we want to use in everyday development. Perl has always been the glue language. Well, now we're moving up the stage and Parrot becomes the glue language.

LXF: **Are there particular parts from other languages that have caught your eye?**
DC: There are some tremendous modules in Java, for example. There's stuff out there that allows you to build enterprise applications that we just don't have anything like in Perl. There are some tremendous things in Python, hidden away in the Vaults of Parnassus; some modules that we just don't have any equivalent for. There are some amazing things you can do with PHP that are really, really hard to do with Perl. I'm sure there will be great things you can do with C#. I don't know what it will be, but I'm sure it will be great, even if just means you don't have to write Visual Basic for Applications any more. I want to be able to do all of that, and I want to be able to hook in a C program that doesn't require me to sell my soul to XS to hook it into a Perl program.

I want to be able to grab some of the great C++ code out there that does really cool stuff very quickly, and Parrot is going to give me all of that. It's going to make all of that easier and it's going to give me a way of taking a Java class and inheriting from that Java class in Perl. The ability to inherit from a Java class, and maybe multiply inherit from a Java class and an Eiffel class and put those things together in a sensible kind of way in my Perl program, which is where I love to program... the kind of power that's going to give me, I can't even begin to imagine.

So, to me, that uniform platform that everyone is going to be able to work on, and is going to work everywhere – not just on proprietary Microsoft systems, or pseudo-Open Source Sun systems, or Linux boxes or whatever – the thing that's going to work on my Palm Pilot in five years' time, that's going to work embedded in my refrigerator. Man, I would love to be able to program my refrigerator in Perl! And maybe I'll be able to. And maybe I'll need to use the Java classes that have been written for refrigerator control, but I'm going to be able to control it with Perl. So, I think that's the fourth big thing.

"MAN, I WOULD LOVE TO BE ABLE TO PROGRAM MY REFRIGERATOR IN PERL!"

LXF: **Looking at the kind of Perl 6 code we've been seeing so far, the most apparent difference is visual: half the code has gone. That is, what used to ten lines of code can now be done in five lines of code. But, and this is a big but, it does it at the expense of adding mojo – adding magic to the code that "just works". Do you think that increases the barrier for entry for new programmers?**
DC: What you've got to remember is that when I'm showing that kind of code, I'm showing them and explaining that this is half the difficulty. The thing that I almost always say is that what I'm showing is a "diff" – a diff against Perl 5. I'm showing you the things that are different, and that sometimes freaks people out, because I spend an hour or sometimes five-and-a-half hours, and it's just new thing, new thing, new thing, thing that's different, new thing, new thing, thing that's different, new thing. That's why I fully understand why people think it's going to be a totally different language to Perl 5.

The next level up and you say, "Yeah, that's Perl-ish. That's probably how it should have been in the first place." However, the feeling is that "everything is different. What people forget is that I'm showing them about 15% of the language. About 85% of the language looks and works exactly the same as before.

What I'm showing when I'm talking about the new ways of doing things is demonstrating the optimised way of doing it and then the native, idiomatic way of doing it. What I occasionally do, but not often enough, is say, "Okay, here's the literal translation between Perl 5 and Perl 6 of this." And what you see then is you have a page of text and, like, 10 characters change between the two.

We really have worked very hard not to pull out things from Perl 6 that didn't need to be taken out. There were some things that absolutely had to go – they were just wrong and were leading people astray.

The most significant change is that the sigils on variables – the dollar, the at sign, the percentage sign – stay with their variables no matter how you use them. They don't mutate depending on how you use the variables any more. That causes the code to change a bit, but it actually makes the code more readable.

LXF: **So you can essentially use Perl 6 as if it were Perl 5, but there's also a better, Perl 6-specific way?**
DC: I have this talk where I say, "Okay, here's this in Perl 5, here is the straightforward, literal change to Perl 6. Do you notice how few characters change? And now here's the idiomatic way that we're adding on." So, the answer to your question is that you're still going to be able to take the same sort of baby steps, we're still trying to make that learning curve as gentle as it is in Perl 5. Because it is very gentle: people can spend years and years getting confident before they ever use a *map* operation.

You can do that in Perl 6, absolutely. In fact, in Perl 6 it may even be easier to do because it may be fairly trivial for people to write modules that restrict you to a subset of the language. The problem with doing baby steps with Perl 5 is that if you teach people maybe 10% of the language, which is enough for them to do real applications, they'll do that but they'll accidentally make a mistake that will be meaningful. This means they'll get weird behaviour or error messages they don't understand.

In Perl 6, you'll be able to take the Perl 6 grammar that Perl 6 actually uses and derive a new grammar from it that cuts out 90% of the rules, so that anything you do that's out of the normal, you can send back a message that says, "Perl knows what this is, but you don't, so don't do it." Or just "invalid syntax, don't do this".

We're going to give them ways of building themselves safe sandboxes of varying sizes. There's no one on board to actually do that yet, but even so it's still going to be that process of gradual learning that allows you to get better. The thing about it is that the curve is just going to get longer – it's going to get a lot longer because there's a lot more in the language.

There's a lot more features and things you can graduate to. I don't expect everyone to start using junctions, except that you can use junctions without understanding all of the quantum physics underlying them. You can just write: "If any of these is less than 10." You don't have to understand how it works behind the scenes, because the fact that it reads like that allows you to use it. As such, a lot of the things we're adding in can be explained to people without explaining the deep semantics, and when they need the deep semantics they can learn them, but slowly.

LXF: **So how about something like PCRE – the Perl-Compatible Regular Expressions? In Perl 6 there's a whole new way of doing regular expressions that many**

have said is a lot better. Do you think PCRE will stick with the Perl 5 standard, or move to Perl 6?
DC: I think a lot of people will stick with what they know, because Perl 5 regular expressions are at least close to standard Unix regular expressions. They are different enough that they do actually confuse people occasionally – the difference in behaviour of the caret and the dollar sign, for example, occasionally knocks people around.

In most of the other Unix applications, the caret and dollar signs mean the beginning and end of a line, but in Perl they mean the beginning and end of the string, which isn't necessarily the same thing.

LXF: **That's fixed now, though?**
DC: Yeah, in Perl 6 that's fixed so it works how people would expect. The thing about that is that a lot of people will stay with PCRE because they're happy with it and they're comfortable with it, but Perl has always set the benchmark for regular expressions – for power, for flexibility. And when we move on, when we improve ourselves, other people will want to follow.

The thing that we're looking at right now is how we can implement the Perl 6 regular expression engine so that it will be easy for people from other languages to pull that bit out and plug it straight into their language. We have ideas about that, and the idea basically revolves around the notion that we think the Parrot engine is a virtual CPU, that we have an assembler language we code for.

I want people to think about the regex engine that will be in Parrot as being a virtual co-processor, that won't be deeply integrated into the Parrot internals, that will have a relatively small interface back to Parrot, and that interface will be vanilla enough for people to be able to rip that box off the side and plug it into their interpreter for their language.

They'll have to do work to make the syntax of their language reflect the new syntax and semantics of Perl 6 regular expressions. I'm not saying you'll be able to just plug it in and everything will be just hunky-dory, but we should give them a way of doing that with a minimum amount of effort.

I think that's been the incredible thing about the PCRE project – you couldn't pull the Perl 5 regular expression engine out from the Perl 5 interpreter. You just couldn't do it. It was impossible. And so, all of the extra features that Perl 5 provided were for a long time denied to these other languages simply because there was no way of going out and actually implementing it.

You see now that a lot of languages have "Perl 5 regular expressions compatible!" among all the other features listed on the box or in a splash in a corner, and I think that's largely attributable to PCRE. I think that's a wonderful thing that was achieved there.

"IT TOOK US FIVE YEARS TO DEVELOP PERL 5, IF YOU COUNT PERLS 1-4 AS BEING PART OF DEVELOPMENT"

LXF: **Larry said a while ago that Perl 5 was his rewrite of Perl, and he wanted Perl 6 to be the community's rewrite of Perl, but some might say that the long wait for Perl 6 is indicative of the 'second system effect', described by Fred Brooks in his book *The Mythical Man-Month*. What do you think – is Perl 6 biting off more than you can chew?**
DC: I don't believe so. I think that I can understand people having that impression, that we bit off more than we can chew, that it seems to be taking forever, and that we must be thrashing or whatever. However, the thing that people forget about is that when you think about Perl 5, it took us five years to develop, if you count Perls 1 through 4 as being part of that development process.

Perl 5 is what most people think of as Perl – it took us five years to do that. For a lot of people, Perl 5 is really only Perl 5.5.3 and later. That was when people started thinking that Perl was stable enough and you could write applications using it. If you call it that, then you're saying it took ten years to develop Perl 5. We've only been going for four years. If we can get it out within a year, we think we're pretty much ahead of the game!

People are saying that they had Perls 1 through 4 during that time and they could do stuff with them, but the very act of doing that constrained us to be backwards compatible at every stage, and that's what led us down so many of the garden paths that we're now trying so hard to correct.

We get one go at changing the syntax of Perl, at bringing in these backwards incompatibilities to fix where we went wrong. We only get one chance at that because people aren't going to stand for it a second time. *I'm* not going to stand for it a second time.

That being the case, it behoves us not to move too soon. It behoves us not to rush in with the first idea we had, because that's how you get second systems. It'd be great to do this, let's bolt it on the side and let's bolt this other thing on the other side, and if you bolt enough things on enough sides you've got Frankenstein's monster. And it looks like it, and it works like it.

Now I have enough time to really explain how Perl 6 works. In Toronto I gave a five-and-a-half-hour lecture just on how Perl 6 works and people stayed, and the reason people stayed was because by hour three, they started to see how neatly this all fits together. They saw how eloquently the features I showed up front combine together to give us things we don't have in other languages, or things that we do have in other languages but have to have their own syntax.

We feel that we're primarily developing this language so that it will be up to the needs of the next 20 years of Perl programmers. That's our outlook. If we can't take five years, one quarter of that time frame, to get it right, then we won't get it right and we'll have to do Perl 7, and everyone will have to go through the whole agonising process again. We would rather do it right than do it right now.

LXF: **So what you're trying to say is that there are no plans to try to speed up development?**
DC: Right. What we plan to do is make sure that everything that goes in there can be done quickly and can work in the way we expect it to work, and provide an increase in power and flexibility, while also providing an increase in both the maintainability and readability.

LXF: **So, just one last question then... when *is* Perl 6 coming out?**
DC: Aha! I could tell you, but then I'd have to kill you... LXF

This interview first appeared in the January 2005 issue of **Linux Format** *magazine.*