AN • INTERVIEW • WITH

Trenton Schulz

by Allan Jude and Benedict Reuschling

The *FreeBSD Journal* Editorial Board suggested that some of the outstanding interviews from the BSD Now series might be of interest to readers as they reflect the state of technology when the interview took place. Here, we've transcribed and excerpted from an October 2019 interview with Trenton Schulz by Allan Jude and Benedict Reuschling.

BENEDICT REUSCHLING: We have a special interview with Trenton Schulz about his early days with FreeBSD, the Robot Operating System (ROS), Qt, and more.

Hello, I'm your host, Benedict Reuschling.

ALLAN JUDE: And I'm Allan Jude.

BENEDICT: Trenton Schulz will talk with us about Robot OS on FreeBSD. First, welcome to the show, Trenton. Can you tell us a little bit about yourself and how you got started with BSD?

TRENTON SCHULZ: Sure, well, thank you very much, Benedict. I work as a researcher at the Norwegian Computer Center. Previously, I was a PhD student and a software engineer. I got started in BSD when I was working on my bachelor's degree back in Minnesota, in the late 1990s, and it was a game night at one of the computer labs when everybody was allowed to bring in and play computer games on the lab computers.

Except there were a couple of people in the back of the room who had brought in their own computers. And they had done that so they could access the network, and they were installing this weird DOS thing on their computers, or at least that's what it looked like to me. And so, I asked, "What are you doing?"

And they said, "Well, we're installing FreeBSD." And I responded, "What is FreeBSD?" And they explained, "It's an operating system."

I said, "You mean like Windows, or whatever?" And they said, "Yeah, but it's UNIX," or something like that. And I had heard about UNIX at that point, so I was kind of interested. Within a week or two, I had brought in my computer and installed it. And then there were about three of us at the university who were working with FreeBSD and sending questions back and forth to each other or looking in the mailing lists, and things like that.

I think I installed FreeBSD 2.2.5, and of course, it was interesting just getting the software to work. I remember complaining that the sound didn't work when I wanted to listen to music or something like that. And that meant upgrading to -CURRENT.

[One of those] friends had no problem running build world on my computer and upgrading it to -CURRENT. Of course, then other things didn't work.

I got to learn a little bit about how these operating system things work.

ALLAN: Hmm, tell us how you got your first job related to BSD.

TRENTON: Well, it was sort of tangentially. While I was doing my computer science studies, I was interested in being able to do graphics on computers. And since I was using FreeBSD and the X Windows System, I wanted to do graphics on that. I was looking for toolkits that I could

work with to do that, and since at that time at that college they were teaching us how to do C++ programming, I found the Qt library. Or "cute" as it's called among all the people who program for it.

I started working on some small Qt programs on my own, and then I kind of rolled them all up into a little package, sent them off to Trolltech, with my resume that I had written in plain text. But they were looking for employees and I got hired. At that time, Norway kind of had a shortage of C++ programmers who weren't in the oil industry.

I ended up coming to Norway, and I got a FreeBSD box when I got to Trolltech. At that time, I was helping out with the Qt/embedded port, which was using the frame buffer. Of course, I was interested in trying to get Qt/Embedded to work with FreeBSD, but I didn't know nearly enough [about device drivers] to do that.

There was a virtual frame buffer that worked on X11 that was just a shared memory so you could actually display things and use the Qt/embedded windowing system [and X at the same time].

And I remember the first weekend I was in Oslo, or the first couple of days, I was at Trolltech fixing the Qt virtual frame buffer to actually work on FreeBSD. In the end, it was just using some Linux-specific semaphore calls and then just changing them over to use the FreeBSD ones. And it worked on both Linux and FreeBSD. I think that was probably the first commit I did to Qt.

I actually shared an office with a guy named Brad, who was also a giant FreeBSD person. He had worked on the Blackbox window manager, which was the forerunner to the Fluxbox window manager.

We were the two FreeBSD people at Trolltech, and the idea was to make sure that Qt was somewhat semi-operable on other operating systems. Or other UNIXes I should say.

BENEDICT: Oh, that's great.

ALLAN: How long were you there?

TRENTON: I was at Trolltech for—well, I guess seven years until they became part of Nokia, and then I stayed on for a year at Nokia as well. At that point I had moved off FreeBSD and was working with Mac.

[The switch to Mac was because] sometimes everything would get out of whack with FreeBSD, and then you would run this portupgrade utility and try to rebuild all your ports, and it was really getting annoying. There was nothing like package or anything that we have these days.

ALLAN: Exactly.

TRENTON: I think it was a point there where I was willing to let a vendor do all that extra maintenance work so I could program. Trolltech had hired me to program on Qt, not maintain my system.

ALLAN: Yes, sort of maintaining an operating system's use of Qt and so on.

TRENTON: I moved over to the Mac for a while then. Then, I finished one of these big projects and I felt that I needed to try something different. So I jumped into research, which is interesting in its own ways.

ALLAN: What kind of things have you been doing research on?

TRENTON: Well, I just turned in my PhD dissertation on human-robot interaction.

ALLAN: Ooooh.

TRENTON: There I dealt with two separate points. One was looking at this idea of having robots in the home and then what you would do in those cases. You have to be concerned about the privacy issues that arise with having a computer with lots of sensors moving around your house.

And then the other aspect goes back to my GUI programming days at Trolltech. I was interested in animation techniques and ways that robots can move to make them a bit livelier or a bit more entertaining. I looked at how you could use animation techniques to move robots and how that affects people interacting with them.

BENEDICT: Is that related to your work with the Robot Operating System?

TRENTON: In a way. Like most things for me with FreeBSD, it's always kind of a side hustle, it seems. When I started at the University of Oslo, the Informatics department was pretty open about what you could run—whatever operating system you wanted in general. If it's a Windows machine, they, of course, will administrate everything. But if you're running a Mac or something else, you pretty much have free reign.

I said, Okay, I'm going to get one of these nice, new Thinkpads and try running FreeBSD on it. At that exact point the new Macbooks had come out and I was unimpressed with what they were offering as far as something that I could open up and repair.

BENEDICT: I hear you.

TRENTON: Having done that [opened and fixed] Macs before, I was less than confident that that wouldn't be required [again]. I got a Thinkpad and installed what was called at the time TrueOS to get the wireless card and everything working.

And I used that for about a year, and there was really no problem for the work I was doing, and then I would use virtual machines to run the Robot Operating System or ROS.

I guess I should maybe stop and explain what ROS is before going further.

ROS is a middleware that can communicate between different nodes on a network that it builds. And you can create topics that publish information, for example, pictures from a video camera, sensor readings, or things like that.

And the thing is it can be distributed so you can run it on different computers, and at the same time, it's a very standardized interface, so once you have a driver on your robot that can publish information in a good way, you can just hook it up to the node and everyone can use it. And it's an open-source project, so it's easy to get all these packages.

It's a pretty nice little system of packages and communication, and it was set up to only work on Linux. You could run it with Homebrew on the Mac and using, I guess, the Linux layer on Windows 10. But it currently was set up so ROS would do distros that followed Ubuntu.

When Ubuntu did an LTS release, ROS would do an LTS release, and they would support that on all the robots or whatever until the time ran out for Ubuntu's support period.

So I had this FreeBSD box, and I had to start working with one of these robots that was using ROS. And I thought, well, I don't want to have to do all this network hopping to get a virtu-

al machine to talk directly to this robot. And I thought I would try to port it.

I downloaded the sources and started working on building it. Almost everything is written in C++ or Python. [The build system is] all based on Cmake stuff, and as far as I could tell, most of that was already in FreeBSD [ports]. If I could follow these scripts, it would probably work okay.

I got pretty far—I got the basic communications stuff done. And I thought, okay, this might actually be something I could do. I could get this up and working because I managed to get that done in a day by just having that compile in the background while I worked on other stuff.

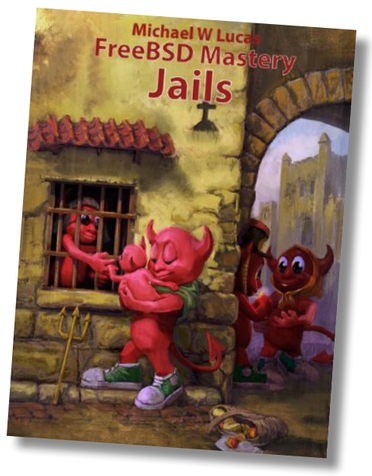
But then I realized I needed the graphical components that are also part of ROS. ROS also has a lot of debugging tools and ways of visualizing data. And simulating robots as well because you don't want to blow up a robot by putting bad code on it. You want to simulate it first.

I started to think, okay, I can start to try to port these other programs as well to get a desktop version of ROS. It was really tough because one of the first things that needed to be built was OpenCV—a Computer Vision Library. And at that time, FreeBSD [ports] had Version 2, but ROS needed Version 3.

And that was just a giant thing to be [ported and] built, and at the same time, it needed Python Qt bindings, and it didn't say for sure which ones it needed. If you know the Qt community, there are at least two different types of Qt bindings that are available for Python.

It just got to be a mess. I spent a few days on it. But then I realized I had to get a PhD done rather than spending time on getting everything to compile. Sadly, I had to bite the bullet and install Ubuntu on the computer and just go and get things done.

Then I realized, well, I could give this another shot because things have happened since I tried in 2017. We're in 2019. I saw that OpenCV [3] had come in. We had modern Qt and everything, so I figured that this could just work.



I had also read Michael W Lucas's book on jails which gave me a much better grasp on how to use jails.

I was able to create the Jail. Copy or download the sources. Install all the prerequisites and actually get compiling. And, of course, I had created ports for some of these building tools that you need for ROS, and I put those in the port's tree as well.

It was really quite nice to build ROS in jails. It got rid of issues about knowing which Qt bindings you needed and stuff like that because you had a clean environment.

ALLAN: It doesn't conflict with the version of Qt you're trying to use for your desktop at the same time?

TRENTON: You got it! I was very happy and wondering why I hadn't done this earlier. But it's difficult to just step into jails. You have to do some reading to really understand how they work.

ALLAN: Yeah, Michael Lucas discovered that when he went to write the jails book, he had to write six other books first.

TRENTON: Indeed! Indeed. And I've read those other six books. And I guess the last thing I'll say is that after reading his book I was finally able to build a CUPS Jail, which I had tried and failed multiple times previously.

ALLAN: Yeah, you might not think you need it, but once you read it, you'll realize there's all kinds of ways you could be using it to make your life easier.

TRENTON: Yeah, but back to the ROS stuff—using those jails, I was able to build up the basic communications parts again. And then I built up the desktop side of it as well. And yeah, I got a graphical ROS distro so that I could use the basic tools, the graphing tools, the visualization tools, and so on.

I still didn't have the simulators because nobody ported those yet. Those simulators are specialized for robots only. If you don't have all of ROS there, there's not much point in porting the simulators on their own.

The simulators are Gazebo, which is the big one, though as far as I know, some people are considering going over to using things like Unity or Unreal as well. But I don't think they have the same physics engines as the gazebo simulator.

Then I thought this might be an interesting talk for a EuroBSDcon. I wrote up an abstract for that and sent it off to EuroBSDcon.

BENEDICT: Was EuroBSDcon your first BSD conference?

TRENTON: Yeah, it was. The nice thing about EuroBSDcon was that it was coming to Lillehammer, Norway.

It was my first conference, and yeah, it was quite enjoyable.

ALLAN: What would you say was the thing you enjoyed the most about the conference?

TRENTON: Well, let's see. One thing I enjoyed was the tutorial the first day. That was Tom Jones's tutorial on FreeBSD hardware hacking. I had never done a lot of that stuff myself. I've always been involved with people who have, but I've never actually got to sit down and try out some of the stuff myself.

The thing for me was just to be able to hang out a bit and talk with people and learn some new things about what's going on with current developments in all of the BSDs. I think it's also kind of fun to sit and see some of those talks instead of finding them later when they're posted online. Because you also get a chance to at least ask some questions and stuff that you might not have had a chance to otherwise.

I would recommend it, especially if it's in a country that you're living in and it's not a horrible burden to get to it, I would say definitely do it. It's a nice conference. And well-run.

BENEDICT: Switching gears a little bit, since you started with the BSDs and then had a little break and then came back to it, do you have some tips or advice for people who are starting out with the BSDs, what they should do, or what they should avoid? Some pitfalls maybe?

TRENTON: Hmmm. When I started with FreeBSD, it was important to be able to use things that friends were using as well. The Internet was much different in the late '90s than it is today in terms of being able to find information.

The nice thing was that you could get real-time help right then and there. I guess I would say one thing is if you can get into BSDs with a friend, that will be helpful because then you can help one another as you discover things.

If I were to offer a tip—if you are going to use ZFS, you should really set up the zfstools port early when you get started, so you are automatically doing snapshots so that you can actually roll back, because I think it's easy to say, "Oh, you just rolled back to a snapshot." But if you forgot to take a snapshot, you're in trouble.

ALLAN: Yeah, in addition to whatever manual snapshots you might decide to make, please also make a snapshot every 15 minutes to last 4 hours and then some grandfathering test scheme.

TRENTON: Yeah. Set that up because then you can be much more fearless in what you're doing. You don't have to be too worried about breaking stuff. And then you can use boot environments as well.

ALLAN: Yeah, I guess I hadn't really thought about the fact that in the 1990s, if you were going to put FreeBSD on a machine, it was probably going to be the only OS on the machine. Or you'd fight with dual booting or whatever. But you were pretty much giving the whole machine over to it, whereas with the virtual machines you can completely risk-free spin up a FreeBSD machine without having to risk messing up your computer and having to fix it.

TRENTON: Exactly, exactly. I think the only thing that becomes a constraint then is just having enough disk space to kind of dedicate to it. If you want to install a lot of software and see how that works, you have to figure out that upfront. But otherwise, it's really nice.

When I first started to look at FreeBSD again, the first thing I did was load up Virtual Box and put a FreeBSD image on there and just tried to see how it worked so I would have an idea of what I should do before I did it for real.

ALLAN: Yeah, one option we'd like to add someday is being able to have the installer say, Oh, I see you already have a FreeBSD installed. Maybe you've broken it. Would you like us to do a fresh install as a new boot environment but leave all your old files behind so that you can still get at them? But you'll have a system that boots cleanly again.

TRENTON: Yeah! That would be nice.

BENEDICT: Yeah, more work for the future.

ALLAN: And actually, before we let you go, I had another question about the Robot OS. You just talked a little bit about it and how it's distributed. Would one robot consist of multiple computers, or is it more about making many robots work together?

TRENTON: Hmm, I guess it could be both, if you wanted. Basically, you create the nodes and each node has certain topics. A robot could have lots of nodes. In one of my research projects, I was using a turtle bot, which is a robot that has a LIDAR [light detection and ranging] sensor on top that spins around to find its location.

And it also had wheels to turn and so on, and each of those things was a topic that would say what speed the robot was going and how it was turning or moving. There was another topic that was getting the data from the LIDAR, and the whole point was that you could write another node that would say, oh, I'll listen to this information from the LIDAR and use that to figure out a map location, and then I'll use another node which has the planner for the robot and will tell the robot where to move.

You end up creating a lot of different nodes, and most of the nodes are written in either C++, Python, or Common Lisp. And in general, they'll be fast enough for what you're trying to do, as long as you're not trying to do real-time stuff. That's what ROS 2 is about.

It's an interesting system and used for a lot of robots nowadays. But like a lot of these things, it is very research-oriented, and most of the ROS stuff—at least the ROS 1.0 version—

does not consider security. It's sort of using the FTP model, where you'd go to a known port and then it gives you another UDP port for listening, but you don't know what the UDP port is.

If you have a firewall, the first thing they tell you to do is create a VPN between you and the robot so that you can talk to each other.

I believe they've tried to fix that in later versions, but I remember that was an issue here at the university because the wireless network in general needs to be closed down. Suddenly, you need to open all these ports, and you don't know what they are until you actually make a connection.

ALLAN: They kind of assign randomly when you try to connect to the robot?

TRENTON: Exactly. And that doesn't work very well. You have to create a very specific, small, closed network. But, in general, I would say most robots should probably be on a closed network [for security reasons].

ALLAN: Yeah, I guess it gets back to more of the research you were doing about how to apply privacy to the robots. If a robot has cameras for eyes and is recording what it is seeing, how do you decide what to do with that and so on?

TRENTON: Right, and there's the other question of what is actually happening with the camera images? If the camera image is something that's just being used locally, for example, if it's a depth camera, it's just trying to figure out the distance between different objects. And all that's happening is finding the robot's location and planning where it's supposed to go, which will never involve going off to the Internet. It's all being processed locally.

In that situation, it's not such a major privacy issue. There are a whole bunch of issues that need to be looked at, and that was one of them.

BENEDICT: Yeah, you'll have to leave something for the future scientists. They're still work…

TRENTON: There is plenty to do. I was at a robot conference just last week, and one of the points discussed was that there are so many research problems that have not been solved and will not be solved for a long time. No question that there will be plenty of things for the future scientists.

BENEDICT: Excellent.

ALLAN: Trenton, is there anything else you'd like to talk about before we let you go?

TRENTON: I can't really think of anything other than thanks very much for the interviews you do.

ALLAN: Thank you very much for being willing to talk with us.