Recollections:
An Interview with Warner Losh (imp@)

BY TOM JONES

TJ: Can you tell us what you were doing in the late eighties and early nineties in the buildup to the FreeBSD project?

WL: In the late eighties, I was getting my degree in computer science and mathematics from the New Mexico Institute of Mining and Technology. We had a Vax 11750 with one or two megabytes of RAM that supported 20 or 30 users for the computer science department. Around this time, we also got 18 or 20 SUN workstations. We were transitioning off old DEC System 20 running TOPS-20, and so most of the campus was Unix around this time.

After college, I got a job at The Wollongong Group doing tech support for their TCP products. The Wollongong Group had a connection to Unix that I wasn’t aware of when I joined or didn’t realize the full significance of. The first Unix port was done at the University of Wollongong in Australia, and this company had bought the rights to that port.

I was working there and doing tech support for VMS TCP/IP—their product—and didn’t realize the opportunities I was missing out on to meet some people that had been involved in early Unix.

After that, I went to work for Solbourne Computer Inc. that did Sparc Station and Sparc compatible servers running their version of SunOS. It was about this time that I started to get involved in open source. To do my development work, I had bought a PC and was running Linux on it because there wasn’t a FreeBSD distribution and the patch kits were difficult to obtain. They weren’t well publicized, so, I didn’t really know about them.

There was a war on at the time about which graphical interfaces to use. Jordan Hubbard had gotten in touch with me because he was using the same graphical toolkit. We had developed a toolkit that presented in both styles, so you could write your software once, run it anywhere, and life would be good --was the theory. And Jordan was one of the users of that.

One day, he contacted me and said: “Oh, by the way, I’ve got this operating system I’m trying to bring up called FreeBSD. You should port to it.”

I said, “How do I install it?” And he goes, “Well, send Rod Grimes a disc or better yet just buy a disc and I’ll send it to Rod Grimes and he’ll put it on there and send it to you and you’ll be able to run FreeBSD.” This was pre-1.0.

The first Unix port was done at the University of Wollongong in Australia, and The Wollongong Group had bought the rights to that port.
During that time, I was also contributing patches to things like tcsh. I would find bugs and I would fix them and I would send a patch and the tarball readme file. Even before I started FreeBSD work, I was doing open-source work.

TJ: I've asked other people how they came across FreeBSD, but I guess you were the result of direct marketing by Jordan.

WL: Jordan wanted to do the same thing I was doing. He wanted to run this toolkit and this GUI builder on his FreeBSD machine for the company. It was in his best interest that I do the port and that’s kind of how he recruited me. And I’ve been using FreeBSD ever since.

TJ: So, once you were using FreeBSD, how did you keep track of the conversations wherever they were happening?

WL: So—dragging up old memories—I think some of it might have been posted to Usenet. I believe my employer did have some access. We certainly weren’t on the Internet, but we had a connection to Usenet.

So, that’s how I stayed up to date with what was going on with FreeBSD and had a chance to argue a little bit online. That’s a trait I’ve had for a long time. I’ve gotten better about arguing for arguing’s sake. I try to make a point these days. There were a lot of flame wars in the early days, some of which litigated good points and others that were the most crazy, silly minutia you could imagine.

TJ: What do you think kept you involved in the technical side of the project early on?

WL: Early on, I did some initial cleanup work of warnings--there were thousands of them when we first started. I think the most significant thing I did was taking on the role of PC card maintainer and getting CardBus working on FreeBSD. These were emerging technologies at the time and very important--this was like the 3.x timeframe.

In the early days, releases happened fairly frequently. I think it was like a year or two or three after the project started when I got my first laptop and started doing this. That was probably my first large and significant contribution to the project. Up to that point, I had a number of less ambitious things, mostly dealing with cleanup and bug fixes for things that I would encounter when I was trying to get stuff done with FreeBSD.

TJ: Do you know the background of the lift that happened between FreeBSD 1.0 and FreeBSD 2.0?

WL: I know a little bit about the background and the lift that happened, but I wasn’t directly involved in that. I know a lot about the lawsuit. I learned about it at the time, and I’ve since studied it in detail. Briefly: Berkeley produced a release of 4.x that had all the AT&T...
code stripped out called Net2. Bill Jolitz took that and created 386BSD, and then created a company called BSDI with some other people. So, there was a free version and the commercial version, and Bill Jolitz kind of disappeared from the free world.

Jordan Hubbard, Nate Williams, and a bunch of other people were involved in creating a number of different patches for the system. They produced FreeBSD 1.0 and 1.1. AT&T got upset with BSDI for using the 1-800-ITS-UNIX phone number and sued them for trademark infringement. That evolved into a big, nasty fight about trade secrets and copyright and all of that.

A lot of the rulings were going against AT&T, so they settled that suit and imposed a settlement in the free community where basically FreeBSD—the principles of FreeBSD—agreed to not do any more releases based off Net2, but, instead, do the releases from Berkeley’s 4.4 Lite. Berkeley sanitized everything and got AT&T’s sign off and there were still some missing bits.

Everybody that was involved with FreeBSD at the time grabbed the new sources, grabbed the drivers that we had written for 386, and grabbed some of the code we had written and modified for 386 that we were allowed to use. We rewrote bits and pieces—about a half dozen files—that were missing from the system.

FreeBSD did this and NetBSD went through a similar process at the time. It’s one of the reasons some of the low-level stuff is different between the BSDs. Each of the projects rewrote it on their own because by then the two projects had split and FreeBSD was focused on x86 and making it fast, and NetBSD was focused on portability.

So, there were different sensibilities and different, very strong personalities that precluded effective collaboration—which is probably the sanitized way of describing all the fights and flame wars and stuff that went on at the time. The lift to make this happen occurred over the course of weeks or maybe a couple of months.

You can go back and look at the early history when we converted to Git. The history goes all the way back to 2.0 but no further because it was hard to pull in the 1.x history and there was a break between the two.

You can look in Git and see that things were going fast—people were committing fast and furious. David Greenman was involved, Jordan was involved, Rod Grimes, I think even Poul-Henning Kamp was also involved by this point in doing the lift to make everything happen.

**TJ:** Can you tell me how you got more involved with the organization of the project?

**WL:** One of the problems with the project early on was that it was centered around Jordan and the Core team. People were coming and going on the Core team, but it was very self-selected and there were a growing number of developers that were dissatisfied with that.

We felt that the Core team wasn’t representing the project well enough. There wasn’t enough representation from the different parts of the project that were organizing the
ports effort. Jordan had written bsd.port.mk and Satoshi Asami had taken it over and Jordan was on the Core team, but there wasn’t a lot of other ports representation.

So,—rightly or wrongly—people were starting to feel that the Core team was an elitist institution. Core was then, as now, probably even worse than terrible at communication, terrible at decision making, you know, their operating processes weren’t well defined at all.

We hit on the idea of having Core be an elective body—Poul-Henning Kamp, Wes Peters, and I. We got together and wrote a simple set of bylaws. And we decided that the fundamental tenant of the project was: There is trust in the project, you have to trust the Core team, otherwise everything falls apart.

The bylaws were spartan and minimal: This is how you will elect a Core team, they’re what you will get, they are running the show. In hindsight, a procedure for making and updating the bylaws was needed. There are some problems with this—Core is too big. Initially we thought, oh yeah, we’ll have Core be big and anybody can act on their own. The redundancy will give people a chance to be involved around the world. But once Core was elected, they started saying everybody has to be involved in things. With everybody involved in stuff, it made decision making hard and slow, and that never really changed from Core to Core.

A lot of the problems that we set out to solve remained, and, in some ways, remain to this day. The bylaws are too hard to amend. We should be tweaking them from time to time and it’s impossible to change them at all.

I organized an election, and we went to BSDCon in San Francisco for the first time and that was where the new Core team was introduced. The old Core team was there, and we had a kind of handoff. This was also one of the first times I got to meet people face to face. I’d interacted with everybody online and by this point the project was five or 10 years old.

A lot of people have very strong interests, and there is a lot more commercial interest now. Maybe it’s time to get some new bylaws or amend the bylaws to make things better. Whenever it happens, it’ll be an interesting discussion. It’s not a matter of if it’ll happen, it’s when.

**TJ:** You’ve done stints on Core throughout the lifetime of the project now. How has the project changed during this time?

**WL:** In the earliest days it was, “Hey, see a problem, fix it” and it was a mixed bag. A lot of problems got fixed, a lot of things got done, but it was a very lone wolfish kind of thing. There were instances where one person was working on something and only they did it. But if two people were working in the same area, sometimes it worked okay and sometimes it led to a lot of conflict. Some people left the project because of the conflict and the strife around it, and from that perspective, the project wasn’t well served. Over time, a more cooperative attitude has developed.

The project has changed in the way we’ve adopted different tooling. People forget that FreeBSD was the first project that used Clang as its compiler. We were a very early adopter of Clang and that has served us well. We were way ahead of Linux early on, getting every-
thing going, building, and working with Clang. The project has been evolving from more of a, “yeah, just do it and be a cowboy” to “we’ll talk to people.”

One of the things that has developed in the last five or ten years is a culture of more extensive reviews. If you go back and look at the project, things were reviewed from the earliest days. We’ve gone from having maybe 5% of the commits reviewed to maybe half of the commits reviewed out, maybe a little more. That’s more of a collaborative process. It’s about growing relationships between people. You can’t just say, “hey, do reviews.” You need to cultivate a pool of reviewers. To get people to review your stuff, you have to review other people’s stuff.

This takes time to grow and develop and it’s been growing and developing with a newer intensity for the last five or ten years. Particularly since we introduced phabricator, which allows more people to submit things for review. Some things can be done better with it, and there are other things that it does terribly. For smaller changes that are almost ready, it is great. Larger changes can be more of a challenge.

We had some PR issues like those around WireGuard and so on because we hadn’t— as a project—developed a good culture of review. But since then, our review culture has gotten much better. It’s much easier to get reviews. People are developing additional tooling to make it easier to submit reviews and manage reviews.

We’re working on ways to do CI. We’ve done CI as a project for years now, but it’s after the fact, you commit stuff, and you find out what broke. We’re working on taking it from fix it after the fact to how do we fix it before the fact?

There are a number of challenges to using a large system to run all the tests as you normally would in the classic CI model. You compile it all, you run all the tests, you do it in all the environments, it takes 10 or 15 minutes, but hey, it’s worth it to keep things working.

If you ran all the tests and all the environments with all the build permutations, it would take centuries. So, you have to subset with FreeBSD somehow and that’s been a challenge for the project.

There are some efforts underway to try to make this better, try to smooth things out, try to improve the situation, and there have been some growing pains with that. The project throughout its life has been like that.

There are things that we’ve done poorly in the past that we do well now, and there are things that we’ve done poorly in the past that we know we need to change, and we are stumbling our way through them. Then there are a few things that we’ve gotten lucky with and never did poorly. We managed to come up with good solutions for early distribution things like CTM and CVSup and so forth that were the state of the art for the really crappy, dial-up world that the project landed in.

We’ve continued to update our tooling—sometimes quickly—sometimes less quickly.

Our move to Git was probably a little late, but one of the things the project likes to do is work on the code. People want to do new kernel things. They want to optimize the buffer cache for streaming, streaming it out in a more efficient way, or anticipating work so that
idle periods are better used, or work on some interesting problem with virtual nets or jails or something.

Nobody wants to do the boring nuts and bolts of source code management. There are a lot of people with very strong opinions, some of them very firmly held, and some are absolutely 100% sure they’re right—and they do zero work. And so instead of actually implementing it—they insist they did this thing in a past company, and they know it’s the one true way. Well, when you’ve got 10 people with 10 one true ways, it becomes hard to make progress.

That’s not listening to the unique needs of the project. And sure, there are a lot of things that are universal, but this project is kind of its own organic thing. And the more you tell people do it this one true way and hammer it in, the more people you lose.

If your question was about how the project has grown over the years, my answer is we’ve grown a lot. There have been the phases when we do something cool, people adopt it, people grow it, it becomes set in stone. And then people blow up the thing that’s set in stone and do something new or do a bunch of new things, and then have to blow up the thing that’s set in stone because it’s getting in the way. The project, I believe, has grown by being adaptable over time.

**TJ:** What is the lasting legacy of FreeBSD?

**WL:** FreeBSD has spurred innovation in a lot of areas. FreeBSD did a lot of innovative work in buffer cache design. We did it in public, in the open-source arena. We were one of the first, and then NetBSD came and did it a little bit better. And then we saw that, and we did ours a little bit better. The legacy isn’t necessarily a bunch of code. It’s the conversations and technologies that have spurred the development.

FreeBSD is a vibrant and thriving community that has used strife and conflict arising from the diversity of opinions to make the world better. And we’ve learned through strife and conflict how to make FreeBSD better.

At times, the strife and conflict have been at a cost higher than the benefits. At other times, the strife and conflict have been low and we weren’t innovating quickly enough. So, there’s a good middle ground that makes for a vibrant community rather than a stale community that just does the same thing forever, or, conversely, a community that fights so much, they can’t get anything done and breaks up.

One of the strengths of FreeBSD is that we’ve learned how to have the conversations, advocate for different positions, and then once there are current winners and losers, move on to the next fight.

**TOM JONES** is a FreeBSD committer interested in keeping the network stack fast.