I'm Not from Yorkshire, I Promise!

BY POUL HENNING-KAMP

My first draft of this article is filed away under the filename "Unix_from_Yorkshire.txt", because the longer it became, the more it read like a UNIX version of Monty Python's "Four Yorkshiremen" sketch:

Running an entire oil-company on a Zilog Z8001 16-bit CPU with 4MB RAM?

Installing UNIX from 103 single-density floppies, where number 92 was bad?

1200 bits per second modems?

No source code?

Binary patching of TZ rules?

Email addresses with exclamation marks in them?

X.25 instead of Internet?

But you try and tell the young people today that... and they won't believe ya'

And they probably would not read it either, because while such tales make decent entertainment when beer is quaffed, it makes for dull and monotonous reading in daylight.

But I guess I should present myself:

I'm phk@FreeBSD.org, my laptop "critter" has run FreeBSD-current for almost 30 years, and for more than a decade I committed to the FreeBSD src repository every 18 hours on average, only to disappear when some Norvegian Newspaper had HTTP performance problems.

How to Make UNIX Not Suck

I wound up in FreeBSD because UNIX truly sucked, because I was good at making it not suck as much and was determined to do so.

The UNIX I was used to, around 1990, was bastardized by some or other mini- or micro-computer vendor, who had jumped on the UNIX bandwagon, wanting to lure in new customers with promises of "portable UNIX", and then hell-bent on locking them in, with their own "enhancements".

As a rule, the vendor support was useless. In many cases they were newer to UNIX than their customers.

There was not one of the dozen or so UNIX machines I had been root on, where I had not been forced to disassemble some program or the kernel, in order to find out what was going on, and more often than not, to patch in binary, just to make things work.

I downloaded and tried 386BSD 0.0 when it was released in spring 1992, but it was too fragile to be useful for me. I did print out a lot of source code at work, to read on the train home.

In summer 1992, 386BSD 0.1 was released and was followed within days by 386BSD "0.1-newer", and that looked a lot more promising.

To give it a workout, I installed it on a discarded "UNISYS Professional UNIX Workstation", (A wildly overpriced 486 PC with a QIC-120 cartridge tape drive and a UNIX binary license, for a list price north of $10K at today's prices) hauled it into the PBX room and set it to work logging and accounting traffic.

As a harbinger of things to come, my first `comp.unix.bsd` posting two weeks after the release was about disk-drive geometry trouble, due to the infamous 1024-cylinder limitation.

In September, my first patch was the nineteenth and final patch in the first installment of Terry Lambert's "unofficial 386BSD patchkit":

> patch00019
> PATCH: CLEAN UP SLIP INTERFACE TO KEEP FROM HANGING
> AUTHOR: Poul-Henning Kamp (p...@data.fls.dk)
> DESCRIPTION:
>
> Here is a patch to clean up the interface between the tty-drivers, in particular the com driver, and the sl# interfaces, this is not a work-around but a genuine bug-fix.
>
> Symptoms: after a number of "com#: silo overflow" SLIP ceases to work.
>
> Overview of the problem: the slip interface will disregard any notice from the tty-driver on problems (parity errors, framing errors or overruns), which basicly means the one might as well throw the packet away right away. Also overrun in the packetizing will go relatively unnotized.

As it says in the title: I'm not from Yorkshire. I'm from the unfashionable and poor southwestern corner of Sjælland, the largest island in Denmark.

But in 1992, I had TCP/IP connectivity on my home-PC.

I called the call-back security-box at work, waited for it to call back, pressed the button on my V.22bis modem, logged into the 386BSD machine at work, started SLIP there, started SLIP on my PC, and I was on the Internet via my work's connection to the Danish Unix User's Group's network "DKNet".

Life was good at 200 bytes per second, but then things got weird.

Bill Jolitz, who had ported the NET/2 sources and added what it took to make it run on a i386, did not respond to the many people who offered him help and patches, the promised 0.2 version never materialized, and the patchkit grew and grew, topping out at 138 "official unofficial" patches, 3 1/2 of which were mine.

In spring 1993, Chris G. Demetriou et al. gave up waiting for Bill Joliz and founded the "NetBSD" project, starting a new CVS-tree consisting of 386BSD+patchkit.

Then, three months later, 30 years ago, Jordan Hubbard et al. did the exact same thing, under the name FreeBSD.

Why?

I have no idea: I was on the other side of the planet, and, at that time, a very minor contributor.

To me NetBSD seemed to be more idealistic: They wanted a very portable, pure BSD UNIX, running on all sorts of hardware: HP, Sun, Digital, IBM PCs, most of which I had neither access to nor an interest in. The NetBSD crew also seemed very academic and more concerned with doing things "right" than getting them to work soon.

In all fairness: They have very competently done exactly what they set out to do, so a big hat-tip from here to our marginally older sibling.

FreeBSD focused exclusively on the i386 PC platform, which is what I had access to, so I gravitated towards the FreeBSD project where people also seemed to be "production-oriented" like myself.

In late November 1993, FreeBSD 1.0 was released, and I had enough patches in it to be listed under "Additional FreeBSD Contributors", and I even received a free courtesy CD-ROM from Walnut Creek CD-ROM some weeks later.

On February 26, 1994, almost precisely 29 years ago as I write this, Jordan gave me the 21st commit-bit in the FreeBSD project.

Your Mission, Should You Choose to Accept It

According to the CVS log Jordan did this: "… so that he can help me with the ports cleanup," and as the 1.1.5.1-tree CVS commit-log as my witness: I did make two commits to ports before I nonchalantly sauntered into src and committed a bunch of my time-nuttery patches to the kernel and NTPD.

Next, I committed an unsanitary concoction of shell and Tcl scripts that could munge a GCC 2.5.8 compiler source tree into the form expected by the FreeBSD Makefiles in case anybody else wanted to upgrade from GCC 1.39.

At the same time, I was negotiating a new job as on-site SW-engineer for a very large document imaging system for "Giro" post-cheques, to be delivered to the Danish Post Bank on a very aggressive schedule by "TRW Financial Services" from Oakland, CA.

Julian Elischer, who I knew from 386BSD & FreeBSD, worked there, and got himself a referral bonus by pointing the "DGB" project my way. That probably made me the second person to land a job because of FreeBSD, Jordan being the first, working at Walnut Creek CD-ROM.

I landed the job and landed in California where I happened to settle into an apartment in Pleasant Hill, just a few miles from Jordan and Walnut Creek CDROM. That soon came in handy.

Nominally, I was in the USA for training, but I soon found myself responsible for installing Solaris and 3rd party software on five presumably performant Sun Sparc 1000 servers with a literal ton of disks, configuring a couple of Cisco 7010 routers, an ATM switch made by Fore Systems, and dozens of ethernets to the 135 PCs running NT.

The USL-BSD lawsuit settled out of court in February 1994, with a 3-month grace period which also applied to FreeBSD 1.0.

So, we had to start over, and did, as soon as we could get the "4.4 BSD(Lite)" release from UCB, and then we had to reintegrate the "unencumbered" bits and pieces of FreeBSD 1 as best we could, just to get feature parity with FreeBSD 1.0.

Rod Grimes did most of that work and, as I remember it, he had the new CVS tree ready for us sometime in May 1994.

Not long after, I was inducted into the core team and almost immediately appointed release-engineer for 2.0, together with Jordan.

In FreeBSD, the releng@ is the designated pilot responsible for rowing the locomotive onto firm ground, and Jordan and I set a precedent that almost unlimited powers come with the job.

As release engineers, we had to write the code to build the release, pack the release into media-agnostic distribution files, the code to start the installer, the installer itself, the code to partition the hard disk, the code to extract the distribution files from all sorts of media and network services, and the code to do the magic boot-block swizzle stuff.

But we only had to do that once. Subsequent releases were a breeze by comparison, and it took over a decade before somebody finally had had enough and wrote **`bsdinstall`**.

Jordan wrote almost all of the **`sysinstall`** program, and it lived up to our design goal of asking all questions up front instead of sometime later when the answer was needed--like Windows or Solaris did, much to everybody's annoyance. What's the point of installing from a CD-ROM or tape, if you have to babysit it all the time?

However, the disk-partitioning editor fell to me to write, causing several people to argue that I should be prevented, by injunction, if necessary, from ever writing user-interface code again.

In FreeBSD, the releng@ is the designated pilot responsible for rowing the locomotive onto firm ground, and Jordan and I set a precedent that almost unlimited powers come with the job.

See the `make world` with Releng Cruises

One thing that took a lot of time and commits was "sterilizing" the release process so that no "dirty laundry" leaked from the system on which the release was built into the distribution files of the release.

For instance, a fair number of Makefiles would have:

```
CFLAGS+= -I/sys
```

instead of the correct:

```
CFLAGS+= -I${.CURDIR}/../../sys
```

We did not quite get to reproducible builds, because back then it was surprisingly common to compile transient information such as `__DATE__`, `__TIME__` and similar things into programs.

Three weeks before the release, I slammed GCC 2.6.1 into the tree, hoping it would solve some of the many issues we had, with a commit message (fe7dee47009525) making it clear that this was not up for discussion. GCC 2.6.2 came out a week before the release, I rushed the changes in, and FreeBSD-2.0 came out with a non-crummy C-compiler.

ITAR dual-use export regulations forbade export of cryptographic software without an explicit license from the US government, and we had neither the time, the money, nor the lawyers to even contemplate that, and we were unlikely to get permission

to export the sources, and certainly not in a relevant timeframe.

The futility of this regulatory scheme was obvious: A copy of the relevant source files were in South Africa, from where Mark Murray made them available via anonymous FTP, and anybody in the whole world, including the USA, could freely fetch those files and use them, because the USA did not forbid import, only export.

Personally, I would just have ignored ITAR, pointing to the global availability if trouble came later, but Jordan vetoed that because, while the FreeBSD project might get away with that, Walnut Creek could not put "contraband" on their CD-ROMs if they wanted to sell them abroad—or in case anybody else did.

The single place where this really mattered was the `crypt(3)` function which is used to scramble passwords for storage: In traditional UNIX, it was derived from the DES encryption algorithm and thus the source code could certainly not be exported.

Somebody, and I am not entirely sure who, had written an ersatz `crypt(3)` function during FreeBSD-1 days, but it really was no good, and we were certainly not going to ship it in 2.0. Since nobody else had done anything, I did, two weeks before the 2.0 release.

They produced a new batch of CD-ROMs with a "January 1995" date, and distinguished the cover by giving Beastie the bright green sneakers, he has proudly worn ever since in FreeBSD-context.

I had previous experience with brute-forcing traditional UNIX passwords. In one job I found that the company policy of "letters and digits" made everybody use their license plate as a password. In a later job, I found out that a lot of people at Olivetti's research center in Ivrea used "Gloria" as a password, and having met her, I did not blame them.

First, I had to do some experiments to see what I could get away with. How much code assumed the length of the scrambled password? Up to 80 characters was not a problem. What characters could you get away with using? Pretty much all the visible ones. And so on.

I made hacking passwords a lot harder. I made the "salt" much bigger to thwart precomputed dictionaries. I designed the algorithm to take much more CPU time, 34 milliseconds on a 60 MHz Pentium, and to be very hard to implement in FPGAs.

To mark the new kind of passwords, I picked the prefix `$1$`, committed the code with a suitably scary commit message—and continued to the next item on the still far too long TODO list.

The resulting code, far from perfect, became known as "md-5crypt". Because it used the already RFC-published MD5 cryptographic hash function, we were free to export it, source and all, and we never got into any trouble.

BSD, BSD, BSD, GNU, BSD and Beer

That is: If you ignore the license.

There had just been one of the periodic GNU vs. BSD license mail storms and just to tick off everybody equally, I revived my old "beer-ware" license and slapped that on md5crypt before I committed it.

I have never fully grokked why md5crypt was copied all over the place, but it was. All sorts of FOSS imported it, from Perl to Apache. A version with the serial number filed off appeared in GNU libc. Somebody wrote md5crypt in COBOL. Somebody produced a JavaScript version, including the underlying MD5, using some kind of C->JS transpiler.

Closed source also embraced it: Macromedia Flash used it. Cisco put it in IOS, and eventually, even Microsoft supported `$1$` passwords.

But as it spread, it also landed in front of lawyers doing "due diligence" before some FOSS-based company or other got bought, and some lawyers had a really hard time figuring out precisely who owed who how much beer.

IBM bought Whistle Communications, makers of a FreeBSD-based communication appliance. A polite junior IBM lawyer called me, asked for permission to record the call, apologized for taking my time, and ran me through the Very Serious Legal Questions prepared by the Senior Lawyers. We both giggled a lot.

But I also received a registered letter from Very Serious Senior Lawyer demanding that I reply "forthwith", in writing and notarized, to the following questions: Was I the sole author of the md5crypt function? Did I have full legal control of the copyright to offer licenses? Was I offering the beer-ware license to $BigCorp? If so, would it be the specific division of $BigCorp, all of $BigCorp, $BigCorp's distributors and agents, or was it the end-users of $BigCorp products who owed me beer? And if so, how much beer did they owe me, how was it to be delivered and who would be responsible for customs, levies, and fees thereon? How did the amount of beer owed relate to the amount of use and/or perceived usefulness of the software to the legal entity in question? Could other alcoholic beverages be substituted if beer was not available in the relevant jurisdiction? Were people unable to send me alcoholic beverages, for legal or any other reason, prevented from using my software? He had clearly thought a lot about it. I did not reply.

Interestingly, not one single lawyer ever asked me if I were willing to make their job easier by offering md5crypt under a standard FOSS license. I guess that is not how lawyers roll?

Finally on the 22nd of November 1994, we released FreeBSD 2.0 right in time for the dot-com boom already in progress.

Walnut Creek CD-ROM pushed out FreeBSD 2.0 on CD-ROM, aiming for December 1994, missed the fabrication deadline, edited the date to "January 1994", produced the CD-ROMs, discovered the mistake, produced a new batch of CD-ROMs with a "January 1995" date, and distinguished the cover by giving Beastie the bright green sneakers, he has proudly worn ever since in FreeBSD-context.

After some more releases, I got out of releng@ on good behavior, but Jordan was stuck with release engineering as long as he worked at Walnut Creek, because FreeBSD became a major product for them.

Critter Always Runs -current

I continued hacking FreeBSD in my copious spare time and bought a Gateway Handbook/486 so that I could spend the daily "BART" commute productively. That was the first "critter" and the best laptop I ever had, but the name was passed on to the dozen or so laptops I have had ever since, all of them running FreeBSD-current.

In 1995, RAM prices had gone through the roof because everybody bought PCs, and with only 4MB in "critter", I noticed that FreeBSD was not using RAM too efficiently. I dived into it and found a spot where the "VM revolution" had passed UNIX by.

The traditional implementation of `malloc(3)` can be found on pages 173-177 in the K&R Old Testament, and it works great on a pre-VM system where entire processes are swapped in and out. But with virtual memory, keeping the metadata about free chunks of memory in those chunks of free memory means you might page in a lot of unused pages, just to free even more memory.

Somebody between K&R and FreeBSD had "solved" that by not really bothering with reusing free memory, until the kernel refused to hand out more with `sbrk(2)`.

After some hacking around, partly inspired by a very old computer's filesystem, I came up with `phkmalloc` which was one of the first truly page-organized `malloc(3)` implementations, and it made my tiny laptop noticeably faster.

One side effect of the design was that several classes of wrong use, notably double frees and freeing modified pointers, could be deterministically detected before they created any havoc, so I added several debugging flags to `phkmalloc`, where `A` meant `abort(2) on any trouble`, `J` meant `fill with Junk` and so on.

When I set `AJ` globally on "critter", `fsck(8)` dumped core on the next reboot, and so did far too much other code over the coming years, including `inetd(8)`, `cvs(1)`, `getpwent(3)`, `BIND` and `OpenSSH`.

The Great Wizard Convention of 1998

Summer 1998 was peak-dot-com-mania, and thanks to sponsors swimming in virtual money, we were able to gather the entire core team physically, for the first and possibly only time ever, at the USENIX Annual Technical Conference in New Orleans.

Bruce Evans from Australia resisted the idea for a long time, eventually revealing that he was in fact almost totally deaf, but he was persuaded to come anyway and got at least one Wayne's World style "We're not worthy! We're not worthy!" greeting.

Bruce died a couple of years ago, but he was already a legend back then. Andrew Tanenbaum had thanked him for Minix386, Linus Torvalds had thanked him in the original Linux announcement, and we in FreeBSD were eternally thankful for his countless code reviews, patient explanations, and wry humor. May he rest in peace.

For good measure, I had proposed a talk about `phkmalloc` and it was accepted for the "FreeNix" track, which were clearly meant as the kid's side-show to the grown-up's serious UNIX conference.

I am chronically unable to remember names and faces, but I instantly recognized Kirk McKusick's trademark mustache in the front row. His picture had been over numerous articles in USENIX and European Unix Users Group's magazine over the years.

I had never mingled with UNIX nobility before, and I decided to skip foil 20, where Kirk's `fsck(8)` program topped the list of buggy software `phkmalloc` had spotted so far. But when I got there, I was so high on adrenaline, I just plowed through, until solid laughter from both the first row and elsewhere in the room stopped me. Kirk was not the only person in audience with code on my list.

All the "victims" I talked with were nice about it, and thought it was perfectly fair to show that list to document that this was a real and present problem.

Kirk turned out to be a particularly cool guy and I would get to work on UFS2 with him a few years later. At Kirk's and Eric Allman's "secret wizard party" I learned that UNIX nobility knows how to party. If the punch had been poisoned that evening, you would all be running Windows NT today.

For me personally, the highlight of the trip to New Orleans was running into Dennis Ritchie at the breakfast buffet, and talking UNIX history, timekeeping, device nodes and networking with him, for so long that we missed the first talk of the morning. Getting his blessing on what I did with DEVFS meant everything to me.

I hope someday,
there will be a definitive book
about the history of FreeBSD,
written by a
competent historian.

Disks, Partitions, Buffers and VNODEs

The `sysinstall` disk editor had revealed to me that the interface between filesystems and storage devices was fairly kludged together. For instance, the disk partioning was implemented in the individual device drivers. Hard disks mostly did, but ramdisks, floppies and optical media did not.

If you look at pre-VM UNIX, it's all neat and sorted, but the way virtual memory was added on top, made me coin the slightly derogative term "phd-ware". A pile of proof-of-concept source code dropped on the doorstep, like an orphan at the monastery before sunrise.

When I started, we had three instances of phdware in the storage-domain: Heidemann's extensible VNODEs, Seltzer's LFS, and Kirk's FFS/UFS. To his credit, Kirk came back later, and he maintains FFS/UFS to this day.

At work, I wrangled a bass-ackwards disk layout. Veritas Volume Manager forced us make 9 slices per disk (for performance) and then mirror the slices pairwise across disks (for reliability), before finally concatenating the mirrored slices into ten partitions for our image storage filesystems.

It would have been so much more sensible and much easier to setup, manage and operate if we could have mirrored the disks pairwise first, slice those mirrors, and then concatenate to partitions, but Veritas could not do that.

To make matters even worse, we had five servers but only four disk cabinets, three disks per SCSI-bus, but two SCSI-busses per power-supply.

Within a year of starting production, that particular architectural "limitation" in Veritas was the root cause that wiped out half a million images of financial transaction documents.

Fortunately, they were still available for rescanning, but production was delayed for most of a week.

That incident, more than anything else, caused me to desire and design a Lego™ style volume management system, without any kind of silly restrictions.

But before I could implement it, there were a lot of weeds to whack, which I did, and so much so that "Danish axes" became a standing joke on the core team for some years. I like to think of it as retiring technical debt before that became cool.

I cannot claim to have worked from a master kernel architecture blueprint, it was more a long sequence of "well, that could be smarter…," in many cases circling back over the same parts of the kernel multiple times to implement another set of improvements.

Finally, the core parts of GEOM were developed in 2002 on a DARPA contract under the "CHATS" research program, thanks to the young and ambitious Robert Watson, who also coauthored the paper about jails with me.

Later, I again found voids in my one-man-company's calendar for the second half of 2004 and decided to see if "community funding" could keep my FreeBSD-habit going. That succeeded beyond all expectations, manifesting in 530 commits that cleaned up ttys, linedisciplines, filedescriptors, VNODE-switch, VFS-switch and more.

Core's Dirty Laundry

But I also had another role in the project—I was on the first core team.

The core team was the nominal governing board of the project, but there were no formal definitions of any of those things, we had no bylaws, and we had three huge internal handicaps:

First, we were global, and the culture differences were very real.

In Denmark, we joke that if three people wait more than 5 minutes for a bus, they will have formed an association by the time it arrives. In other countries, forming associations and organizations were very formal events, involving lawyers, notaries with stamps, filing fees, or downright governmental preapproval.

Second, and much worse, we were all primarily very good coders, a demographic rightly not famous for interpersonal skills, and we were all used to being the smartest person in the room, building and probably also zip code. Take it from me, lynx are no good at cat herding.

Third, the FreeBSD core team grew organically on the general principle of "he does good work, pull him in!" But out of some kind of misguided respect, we never retired anybody, even if they had wandered off and we hadn't heard from them for years.

Eventually we peaked at 18 nominal members, of which much less than half could be expected to cast a vote when it came down to that, often causing the losing side to argue that no quorum was reached.

And with these three handicaps, the core team had work to do, most importantly, handing out commit bits, something we should never take lightly. In a few ugly cases, it involved taking them away again, when strongly encouraging the misfitting or misbehaving person to volunteer it back had failed.

If You Can Keep It…

A particular "personnel-issue" brought the legitimacy crisis of core@ to a breaking point, and while various minor patches were proposed, they would not, and could not, have fixed the fundamental problem: Core's unwillingness and inability to make unpleasant, but necessary decisions.

Eventually, I proposed that we should simply let the committers elect a new core team, but that idea did not appeal to people from countries with a history of less than fair elections. It would only lead to "politics", "campaigning", and "corruption" they extrapolated.

I suspect the argument which finally sold them on my idea was when I said that I would not be a candidate if an election were held, but otherwise I would cling to my core bit to the bitter end, or until they pried it away from me, in bright daylight and in public.

This would not be a hard promise for me to fulfill, as my personal life was quite chaotic. My (then) wife's mental illness had forced me to become self-employed and work from home to take care of our two toddlers.

One way or another: I eventually got my election after we had agreed to a very minimal set of bylaws for FreeBSD.

At the BSD Conference in Monterey in autumn 2000, the first core team election result was officially announced, a competent 9-person core team was elected, and the "press-release" email contained:

> Departing Core Team member Poul-Henning Kamp said, "I'm very proud of what we have done together in the Core Team over the last 8 years. The new Core and the fact that they are elected by the committers, means that the project will be much more responsive to change in the future."

In the past 24 years, the committers have taken their job as voters seriously, we are on our 12th duly elected core team, none of which have been any worse than we were on "core.0" and many of them have been obviously better.

To me, democracy was my most important contribution to FreeBSD.

And That's Enough of That…

I think I managed to avoid a copyright claim for planking "Four Yorkshiremen" in this second attempt, but it still feels fundamentally wrong to me, because there is so much more to FreeBSD's history than the haphazard selection of the egocentric memories I have laid out above.

The number of people I have mentioned by name is the least I could get away with, because there are literally far too many who deserve to be mentioned to fit into an article of this length.

I hope someday, there will be a definitive book about the history of FreeBSD, written by a competent historian, with no or little beef of his own in the pot, where everybody who contributed, including myself, gets their fair bit of attention.

Until them, to all FreeBSD Friends, past, present, and future:

Thanks for all and keep caring!

/phk

POUL HENNING-KAMP is phk@FreeBSD.org, his laptop "critter" has run FreeBSD-current for almost 30 years, and for more than a decade he has committed to the FreeBSD src repository every 18 hours on average, only to disappear when some Norvegian Newspaper had HTTP performance problems.