13 Years of Linux

by Tim Bird

Introduction

It has been 13 years since I was first introduced to Linux. When you saw the title of the talk, you probably assumed that I was going to talk about the entire history of Linux, and that I got my math wrong. But I'm just going to talk about my history with Linux. Linux was introduced to the world by Linus Torvalds in 1991, and has been kicking around for almost 15 years now. However, I got a bit of a late start on Linux, and, as you will learn in my talk, I have often to work on the periphery of Linux development, rather than at the center. Although I haven't been a core community developer during the last 13 years, I did see a lot of interesting things happen during this time.

I have some observations I'd like to make about Linux, the open source method, the importance of standards, the community effect... just ramblings on a lot of topics, really. This whole conference is arguably **my** show, and, gee, look, I've made myself the keynote speaker. How arrogant is that? However, that's my prerogative as the organizer of the show. But making myself the keynote speaker is really not as self-serving as it sounds. I'd much rather be sitting in the audience right now listening to someone else talk. But it turns out that I'm a terrible delegator, and my first choice for the real keynoter couldn't make it.

And so, hear I am. And here are some stories of my experiences over the last 13 years with Linux. Hopefully, some of my stories will remind you of the reasons you got into Linux development, or help you avoid some of the errors I made along the way, and point the way to using Linux more effectively in your products in the future.

Novell

It all began when I was working at Novell in 1993. I had done some perfunctory work on Unix at Novell as part of their "Portable NetWare" project. In the process I had gotten somewhat familiar with the Unix guys at Novell. These were guys who hung out on the Internet, using programs like gopher and archie. I remember very clearly the day that one of them, a guy by the name of Bryan Sparks, the leader of the Portable NetWare project, came to my office very excitedly, and talked about a new system of using the Internet, called the "world wide web". He described how you could view a document from all the way around the world, just by clicking on a link. I told him we could do that with

NetWare, and I just didn't get too excited about it. He talked about the ability to synthesize information, in real time, in response to a request. I told him I didn't get the point. Why would you “make up” data at the time of the request? Retrieving existing data, I understood. Or fulfilling a query by selecting or formatting data. But making it up on every request? How would this scale? And what good would that be?

This was not my most visionary moment as an engineer.

Bryan was also into other weird stuff, like operating systems. He had been playing with NextStep and Mach, and, although I didn't know it at the time, he also dabbled a bit in a new OS called Linux.

Secret Project X

Despite my early skepticism, within a few months, I found myself agreeing to join a super-secret project being initiated by Bryan at Novell. As is customary for such things, we chose a codename for the project. We called it "Secret Project X". We secured office space at an off site location, in order to keep our project secret even from other Novell employees. This was all very cool cloak and dagger stuff, and I was pretty excited at the prospect. We were planning on creating for Novell a desktop operating system based on Linux, which incorporated a world wide web browser as part of the graphical desktop of the system. You have to remember the circumstances of the time. I can remember when the world wide web reached 50 servers. Microsoft was a punky little company that had yet to release Windows 95, which, in my not-so-humble-opinion, is when the tables started to turn towards them gaining an OS monopoly. In these days (1993), the desktop was still fair game for new OSes, and companies like QuarterDeck, Digital Research, and HP were trying to build their own graphical systems on top of DOS.

In any event, it was still possible for Novell, or Apple, or some other enterprising company to achieve something in the desktop market, and Bryan had convinced Novell that they could do something, using Linux and somehow involving the world wide web.

Just as an aside, on my first official day on the brand new, super-secret Novell project, I was driving to the new off site secret office, and I happened to be stopped at a traffic light next to another Novell employee who I knew. While we were stopped there, he saw me and we exchanged greetings. He commented on getting to work, and I replied that I wouldn't see him there, because I had business elsewhere that day. Then he said "Oh, you must be going to the new off site location for that secret project Novell just launched", and then he drove away. So much for corporate secrecy.

First encounters with Linux

In any event, we got started on the project, and quickly found out that Linux was truly a huge pain in the neck. I remember installing Linux from 45 floppies, from a distribution (back when the term hadn't even been invented yet) called SLS (I think this stood for Soft Landing Systems). Things were very delicate in those days, and whenever someone successfully got Linux up and running (especially with X, with correct modelines), there was much rejoicing. I remember spending hours fiddling with modelines to get X working, every time I installed Linux on a new machine. Thankfully, those days are over.

At the beginning of the project, we sat down as a team and had lots of discussions about the best OS to use as the base for the project. Linux really was immature at this point, and at one point we seriously considered switching the project to BSD. However, one key observation we made was that although BSD was more mature at the moment, it was clear that the pace of development with Linux was much better. There seemed to be much more activity and development effort being expended on Linux. We projected that soon, Linux would surpass BSD in terms of supported hardware, development tools, robustness, and several other dimensions of quality that we were concerned about in building our product. With this trend, we decided to stay with Linux and see what happened. It is clear, in hindsight, that this was the correct decision.

DOSEMU

On Secret Project X, I was involved with a few sub projects related to compatibility with other operating systems. I had extensive experience with DOS, and at the time compatibility with DOS programs was critically important. Therefore, one of the projects I worked on was DOSEMU. This was the DOS emulator for Linux. DOSEMU ran using the vm86 instruction mode of the 386 processor. Some of the things that I contributed to the project were support for the DOS NetWare client, some debugging and instrumentation capabilities, and a feature to dynamically map drive letters from inside DOSEMU back to the Linux host file system.

Cross-platform APIs

Another project I worked on involved supporting an API abstraction layer that allowed someone to write a single piece of software, and compile and run it in Windows, Macintosh, and Linux environments. One way that Novell planned to deal with the application compatibility problem (at least going forward) was to change the rules, and create an API that would support all platforms simultaneously. This was source-level compatibility, as opposed to binary-level compatibility as is now offered by Java with its JVMs and class libraries, but that's arguably a better way to go anyway. In a lot of ways, this project was way ahead of its time.

Almost hiring Linus

I should add here a little-known tidbit of Linux history. Novell actually offered Linus Torvalds a job in 1994. We flew this young Finnish college kid to Provo, Utah, and offered him the chance to work on Linux full time, and be paid for it. (I can't remember whether we were going to require him to move to the U.S. or not, but I do remember when he came for the visit, we took him to lunch, and he did NOT like US root beer. Apparently no one else in the whole world likes root beer. Go figure.)

But in any event, Linus turned us down. His reasons for doing so showed great wisdom, and what turned out to be a certain degree of clairvoyance regarding the future of Linux. The main reason he gave us was that he wanted to stay independent of any single company, to avoid Linux becoming the product of that company. I'm pretty sure he turned down a position at RedHat later on, for the same reason.

I personally think that this was a very good decision, and one which ultimately did preserve Linux' ability to maintain a steady stream of contributions from diverse developers and companies. Linus has often had somewhat strange rationale for doing things, but over time he has shown an uncanny knack for choosing the right thing. (Except for not putting a debugger in the kernel. I mean, come on!)

Linus was right for another reason as well, however. When we made him the offer of a secure job at a large, profitable and well-funded company, we honestly believed that this would be a long term deal. But he declined, and it didn't turn out that way.

The project dies

Within a year of working on the desktop operating system, Novell decided it was not really interested in the project, and canceled it. It really was too bad. After only a short time (a little over a year) we had a well-working Linux system with a graphical desktop, our own web browser called "Ferret", and many other great improvements. We helped add NetWare protocols to Linux, and had rounded out the product in many ways too numerous to mention. But alas, Novell could not see the potential. Novell didn't think that either Linux or the World Wide Web were going to amount to much. (hmm.)

At this point, some of the project members departed and went to pursue other endeavors at Novell. I went back to programming the NetWare client for DOS. However, Bryan did not want to see the idea die, and he decided to seek venture capital to start a company outside of Novell to continue the work. He got it, from the ex-president of Novell himself, Ray Noorda. Six months after I had left the project, Bryan convinced me to leave my secure position at Novell, and join him in the wild and dangerous world of startups. In the spring of 1994, Caldera was born. I was employee number 9.

Caldera

Now the mere mention of the word "Caldera" may cause some of you to recoil in horror. Yes, this is the Caldera that eventually turned into SCO, and became the bane of the Linux industry with their legal suit against IBM and others over alleged misappropriation and introduction into Linux of proprietary Unix System V code. I won't comment on the merits of that case here, other than to say that from what I can tell, SCO appears to be using some very hard drugs. No evidence has shown up in the case yet, despite it's having dragged on for 3 years now.

(Notice that I didn't say "no solid evidence", I said “no evidence”. It's true. After 3 years of litigation, there's really no evidence whatsoever that IBM did anything wrong.)

I want to assure you that the company that filed that lawsuit bears no resemblance to the happy and community-friendly company that I worked for after leaving Novell.

Novell didn't believe in Linux, part 2

As an aside, some time later I was offered the possibility of returning to Novell to pursue some special projects there. I was conversing with Drew Majors, one Novell's founders, and CTO at the time. I told him that I planned to continue

working with Linux to see what would become of it. I can remember very distinctly, in the Lobby of building "C" at the Novell campus, Drew Majors saying to me, "Tim, do you really think the world needs another OS?". I think this was 1995. In a way, of course, Drew Majors was right. The world didn't need another new OS. But Linux was more than just another new OS, it was a new way of developing an OS, and I believe that this was and is what the industry needs to move forward in certain ways.

I occasionally wonder what would have happened in the industry if Novell had really put its backing behind the project we started. Our little team consisted of 15 people, and, with the help of open source, we had made a very respectable product. I can't help but be amused as I look at the industry approximately 12 years later, and see that almost all other OSes have died or dwindled. But Linux has continued to improve and increase. And Linux is now one of Novell's core business products (if not it's most important product). They could have had a head start on the rest of the industry. (But let's not cry over spilt milk – That's an English expression which means it's no use worrying about mistakes made in the past. But it also doesn't mean we shouldn't learn from those past mistakes.)

But I suspect also that had Novell actually built a respectable desktop product using Linux, they still might not have embraced the open source movement. The whole concept of open source was pretty new to most people back then, so it's hard to say what they would have done. But following the open source methodology has been critical to Linux' success. As I read about what Novell is doing now, I think they understand this, but back then I think they would not have.

Caldera focuses on business

Anyway, back to Caldera. While I was there, Caldera really focused hard on making Linux acceptable to business users. To some degree, we shifted our focus away from the desktop, and tried to get into the vertical application server market. We saw the success of Novell with its Value Added Reseller program, and thought that building a channel like that would be the best way to get product into the hands of business users. We focused a lot of effort on adding commercial polish to the installation, and to recruiting vertical application vendors to get their products ported to Linux. We also directly developed software ourselves, in an effort to plug gaps that we saw in the total Linux offering. Often times, unfortunately, we tried to plug the holes we saw with commercial software. I personally worked on the Looking Glass desktop. (I know that “Looking Glass” is something you've never heard of - that's a point I'll get to later). I also did the port of the Netscape commercial web server to Linux, and worked with several 3rd party vendors to put together a catalog of offerings (including WordPerfect for Linux). By the time we were done, we were very proud of the 3rd party alliances we had built, and the exclusive deals we had

made, which positioned us as the premier Linux distribution, in terms of commercial functionality and features. We spent a lot of time and effort marketing Linux the traditional proprietary way - we attended trade shows, we recruited VARs, and we advertised in traditional IT magazines.

Unfortunately, we had learned the wrong lesson from Novell. It is true that Novell was resoundingly successful in the 80s and early 90s, due in large part to its VAR channel. However, one key to Novell's success was that in the early days, NetWare kind of crept in the back door of organizations. Instead of going into the organization through the board room, through the IT department and purchasing and what-have-you, NetWare had this way of just showing up inside individual departments. It was cheap enough, and very capable, and just solved the local filesharing and printing problems that departments had. Ironically, Novell became a raging success against the wishes and in spite of the best efforts of most IT managers to keep it out of their organizations. Linux was about to follow the same path.

RedHat invades

A few years after Caldera began, we started to notice things going horribly wrong. After a LOT of effort and expenditure on Caldera's part, we saw that Linux was making its way into lots of departments, the same way NetWare had. The only problem was that it wasn't OUR Linux. Increasingly, we saw that an upstart distribution called RedHat, which was released freely on the Internet, was being installed at businesses. There were numerous times when we went to a company, convinced them at the IT level that Linux was mature and safe enough to use, and then when the company looked around internally, they found that someone in the company was already running RedHat.

Within a year, despite relatively meager revenues, RedHat had gone public, and dominated the market for Linux distributions. There were two key factors that led to this result: First, the RedHat distribution had been freely available, and had established a level of grass-roots networking and support that was difficult to match through commercial efforts. RedHat not only leveraged the open source community in developing their product, but also in supporting and extending it. The second important factor in RedHat's success versus Caldera was that the open source community had come out with reasonable software to fill the gaps in the Linux offering. KDE came out as a great desktop system. Apache became the de-facto web server of the Internet. And even the browser was lost as an exclusive offering by Caldera, when Netscape decided to open source Netscape Navigator. This product became Mozilla, and then over time morphed into FireFox. Almost all of the differentiation factors that Caldera pursued were eliminated over time. RedHat, with its grass roots invasion, had cemented the loyalty of early adopters, and the open source community had made the offering good enough.

An important lesson I learned from this is that you need to be careful with the software that you want to use to differentiate your product. Unless it truly is something remarkable, you may find that it has been replicated, well enough, in open source. I have consistently been surprised; both by what open source can produce, and by what it doesn't seem able to produce. Finding the right mixture of non-differentiating open source and differentiating proprietary software will always be an interesting problem. And note that you will have to deal with this problem whether you use open source or not, because if your competitors are using open source, they'll benefit from the output of the community whether you do or not, and things you thought you could use for differentiation will be lost to you eventually.

Caldera was doomed as a pure Linux vendor (which was unfortunate, because there really were some great people there, and we had some highly exciting times.) But my own involvement with Linux now took a twist. As part of Caldera's effort to support DOS programs, it had not only worked on DOSEMU, but also acquired rights to Novell DOS (formerly DR-DOS by Digital Research). For those of you who remember, DR-DOS was a very strong competitor to MS-DOS, and for a time had Microsoft running scared about losing their market share. (There's a whole other story about how Microsoft cheated in the market to combat DRDOS, and how Caldera finally called them on it with an anti-trust lawsuit. I was involved in that lawsuit, and learned some pretty interesting things about the importance of network effects in business as well as in open source software. I won't go into the details of the result of the lawsuit, but I will say that when the case was eventually settled out of court, everyone involved on our side took a VERY nice and expensive vacation.)

Enter DOS

Novell DOS was originally acquired by Caldera to provide "out-of-the-box" DOS program support, which no other Linux distribution had. Our DOS support included our own bootable DOS image, which was another of those key differentiators that Caldera pursued. In any event, we discovered after acquiring DOS (and hiring some of the original DR-DOS developers), that DR-DOS had been re-targeted at the embedded market. x86 processors had commoditized to the point where it was now feasible to deploy them in fixed-purpose devices, and DOS was just the sort of small, boot-the-machine-and-then-let-you-touch-the-metal OS that was needed.

You may laugh now, but I bet you didn't realize that for a period of time DOS held the top market share position for the OS for Karaoke machines.

Some of us at Caldera saw an opportunity to branch out, so in 1997 part of the

people there took DOS and Linux and formed a new company, which eventually became Lineo. This is where I got my introduction to embedded Linux.

Lineo

Lineo was a VERY interesting company to work for. While there, I learned a lot about Linux, and a lot about business – and a lot about how NOT to run a business. Lineo built itself up very quickly by acquiring lots of small Linux companies. Lineo raised a LOT of venture capital. It was easy in those days, after the RedHat IPO and the VALinux IPO. Some Linux companies profited very well as part of the general dotcom boom. People were actually offering Lineo venture capital money that we turned down. So, Lineo had enough capital to make some significant progress with Linux.

Lineo made some fairly big business mistakes, but that's a different subject. Get with me sometime during the conference, and I can tell you some pretty interesting stories about how Lineo spent its money.

The important thing about my experience at Lineo was my getting involved with Embedded Linux.

One of our first projects with Linux was trying to build a radio-based router. We really had no clue what we were doing, but we muddled through and figured out how to cut the kernel size, and how to configure the protocol stacks. After a few embedded projects, it became clear that we had some significant problems to overcome, in applying Linux in the embedded space. Some of the problems were technical and some were marketing problems.

Some of the technical problems were, maybe surprisingly and maybe not, some of the same ones we have today. At Lineo, we once again embarked on our strategy of identifying the gaps in the (this time Embedded) Linux offering, and trying to fill those gaps. However, instead of fill the gaps with our own engineering, using proprietary software, we filled the gaps by acquiring companies, and using open source.

Lineo Acquisitions

We acquired a company called Zentropix which sold a realtime system for Linux. They had started with RT-Linux, and at the time we purchased them were selling an x86-based dual-kernel realtime system called RTAI. We also acquired a company called RT-control, which had the people who created a version of Linux which could run on processors without an MMU, which was called uClinux. We began serious work on a small C library and program to offer a small utility set.

One of Lineo's employees, Eric Anderson took over work on uClibc and on busybox. Neither of these projects was created by Lineo, but we carried these open source projects forward, as primary maintainers, in a significant way. I think it's safe to say that at busybox is one of the most important pieces of software in the embedded Linux market, and Lineo had a big hand in making it what it is today.

Also, Lineo acquired an embedded contracting company in Japan (formerly USE), in order to establish an embedded Linux engineering and sales team in that country. And we acquired a company called Moreton Bay, which built VPN routers using Linux. There were other companies, but they weren't as interesting as the companies I just mentioned.

Technical problems that needed to be solved

If you look at what those companies brought to the table, you see that Lineo was focused on solving several key problems:

- real time
- small flash and RAM footprint
- networking and networking security issues
- customizations for vertical markets

It should be noted that although the US operations of Lineo eventually went away, many parts of Lineo still exist today. In particular, the developers and products of Moreton Bay eventually spun back out and became a company called SnapGear, which continues to develop and maintain uClinux. And the Japanese group of Lineo separated out and became Lineo Solutions. They continue working on embedded Linux products and solutions in Japan. I should mention that the last vestiges of what was the original Lineo were acquired by Metrowerks, a division of Motorola Semiconductor Division, which has now became Freescale. Lineo alumni are scattered far and wide, and many are still involved with embedded Linux. And with the industry being something of a small world, a number of them are here at the conference today. I look forward to reminiscing with them.

During and after acquiring these companies, Lineo also focused on building its own distribution of Linux called Embedix, and we worked hard on tools, and on Linux standards. (Here, I'd like to take a moment to comment on the name “Embedix”. That was my idea for a name for the product and when I came up with it I was quite proud. (I'm still proud.) I still think it's a great name for an embedded Linux product. I should really check to see if anyone's still using the name and see if I can reclaim it. OK – sorry – that was just a moment of nostalgia.)

Marketing problems

Besides these technical problems, Lineo also had to deal with some interesting marketing problems. One of the marketing problems was the weird license used with Linux – the GPL. Almost everyone had big concerns about the GPL because it was new and not well-understood. Large companies, in particular, were wary of trying something new legally. Everyone in the industry (e.g. MontaVista, RedHat and Lineo), spent a good deal of time trying to educate companies on the dangers and non-dangers of the GPL. At Lineo, we even tried to deal with the licensing issue programmatically.

License

One thing I worked on at Lineo was a license verification tool. This tool analyzed the licenses you were using on your product. Among other things, it analyzed your distribution packages, and your kernel source tree, and it conducted surveys of your engineers, and made recommendations for avoiding problems with the GPL. It was a truly revolutionary product. Can I tell you something, though? - Nobody cared. At least nobody cared enough to actually pay for it. For a while there, everyone was upset by the GPL, and its terms. However, nowadays most experienced developers know where the boundaries are, and we just work within them or outside them. If you don't do stuff in the gray area, then GPL is not really very scary at all. The Free Software Foundation has proven to be pretty reasonable in enforcing the license. (Note, however, that now the FSF has pretty much gone off the deep end with the the GPL 3.0 draft - but that's a different story).

New market

Another big marketing problem in those days was just the fact that Linux was a brand new product in an older, well-established market. Linux really did lack a certain amount of maturity in the embedded market. It was hard back then to get people interested in Linux. To some degree, we were ahead of our time.

One aspect of this was that Linux just didn't fit the custom-OS model. At the time Lineo entered the market, 50% of embedded projects were done with roll-

your-own operating systems. I can only imagine all the scratching and clawing that was required by WindRiver and others just to get to that point. Think about that for a minute. 50% of all embedded projects were completed with in-house OSes!

But we kept telling people that this ratio was simply NOT going to hold. The same arguments we used then are still applicable today. I can still repeat them easily today, because I said them so often when I was CTO of Lineo: As hardware capabilities (in memory, speed, etc.) increase, and particularly as software requirements and complexity grow, there will be more and more need for a general purpose OS. To deal with modern product requirements, you are going to need something with virtual memory, memory protection, tuned, mature network stacks, support for the latest disk, network, and bus standards, and lots of drivers. No company can keep up with this type of onslaught. And really, with the advent of Linux, why bother?

On using a general-purpose OS in embedded

There was often the issue with customers about whether a general purpose OS was really needed for their product. I remember trying to sell Linux for a VPN router project to a large communications company in the US. The customer we were trying to sell our product to said "you can't expect me to believe that a general purpose OS like Linux is going to perform as well as a custom-built OS." Well, actually I did expect him to believe it. I believed it myself. While the custom-built OS has some advantages for performance (no memory protection, shorter code paths), the general-purpose OS has advantages in other areas (e.g. memory protection and resource tracking lead to ease of development and robustness). Also, this particular general-purpose OS, Linux, has been openly tested and refined by thousands of people. I'm pretty sure that the custom OS probably just ripped off the BSD networking stack (which everyone seemed to do, including Microsoft). Linux also ripped off the BSD networking stack, but then really focused on it and continued to improve it. (Note that BSD's stack also improved at the same time).

At the time, hashed sockets had just been added to Linux's networking stack. Also, there were experiments with zero-copy transfers. With those features, and Linux's greater memory-handling capabilities, it was perfectly reasonable to believable that Linux could outperform even an OS designed specifically for communications. (At the time, we even had some benchmarks showing that Linux was AT LEAST as fast, and possibly faster than the competitor for this project.)

Today, most new networking research is done on Linux. Have you seen Van Jacobson's new work to migrate portions of the networking stack out of the kernel? This is very hot stuff, and will be very interesting to see develop in the next few years. This type of stuff isn't happening on other OSes, because Linux

is more open to researchers, and is constantly being improved. You can't compare the ability of an in-house team, or even the ability of a large, talented team at a major proprietary software vendor, with the ability of almost ALL the world's network researchers. So yes, a general purpose OS can be better.

Standards

One other thing we tried to do at Lineo, besides tackle technical and marketing issues, was to try to build an ecosystem for embedded Linux, via standards.

In 1999, Lineo got together with Motorola and tried to create a standards group called EMLAB ("Embedded Linux Advisory Board") I think we chose this awkward name mainly because the acronym was available as a domain name - emlab.org.(see http://lwn.net/1999/features/ESC/ELadvocacy.php3). But we didn't originally include MontaVista or RedHat. We did invite them soon after the forum was created, but only after we had set up some of the infrastructure for the group, and they, feeling snubbed by this, were very cautious about participating. But Lineo was sincere about wanting to create an organization to promote embedded Linux, and about inviting other vendors to participate. After this initial false start, Rick Lehrbaum, the founder of linuxdevices.com, helped all of us to jointly create the Embedded Linux Consortium (ELC). You may have noticed that “E” “L” “C” is the acronym we're using for this conference. There's no conflict, because the ELC-consortium died and went out of business last year. I believe it served a great role in getting the message out about the viability of Linux as an embedded OS during its first years (2000-2001), and in that regard, as a promotional agent for embedded Linux, it was a success. However, ELC never accomplished a second, and what I consider critical, technical mission, to standardize certain aspects of embedded Linux.

Although it was easier to avoid fragmenting the kernel, due to the kernel being monolithic and available under GPL, several other parts of the total Linux software system were NOT standardized, and this caused real problems in creating 3rd party middleware and application software for embedded Linux.

The ELC only every produced one specification, and the specification covered an area (the C library) that had already been specified in great detail, and was mostly not fragmented already. The truly revolutionary thing that ELC should have tried to accomplish was standardizing the hard stuff, where Linux WAS fragmented. But ELC was mostly a vendor driven consortium. That is, the board was heavily dominated by companies that were interested in selling Linux, or Linux-compatible OSes. So the ELC basically went about trying to set up a standard to keep the C API from fragmenting. This seems like a reasonable thing if you're coming from outside Linux, and are faced with making a compatible API for programs to run on both Linux and some other non-Linux OS. However, that was not what developers and companies inside the Linux space needed. They needed to 1) fix basic technical problems with Linux, and 2) tame the areas that

really WERE fragmented, and these existed mostly in user space middleware.

I can remember at one of the ELC meetings, a developer from a JVM company stood up at the meeting and said, in essence: “ If you don't standardize the graphic APIS for Linux, Microsoft will be laughing all the way to the bank.” That request, issued several years ago, still haunts me, because I believe there's an element of truth to it. But the ELC never addressed this, and both the ELC and Lineo are no more (for different reasons).

Sony

Which leads me to my work at Sony. After the demise of Lineo, I started doing independent consulting. Sony had been doing a lot with Linux, and had actually started working with Panasonic on some joint projects to develop fixes for some low-level Linux problems. These projects included things like irq thread preemption, tickless systems, special memory management based on the memory type, and optimizing and parallelizing the boot process. Sony hired me to evaluate the status of the Linux industry, to determine if the time was right to build a greater group of collaborating companies, in a formal consortium.

I gave them my analysis (which was “yes”), they hired me, and the CE Linux Forum was formed. In my current job at Sony, I work on CELF stuff, doing things like building the test lab and organizing this conference. I also work to maintain a Linux kernel that is used by some Sony teams in their products. I can't talk too much about that latter thing, so I'll say a few words about the forum.

And now, CELF

This talk is not primarily about the CE Linux Forum, although I can talk about the forum for hours if given the chance. Just ask anyone who was at the Plenary meeting yesterday. But I would like to say a few things about how CELF is operating, and what I see for the future.

CELF has been operating for 3 years now, and we've had our ups and our downs. For those of you not familiar with the forum, we have 8 technical working groups, and last year we hosted or significantly participated in about 9 conferences. We have extensive online resources, on our wiki, that are intended to be an aid to anyone working on embedded Linux. We've produced a lot of good research, and we've published or exposed a lot of good patches. CELF member companies have been active in developing Linux features and technologies to address the technical needs of our products, but the forum itself has struggled in a few areas. For example, we've had difficulty getting some of our features back into mainline. But we've hired a community-experienced

developer, Matt Mackall, who's been helping us with this recently.

Overall, I think CELF is headed in the right direction. Our conferences in Japan have turned into very successful events.

Benefiting from the forum

At a company level, Sony puts a lot of effort into the forum, but we also get a lot out. As an example of this return on investment, I'd like to comment on my work as the Chair of the Bootup Time working group. When I volunteered for that position, about 2 and a half years ago, it was basically because no one else raised their hand. I worked in that capacity for 2 years, and in that time there was a lot of information shared, some patches created, and some instrumentation systems explored. I now find myself to the be the primary maintainer of a system called KFT – Kernel Function Trace. It turns out that this system is useful for lots of other debugging and investigation tasks in the kernel. I gave a talk at OLS 2 years ago, and in the process collected a lot of the different techniques and materials into a paper. I learned more about boot up time writing that paper, than I ever would have as a passive observer of the Working Group.

I am benefiting in a similar manner as Chair of the Architecture Group. I see lots of different developments by member companies that a lot of other members of the forum, and members of the open source community at large, aren't aware of. I'm in a unique position to see all the great work that is going on, and to benefit from it. That's one of the reasons I am optimistic about the future of the forum, despite the appearance in some areas that things are going slowly. I think you'll see today and tomorrow that there's a ton of great stuff going on.

One last final thought, unrelated to the rest of my talk:

Penguins... Why did it have to be penguins?

Over the years I have accumulated a lot of penguin junk. While I was at Caldera, I wrote a book, and was forced by marketing considerations to put a penguin (TUX) on it. Here at the conference, we are using a modified TUX on the signs and programs. I have accumulated dozens of T-shirts, many with penguins on them. It's not just a career, it's a wardrobe.

Just to get some use out of them, I've brought a small sample of my penguin collection to show you today. I've had people remark that my stuffed penguin actually scared them. This is what the marketing department at Macmillan publishing came up with to promote my book. It's an attractive penguin that is ALSO a hat. (put hat on). If there's a more dignified way to promote a book, I don't know what it is. I have my penguin clock, my penguin superball. I was going to bring my penguin Christmas lights, but they're packed away and I didn't

want to get them down. When will the madness end? Must we have so many penguins?

Actually, you should know that the whole "penguin as a mascot" thing is a stroke of marketing genius by Linus. I honestly don't know whether he consciously did this or not, but using a simple motif, which can be altered and customized for specific uses, is a great marketing aid. I can show you a penguin with headsets and a vague, hard-to-identify device (is it a PDA, a remote control, an MP3 player, or a 1200 baud modem?). You still know, despite the customization, that this will be something to do with Linux. So there – Linus is a marketing genius as well as a programming God.

By the way, as an added bonus at this conference, we are NOT giving you a T-shirt with a penguin on it.

So in summary, here are some lessons I've learned over the years working with Linux: (They are my only slides):

Lessons:

Lesson 1 - Go with the software which has the activity and effort.

Pace is more important than position.

Lesson 2 – You can go far, if you leverage open source.

Your manager will think it's you making all that progress.

Lesson 3 - Don't bet against open source software.

Things will commoditize over time, even the stuff you write.

Lesson 4 - Go where you see everyone else is

It's more fun there anyway.

Lesson 5 - The license is NOT that big of a deal

Now just tell that to your lawyers.

Lesson 6 – General-purpose OSes rule!

Or, they will shortly.

Lesson 7 – You get out of something what you put into it.

Lesson 8 - Learn to accept penguins

(penguin slap animation)

CELF is working

I'm still a bit haunted by the statement made in ELC years ago, about the need

for a standard graphical API for embedded Linux. We still don't have such a standard today. But we're getting closer.

The solution, of course, is to create some standards. But for some things, creating a standard is hard. But CELF can nudge things along.

I am convinced that the existing technical problems with Linux will be addressed. I also absolutely believe that the level of non-differentiating software will slowly but surely rise in our products. This will be a good thing, because it will free up resources to work on more interesting stuff that our customers want.

And I believe CELF will be part of making this happen. We can do a lot if we work together. That's why I'm here today, and I hope it's why you're here as well.

Welcome to the conference

So I say welcome to the CE Linux Forum's Embedded Linux Conference.

I have just a couple of messages on logistical things. Please be advised that each session is supposed to end 10 minutes before the next session begins. The session end times are not printed in the program, but hopefully this is a really simple rule to follow. If questions for a particular session run over the allotted session time, please take the questions into the hall. There's a central congregating area in the middle of the building, where you can continue to talk to the presenter about his talk. This is encouraged – that's why we all came here.

The demos will start at 5:30 tonight, in the San Jose room. We'll also be having a reception in the main area. The reception will be absolutely fabulous. There will be plenty of GOOD food and drink. We'll even have an ice sculpture of... a penguin...

Thanks for coming, and Have fun!