Written Interview for Kernel Engineer

Takahiro AKASHI
takahiro.akashi@linaro.org
akax.hiro@gmail.com

Engineering Experience

What kinds of software projects have you worked on before? Which operating systems, development environments, languages, databases? That is not a checklist, just some suggestions of what to describe you have worked with.

In the last ~20 years, I have worked for the Semiconductor group/company. (The same organisation, but it has been spined off from Fujitsu in the middle.) For the last 10 years, I have been working for Linaro as a Linux engineer dispatched by the company.

On various Arm/Arm64-based SOC's,

- Network performance (latency) analysis on TSN (Time Sensitive Networking) under VM environment (Linux, kvm)
- Prototyping of hypervisor-agnostic virtio backend server framework (Zephyr+Linux, Xen)
- Trusted Boot/Feature enhancement for UEFI support on U-Boot
- Feature enhancements for Arm64 support (Linux)
- Development of BSP (Board Support Package) for Arm-based ASIC boards (Linux)

On FR-V (Fujitus's proprietary 32-bit microprocessor),

- Multimedia Codec solution, Development of IPC library (uITRON)

Before moving to the semiconductor group, I have worked for the Computer system group in Fujitsu, which is one of the largest computer company in Japan. My role here was a kind of researcher or software architect for pre-production technologies.

On PRIMEPOWER (Fujitsu's SPARC server),

- Prototyping of microkernel-based fault-tolerant system (Chorus, Unix SVR4)
- Porting of microkernel to SPARC server (Chorus, Unix SVR4)
- Database tunings/Development of Distributed Shared Disk for clustering (Unix SVR4, Oracle)

In my early days, I had been dispatched to Unix System Laboratories (NJ, USA) by the company and worked on

- Multicore enhancement for UNIX SVR4 (file systems, libraries)

Throughout my career, I have been a "C" programmer with the background of Unix/Linux operating system.

Would you describe yourself as a high quality coder? Why?

As far as my deliverable works in OSS contributions are concerned, yes, I think so since I always follow the community's standard and design disciplines in particular upstream projects to make the code (patches) best adapted and accepted yet fulfilling development's goals. There are many records as I mentioned above.

Overall, I'm quite adaptive to any type of development works and good at implementing expected features timely and precisely based on good understandings of either hardware requirements, standard specifications or existing software stack in layered frameworks.

When I used to be engaged in my company's projects, our engineers were guided to conform to the legacy "water-fall" model of software development (even for Linux BSPs), producing a bunch of detailed design documents and exercising extensive test scenarios aiming to 100% code coverage.

So I was initially puzzled with the way of development & reviews within OSS community when I was dispatched to Linaro. While I have no longer followed the legacy method, I still try to rule myself by practicing as many different test scenarios as the time allows locally/manually, though I don't perform an exhaustive analysis of code coverage.

Would you describe yourself as an architect of resilient software? If so, why, and in which sorts of applications?

My knowledge and experiences in early days of my career were focused on old platforms/technologies and won't be directly applied to the modern cloud systems (either HW or SW). In this sense, I don't see my self as an "architect" in this field, but would say that I hold a technical sympathy with backgrounds of this kind.

When I was in Unix server division, the organisation's aim was to promote their Unix server lines as a replacement to mainframe machines. So our team in research & planning section started some preliminary studies on, what is called, fault-tolerant systems, most popular ones were Tandem Computer and Stratus. So I have some knowledge about their architectures, at least, at concept level.

A few years later, our section started another software research project, aiming to provide "fault-tolerant" Unix OS based on micro-kernel technology. Most of prototyping works were done by our research lab but I got involved in the design and review phase. In this system, for instance, the file system server is executed in a pair of actors (or processes in Linux/Unix jargon) and the primary one creates "checkpoint" data at any critical point and synchronously transfer them to the secondary actor. Once the primary actor fails due to HW or SW, the secondary actor immediately takes over the role and "replays" the transaction as it always keeps track of the same state in the ex-active one.

This checkpoint-recovery mechanism is quite heavy to implement and didn't perform well as expected. In contrast, today's cloud environment take an opposite approach, in which the system are composed of clustered servers with a load-balancer to provide high-availability in total. They are loosely coupled over "stateless" protocols and a transaction may fail, but we can retry it later after switching over to another server.
That's it. Simple.

What software products have you yourself lead which shipped many releases to multiple customers? What was your role?

I have never developed software "products" in a strictly-commercial sense. When it comes to software which was released to customers, I developed Linux BSPs (Base Software Package) to support a couple of generations of ASIC evaluation boards in my current semiconductor company. They were provided to customers as “as-is” reference code in a form of diff patches rather than in binary packages, though.

In these projects, I played a role of a leader of Linux team whose main responsibility was to manage the project progress and mentor younger engineers who had little experiences with Linux (and device drivers). I was also in charge of bringing up boards and developing some device drivers.

Since those were some of our first experiences in developing BSP in-house, I struggled to establish make-shift integration rules and workflow, which were quite basic and primitive, mimicking merge process in upstream projects, without automated CI/CD.

What is your most senior role in a software engineering organisation? Describe your span of control, and the diversity of products, functions and teams you led.

As summarised above, my most senior role is a tech lead of Linux team in our ASIC division. I played on it for 3+ years. Since I was dispatched to Linaro, I have never been titled with any positional role.

What is your proudest success as an engineering leader?

While I'm not convinced that I can name it a success, the experience as a tech lead mentioned above is undeniably the turning point in my career.

It was the time that I moved from Computer system group to Semiconductor group. During several years before and after this transition, I was not a developer in any sense and struggled to find out my own competency as a professional engineer even though I had had experiences in research works in the past.

Getting a leader position at the end makes me confident that this is my job that I have been looking for. Dealing with open sources, in particular linux kernel, allows me to leverage my old (even obsolete) skills and knowledge in OS fields and promises me many opportunities for being engaged in cutting-edge technologies and projects in years to come.

After I was dispatched to Linaro, I strengthened this confidence as I can now feel that I'm contributing to software evolution and am a recognised engineer or in part as an expert in some area.

Outline your thoughts on open source software development. What is important to get right in open source projects? What open source projects have you worked on? Have you been an open source maintainer, on which projects, and what was your role?

To be patient and be a good neighbour in the community. I'm serious.

Different people may have different opinions, and sometimes oppose to you with offensive comments. Otherwise, most people are friendly and yet lean to their own ideas and approaches that are slightly conflicting to my approach. So having good discussions with other engineers and listening to their voices would be crucial especially when you try to push your patches upstream. This might be even true in in-house development, but I always try to keep this attitude in my mind due to the nature of community's diversity.

95% of my contributions to OSS belong to either Linux kernel or U-Boot, while the others are derivative patches to related projects, for instance, some utilities (kexec-tools), libraries (Xen library) and so on.

I don't work as a maintainer in any projects now, but intend to be a good reviewer in projects that I used to be actively involved in, notably UEFI subsystem on U-Boot in these days.

How comprehensive would you say your knowledge of a Linux distribution is, from the kernel up? How familiar are you with low-level system architecture, runtimes and Linux distro packaging? How have you gained this knowledge?

I'm not sure what "low-level" system architecture and runtimes means, but as a developer on Arm platforms, I'm using and working on Arm's standard boot sequence, say, TF-A (Trusted Firmware for Coretex A), U-Boot (as BL33) and linux based on device trees.

While I'm not specifically an expert of TF-A, I have a good understanding about how it works. As I mentioned in the questions above, I have been working on Arm port of linux kernel as well as UEFI system in U-Boot and have good working knowledge for implementing additional features on these existing frameworks.

I have several years of experiences in board bring-ups/hardware enablement as well as device driver development for in-house Linux BSP releases. (although Arm linux at that time was a bit old-fashioned.)
Regarding distro packaging, our Linux BSP projects adopted buildroot and later yocto, but other than that, I have no specialised skills on packaging. I am simply a user of dpkg/apt commands in my daily development work.

If I add one comment, I'm not quite good at x86 architecture and ACPI things since I favour simple, clean and modernised (i.e. less-constrained) architectures.

Describe any experience you have with low-level embedded systems engineering, on Linux or other embedded operating systems

I have been engaged in Linaro's internal "dependable boot" and later "trusted substrate" project, particularly UEFI subsystem on U-Boot, as mentioned in my first answer. The project goal is to establish all the software stack for securely booting the Arm-based IoT/edge devices.
I used to be one of most active developers of UEFI subsystem for 3+ years. Features I implemented include secure boot and capsule (firmware) update mechanism and I contributed to improving and refactoring the code in many ways. Even now, I am still engaged in U-Boot community and try to review UEFI-related patches and follow up any issues/bugs.
(Not sure, though, that those are what you ask in this question.)

When I worked for FR-V (Microprocessor) division, I was also engaged in multimedia/Codec solution projects as mentioned in my first answer. The system was equipped with multicore 32-bit processors plus an integrated hardware accelerator as a low-power and parallel solution. In one project, I designed and developed IPC (inter-processor communication) framework on RTOS (uITRON) for Codec processing.

Beside this assignment as a support developer to help stuff engineers for this specific task, my main role was a product planning/marketing for the product lines. So this is not a engineering thing.

Outline your thoughts on quality in software development. What practices are most effective to drive improvements in quality?

It depends and varies on the definition of quality, the criteria of required quality and the scale/complexity of the software stack and of course, the cost. (I don't assume that you have, say, functional safety things in mind.)

In general, however, there is nothing special I think.
Simply follow basic steps:

- Document, at least, external interfaces or semantics
- Review the code with many eyes
- Run extensive tests either in unit tests or integration tests based on the documented behaviours (test-first?)
- It might be a help to use static/dynamic analysis tools like coverity or valgrind. (Not sure how effective they are.)

Although I'm not specifically attracted by agile programming style (with bottom-up design), I practice iterated developments, hoping to prevent any trivial mistakes.

Once released,

- Get as many feedbacks from users as possible to improve
- Run CI/CD regularly for continuous changes/updates

Describe your experience with public cloud based operations

I have no or little experience in managing a system on public cloud as I have been working in literally embedded segments for the last 20 years.

Recently, I started to learn about cloud operations by attending some seminars by AWS. This is not for acquiring my operational skill, but rather for learning about the basic in cloud system developments since edge computing, connectivity and cloud-native DevOps are becoming key words for embedded solutions (like automotive).

Outline your thoughts on documentation in large software projects. What practices should teams follow? What are great examples of open source docs?

I'm not sure how large you assume are the projects, but if a couple of engineers get involved in the development of one software, I would prefer to define layers and modules (software stacks) and ask

them to document the interfaces between the components to avoid any possible mismatches which may lead to potential bugs and re-engineerings. Documentation is also helpful for other test engineers to create black box tests to ensure that the software conforms to the design.

Apart from design docs, I believe it important to create test specification documents in details. Those documents should be thoroughly reviewed by external engineers to assess the coverage as well as validity of verification.

Outline your thoughts on performance in software engineering. How do you ensure that your product is fast?

As a whole, I tend to dedicate most of my efforts in creating clean and well-structured code rather than performance-centric tricky code. This is because I'd value the stability and maintainability of the code. If there is any real concern on performance, I will write a benchmark app and take on the performance analysis, then try to refactor and tune the code.

What matters in performance tunings is that we are the users of the software or at least understand concrete use cases for that software. Otherwise, the tuning work can easily end up with endless efforts.

Outline your thoughts on security in software engineering. How do you lead your engineers to improve their security posture and awareness?

When we want to develop a security-critical software with high privilege, we should start with considering the security thread model under concrete use cases to identify possible attack surfaces. Techniques used here, either in formal or informal way, may vary from project to project.

In my experience (UEFI secure boot on U-Boot), we tried to review the design and patches from a couple of aspects in our internal discussions:

- confirm that U-Boot itself is safely started by the previous stage (TF-A, for chain of trust)
- restrict user's interactions to prevent unauthorised system operations (disabling some commands)
- put critical data in secure storage and limit the malicious accesses (public key/certificates are handled by secure world code (OP-TEE))
- keep the firmware updated (of course, FW update is another interesting topic.)

I think it is important to learn and share those perspectives in design and review processes in the team.

I'm not sure that we should be very conscious of security in normal software project. Even in privileged software, however, we may create features which are not directly linked to privileged operations. In such a case, I would encourage developers to perform some code verification by using static analysis tools to mitigate any potential vulnerability in program logic. Fortunately, for U-Boot repository, coverity is occasionally exercised by the top maintainer and the results are circulated and reviewed during release candidate intervals.

The Linux kernel is heavily entwined with the Git DVCS, can you describe your level of Git experience?

Git is always my primary tool for version control in day-to-day developments for the last 13+ years or so.

As an upstream developer, I make heavy use of git from cloning the existing repositories, checking out a specific version of code, looking at the history of commits to maintaining my developed code, including creating branches/tags for checkpoints and (possible) reverts, rebasing/modifying the existing commits, merging patches from others, and preparing patches for upstream. Much less frequently bisecting commits for spotting a malfunctioned patch.
Anything else?

Since I'm not a maintainer, I don't have many chances to deal with pull requests and patchworks in merge operations.

Have you worked in real-time systems? In Linux or other RTOS and if the latter which one(s)? Can you elaborate on some of the challenges real-time programming faces, and your experiences with them?

It may not be a 'real-time' feature, but I was engaged in a system using Zephyr. In this task, I developed a PoC (Proof of Concept) of virtio-based block device server on top of Zephyr in guest virtual machine (Xen). The aim was to create a prototype framework for a hypervisor-agnostic virtio backend implementation and evaluate the feasibility for future work.

I have another experience of using real-time linux. In this study, the goal was to assess TSN (Time Sensitive Network) performance under virtual machine environment. The efforts include evaluating network latency between remote guests and analyse the potential bottle neck imposed by the hypervisor (kvm in this study).

One of insights that I've observed is that, even though linux kernel is monolithic, there are a couple of kernel threads involved in processing network packets from a physical port to a virtual guest port (a tap device). We need to deal with coordinating and prioritising those threads in the system in order to achieve some level of determinism in latency.

Please describe any experience you have working in operating system kernel internals, Linux or other.

For linux, I implemented ftrace and kexec/kdump for arm64 and additionally contributed to audit, seccomp, kgdb, and livepatch support for arm64 as well. Those were part of efforts by Linaro to boost arm64 ecosystem by filling the disparity of functions that used to exist between x86 and arm64 architectures. (in 5+ years)

When I was in Unix server division, I have worked on microkernel (known as Chorus OS) based projects. (not sure, but in 3+ years)

In the first project, my team (a group of three?) ported the OS to our SPARC server for demonstration purpose to the product line management. The task was not quite difficult as Chorus provided us the base SPARC port and we already had our commercial version of SVR4 on the server running. In the second project, I have been engaged in prototyping a "fault-tolerant" operating system in collaboration with our research laboratory. (I mentioned this in my third answer.) Unfortunately, the output from those projects had never been productised.

In addition, in my early days, I was involved in multicore enhancement project for UNIX SVR4. This was a collaboration work under the industry alliance, named Unix International (UI), to promote one variant of UNIX OS (SVR4 ES/MP). My role was to modify the existing code of UNIX file system and some user libraries to make them re-entrant and multicore-safe. (in one and half years)

How extensive is your experience with C? What type of software components have you used it in? What about C++? Why would you use one over the other?

I am always a C programmer, either in product-grade development or research work, throughout my career. Almost of all my works in C are low-level software, operating systems (kernel/drivers), boot loaders (TF-A/U-Boot/EDK2) and related utilities.
No other choice than C.

I have no experience of C++ in developing product-grade software. I have working knowledge of basic syntax to some extent so that I can understand existing code.

How extensive is your experience of Python software engineering? How do you test Python applications? Outline the applications that you have led in Python, and your takeaways from that experience.

I have no experience in writing Python programs from the scratch, but rather am a user, particularly, of pytest when creating test scripts in U-Boot development.

Describe any experience you have with Rust

I am still in early stage of learning and aiming to get future jobs partly since Rust support for modules

was integrated to recent Linux and partly because Linaro has already organised a virtio-related project in which there may be some chances for me to work on tasks using Rust-vmm (virtual machine management framework) library. Nothing decided yet.

I have self-learned the basic syntax with "Rust Programming Language". My next experiment was to convert a C program (~700 lines of code) I wrote as a U-Boot utility to Rust. I will have to take more practices, especially on the usage of 'unsafe' feature for low-level programming with Rust.

Have you ever worked directly with ACPI and/or DeviceTree? Can you elaborate?

If you mean some kind of contributions to ACPI or DeviceTree project, no.

I always play with a device tree as part of my development tasks on linux and TF-A/U-Boot which are to be configured with a device tree. I haven't worked on server-specific, that is Linux+ACPI, projects in the past.

What kind of packaging and container formats, such as debian or snap, have you created and/or maintained? Can you describe your experiences and challenges? This is actually working with the format, not experience as a consumer.

I have no experience in working with/maintaining packages or containers.

Education

In high school, how did you rank competitively in maths and hard sciences? Which was your strongest?

I remember that I was at the top, either for math, physics and chemistry, for the term-wise exam's. I would say math is my strongest.

In high school, how did you rank competitively in languages and the arts? Which was your strongest?

I remember that I was among the top group (~10%) for Japanese(-classic) and Chinese-classic. I was at the top for Foreign language (English) and reading & composition in English was one of my strongest subjects.

While I enjoyed arts class (in my first grade), I don't remember the score.

What sort of high school student were you? Outside of class, what were your interests and hobbies? What would your high school peers remember you for, if we asked them?

I literally devoted my high school days to preparing for university admissions tests. Most of all my school mates remember this as the result (ranking) of each term's exam was published. The only fun to enjoy outside of class was to play Shogi games (Japanese chess).

Which university and degree did you choose? What others did you consider, and why did you select that one?

I have a master degree of Information Engineering from Tokyo University, and a bachelor degree of Electrical Engineering from Tokyo University, I didn't consider any other universities to attend since I believe that Tokyo University is recognised as the highest ranked educational institute.

At university, did you do particularly well at any area of your degree?

To be honest, I don't remember.

Overall, what was your degree result and how did that reflect on your ability?

I don't remember my degree result (or score?). If there is something that might affect my careers, it must not be "information theory" or mathematical computer algorithms that I learned in the class.

Please note that, during my bachelor degree, I learned the programming in FORTRAN and Pascal on a green-screen dumb terminal (vt100) which was connected to a mainframe machine located in the university's computer centre.

When I was in a laboratory for master degree, some innovative technologies and computers fascinated me and convinced me of the arrival of new age of computers.

On software side, I was very much impressed when I first read K&R's "C Programming Language" in Japanese edition and knew about Unix operating system through the book, furthermore when I first learned object-oriented models with elegant Smalltalk on Lisa.
On hardware side, I was so excited when I had chances to play with Macintosh (classic) and Sun's workstation (both running on 680x0 at that time).

Such experiences with new types of computers propelled my desire of developing such modern products, and then I entered the computer company, Fujitsu.

So I think most of my practical skills and abilities on computers were acquired after I entered the company.

In high school and university, what did you achieve that was exceptional?

I was enthusiastic with bodybuilding in my university days and am proud of having a title in one college-level contest.
This may not be what you want to ask in this question, though.

What leadership roles did you take on during your education?

I have nothing special to mention.

Context

Outline your thoughts on the mission of Canonical. What is it about the company's purpose and goals which is most appealing to you? What do you see as risky or unappealing?

I enjoy and am satisfied with Ubuntu (Desktop) as a platform for all kind of my development tasks in embedded fields even though Canonical's core business and its success focuses on enterprise IT systems in cloud.

I believe that it is Canonical's strength point that Ubuntu commits to OSS and offers its distribution in a long term, covering a range of IT segments from embedded (IoT/edge) devices, desktop to servers/distributed systems. This strategy, I suppose, drives and convinces knowledgeable users of Ubuntu to easily adopt Ubuntu in successive systems/projects as well, especially in the modern era where everything is connected to the internet.

That said, I have been sometimes bewildered with Canonical's ambitious and yet a-bit-too bold projects, like Ubuntu phone and Mir for the desktop.

In addition, I would expect more visible contributions to Arm-specific portion of software in the ecosystem by Canonical, while I know that Ubuntu has been supporting Arm/Arm64 architecture since early days.
(I simply wonder why Canonical left Linaro years ago.)

Who are Canonical's key competitors, and how should Canonical set about winning?

I can easily image RedHat and SUSE.
After a quick search, I found that Canonical is much smaller than RedHat by an order of tens either in annual revenues or numbers of employees. (To be frank, I haven't been aware of this fact until now.)

I'm not sure that this is a problem to Canonical or whether Canonical wants to catch up with the gap eagerly because it might have its own business strategy. Speaking to Japanese market, although I'm never an expert in business, I think that Canonical's support business is not so popular than RedHat as they have been leading the market and a dominant name. Japanese conservative people may prefer a major bland (in my own opinion, of course). I guess that Canonical should seek for and attract more local resellers and channels to approach customers.

I have no specific idea from a technical perspective for "winning".

From my backgrounds, I hold my interest in embedded segments but no working knowledge about what a role Canonical wants to play to enhance its presence there by leveraging Ubuntu Core. If this job interview process goes well and I have a chance to talk to some of Canonical managers, I'd be willing to hear more in details on Canonical's strategy in this market.

Why do you most want to work for Canonical?

My sole desire, as I am supposed to retire my company at the latest in next March, is to continue to get involved in open source communities and contribute to OSS development, in particular Linux kernel, as a professional engineer.

I believe that Canonical offers many opportunities to allow and even encourage me to do so as Canonical has a good balance of missions; Canonical holds strong and long-term commitments to key OSS's, while it retains strong relationships with influential solution partners and customers to accelerate technologies.

In addition to technical aspects, I love Canonical's work environment or "remote-first" work ethic with which Canonical hire engineers globally and even advocates that people be able to (fully?) work from home.

Since the first pandemic started three years ago, I have stayed at home, yet working as an assignee of Linaro. This experience convinces me that remote-work style should best fit to open source software development and give me the maximum flexibility as well as the satisfaction with my own performance from the viewpoint of work-and-life balance at my age.

Last but not least, Canonical is yet a UK-based company despite of its distributed nature. This is another point appealing to me.

What would you most want to change about Canonical?

I don't know at this point. I would figure out things to hunt down once I become part of Canonical and get deeply engaged in some project. Anyhow, hopefully, they would not be "change something", but "add something" to Canonical's goals/targets.

What gets you most excited about this role?

I will echo my answers above.

First of all, the role of a kernel engineer in Canonical enables me to stay being part of linux community and continue to get involved in related open source projects. It is my pleasure to work within growing OSS ecosystems, discussing with partner/customer engineers from different backgrounds, who bring us, platform engineers, a bunch of valued feedbacks and insights in real use cases and practical issues they are confronting now.

I will also be thrilled with all the experiences which could give me more incentives to learning about cutting-edge technologies and chances for expanding my own abilities as an kernel engineer and expertise in advanced technology areas. Canonical is, I hope, such a company that would convert my willingness to reality, aligning to its business.

As a bonus, if I understand correctly, Canonical holds in-person meetings of employees at various

venues across the world a few times per year. I am also so happy about a chance of this kind for visiting foreign countries where our colleagues with various backgrounds meet up together at one place and build up much closer friendship for future co-working.
I love this culture along with the diversity in its nature.

FIN