Distro Integration WG

About

Work is focused on two kinds of projects.

- Issues common to multiple Linux distros (e.g. bugs/features in common packages)
- Enabling a specific Linux distro, provided the Linux distro has a champion/vendor.
  - The champion identifies the directions/projects
  - The WG works to prioritize, unravel dependencies on other WGs or file projects in other WGs

Meetings

The Distro Integration working group meets bi-weekly with following dates in 2024:

- 2024-01-02
- 2024-01-16
- 2024-01-30
- 2024-02-13
- 2024-02-27
- 2024-03-12
- 2024-03-26
- 2024-04-09
- 2024-04-23
- 2024-05-07
- 2024-05-21
- 2024-06-04
- 2024-06-18
- 2024-07-02
- 2024-07-16
- 2024-07-30
- 2024-08-13
- 2024-08-27
- 2024-09-10
- 2024-09-24
- 2024-10-08
- 2024-10-22
- 2024-11-05
- 2024-11-19
- 2024-12-03
- 2024-12-17

The group has agreed to make it's meetings open to members the guest link is available here:

Join the Meeting

Resources

- Google Drive Directory

Working Group Agendas

- 2023-12-10
- 2023-12-19

Updates

19 Jul 2023

Fedora has completed building approximately 90% of it's package set for Fedora 38. More details can be found on the Fedora wiki page.

15 Jun 2023

Brian 'redbeard' Harrington shared a number of concerns with the mailing list, related to both direct concerns held by distros as well as desired output *from* distros:

Mailing List Updates:

Concerns from distros:
1) How will we handle packaging of libraries and binaries? - We should engage in a public, community discussion of packaging concerns. Things like 'What are the "trade-offs" of "fat" multi-arch binaries?', 'How do we avoid having to maintain more than a single "boot image" for RISC-V?', etc.
2) What does the system provisioning process look like? - Deployment patterns over the years have oscillated back and forth between installation and image based deployments. An installation workflow allows for more checking and recovery while an image based deployment can be faster. Do we need more ratification around how net booting is handled for example?
3) Who "owns" the ELF specification? As far as I'm aware it's still (at least on paper) the remnants of the Santa Cruz Organization (SCO). I've talked about this with Uli. Can anyone else disabuse me of this notion? Should this be under the purview of RISE?
4) Are there common patterns we can align on around workflows for kernel updates (for example workflows like KSplice) which may be used due the intersection of HSM kernel patches and better HART support in the firmware?

What third-parties want distros to contribute:
1) Provide public test harnesses for performing validation. i.e. "Can Red Hat provide and maintain a public provisioning manifest where Rivos/Si-Five /Redbeard's Fictitious Server Company can pull these assets and perform full system installs as a part of internal continuous integration testing?")
2) An agreed upon/trusted "SDK" for building downstream projects. This could look like a compiler toolchain hosted by RISE. Personally, I've gone back and forth mentally on this and just keep thinking about Ken Thompson's paper "Reflections on Trusting Trust".

Distro Updates:

Fedora (via David Abdurachmanov) currently has over 80%+ of package builds with AMD64/x86_64. This currently includes Kernel (6.26), GCC (13.1.1), LLVM (16.0.5), CMake (3.26.4), nasm (2.16.1),

01 Jun 2023

No objections to refocusing the WG on common issues to all distros. Next steps?

- "Top 10" issues from Red Hat?
- Reach out to other distros?
- Ownership of "complex" projects affecting multiple WGs, like multiple profile/extension support (eg. https://wiki.riseproject.dev/display/HOME /Profile-+and+Extension-+Optimized+Distros)

We didn't get to Profile- and Extension- Optimized Distros, but Ulrich Drepper left a lot of comments to digest and follow up on.

Brian 'redbeard' Harrington proposed some kind of test suite for Linux distros and will follow up via the WG mailing list with more info.

31 May 2023

Posted Profile- and Extension- Optimized Distros.

During 05/25 TSC call, someone brought up lack of an easy-to-use test environment for upstream project maintainers, to allow them to easily validate submitted RISC-V enablement patches. Something like a "getting started OS image", bundling qemu + OS and with support for extensions missing from existing hardware. Interestingly enough, Intel has a project going through internal validation that amounts to exactly this. Andrei Warkentin posted some slideware and a demo for review at https://drive.google.com/drive/u/0/folders/1D3YOm_6Vlpk-bVgtv0ugt1c0wUkh3gsF and Intel is interested in partner feedback:

- Is this a solution to a problem?
- Would there be interest in Intel putting this out there for public use?
- Is there an interest in making this a RISE thing?
- Any other feedback / suggestions / improvements (at least based on PDF + demo video).

24 May 2023

Started discussion to refocus / repurpose the WG - https://lists.riseproject.dev/g/RISE-Distro-Integration-WG/topic/distro_integration_wg/99118959?p=,,, 20,0,0,0::recentpostdate/sticky,,,20,2,0,99118959,previd%3D1684965620222749443,nextid% 3D1684965620222749443&previd=1684965620222749443&nextid=1684965620222749443

Instead of aiming to improve specific distros (which is hard to do if you're not the OSV), the goal should be to identify and fix common pain points and limit the focus to Linux (no embedded OSes / BSDs today).

Heinrich Schuchardt (Canonical) brought up distro support for RISC-V extensions being an important problem to be solved. On AArch64 "ifuncs" were used, for example, to select the right atomics implementation (e.g. to support ARMv8.1 LSE). I found https://lpc.events/event/11/contributions/1103 /attachments/832/1576/Puzzle%20for%20RISC-V%20ifunc.pdf, but unsure how old this is.

Kumar had previously sent a list of possible projects, and this included something named "Standard RISC-V framework for all distros" - including boot (DT /ACPI), Interrupt controller/IOMMU (AIA, IOMMU) support and such. Now obviously this is kernel work, but maybe this means the "distro integration" projects are more about identifying and tracking all the meaningful dependencies between kernel / libs / apps that may exist to deliver a specific feature /experience?