Nuvia's Jon Masters Talks Up Their Linux / Open-Source Support Plans

Following the virtual Linaro Tech Days this week, Nuvia's VP of Software, Jon Masters, has begun talking up the Arm server start-up's Linux/open-source support plans.

While little is known publicly about Nuvia beyond being another entrant into the Arm server space and having some big names behind the company, their Linux/open-source plans look enticing if they are indeed met. While they may not be looking at complete open-source support for every aspect of the CPU, for those concerned just about out-of-the-box experience and practicality things are looking up for Nuvia.

MIPS Loongson 3 Seeing Support Improvements With Linux 5.7

Queued as part of the MIPS architecture work for Linux 5.7 are a number of Loongson improvements, in particular for the Loongson 3 series.

The additions for this next version of the Linux kernel include a generic Device Tree for Loongson 3 devices, Desktop Management Interface (DMI) for MIPS (generic to the MIPS architecture but contributed By Loongson engineers), a Loongson I/O local interrupt controller driver, a HyperTransport PIC controller driver, and various other changes currently staging within the MIPS development tree. The generic Loongson 3 DTS support should help in allowing mainline Linux images to run nicely on more devices.

AMD PassThru DMA Engine Driver Still Pending For The Linux Kernel
In addition to the AMD Sensor Fusion Hub driver that we are hopeful could land in Linux 5.7 albeit not yet queued in the iio-next branch, another AMD driver that has been around for a few months in patch form but yet to be mainlined is the AMD PassThru DMA Engine driver.

The AMD PassThru DMA Engine driver was volleyed last year and saw two additional rounds of revisions but has been quiet since the end of January. As of writing, it hasn't yet made it into the DMA-next area ahead of the Linux 5.7 merge window expected to open in early April.

Testing is an obligatory step in software development, though sometimes engineers may skip or undervalue it. Not only do you want to make sure things are working as you planned, you also want to make sure that you did not break anything that was previously working (i.e. you don't want to introduce any regressions).

If you wait until your code is merged before properly testing it and find that something is wrong, you will have to spend more time fixing it than if you'd spotted the bug during development. If it reaches the mainline code base, other developers may be impacted by the bug. If the bug is found only after a release, a much wider group could be impacted and may require a fix to be backported and provided as a bugfix. A simple fix may become a more complex one as time passes, as changes made by another developer to other parts of the code base may expect the buggy behavior. When testing is taken seriously, more bugs will be discovered earlier, potentially before the code is merged, avoiding the above added work.

Let's focus on a specific category of bugs, those in kernel system call implementations (or just syscalls, for short). They are the main entry point for users to access functionality and resources provided by the operating system, from opening files to configuring a device. What happens when a user's input is not what the kernel expected? The only correct answer is that the kernel should warn the user that they have given invalid input, by returning an appropriate error code. User input that leads to unexpected behavior; that crashes the system; that gives permissions incorrectly or unexpectedly scales privilege is a kernel bug. Given that, syscalls are an important part of kernel testing, since they are a potential point of failure.

The code base of the Linux Kernel project changes rapidly and is deployed in devices around the world, thus performing proper testing is crucial. As Linus Torvalds says, the first rule of kernel development is that we don't break userspace. This means that if a user application is working in a release, it should work in the same way in any of the following releases. Despite current efforts, the state of kernel testing is not enough. The code base has almost 3 million lines of source files, but only a small part is being tested during development.

Should you still have an HP 100BaseVG AnyLAN network adapter from the mid-to-late 90's,
the mainline Linux kernel is finally preparing to eliminate its driver.

The long-standing HP100 Linux network driver that has been around for nearly two decades is finally set to be retired. The HP100 Linux driver is for supporting the 100BaseVG AnyLAN hardware from Hewlett Packard.

VirtIO Video Driver Coming Together For The Mainline Linux Kernel[7]

VirtIO-Video is a VirtIO-based video driver for a virtual V4L2 streaming device with input/output buffers for sharing of video devices with guests. VirtIO Video has existed for a while now but it looks like it could be getting close to upstreaming in the Linux kernel.

This 2018 presentation (PDF) by OpenSynergy outlines VirtIO-Video for handling video streaming devices like video cameras, stream capturing, and other functionality within the context of virtualized guests. VirtIO-Video still supports hardware video acceleration of the host system and this virtual driver is basically about handling of input/output buffers of video streams.

Source URL: http://www.tuxmachines.org/node/135669

Links: