this month In FreeBSD

BY DRU LAVIGNE

Over the next few months, we'll be taking a closer look at some of the new features making their way into the 2016 releases, and the developers behind those features. This month, I had a chance to interview EDWARD TOMASZ NAPIERALA about his journey from FreeBSD user to ports committer to Google Summer of Code student to src committer. He also discusses the development of the root remount feature for FreeBSD. You can read a bit more about his past projects at his FreeBSD wiki page (*https://wiki.freebsd.org/EdwardTomaszNapierala*).

Q Tell us a bit about yourself. How did you get started with FreeBSD and what is your involvement with the FreeBSD Project?

A I'm a physicist by trade, but all of my actual work has always involved software.

I decided to give FreeBSD a try when I was in high school, after Linux ate my files for the n-th time (oh, the joy of of the 2.3.X development kernels). For a long time, I was just a user and occasional sysadmin. Then I became involved in ports, and eventually got a ports commit bit. In 2008, I decided to participate in Google Summer of Code (GSoC), working on NFSv4 ACLs. Quickly afterwards, I received a FreeBSD src commit bit and have used it since then.

Q Currently, you are working on the root remount project. What benefits does root remount provide and when would a user use this feature?

A It gives you a way to boot with a temporary file system, such as a memory disk image preloaded by `loader(8)`, and then replace it with the proper one. A typical example would be an iSCSI boot, which presents a chicken and egg problem: setting up an iSCSI session requires running `iscsid(8)`, and thus mounting `rootfs`, as it contains iscsid. With reroot, you provide a ramdisk with the necessary binaries (`eg /rescue`) and a script to set up the iSCSI session, and call "`reboot -r`" after the real root device becomes accessible. The kernel changes the root device and `init(8)` continues with the usual startup scripts.

You can think of it as a FreeBSD analogue of `pivot_root()` and `initrd`, as seen on Linux.

Note that you can also use remount root as a way to quickly reset the userspace, which is useful when experimenting with startup scripts and configuration changes.

Root remount support has been committed to `11-CURRENT` and there will be a merge to `stable/10`. Reroot support is planned to be included in FreeBSD 10.3 which is currently scheduled for release in March 2016.

Q From a developer's perspective, how difficult was it to create the root remount feature?

A It was quite an adventure. While the whole idea is simple, several approaches had to be tried, and a few unrelated things got fixed in the process.

I didn't go with the `pivot_root()` approach as it's an admin-unfriendly interface to work with. It also has a problem with `devfs:` after `pivot_root()` you wouldn't be able to unmount the old `/dev`, because of all the opened device nodes for pseudoterminals and mounted disk devices. Instead, I decided to just repeat the last phase of system boot, from the point of mounting the root file system onwards. Somehow the first attempt was naive: a function that called `vfs_unmountall()` which unmounted all the file systems and also killed `init(8)` in the process, because a process cannot survive the forced unmount of the file system that contains its executable file. I then repeated the steps normally performed during startup, such as mounting "`/`" and "`/dev`" and starting init. I modified the `reboot(2)` syscall and the `reboot(8)` utility to get that function called. I also needed a way to make it possible for the new `init(8)` to have `PID 1`.

This experiment only required a few hours of work, but it turned out there was a bug that

caused a panic after a forced unmount of a file system with a running executable, deep within the virtual memory management code. Since this was way outside of my area of expertise, I reported the bug and wrote a workaround that killed off all the processes first. The result kind of worked, but afterwards the system was...slow. It turned out that sending a signal to one particular kernel thread resulted in it burning CPU cycles indefinitely. So, I reported the problems and they were fixed by someone with better knowledge of those parts of the system.

But there were more adventures. The thing I didn't foresee, and in hindsight I should have, was that the `init` process is special, as it reaps orphaned processes. Because of that, the kernel treats it differently from other processes. And that treatment, like reparenting the orphaned zombies from the old `PID 1` to the new one, would need to be repeated in the reroot code. And that code is rather complex.

Another problem was that after rerooting, there was a leftover `/dev` mount. It turned out that `devfs` ignored the forced unmounts. This section of code was something I was reasonably familiar with, and I made it work by fixing a few problems in the GEOM framework. I then realized that it didn't actually get me any closer to my goal: one cannot just forcibly unmount `devfs` before `rootfs`, because it would be like yanking a drive without unmounting it, resulting in a dirty file system. So, the actual fix for forced `devfs` unmount required rearranging the code in `vfs_unmountall()` to make sure it doesn't try to unmount `devfs` before unmounting the `rootfs`.

With all those fixes, the code finally worked correctly, but it was ugly and not very maintainable. And the `init` part was hellish to debug as the forced `devfs` unmount revoked the terminals, so even the simplest form of debugging, using `printfs`, couldn't work. And `ddb(4)`, the kernel debugger, didn't help either, as it cannot debug user-

> "There are several FreeBSD committers who started by participating in GSoC, me included. It's like a virtual internship at FreeBSD."
>
> —EDWARD TOMASZ NAPIERALA

THROUGH JANUARY 2016

BY DRU LAVIGNE

2016 Events Calendar

The following BSD-related conferences will take place in January 2016. More information about these events, as well as local user group meetings, can be found at **www.bsdevents.org**.

SCALE • January 21–24 • Pasadena, CA

http://www.socallinuxexpo.org/scale/14x • The 14th annual Southern California Linux Expo will once again provide several FreeBSD-related presentations, a FreeBSD booth in the expo area, and an opportunity to take the BSDA certification exam. This event requires registration at a nominal fee.

'16 FOSDEM • January 30 & 31 • Brussels, Belgium

https://fosdem.org/2016/ • FOSDEM is a free event that offers open-source communities a place to meet, share ideas, and collaborate. This annual event attracts over 5,000 attendees each year. This year's event features a BSD devroom, a FreeBSD booth in the expo area, and an opportunity to take the BSDA certification exam.

space processes.

The second attempt preserved the ideas that worked: to unmount everything, then mount the new `rootfs` using the code already there in the kernel to mount it during boot. This attempt replaced the bad part: namely, restarting `init`.

The reason the init was killed was that when you unmount a file system that contains an executable file for a running process, the process will (probably) die. The process might be reading some page containing the machine code from the binary, and can't because the file system is no longer there. And `mlockall(2)` doesn't help in this case.

So, what I could do instead was to make `init(8)` copy itself into a safe place before unmounting `rootfs`, then `exec(2)` the copy. Remember that `exec` doesn't create a new process; it just replaces the running one, so the new `init` would run from the new executable file, but still keep `PID 1`. And, after the whole process is done, it will execute the `/sbin/init` from the target `rootfs` once again.

Another change compared to the first prototype was that instead of unmounting `/dev`, it was preserved, moved from its old mountpoint in the old `rootfs` to the new one. While it wouldn't be the best piece of functionality to expose to the userspace, it was already there used by the existing root mount code.

And that's how it works now: the `reboot(8)` utility sends a signal to `init(8)`, which then mounts a `tmpfs onto /dev/reroot`, copies the `init(8)` executable there, executes it, asks the kernel to switch file systems, executes the new `/sbin/init`, and cleans up. Then, `init` starts up as usual, running its `rc` scripts.

Q What other types of development work have you done within the FreeBSD Project?

A I started with ports, both fixing existing ones at a time when hundreds of them required build fixes due to a GCC upgrade, and adding new ones. Then I went to src, and implemented `rctl(8)`, live file system resizing, the native `iSCSI stack`, `autofs(5)`, and finally the root remount. I've also fixed a number of bugs, from ZFS to the `iwn(4)` WiFi driver.

Q In addition to being a past Summer of Code student, you have also participated as a FreeBSD mentor in the Summer of Code program. What are your thoughts on the Summer of Code program with regards to new developers and open-source projects?

A It's a great opportunity to get involved in the project. There are several FreeBSD committers who started by participating in GSoC, me included. It's like a virtual internship at FreeBSD.

One thing to remember, as a potential student, is to get involved a few months earlier. Get in touch with people working in your chosen area of interest. If you can't tell who that would be, just ask some developers as people will usually be able to easily point you at the right person. Finally, try to have a good understanding of how your project is supposed to work. When you do, getting accepted is easy.

Q Has participating in the FreeBSD Project advanced your career? If so, how?

A Sure it did! Actually, most of my career has been closely related to FreeBSD. I got my first FreeBSD-related job offer just after my first GSoC, at Wheel Systems, a vendor of authentication systems and FreeBSD-based security appliances. Several years after that, I decided to take a break from commercial work and make my living developing FreeBSD, under the FreeBSD Foundation's sponsorship. •

Dru Lavigne is a Director of the FreeBSD Foundation and Chair of the BSD Certification Group.