Dealing with upstream: how KDE and the distros manage to keep things together

Upstream projects are the ones that write most of the code that goes into powering your free software operating systems, and to a distribution, they are what makes the whole thing possible. There is a fine but delicate process that happens to keep all of these projects all running as smoothly as they do.

By Troy Unrau | Last updated August 19, 2007 11:00 PM CT

What defines upstream? KDE, GNOME, Apache, even the Linux kernel are upstream for a Linux distribution, and even a totally different distribution altogether may be considered upstream (for example, Debian is considered upstream for Ubuntu). These upstream projects are the ones that write most of the code that goes into powering your free software operating systems, and to a distribution, they are what makes the whole thing possible. There is a fine but delicate process that happens to keep all of these projects all running as smoothly as they do. Distros have been known to fail due to making too many changes to the source without the cooperation of their upstream partners.

You can trace the path even further upstream: KDE, which is already upstream for a distribution, depends on code that comes out of a wide variety of library projects, such as the popular Qt toolkit library. KDE has to keep its upstream providers happy too. KDE even keeps a local branch of Qt within their own source tree which they use to test patches, and apply fixes that haven't been applied upstream yet. KDE has to exercise caution when dealing with their own branch, and cannot fully rely on it for the KDE project to function as there is no guarantee that any of the changes they make will be incorporated into the official Qt source tree. However, if the code is of good quality, and it fixes or improves Qt in some way, it will usually end up incorporated into the next Qt release allowing the two branches to become synchronized once again.

KDE is just one project that has to deal with upstream contributors. Distributions have a much larger task in hand. Rather than only having to deal with only a few dozen upstream sources, distros have to contend with hundreds or even thousands of small open source projects. Not only do they have to keep their packages up-to-date with the new releases that are happening, but they often have their own branches or patch-sets for each of the projects that they are packaging.

This introduces a unique problem that is happening more and more within distributions: what do you do with the changes you make to the upstream packages? Many distributions have encountered this problem as they start to make heavy customizations to their packages. After you've made changes to help fix some bugs, add or change some features, re-brand programs with your own logos, and better integrate some packages that are otherwise independent—you end up with quite a large set of patches that you need to maintain. Some of these patches, especially bug fixes, usually end up in the upstream source trees pretty rapidly, but that is only if the distros maintain a clearly organized set of patches. It is difficult to simply run 'diff' on a big project like KDE to find a single bugfix once you've started re-branding it, especially once that bugfix is more than a single, trivial line of code.

This is actually a major problem for distributions as their code will continuously diverge from the upstream over time. Consider Mandrake Linux, which started as a simple project. They simply took the existing Red Hat release of that time, and added KDE. At that time, it was very simple to maintain, requiring very little manpower. Since then, they have merged with Connectiva to become Mandriva, and have a whole set of custom patches and applications that make it far more difficult to get upstream. KDE 4.0 is scheduled to be released later this year, which will pretty much make any patches they made against the current, stable version of KDE useless, meaning they will have to do it all over again. This problem does not only apply to Mandriva, but to major distributions like OpenSuse, *Ubuntu, Red Flag, and others. They all need to now dump a lot of work that they've done in order to adapt to a major change upstream.

Only one distribution seems to be relatively immune to this process, that being Slackware. Slackware is one of the oldest distributions in existence, and may in fact be the longest running distro of all time. The reason that Slackware is successful, but never popular to the extent of other distros, is that it leaves customization alone. Wherever possible, they ship 'virgin' packages which can be recompiled using upstreams' official source packages. This makes Slackware a good development platform as you can ensure that any problems you encounter are not due to distributions making changes to the source, but also makes it a less popular choice among users who prefer a more polished, integrated experience.

Working Cooperatively

With KDE 4.0 being released later this year, this is a perfect opportunity for the distributions to take a lesson in cooperation, and instead of maintaining all of their patches independently, to try harder to get their work upstream. Most upstream projects have extremely open access to their source trees. For example, it takes only a few hours to get write access to KDE assuming that you have something that would benefit the project. Other open source projects are equally open to contributions from users, packagers, and distro types and are not incredibly difficult to get your changes into (unless of course you're trying to rename Firefox for political reasons). Not doing so could be a huge waste of resources for a distribution, causing bit-rot to set

in for countless otherwise excellent patches.

In fact, the popularity of a distribution can conceivably be linked to how well they work with their upstream partners. Kubuntu, OpenSuse, and Mandriva, for example, have recently been taking huge efforts to get their code upstream into KDE in time for the 4.0 release and that reflects their popularity within the KDE community. The distros that the KDE developers are using will be the ones that they recommend to KDE users at events, on IRC, and in forums. Usually, they will recommend the distro that keeps KDE the 'purest'. This is easy enough for the small distributions to adhere to, since they lack the manpower to maintain independent branches of KDE, but for the large distributions, it can be tempting to try too hard to create a unique product. When a distro stops resembling upstream, it is cutting off its own head by necessitating very expensive code maintenance just to keep up-to-date. When this happens, the popularity of a distro can plummet, which can be seen by the failure of a number of very promising distributions from Caldera (yeah I know, they had other problems too) and Corel Linux in times past, as well as the very limited recent success of the highly customized Linspire/Freespire.

This week, the KDE-based Ark Linux distribution announced that it is basing it's next release on KDE 4.0; they also announced that they would be making every effort to integrate their existing hardware customizations right into KDE's new "Solid" device libraries rather than maintaining them separately as they did previously. This should be applauded as it is a first good step towards ensuring that they have less maintenance to do in the future, and ensures a good relationship with the KDE developers upstream. Some other distributions have already been equally helpful with contributions ranging from OpenSuse's heaps of code and sponsored KDE developers to Kubuntu's contribution of their new control panel.

Users should be excited to see these distributions all working together, and should encourage it whenever possible. The temptation for a distribution to go their own way in favour of market-share is always there, but the whole free software world is much better off when they do their work upstream.

Serving the technologist for 1.0756×10^{-1} centuries