Beta Testing 101

By *Spinlock*
Created *16/05/2006 - 8:03pm*
Submitted by Spinlock on Tuesday 16th of May 2006 08:03:09 PM Filed under Howtos [1]

Ever have one of those days when you can't stand end users? I know I have.

There are several expectations end users have for people creating software, or even Linux distros. They want it to work on their hardware, they want it to be stable, and they want it right now. But, at the same time, new releases shouldn't come too often, or it messes up the feng shui of their systems.

But you can't get them to beta test. I say this from experience in multiple projects, both those I lead and those I've observed. Oh, you can get them to agree to beta test, but I'm not sure that they understand how. So, here's a few tips from someone who's been on all sides of the equation.

1. Get a jumpdrive and backup all your logs. They've come way down in price over the past few years, and it's not prohibitively expensive to buy. And buy you should... get a separate drive just for backing up logs while beta testing. The least experience Linux user in the world can be the best beta tester a project has just by making copies of their logfiles available.

2. In that vein, back up system information as well. Hardware-specific bugs are much easier to track down and kill when the bug-hunter can look for patterns. Make it a habit to save the output from lspci and lsusb into their own log files on that jumpdrive. Set up a folder for each session, cd to that folder, and simply lspci > lspci.log and lsusb > lsusb.log.

3. Further in the logging journey, learn to use dmesg and grep. To find out information in the system log about your agp setup, dmesg | grep agp. This will prove invaluable, and keep you from having to manually parse potentially hundreds of lines of text to find two lines of interest. Once again, you can concatenate the output to a logfile: dmesg | grep agp > agpinfo.log.

4. Finally, learn to use stdout and stderr. These are the output and errors that come from running a program. To help diagnose dependency errors and the like, the output from a malfunctioning program can be redirected to a log file like so: make > makelog 2>&1. This can turn a single run into a goldmine of information for the harried developer with people breathing down his neck about the next release.

5. Try things you normally wouldn't. Play the games. Write a letter to your Great Aunt Ruth. Listen to that Hanson cd you won't admit you still own. Burn copies of it and leave it unmarked in your friends' cars so they get curious and pop it in. Cross-compile the kernel for a 386. Test your dial-up modem. Scan for wireless networks. Any number of these things will be attempted by the end users of this project, and your input may keep a bad situation from ruining a release.

6. Go to Wal-Mart and buy one of those cheap marble composition books. Take notes in it while you use the computer. Having a hard copy in case of major system meltdown can be worth more than gold, and it will give you something to look back on later. Notes files get deleted, but you can keep notebooks for 20 years if you take care of them.

Well, that's it for this lesson. I hope that it's useful for potential beta testers, software developers, and even the end users (maybe now you can understand why some things aren't being released yet.)

And that's life when you're in spinlock.

Howtos
