24
Nov
2009

Crossroads in FOSS Projects: Some Business Considerations

At our Seminar last month, Managing FOSS to Lower Costs and Achieve Business Results, several participants asked about the dynamics of FOSS (Free and Open Source Software) projects that reach a crossroads (a failure, a merger, loss of key personnel, etc). I had not expected that concern because with commercial software, it seems to me, the problem is more severe. When you have the source code and the right to modify and redistribute it, the source gives many more options (and its freedoms provide many more protections) than when commercial software goes bankrupt or gets bought by a competitor for instance.

But the reason for the questions may be a lack of understanding about how FOSS projects work. They involve individual human beings, perhaps just a single person or, more likely, several people from many organizations and even different cultures around the world joined in common purpose. For various cultural reasons, the project may be “owned” by an entity — usually a non-profit, but some are for-profit or even government owned, while others may simply be an “ad hoc initiative”. Some projects have explicit constitutions and defined processes for organizing the work and handling problems others are more informal.

At any time, any human social structure can experience a **crossroads** that could lead it to fail suddenly or wither on the vine in a gradual descent into “oblivion”. The cause of the failure will shape the results, but a very common situation is that conflicting visions or approaches for the project result in a “fork”. Then a sub-group of the original project takes the source code and starts a “new” project to develop the code in a new direction. Sometimes the original project “dies” and sometimes both continue resulting in two projects. Since multiple FOSS projects serving the same function or market incur inefficiencies due to duplicate development, there is a strong cultural value in the FOSS world to try to find a way to accommodate everyone in the project and prevent forks. When it works, the result is great software that meets everyone’s needs. But the reality is that often it is more effective to have multiple implementations of the same functionality so that each can be optimized for distinct objectives. Frequently one cannot know which approach will be best until many years of development and evaluation have transpired.

I recently learned about a FOSS project that forked when a friend asked me to copy some files to his new “My Book Essential”, a Western Digital product that provides 1TB of USB (Universal Serial Bus) storage. The My Book uses the poorly documented, non-free NTFS (New Technology File System). Linux has three projects that support NTFS: an in kernel driver, ntfsprogs (the Linux-NTFS project), and NTFS-3G. It turns out that all three were available for my Debian Lenny (5.0.3) system. First, I tried the in kernel support and learned that it was still read-only. Then I tried ntfsprogs which failed to mount the My Book:

```
NTFS-fs error (device sdc1): load_system_files(): Volume is
dirty. Mounting read-only. Run chkdsk and mount in Windows.
```

I realized that since it was a new device it probably did not ship from the factory with a dirty volume. It was probably a bug. So I tried NTFS-3G which worked very well. In my research of the situation I was able to determine that both NTFS-3G and Linux-NTFS are under active development and have features missing from the other. So each has value and I’m glad my distribution included both. In Debian Lenny, the NTFS-3G driver has better support for writing files.

This illustrates one of the benefits of a **crossroads** in a FOSS project: you can end up with **two** good tools to add to your toolbox!

Posted by CJ Fearnley *in* Debian, FOSS Community, Tech Notes, *0 comments*