Distribution-Independent Printer Driver Packages without LSB

Till Kamppeter, Openprinting

THE LINUX FOUNDATION

The Problem

- **Years ago, we have agreed on to use the LSB to create distribution-independent printer driver packages**
  - LSB describes common programming interfaces of the currently released enterprise distros
  - Most distributions have compatibility meta packages to let LSB-based software packages to run on them
  - Printing-related interfaces got added to the LSB
- **Debian dropped LSB support, dropping its meta-package**
  - https://lwn.net/Articles/658809/
  - Ubuntu lost support, too as they sync the meta package
  - Ubuntu will do temporary solution for 16.04 LTS only
  - To few software vendors make use of the LSB (probably only Epson with their printer drivers)
  - LSB very complex, therefore LSB compatibility awkward to maintain

Possible Solutions

- **Best would be: IPP Everywhere now and for every printer, do away with drivers at all!**

- **We probably still need distribution-independent driver packages at least for some time, but**
  - LSB not available in all distros, and perhaps soon fading away then
  - LSB too complex for manufacturers to develop under
  - Needed: Simpler way for making distribution-independent binaries, ideally without need of compatibility meta package and no complex standard description
  - Should be easy to implement to attract more printer manufacturers

- **Also possible: PPD-only package for PostScript, PCL (as Ricoh does), or other known PDL**

Static linking

- Easy to install

- Large executables
- Security bugs in built-in libraries do not get fixed
- Executable cannot dynamically link libraries at run-time
- Especially libc links dynamic options at run-time, with libc statically linked only objects of that libc can get linked
- You cannot simply statically link existing code, it often needs to get adapted

Replace libc by musl

- http://www.libc-musl.org/
- musl replaces libc
- musl itself does not dynamically link anything, allowing for totally statically linked executables
- Single binaries running on any machine with the appropriate processor architecture
- musl is under permissive MIT license

- Code needs to get prepared for static linking, but probably not different tocode using libc

Partial static linking

- Never statically link libc
- Statically linking libstdc++ is safe
- Be careful also with X11 and openGL libraries
- Link everything else statically
- http://blog.sagargv.com/2014/09/on-building-portable-linux-binaries.html
- More links on this and building instructions in the blog post

Ship with libraries

- Drop in the library files your application needs to the same directory as the binary and include $ORIGIN in the rpath when linking.
- Use automatic tool, like CDE: http://www.pgbovine.net/cde.html

- No compatibility problem with static linking
- No license violation in closed-source drivers
- No “dependency hell”

What to do?

- Should we require one of the show methods?
- Or should we leave to the individual printer vendors which method to apply, only require distro-independent package?
- Should we stay with RPM/DEB packaging or make tarballs of the binary files dropping them into the standard locations of CUPS?