

Qt BY TROLLTECH



LSB

Harald Fernengel
Trolltech ASA





Challenges for deploying on Linux

- Fragmentation of Linux Distributions
- Multiple architectures
- Multiple Desktops / Windowing Systems



Fragmentation of Linux Distributions

- The Linux Standard Base (LSB) defines a consistent ABI for Linux for 7 architectures
- The LSB provides build tools and certifies software and distributions



Qt

- Qt 3.3 is part of the LSB standard, Qt 4.1 a module



The standard

- LSB is a trailing standard, no proactive
- A technology has to be shipped on all major Linux distros before it can be standardized.
- Builds upon existing standards (POSIX)



The standard

- LSB is a trailing standard, no proactive
- A technology has to be shipped on all major Linux distros before it can be standardized.
- Builds upon existing standards (POSIX)
- ~~Everything must be LGPL or less~~



Politics

- Elected Chair: Ian Murdock
- Steering Committee:

Brian Aker (MySQL)

Jeff Bailey (Canonical/Ubuntu)

Paul Gampe (Red Hat)

Marvin Heffler (IBM)

Ian Murdock (FSG/LSB)

Matt Taggart (HP)

Mats Wichmann (Intel)

Jeff Ayers (RealNetworks)

Rajesh Banginwar (Intel)

Andrew Josey (The Open Group)

Thorsten Kukuk (Novell/SUSE)

Nick Stoughton (Usenix)

Kay Tate (IBM)



How to use the LSB?

- Install the SDK from `linuxbase.org`
- add `/opt/lsb/bin` to the path
- set the `LSB_MODULES` environment variable to `Qt4` for Qt4 apps
- run `qmake && make`



So what happens?

- `lsbcc` and `lsbc++` is used instead of `gcc` and `g++`
- All libraries that are not standardized in the LSB are statically linked
- The resulting binary runs out of the box on all modern distributions



But it doesn't build!

- `lsbcc` and `lsbc++` redirect the standard include pathes to LSB's stub headers
 - Grep in the headers in `/opt/lsb/include` for the missing symbol
- Similarly, the application links to LSB's stub libraries
 - Do an `nm` on the libraries in `/opt/lsb/lib` to find the symbol
- Still not found? --> Bugzilla :)



So I have the binaries, but where do they go?

- File Hierarchy Standard standardizes the Unix filesystem
 - Specification: <http://pathname.com/fhs>
 - `/opt` is reserved for add-on applications
 - Choose to install into `/opt/<package>` or `/opt/<provider>`
 - `<provider>` must be a LANANA registered name (<http://www.lanana.org>)



The Application Icon

- Let your artist create a `*.png` file
- Read the Desktop Entry Specification on freedesktop.org (not part of the LSB yet)
- Create a `theapp.desktop` file



A typical app

`/opt/theapp/theapp`

`/opt/theapp/theapp.png`

`/opt/theapp/plugins/`

`plugin1.so`

`plugin2.so`

`/usr/share/applications/theapp.desktop`

The executable

The icon

Plugins subdir

A plugin

Another plugin

The desktop file



Package it

- The "native" LSB package format is RPM
- An LSB installer is planned for a future release
- Debian's `alien` will recognize LSB RPMs and automatically execute the RPM scripts



Services

- LSB has good support for start/stop scripts and their installation
- It is possible to write a service that feels "native"



Need your own library?

- All non-LSB standardized libraries must be statically linked.
- No access to "backend" libraries
 - e.g. Qt can use XRender, CUPS – you can't.
 - Makes it harder to statically link to a custom Qt version / any other lib



How to get a new library into the standard

- Library must be shipped by all major Linux distributors (SLES, RHEL)
- There must be demand from ISVs
- Someone available to do the standardization work



Disadvantages

- LSB only defines an ABI, not functionality
- Not all areas of application development are standardized, e.g. sound or multimedia
- Some symbols (e.g. `syscall`) not in the standard
- Functionality provided at runtime not clearly defined (image plugins, SQL drivers)



Advantages

- Good backing from industry
- Good support from Distributors
- Actively maintained and extended



Thank you for your attention!