



Adding Value through LSB Conformance

Stuart Anderson, Free Standards Group
anderson@freestandards.org

Mats Wichmann, Intel Corporation
mats.d.wichmann@intel.com



Free Standards Group



-
- An umbrella organization for a group of standards-related projects
 - LSB
 - OpenI18N
 - Accessibility
 - DWARF
 - and Affiliates:
 - FHS, FreeDesktop.org, OCF



Linux Standard Base



-
- Open Participation
 - 4 principal components
 - Specification
 - Generic & architecture Specific
 - Tests
 - Runtime & Application
 - Sample Implementation
 - Development Environment



Benefits of Binary Packages



- Easier to install == available to more people
 - Captures/automates install dependencies
 - Removes dependency on build tool chain
 - gcc version, libfoo-devel version, etc.
 - Using a package management tool eases state queries, version upgrades
 - Takes distro/version out of the support equation



Developer Benefits



-
- Making source LSB-clean helps
 - Clarify what assumptions are made
 - e.g. in building a configure script
 - Fewer versions to build/test
 - eliminates field portability questions when others try



The LSB Specification



- The LSB spec describes a stable runtime environment for applications
 - Required shared libraries
 - A description of the behaviour of public interfaces
 - Symbol versioning as needed
 - An LSB dynamic linker
 - Commands needed for installation, startup
 - Guidelines on packaging and install locations



Runtime Compliance



- LSB is a contract between systems and apps
- For a distribution, a promise to provide the functionality to run a compliant application
 - Doesn't mean commands are LSB conforming
 - Doesn't restrict the distro from providing more...
- Prove by running the tests, example applications
- Possible to implement on a non-Linux system
 - E.g. hosted in a VM



Application Compliance



- For an application, a promise to use only the services required of a compliant system
 - Link against LSB runtime linker
 - Use only LSB specified interfaces and libraries
 - Statically link with non-LSB specified libs
 - Or bundle non-LSB shared libs with the application
 - Prove by passing LSB Application Checker



LSB Certification



- The Free Standards Group offers a certification program at <http://www.freestandards.org>
 - Register, submit test results, sign agreement to use LSB logo
- All major Linux distributions are LSB Certified
 - Red Hat, SuSE, Mandrake, Turbo, Connectiva, many others
 - 24 certifications as of 10/2003, including a couple for Itanium[®]





LSB Application Certification



- Certification is available for applications, too
 - In addition to tests, complete the FHS checklist and package in LSB format
- Two other models for applications available
 - Self-certified - follow the model, but don't submit to certification authority
 - *Requires LSB* - needs LSB runtime, plus some extra stuff – probably most applicable to Gelato



Questions





Building LSB



-
- How to make sure the rules are followed?
 - The spec is lengthy – don't try to memorize it!
 - By reference, includes many shelf-feet of other specs
 - Existing projects need to modify their build rules
 - The answer: tools
 - Application checker
 - Compiler wrapper to control compile and link
 - LSB-clean stub libraries and headers



A Trivial Example



- The application checker in action:
 - Hello world (straight up, with a twist):
 - `$ cat hello.c`

```
#include <stdio.h>
#include <unistd.h>
main()
{
    printf("hello world: %d\n", getpid());
}
```
 - `$ gcc hello.o -o hello-1`
`$./hello-1`
hello world: 14250



Checking the Program



- ```
$ /opt/lsbappchk/bin/lsbappchk hello-1
lsbappchk for LSB Specification 1.3.3
Checking binary hello-1
Incorrect program interpreter: /lib/ld-linux.so.2
Header[1] PT_INTERP Failed
Found wrong interpreter in .interp section: /
lib/ld-linux.so.2 instead of: /lib/ld-lsb.so.1
$
```

  - That's not an LSB binary...



# Build Toolkit



- LSB provides compiler wrappers for C and C++
  - Model: LSB building shouldn't be (very) intrusive!
  - Fiddle with command line to “silently” apply LSB build rules, call standard compiler
    - Alternate compilers should be possible (not proven)
  - Effectively a restricted build in regular environment
  - Should be able to set CC to lsbcc, CXX to lsbc++
    - `make CC=lsbcc`
    - `CC=lsbcc CXX=lsbc++ ./configure`



# Headers and Stub Libs



- LSB stub libraries
  - Empty shared libs for link-time symbol resolution
  - Catches LSB name space violations up front
- LSB headers
  - An LSB-clean set of headers matching the stub libs (generated from same DB)
- Compiler wrappers make sure both are used in preference to system versions



# lsbcc Example



- Build, check, run:
  - `$ /opt/lsbdev-cc/bin/lsbcc hello.c -o hello-2`
  - `$ /opt/lsbappchk/bin/lsbappchk hello-2`  
lsbappchk for LSB Specification 1.3.3  
Checking binary hello-2
  - `$ ./hello-2`  
hello world: 14986
  - `$`



# Non-LSB Code



- What if we “break” the source from an LSB viewpoint?
  - `$ readelf -s /lib/libc.so.6 | grep getpid`  
000bb160 g DF .text 00000032 GLIBC\_2.0 \_\_getpid  
000bb160 w DF .text 00000032 GLIBC\_2.0 getpid
  - Change the code to use the internal `__getpid` in `hello2.c`
  - `$ cc hello2.c`  
`$ ./a.out`  
hello world: 11185
  - That’s still a working program natively, but should have an additional LSB conformance problem: a non-spec interface



# Compiling Non-LSB Code With lsbcc



```
- $ lsbappchk a.out
lsbappchk for LSB Specification 1.3.3
Checking binary a.out
Incorrect program interpreter: /lib/ld-linux.so.2
Header[1] PT_INTERP Failed
Found wrong interpreter in .interp section: /
lib/ld-linux.so.2 instead of: /lib/ld-lsb.so.1
Symbol __getpid used, but not part of LSB
```

- The stub libraries prevent linking this:

```
- $ lsbcc hello2.c
/tmp/ccqrdLCg.o: In function `main':
/tmp/ccqrdLCg.o(.text+0x17): undefined reference
to `__getpid`
collect2: ld returned 1 exit status
$
```



# Packaging



- Another portability concern is “packaging”
  - How will the application be delivered and deployed?
  - Choice of packaging format
    - LSB rpm format, ok for rpm-based and deb-based distros
      - Does not require rpm command: converting, as with alien, is OK
    - Can choose another packaging format (discouraged)
      - Need to supply the command (LSB-conforming) with the app
  - Use FHS-specified paths to install: /opt
  - Don't pollute the package name space: LANANA



# Is The System LSB-Ready?



- On delivery, how can we be sure the target has LSB conformance?
  - Require “lsb” in your package
  - LSB conforming systems are required to provide “lsb” (and a version) in their packaging system
    - Most have a package like `lsb` or `redhat-lsb`
      - A “keystone” package which provides `lsb`, requires whatever else that system needs for conformance
    - This dependency is why the LSB pkg format is preferred!



# LSB-Ready (cont'd)



- The LSB is a behavioral spec
  - Thus, “lsb=1.3” covers the whole behavior of that spec version
  - No need to worry exactly which package/version provides any of that functionality
- What do non-LSB installers check?
  - LSB requires a command `lsb-release` which returns the same info



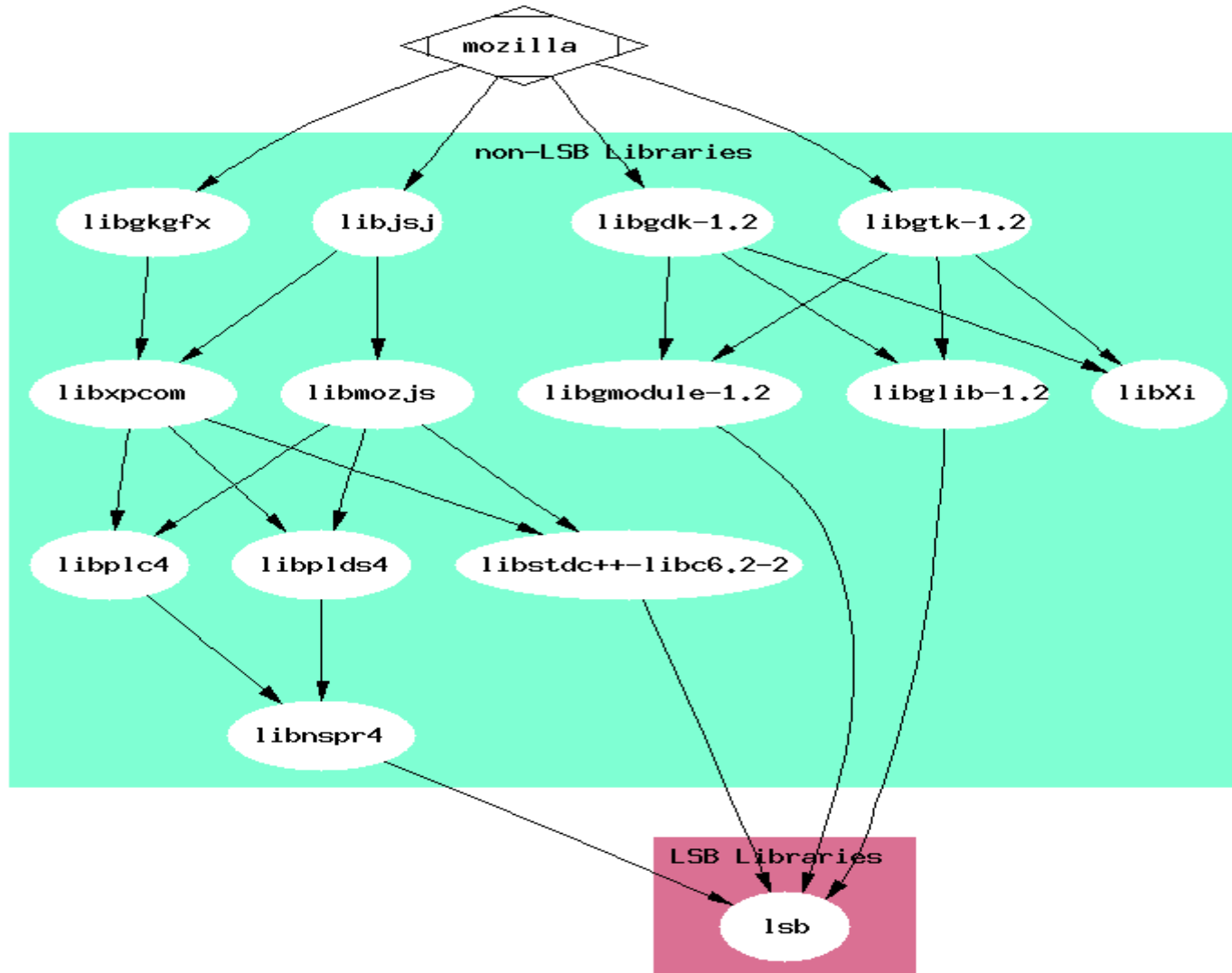
# Porting: the Reality



- It's not going to be as easy as the example...
  - The library selection policy was very conservative
  - Some software needs more libraries
    - The test build of Eclipse did
  - Choices:
    - Avoid those libraries
    - Link them statically: lsbcc will handle this; the binary will still be a dynamic binary
    - Provide those libraries LSB-conforming



# Library Dependency Example





# A Library Collection



- Gelato could build the libraries it needs LSB conforming and place in repository
  - LSB naming rules will help: /opt/gelato (which Gelato would “own”) can hold these libraries
  - One or more library packages can then become dependencies for the applications
    - One LSB system (Connectiva) has apt-get for rpms, their mods are open-source: might help installation
  - Hopefully many will become LSB libraries later



# Summary/Questions



- 
- The LSB enables portable binary software
    - Software becomes more accessible
    - Developers can focus on creating value
  - Tools simplify the build-and-check process
  - Gelato and LSB can cooperate on making more libraries available as LSB-conforming packages
  - Questions?



# Resources



- 
- Everything LSB: [www.linuxbase.org](http://www.linuxbase.org)
  - Mailing lists linked off [www.linuxbase.org](http://www.linuxbase.org)
    - lsb-build for tools, lsb-appbat for porting issues, libraries
  - IRC: [irc.freestandards.org](http://irc.freestandards.org) #lsb
  - Bugs/request at LSB bugzilla: <http://bugs.linuxbase.org>
  - Downloads: [www.linuxbase.org/download](http://www.linuxbase.org/download)
  - FHS checklist: [www.linuxbase.org/test/fhschecklist.html](http://www.linuxbase.org/test/fhschecklist.html)
  - The FHS: [www.pathname.com/fhs](http://www.pathname.com/fhs)
  - LANANA Registry: [www.lanana.org](http://www.lanana.org)