



Producing Standards-Compliant Eclipse Binaries: An LSB Case Study

Kevin Cernekee

<http://gelato.uiuc.edu/projects/eclipse/>



Advancing Linux Itanium



Benefits of Standards Compliance

- ✦ Ease of use and ease of modification by end users
- ✦ Code maintainability and portability
- ✦ Lower support costs
- ✦ Consistency across different software projects



Advancing Linux Itanium



Gelato Standards Compliance

- ✦ Gelato has partnered with the Linux Standard Base (LSB) group
- ✦ LSB creates standards for both application software and OS distributions
- ✦ Gelato is also working on its own source release guidelines
- ✦ Participation by member institutions is voluntary but encouraged



Advancing Linux Itanium



LSB Overview

- ✦ LSB is a workgroup run by the Free Standards Group
- ✦ Aims to provide binary level compatibility between compliant OS distributions and compliant applications
- ✦ Current version: LSB 1.3 – one common spec + 5 processor specific specs
- ✦ Upcoming version: LSB 2.0



Advancing Linux Itanium



How LSB Works

- LSB standardizes library interfaces at the source and binary levels
 - Binary level: standardizes shared library interfaces, exports, symbol versioning, etc.
 - Source level: provides standard header files that contain **only** prototypes, constants, etc. defined in the LSB spec
 - End result: only compliant code will properly build and link in this environment



How LSB Works

- LSB sets other standards for the program's runtime environment
 - File placement is handled according to the Linux FHS standard
 - LSB defines system calls, return values, error codes, and behaviors
 - Codifies standards for calling conventions, executable file formats, and other low-level ABI concerns



LSB Development Tools

- lsbdev-chroot
 - Provides a minimal self-contained build environment
 - Uses bind mounts to expose selected parts of the original root filesystem
 - Instructs the compiler to build against the LSB stub libraries and headers
 - Easy to customize to work around specific issues
 - Best choice for complex setups



LSB Development Tools

- lsbdev-cc: gcc wrapper
 - lsbcc is a drop-in replacement for gcc
 - Rewrites gcc command-line arguments so that resultant objects use LSB stubs
 - Usage is similar to that of a cross compiler
 - Works best for programs that do not perform too much probing of the build environment





LSB Development Tools

- ✦ **lsbappchk**
 - Tests an application binary for LSB compliant usage of shared libraries
 - In general, won't find other issues such as improper use of library functions
- ✦ **LSB Reference Implementation**
 - Provides a small, compliant Linux distribution
 - Can be used for application testing or for troubleshooting a new distribution



Advancing Linux Itanium



LSB Development Tools

- ✦ **LSB Application Battery**
 - Provides several known-good sample programs built to LSB specs
 - Intended as tests to exercise LSB infrastructure and build tools on a compliant system
 - Can be used for as examples for application developers
 - Plays an important role in the certification process



Advancing Linux Itanium



LSB Development Tools

- ✦ **Other tools:**
 - **lsbdev-c++**: LSB compliant C++ libraries
 - **lsbdev-base**: includes all stub libraries and headers used by **lsbdev-chroot** and **lsbdev-cc**
 - **lsblibchk**: tests a Linux distribution for compliance



Advancing Linux Itanium



Building a Simple Application

- ✦ We will use **lsbcc** to build "Hello World"
- ✦ First, ensure that the LSB infrastructure is installed on your system
- ✦ Add **lsbcc** to your **PATH** and compile

```

cernekee@naples:/tmp$ ls /opt/lsbdev-cc
bin doc man
cernekee@naples:/tmp$ export PATH=$PATH:/opt/lsbdev-cc/bin
cernekee@naples:/tmp$ lsbcc hello.c -o hello
cernekee@naples:/tmp$ ldd hello
    libm.so.6.1 => /lib/libm.so.6.1 (0x2000000000004c000)
    libc.so.6.1 => /lib/libc.so.6.1 (0x200000000000e4000)
    /lib/ld-lsb-1a64.so.1 => /lib/ld-lsb-1a64.so.1 (0x2000000000000000)
cernekee@naples:/tmp$ ./hello
Hello World!
cernekee@naples:/tmp$ █

```



Advancing Linux Itanium



Building a complex application

- ✦ Usually easier under the LSB chroot environment
- ✦ Four major steps
 - Satisfy all dependencies recursively
 - Attempt to build the module
 - Install/consolidate files and test compatibility
 - Package and test on the reference implementation

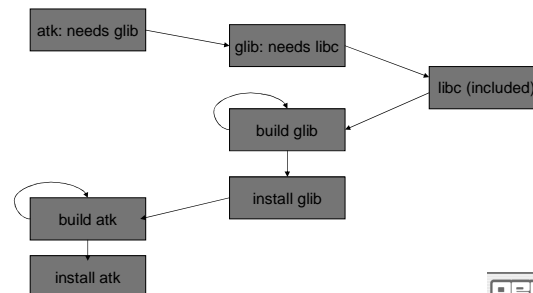


Advancing Linux Itanium



Building a complex application

- ✦ Example workflow:



Advancing Linux Itanium



Building a complex application

- ✦ Satisfy dependencies
 - Many common libraries are included in the development environment
 - Uncommon libraries will need to be built and installed in the chroot environment
 - Installation locations: /opt/\$APPNAME/lib



Advancing Linux Itanium



Building a complex application

- ✦ Building program modules
 - Build errors may be common – developers often do not follow standards
 - Refer to the LSB specifications to determine proper usage of standardized functions
 - Run lsbappchk when in doubt to verify that the generated binary is compliant
 - Take notes on build failures and generate a patch



Advancing Linux Itanium



Building a complex application

- Final steps
 - Test the application to ensure that it still runs
 - Use “ldd” to look for dynamic linking against unsupported shared libraries outside the installation directory
 - Write an rpm spec file and package the software
 - Test the rpm under the LSB reference implementation



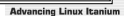
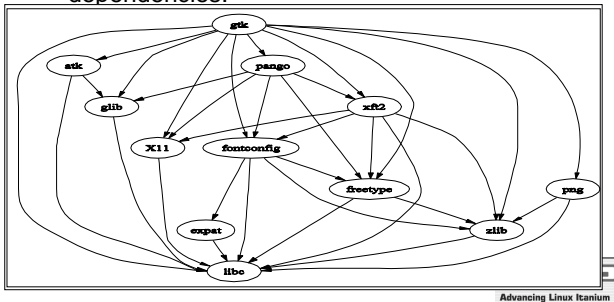
Building the Eclipse binaries

- Eclipse is mostly written in Java, but...
 - There is a platform launcher binary (“./eclipse”) which is written in C
 - Eclipse uses libswt (also C) as an intermediate layer between the Java application and the system’s widget library
 - Most of my work has focused on this code since the Java modules don’t really need to be changed



Building the Eclipse binaries

- The platform launcher and libswt have many dependencies:



Building the Eclipse binaries

- Issues found when building Eclipse
 - libswt is not 64-bit clean, so I applied and tweaked Red Hat’s patches to widen the pointers
 - Platform launcher and libswt source was available, but was scattered throughout the tree
 - Patches and scripts that address these issues are on my site





Building the Eclipse binaries

- ✦ The LSB spec recommends that non-LSB libraries be linked in statically
 - On Eclipse, this was not possible
 - Some libraries dynamically load other shared objects at runtime
 - Some libraries just weren't designed to be linked in statically
 - Solution: build our own shared libraries and include them in the package



Advancing Linux Itanium



Building the Eclipse binaries

- ✦ Source fixups were needed: examples
 - Some programs #include'd <sys/poll.h> instead of <poll.h>, causing build failures
 - XFree86 attempted to use alloca() and getsuid(), two functions not defined in the spec
- ✦ Sometimes "man" pages do not reflect current LSB standards
- ✦ Sometimes using autoconf wisely will mitigate the problem
- ✦ For the most part, these problems will not be obvious until you build under an environment that enforces standards



Advancing Linux Itanium



Building the Eclipse binaries

- ✦ Other unusual problems
 - Hidden dependencies: not all dependencies are shared libraries
 - Libraries that are linked in statically might not be immediately apparent during the planning stage, e.g. zlib, libpng, etc.
 - Some packages don't like to install to unusual locations like /opt/eclipse-2.0.1
 - Some packages don't like to use headers found in odd locations



Advancing Linux Itanium



Building the Eclipse binaries

- ✦ Solutions and workarounds
 - Often, setting LD_LIBRARY_PATH and LD_RUN_PATH helps find libraries
 - Sometimes, overriding environment variables like CC and CFLAGS will fix the problem
 - But sometimes you just need to patch the source code or Makefiles



Advancing Linux Itanium



Packaging the Eclipse distribution

- ✦ In most cases it is possible to write a proper rpm spec file that builds and packages in one step
- ✦ In this case, the build process was rather complex so I just made a spec file that converts a tarball to a package
- ✦ You may need to override the Provides: and Depends: attributes if you include your own shared libraries



Advancing Linux Itanium



Packaging the Eclipse distribution

- ✦ Example spec file:

```
Summary: Eclipse
Name: eclipse
Version: 2.0.1
Release: 1
Group: Development/Tools
License: CPL
Source0: empty.tgz
BuildRoot: /usr/tmp/buildroot
Requires: libm.so.8.1(GLIBC_2.2)(64bit) libdl.so.2(GLIBC_2.0)(64bit) libc.so.8.1
(GLIBC_2.2)(64bit) libpthread.so.0()(64bit)
%description
Eclipse is a kind of universal tool platform - an open extensible IDE for
anything and nothing in particular. This is a 64-bit clean release built
in the LSB chroot development environment.
%define __find_requires %{}
%define __find_provides %{}
%prep
%setup -q
%build
%install
%clean
%files
%define(-,root,root)
/opt/eclipse-2.0.1
eclipse-2.0.1.spec [R0] 1,1 All
```



Testing for Compliance

- ✦ Run lsbappchk to look for symbol name or versioning issues
- ✦ Run the application under the LSB reference implementation if possible
- ✦ Apply for LSB certification (optional)



Advancing Linux Itanium



Getting Help

- ✦ Documentation
 - <http://www.linuxbase.org/>
- ✦ Contact me
 - It's my job to keep Gelato up to date on standards issues and assist developers
 - cernekee@crhc.uiuc.edu
- ✦ Contact the Free Standards Group
 - The LSB workgroup runs several mailing lists and is always willing to help



Advancing Linux Itanium



Future Eclipse work

- ✦ IA-64 ready, LSB compliant Eclipse 3.0
 - Work on the 3.0 beta series is in progress
 - We will post announcements on our site and on gelato.org
- ✦ Eclipse plug-ins are in development
 - Will provide user-friendly GUI interfaces to IMPACT, ecc, perfmon, and other tools we use frequently



Conclusion

- ✦ Standards compliance provides numerous benefits to end users, maintainers, and developers
- ✦ Bringing a large application into compliance can take work, but good tools are available to ease the process
- ✦ Bringing a simple application into compliance can be as easy as “make CC=lsbcc”



References

- ✦ <http://www.linuxbase.org/>
- ✦ <http://www.freestandards.org/>
- ✦ <http://gelato.uiuc.edu/projects/eclipse/>

