Certification Infrastructure for the Linux Standard Base (LSB)

Vladimir Rubanov, Denis Silakov
Institute for System Programming of the Russian Academy of Sciences,
Russian Linux Verification Center,
{vrub, silakov}@ispras.ru,
http://linuxtesting.org/

Abstract. Expanding further success of the Linux operating system is hampered by problems with portability of applications across various Linux distributions. Identifying and standardizing a common subset of mature functionality available in most Linux distributions is the target of the Linux Standard Base (LSB), an interface standard being developed by the Linux Foundation - the leading international consortium to foster the growth of Linux. Current LSB status is described with the focus on the technical certification infrastructure (frameworks for automated development and execution of LSB compliance tests and the online certification system itself). The article emphasizes the critical role of the infrastructure in development of the standard itself and in its successful adoption in real world practice.

1. Introduction

Linux proved itself as an industrial strength operating system. It is deployed on an ever increasing number of computers from supercomputers, servers and desktops to mobile phones and embedded systems. We are witnessing the extraordinary growth in the Linux community including application developers. However, there are some negative issues, which need to be addressed. Application developers are faced with a large and constantly increasing number of Linux distributions from different vendors and they need to take into account the many differences between these packages when they want to develop applications “for the Linux platform”. That is why defining Linux as a uniform platform is crucial for keeping the growth and ensuring vitality of this operating system.

Removing differences in the core platform services provided by different distributions is important for decreasing the costs of developing Linux applications. This directly affects the number of applications available for this platform, which is one of the key characteristics of operating system adoption.

At the moment of writing this article there are 543 (!) different distributions registered at http://lwn.net/Distributions/. And this number does not take into account special versions developed for internal company or individual use. But what

is Linux from the application developer's point of view? It is a combination of system components such as kernel and libraries that work jointly to provide application programming interfaces (APIs) to applications. The problem is that every Linux distribution consists of a unique set of specific versions and builds of such components, which results in that distributions may vary in the number of and behavior of the provided to applications interfaces both at the API and binary levels. That is why applications that work on one distribution may fail on another and supporting multiple distributions becomes a serious problem for application developers. Of course, it is possible to develop specific versions of an application for particular distributions but this is rather expensive and may not be affordable for some developers, which may completely reject supporting Linux platform. This inhibits growth of Linux applications and the adoption of the platform itself as developers want to develop applications "for Linux" not just for RedHat or Suse.

To approach this problem, it was proposed to standardize a common set of functionality in the main distributions and recommend using only these functions when developing portable Linux applications. This would enable portability of compliant applications across all compliant distributions.

This article contains two sections. The first one introduces the main industry standard targeted at solving portability problems for Linux applications - Linux Standard Base (LSB) [1]. The second section presents the experience and results of the Institute for System Programming of the Russian Academy of Sciences (ISPRAS) in building certification infrastructure for this standard. Such infrastructure turned out to be extremely important for successful adoption of the standard.

2. Linux Standard Base (LSB) - the Single Linux Platform

The core idea of the Linux Standard Base (LSB) standard is to describe a subset of Linux interfaces provided by various libraries that constitute "the Linux platform" from the application developer point of view. This subset should be present in most Linux distributions and provide compatible services. Description should include information about binary level symbol (ELF name) of each interface and API-level information (parameters, return values and corresponding types) including behavior specification of the interfaces. To develop such a standard, a non-profit international consortium was founded in 2000 - Free Standards Group (at present Linux Foundation [2]), supported by the leading IT companies including IBM, Intel, HP, Novell, and Oracle. The first version of the standard was published in June 2001 and covered about 3000 interfaces. In the next years the standard grew and matured with each new version covering more and more interfaces (while excluding some obsolete). The current LSB version is 3.2 and it includes over 30000 interfaces from more than 40 libraries. Most of the main Linux distributions have now been certified for LSB compliance.

The important thing about LSB is that it does not try to impose something completely new on Linux distribution vendors. In many cases LSB just refers to

established industry standards and documentation that existing implementations conform to de-facto. Such referenced specifications include SUSv3 (POSIX), ISO C99, ISO C++ Language and various documentations maintained by upstream component developers. And only in the case of missing stable description of an interface, LSB describes it on its own based on real-world implementation of this interface present in the main distributions.

When considering standardization scope, LSB uses the "best practice" criterion. This means an interface becomes a candidate for inclusion in LSB when it is present in all the main Linux distributions and is actively used by real applications. In other words, interfaces for standardization should be quite popular both in the distribution and in the application domains. Also, some technical requirements should be met such as that interfaces should have stable implementation, documentation and tests. It is important to note that the role of a vendor-independent international consortium helps Linux Foundation to take unbiased decisions when developing LSB.

The modern LSB version 3.2 includes 5 mandatory modules:

- **LSB Core** - low-level system interfaces in C (libc, libcrypt, libdl, libm, libpthread, librt, libutil, libpam, libz and libncurses libraries).
- **LSB C++** - standard real-time support library for C++ (libstdcxx library).
- **LSB Desktop** - various functions for working with graphical interfaces and auxiliary services (mainly XML, X11, GTK and Qt).
- **LSB Interpreted Languages** - Perl and Python environment and modules.
- **LSB Printing** - basically libcups library.

The first three modules include both generic (architecture independent) and architecture specific elements that are roughly structured in the following hierarchy: module->library->group->interface. LSB 3.2 supports 7 architectures - IA32 (x86), AMD64 (x86_64), IA64 (Itanium), Power PC 32, Power PC 64, IBM S390 and IBM S390X. Interpreted Languages and Printing modules have only generic descriptions.

An important factor in understanding the purpose of LSB is that this standard is not targeted at *all* Linux distributions and applications. It is for the *most* of them that are quite general purpose. Specialized distributions (like embedded) and some system applications may not need to be fully LSB compliant. Meanwhile, even if an application uses some interfaces beyond LSB then LSB still does matter when developing such application as it allows reducing development costs due to the intersection with LSB that one can rely upon. To enable compatibility for those interfaces outside LSB, one can use separate methods like static linking of necessary libraries or developing special stub proxies that hide differences between different distributions. LSB just allows reducing the number of interfaces for which such special means are needed.

3. LSB Technical Infrastructure

One of the key success factors for developing and supporting an interface standard like LSB is a proper technical infrastructure that automates main processes for maintaining the standard itself and that brings the standard closer to real developers. In the particular case of LSB the main components of its infrastructure include generators of the standard's text itself and associated header files based on a central database, a web portal for LSB developers, analytical and decision support systems, certification tests and frameworks for their effective development, execution, result analysis and finally certification. We will give an overview of these systems later in this section after presenting some historic background.

3.1 LSB Infrastructure Program

The origins of the current team involved in the LSB infrastructure development come from the Open Linux VERification (OLVER) project – http://linuxtesting.org/project/olver. The project was done by the Russian Linux Verification Center [3] at the Institute for System Programming of the Russian Academy of Sciences (ISPRAS) [4]. It was funded by the Russian Federal Agency for Science and Innovation. We analyzed the text of the LSB Core standard for about 1500 Linux system interfaces, delineated elementary assertions and transformed them into formal specifications in the SeC language from which we then generated conformance tests for automated testing of Linux distributions against LSB Core requirements [5].

The OLVER project results turned out to be of high interest for the Linux community and LSB standard body committee (Free Standards Group at that moment), which proposed ISPRAS a long term cooperation for building a new LSB infrastructure to meet the growing needs of the Linux industry for this standard. The weak infrastructure that existed at that moment (late 2006) was the most burning point for LSB community that hindered wider LSB adoption. In particular, it was critical to improve the certification framework and strengthen test coverage. In early 2007, the Free Standards Group merged with Open Source Development Labs and the newly created Linux Foundation extended cooperation with ISPRAS to cover more infrastructure areas and prepare the jump base for promoting wide LSB adoption. It is important to note that all results (specifications, tools, frameworks) contributed by the Linux Foundation and its partners like ISPRAS to the LSB ecosystem are open-source and Linux developers are encouraged to use them and adapt. The first results of the new LSB Infrastructure Program [6] were announced in June 2007 at the first Linux Foundation Collaboration Summit in Mountain View, CA and since that moment they are being continuously improved with regular releases. Next in this section we describe the current (June 2008) status of these developments with focus on the certification related parts.

3.2 LSB Database and Navigator

The backbone of the entire technical LSB infrastructure is the central database (MySQL engine is used) that contains integrated information about the LSB standard itself, about its surrounding Linux ecosystem and various operational matters including certifications. The current database contains 81 tables with over 25 million records. There are three parts of the database:

1. The *standardization* part includes information about LSB elements that constitute the essence of the standard itself.
2. The *community* part contains information about real-world modern Linux distributions and applications.
3. The *certification* part keeps information about the certification status of various products, audit operations, fee payments, etc.

The *standardization part* includes the following elements:

- grouping elements:
  - modules (collections of libraries and commands);
  - libraries (collections of library groups and headers);
  - library groups (collections of classes and interfaces);
  - headers (collections of interfaces, types and constants);
- and leaf elements:
  - commands;
  - classes;
  - interfaces (global variables and public functions);
  - constants and macros;
  - types.

These elements are interlinked into a graph of dependencies of various kinds. Information contained in this part of the database is enough to generate complete header files that define all standardized interfaces.

The idea of the *community part* is to have a single place with “raw” information about the Linux ecosystem from the platform standardization point of view. Basically it allows understanding which interface elements are provided by particular versions of distributions and which ones are required by various particular applications.

The *certification part* supports certification process and is visible to users through the LSB Certified Product Directory (that shows the list of LSB certified products) and through the online LSB Certification System (that tracks and manages certification workflow).

The database is used by various tools, among which one should emphasize over 40 scripts that generate LSB deliverables such as the text of the standard itself, header files and various code pieces that are part of implementation of other LSB tools (mainly SDK and tests).

To efficiently use the database information from the human point of view, we have created a web portal that provides user interface and brings the database closer to people. This is a public portal named *LSB Navigator* (http://linux-foundation.org/navigator/) that provides search, filtering & browsing capabilities to effectively find necessary information about the LSB and the Linux ecosystem. It can be used by Linux developers, Linux distribution vendors, and LSB workgroup to browse, query, analyze and submit feedback. It is important that the LSB Navigator is used by the LSB workgroup for gathering info when taking decisions about the standardization scope. Selected features of the LSB Navigator include:

- Navigation through the standardized LSB elements from modules down to leaf elements like interfaces and constants.
- Global filters for LSB version and hardware architecture.
- Individual home pages for over 1 million Linux interfaces (each is just 2-click away from the main page via search) that include information on:
  - status of each interface in terms of LSB (in LSB, never been in LSB, planned for inclusion, withdrawn, deprecated);
  - LSB info (module, library, header file, etc.);
  - interface signature (parameters and return value);
  - direct link to documentation of the interface;
  - which distributions provide this interface;
  - which applications use this interface;
  - which tests are available for this interface;
  - community discussions related to the interface.
- Distribution info (provided libraries and interfaces).
- Application info (external libraries and interfaces required by each app).
- Statistics on LSB elements (total numbers of interfaces, commands, classes in each LSB version).
- Statistics on interface usage by applications:
  - which interfaces are most frequently used by different applications (list of interfaces with info on how many applications use each interface);
  - which libraries are most frequently used by different applications (list of libraries with info on how many applications use each library);
  - LSB "rating" of registered applications (list of applications with info on the number of LSB and non-LSB libraries and interfaces used by each application).

An important idea in the context of supporting certification is that the LSB Navigator serves as an online reference and a knowledge base for the LSB standard. Other systems can easily use links to specific pages in the Navigator thus providing an integrated environment for users.

3.3 LSB Certification Tests

Ian Murdock, the former LSB Chair, said: "An interface standard is only as good as its test suites". In late 2006, the LSB test coverage was about 15%, which means that

85% of standardized interfaces did not have tests at all. That is why developing new tests that check conformance of distributions against the requirements of LSB standard was the first priority to enable real life value of this standard. The problem is that the number of interfaces in the LSB is too huge to develop tests of excellent quality for all of them. To create a feasible strategy for developing LSB tests, we defined three testing quality grades (the borders between the testing grades are obscure as the scale is actually continuous):

1. *Shallow* – simple tests with the only guaranteed purpose of ensuring the interface does not crash (sometimes it is additionally checked that the interface does not return an error code) being called with some particular correct parameters and in the correct environment. This is close to "existence" or "smoke" or "sanity" tests (but beware - these terms are interpreted differently by different experts).
2. *Normal* – this is the most reasonable level of testing achievable by tests written in plain C. The tests check the main functionality and may check a few error scenarios. Most of the legacy LSB tests are of this quality.
3. *Deep* – this is the level when most of the specification assertions are tested in various conditions/states. This is usually done for most important and critical software.

To develop tests of corresponding grades, we are using different approaches.

For *shallow tests*, we developed a new technology and tools (**AZOV Framework**) for automatic generation of shallow tests based on some description of the interfaces, their parameters, dependencies and default values in the LSB database. The core idea here is to augment the database to contain enough information about the interfaces and their dependencies that would allow automatically building correct call chains representing typical scenarios of interface usage. The first version of the shallow testing framework is ready and is now being used for Qt tests development. The cost of developing shallow tests is mainly in populating the information in the database + debugging automatically generated tests.

Normal tests are basically manual C tests, though we do use some automation here as well. We have been inspired by the generator that we found in the existing LSB tests for GTK – gtkvts. Basically, it allows automatically generating many test instances based on the same parameterized code thus achieving better testing quality without duplicating the code. However, we found particular gtkvts implementation limited and we used only the idea while implemented our own tools and methodology called **T2C** (template-to-C). The first version of the T2C Framework is ready and is now being used for developing GTK, C++ and X11 tests.

Deep level of testing is hard to achieve by manual tests in C; so advanced testing technologies are necessary here. We use our own ISPRAS **UniTESK** technology for this. It is based on a model-based testing technique where requirements for the target system are expressed as formal specifications in a special language (SeC in UniTESK) and then various test actions are automatically generated on-the-fly from

test scenarios. Our former project OLVER (see 3.1 above) was based on this technology and now we are adapting/improving it to become official LSB tests for the LSB Core part.

For the normal and deep tests, it is important to have linkage to the text of the specification assertions. This means when a test fails it should say which particular assertion in the standard's text is violated as well as provide info about particular mismatch (like XX expected, YY returned). Normal and deep tests being developed in ISPRAS do have this feature and analyzing reports and debugging failures is much easier with this approach. Leveraged by a visual execution environment and interactive HTML reports this gives unprecedented comfort for the users. Apart from better reporting, the assertion catalog also enables measuring test coverage in terms of the number of assertions checked by the tests, which gives an advanced level of test quality measurement.

Currently, there is a testing strategy being implemented by ISPRAS to achieve almost 100% test coverage for LSB interfaces by the end of 2008 but with most of the tests of shallow quality. The target for the end of 2009 is to cover most of the libraries with normal quality tests and the most important part (LSB Core) with deep quality tests.

3.4 LSB Certification Tools

To make LSB certification and testing technically appealing, it is important to have user-friendly systems that support these processes. As a part of the LSB Infrastructure program in early 2008 we created a new web-based Certification System that guides people certifying their products through the certification workflow and keeps records for certified products. The new certification system includes three major parts:

- *Certification Management* provides step-by-step instructions on what to do and enables easy status tracking and collaboration with LF staff during the certification process for companies and individuals who want to certify their Linux distributions or applications against LSB.
- *Product Directory* is a public part of the Certification System that contains the current list of LSB-certified Linux distributions and applications with various views and groupings.
- *Problem Reporting* is for online collaboration on solving problems arising in the process of certification. It also provides a knowledge base of various issues and solutions.

There is also *Administration Mode* that allows managing the certification system from the LF side with full rights on administering all the data.

Finally, at the technical level of certification process it is necessary to be able to smoothly execute automated tests and analyze the results. For this purpose, we have developed *Distribution and Application Testkit Managers* (DTK and ATK Managers) for testing distributions and applications respectively. These tools are

web-based with embedded simple web-server. ATK Manager is also known as Linux Application Checker.

The tools represent test execution frameworks that have the following key features:

- integrated user interface for all the LSB test suites (web based and command line);
- selection of tests to run (all, manually selected subset, predefined subsets, etc.);
- saving/loading configured options in user profiles for quick test runs in the future;
- automatic download of missing test suites from the Linux Foundation FTP site;
- 'one-click execution of certification tests;
- unified test reports with links to the knowledge base of known issues and to home pages of interfaces in the LSB Navigator for more information including community discussions;
- management of test results history.

Further, the ATK Manager provides features for application analysis without regard to the certification process. These features include viewing the list of all external libraries and interfaces required by an application with status in LSB perspective (if in the LSB or not). In particular, this allows detecting unused libraries present in DT_NEEDED section of the application under analysis but without actual interfaces used by the application in such libraries.

The online certification system and ATK/DTK Mangers are integrated to provide transparent switches between local and the LF central-server based functionality to provide an easy to use complete certification process.

4. Conclusion

Problems of application portability among different Linux distributions prove to be one of the most important factors that inhibit the growth of the number of applications available for Linux and thus prevent developing further success of the Linux platform as a whole. In this paper we have considered the Linux Standard Base open standard, which is the primary modern effort to address this problem. A number of industry companies initiated this activity to standardize a common subset of Linux functionality that most applications can rely upon. Currently the LSB standard is developed by the Linux Foundation international consortium with funding of such companies as IBM, Intel, HP, Novell, Oracle, etc.

In order to develop a good interface standard such as LSB it is crucial to have a proper technical infrastructure. The Linux Foundation jointly with the Institute for System Programming of the Russian Academy of Sciences are developing such an infrastructure for the LSB standard.

The current results of this cooperation include new production versions of the following systems:

- Central LSB Database & various data transformation scripts.
- LSB Navigator – a web portal on top of the central LSB database with advanced navigation, queries, community collaboration mechanisms, developers' feedback and contribution interface to promote information about the LSB and surrounding ecosystem and also for making decisions on the LSB standardization scope.
- LSB DTK and ATK Managers – to automate Linux distribution and application certification testing in a user friendly way.
- LSB Certification System (integrated with the ATK/DTK Managers) - to support and facilitate LSB certification workflow.
- Misc. auxiliary tools for automating investigation and analytical tasks.
- New testing technologies and tools for automated test development of various quality grades:
  - **UniTESK** for deep testing (ISPRAS owned technology with over 10 years history).
  - **T2C** – methodology and tools for normal tests development (developed specially for the LF on top of TET harness and ideas of gtkvts .inp files).
  - **Azov** – innovative methodology and tools for automated massive development of shallow tests based on extended information from the central LSB database (developed specially for the LF).
- New tests for more than 19000 of LSB interfaces.

The developed infrastructure is being used in real life and has been getting many positive responses. Meanwhile, there are plans to improve many issues - first of all, mature all the tools and achieve 100% test coverage of the LSB interfaces. This will help making the LSB standard the acknowledged "single Linux platform" that really mitigates application portability problems, which will clear the way for further expansion of Linux.

5. References

[1] *Linux Standard Base Homepage*. http://www.linuxfoundation.org/en/LSB/.

[2] *Linux Foundation*. http://linuxfoundation.org/

[3] *Linux Verification Center*. http://linuxtesting.org/

[4] *Institute for System Programming of the Russian Academy of Sciences.* http://ispras.ru/

[5] V. Kuliamin, A. Petrenko, V. Rubanov, A. Khoroshilov. *Formalization of Interface Standards and Automatic Construction of Conformance Tests*. Proceedings of SECR 2006 conference, Moscow.

[6] *LSB Infrastructure Program*. http://ispras.linuxfoundation.org/