Electronic Communications of the EASST
Volume 20 (2009)

Proceedings of the
Third International Workshop on
Foundations and Techniques for
Open Source Software Certification
(OpenCert 2009)

Automatic Analysis of Applications for Portability Across Linux Distributions

Vladimir Rubanov

9 pages

Guest Editors: Luis Barbosa, Antonio Cerone, Siraj Shaikh
Managing Editors: Tiziana Margaria, Julia Padberg, Gabriele Taentzer
ECEASST Home Page: http://www.easst.org/eceasst/ ISSN 1863-2122

Automatic Analysis of Applications for Portability Across Linux Distributions

Vladimir Rubanov[1]

[1] vrub@ispras.ru
Institute for System Programming of the Russian Academy of Sciences,
Russian Linux Verification Center
http://linuxtesting.org/

Abstract: Problems with portability of applications across various Linux distributions is one of the major sore spots of independent software vendors (ISVs) wishing to support the Linux platform in their products. The source of the problem is that different distributions have different sets of system libraries that vary in the interfaces (APIs) provided, with respect to both composition and behavior. And the critical questions arise such as which distributions my application would run on or what can I specifically do to make my application run on a greater number of distributions. This article presents an approach and the Linux Application Checker tool that help to answer such questions by automatically analyzing the target application and confronting collected data with the internal knowledge base containing information about the various real world distributions. Additionally, Linux Application Checker is an official tool approved by the Linux Foundation for certifying applications for compliance with the Linux Standard Base (LSB) standard, the leading effort for the single Linux specification.

Keywords: Linux, Application portability

1 Introduction

There are more than 500 public Linux distributions in the modern world (see for example http://lwn.net/Distributions/). And each of these distributions represents a unique combination of versions/modifications of the base upstream components such as kernel, shared libraries, system commands, etc. In this article, we consider Linux distributions mainly as system platforms enabling operation of third-party applications. In this perspective, the main components of a Linux distribution are the kernel itself and a set of system libraries that provide application programming interfaces (APIs – functions or global data) to the applications.

The problem comes from the fact that different distributions can provide different versions of libraries (including custom modifications introduced by specific distribution developers). This can result in different sets of interfaces both from the composition (entire libraries or some interfaces within the same library may be missing in some distributions) and from the behavioral (the same interface can behave differently on different distributions) points of view. That is why it becomes so difficult to create a portable application that can be run on any distribution out-of-the-box. The task becomes even more difficult if the application vendors want to have their applications portable in the binary form (which is critically important for commercial vendors). 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 so that they might 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.

Distribution developers try to mitigate these problems by supplying several versions of the same shared library in their distributions so that applications could find and use appropriate versions. Efforts to standardize some common subset of libraries and their interfaces that can be found in most of the modern distributions also help to deal with the portability problems (see **Linux Standard Base (LSB)** [LSB], [Rub07]).

However, application developers still have to separately analyze each specific distribution that they want to target their application at to check which libraries and interfaces it provides. And then manually extract the common subset in all the target distributions to understand which functions can be relied upon by the application. The application then needs to be developed/modified to use only this subset of libraries and interfaces. Automating this process of portability analysis and decision support is a critical step to help developers to create portable applications that can be run out-of-the-box on a wide variety of Linux distributions.

In this article, we present results of the **Russian Linux Verification Center** [LVC] at the **Institute for System Programming of the Russian Academy of Sciences (ISPRAS)** [ISP]. Under cooperation with the **Linux Foundation** [LF] we collected data about main Linux distributions in a central knowledge base and developed a tool (**Linux Application Checker**) for automatic analysis of application portability across these analyzed distributions. The data in the knowledge base is constantly updated to be in synch with the state of the Linux art.

The article contains two sections. The first one introduces the knowledge base about modern Linux distributions maintained under the auspices of the Linux Foundation – the leading consortium dedicated to fostering growth of Linux. The second section presents the Linux Application Checker tool designed for automated analysis of cross-distribution portability of Linux applications.

2 Linux Foundation Ecosystem Knowledge Base

To standardize, protect and promote Linux, the leading IT-companies (IBM, Intel, HP, Novell, Oracle and many others) formed an international non-profit consortium **The Linux Foundation** [LF], which now represents the main power in the World that unites efforts and expertise of various organizations and persons that are interested in ensuring the further growth and success of Linux as a platform. The consortium also provides a neutral forum to discuss various issues and make collaborative decisions.

One of the main technical activities of the Linux Foundation is development of the **Linux Standard Base (LSB)** standard. The core idea of the standard is to describe a subset of Linux interfaces provided by various libraries that constitute the single Linux platform from the application developer point of view. This subset should be present in most Linux distributions and should provide the same functionality in each of them. Description in the standard should include information about binary level symbols (i.e. names of ELF (Executable Link Format) symbols) of all interfaces and API-level information (parameters, return values and corresponding types) including behavior specification of the interfaces.

LSB is a living standard, which means it is regularly revisited and new versions are released. Usually, each LSB version corresponds to one generation of the main Linux distributions (e.g. LSB 3.0 is for RHEL 4 and SLES 10, while LSB 4.0 is for RHEL 5 and SLES 11). So one of the key success factors for developing and supporting an interface standard like LSB (covering more than 30,000 interfaces!) is a proper technical infrastructure that automates the main processes for maintaining the standard itself and that brings the standard closer to real developers.

The Linux Foundation (formerly Free Standards Group) already developed a basic infrastructure for the first LSB versions in the early 2000s. But with the growth of the standard, it was realized that a next-generation infrastructure is needed that should include many more components and automate many more processes in the standards development. In 2006, the Linux Foundation jointly with ISPRAS launched a new **LSB Infrastructure Program** [Inf].

In context of this paper, the important direction of this new program was building a new *central database* (MySQL based), which became the backbone of the entire technical LSB infrastructure. The database contains integrated information about the LSB standard itself, about its surrounding Linux ecosystem and various operational matters like active certifications. The current database contains 97 tables with over 89 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 second part of the database (also known as **Linux Foundation Ecosystem Knowledge Base**) is most interesting for the topic of this paper because it is this part that enabled creation of the Linux Application Checker tool for automated analysis of cross-distribution portability of Linux applications. Basically, the information in this part represents the structure of popular Linux distributions in terms of specific libraries and their interfaces provided by each distribution to applications. As of March 2009 the database includes information about 88 versions of Linux distributions.

The contents of the database are open to the community and can be browsed in a user-friendly way using the **LSB Navigator** [Nav] web-portal.

3 Linux Application Checker

Collecting information about the structure of popular Linux distributions in a central database was the principal step to enable automated cross-compatibility analysis of applications. But in order to effectively implement this analysis, it was important to create a tool that automates corresponding further steps:

1. Understand *external dependencies* (external libraries and interfaces needed) of the target application.

2. Compare the applications dependencies with the information about *libraries and interfaces provided by each Linux distribution* from the knowledge base.

3. Visualize *the results* and provide additional hints on how to fix some issues.

The **Linux Application Checker** tool (or App Checker in short) developed jointly by ISPRAS and Linux Foundation serves this purpose. Additionally, it checks some other compatibility issues (e.g. names of ELF sections in binary files) but the main point remains in the match of the libraries/interfaces required by application with the libraries/interfaces provided by distributions.

Linux Application Checker provides visual user interface based on a simple embedded web-server. At the starting page (Application Check), users should select a set of components that constitute the target application package. The leaf components to be analyzed are single binary files in ELF format (executables or .so libraries). Also, App Checker checks Perl, Python and shell scripts for compatibility issues. To facilitate the selection process for complex applications, it is possible to indicate groups of components by selecting aggregate entities:

- whole *directories* (all executable files and .so libraries as well as Perl, Python and shell scripts in the directory will be considered as belonging to the application);

- *installation packages* or *tarballs* in RPM, DEB, tar.gz, tar.bz2 formats (all files of the proper types in the packages will be considered as belonging to the application);

- *packages already installed* in the system.

During analysis, App Checker inspects all the components indicated as belonging to the application (both directly and through the aggregate entities) and performs a number of checks. All the checks can be divided into two groups:

- *cross-distribution compatibility analysis* based on the Linux Foundation Knowledge Base (LF KB);

- *certification checks* on compliance with the *LSB standard*.

The first kind of checks is for the general audience of Linux application developers interested in maximizing portability of their products. App Checker provides a number of reports concerning specific compatibility issues that can prevent the application from running on certain Linux distributions. The most common reason is libraries or interfaces required by the application but missing in some distributions. Information about required libraries is taken from ELF-files of the application by filtering DT_NEEDED records of the .dynamic section. Required interfaces are taken by filtering ELF-symbols from the .dynsym and .symtab sections. Internal dependencies between the components of the application itself are excluded from the list of external dependencies.

If available and appropriate, App Checker provides additional information and recommendations on how to fix detected compatibility issues. These recommendations are taken from the known issues sections in the Linux Foundation Knowledge Base (for example, advice on replacing some deprecated interfaces by more portable modern counterparts or recommendations to include some missing library as a part of the application package).

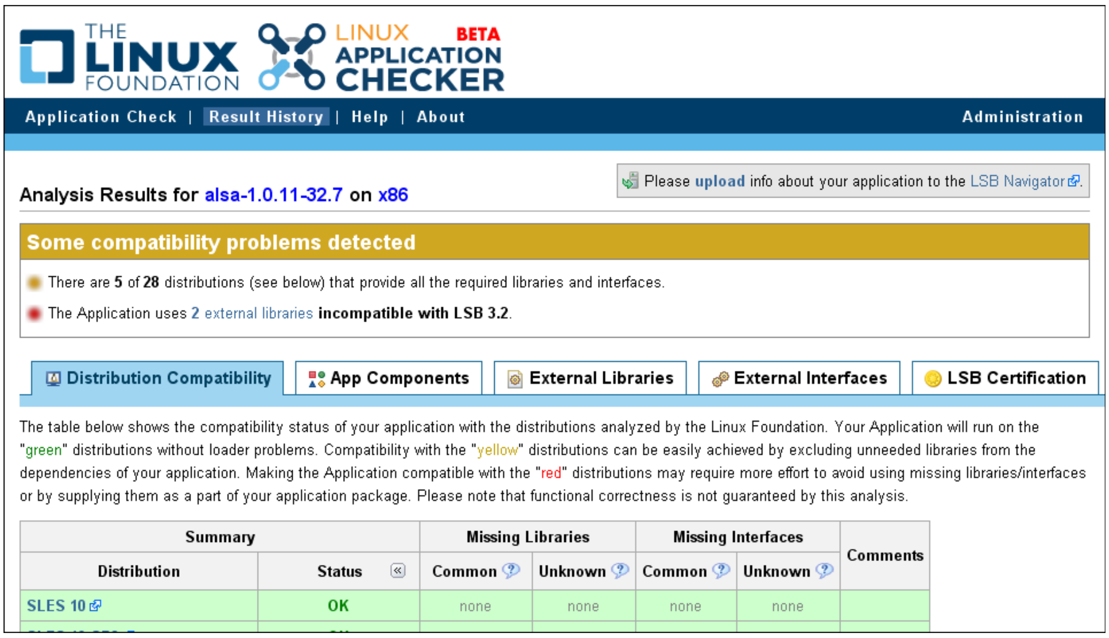
LSB checks are more specific. Their main purpose is to analyze compliance of the application with the LSB standard in the context of formal certification. The report generated by these checks serves as the main input on which to base decisions about LSB certification of applications. Analysis of external dependencies of the application is performed as well but the dependencies are confronted with the libraries and interfaces included in the LSB, not in any specific distribution. Basically, it is checked that the application uses only those libraries and interfaces that are standardized by the LSB. Also, App Checker ensures compliance of other aspects like binary structure of ELF-files and .rpm packages, usage of the specific dynamic loader, etc.

3.1 Using Linux Application Checker Reports to Analyze Applications

At the end of the analysis, Linux Application Checker provides a report structured into a few subsections shown in separate tabs. The general look of the report is presented at Figure 1. There is a short summary at the top of the report starting with a color coded one-line compatibility verdict (green – no compatibility issues, yellow – some issues, red – numerous issues) supplemented with a few lines of statistics on the number of found issues of different kinds like the number of incompatible distributions, unused libraries, non-LSB elements, etc. There are 5 tabs/reports below that present detailed information about compatibility issues in different views. The sections below discuss each of these reports in more detail.

3.1.1 Distribution Compatibility Report

This tab is open by default and contains information about compatibility of the analyzed application with particular Linux distributions (information on which is present in the LF KB). The data is presented as a table with each line corresponding to one distribution. Each line is color coded in green (the application is fully portable), yellow (there are minor issues) or red (there are serious compatibility issues). It is important to understand that compatibility verdicts actually mean possibility to successfully run application but they do not guarantee that the application will actually behave correctly. The compatibility analysis is mainly based on confronting available in each distribution libraries and interfaces with those required by the application. By leaving the mouse pointer over specific lines in the table, it is possible to see a hint that contains more detailed information on the number of compatibility issues of various kinds for corresponding distributions. By clicking on the table it is possible to expand the table and see detailed information for all lines at once (in additional columns). There is a special type of unknown libraries – the presence of these libraries is not guaranteed to be reliably analyzed in all the distributions. This is because the LF KB contains information about presence of just a selected set of so called


THE LINUX FOUNDATION
LINUX APPLICATION CHECKER BETA
Application Check | Result History | Help | About
Administration
Distribution Compatibility
App Components
External Libraries
External Interfaces
LSB Certification

Analysis Results for alsa-1.0.11-32.7 on x86

Please upload info about your application to the LSB Navigator.

Some compatibility problems detected

● There are 5 of 28 distributions (see below) that provide all the required libraries and interfaces.

● The Application uses 2 external libraries incompatible with LSB 3.2.

The table below shows the compatibility status of your application with the distributions analyzed by the Linux Foundation. Your Application will run on the "green" distributions without loader problems. Compatibility with the "yellow" distributions can be easily achieved by excluding unneeded libraries from the dependencies of your application. Making the Application compatible with the "red" distributions may require more effort to avoid using missing libraries/interfaces or by supplying them as a part of your application package. Please note that functional correctness is not guaranteed by this analysis.

Summary		Missing Libraries		Missing Interfaces		Comments
Distribution	Status	Common	Unknown	Common	Unknown	
SLES 10	OK	none	none	none	none	

Figure 1: Linux Application Checker Report Header

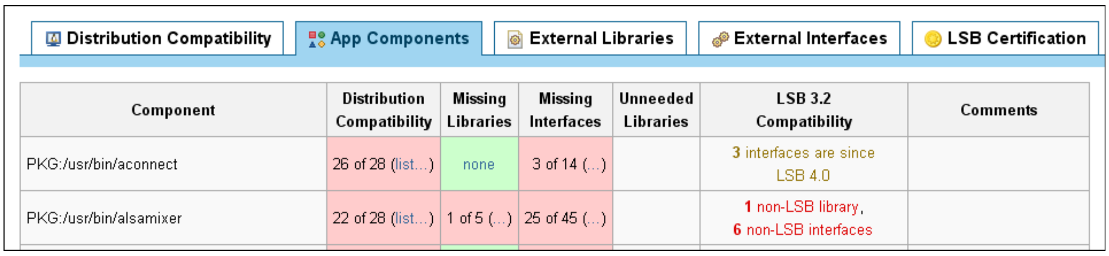
approved libraries as it would consume too much space if we stored information about all libraries present in each distribution. Meanwhile, the list of approved libraries covers those that are most widely-used by applications and currently includes 1177 names. This list was formed by analyzing more than a thousand popular applications and creating a union of all the libraries required by these applications. And we continue adding to the list as we come across new libraries needed by new applications.

So, the first report allows easy understanding of which particular distributions contain all the necessary libraries and interfaces for the application while allowing the user to choose to see the list of missing entries for incompatible distributions (the list is shown in separate window when clicking on a proper links inside the table).

Another interesting aspect that App Checker can also detect is so called unused libraries. Such libraries are included in the external dependencies of some applications components but no actual interfaces are then used from these libraries. Usually, unused libraries can be easily eliminated by just correcting build options of the application, which improves portability of the application.

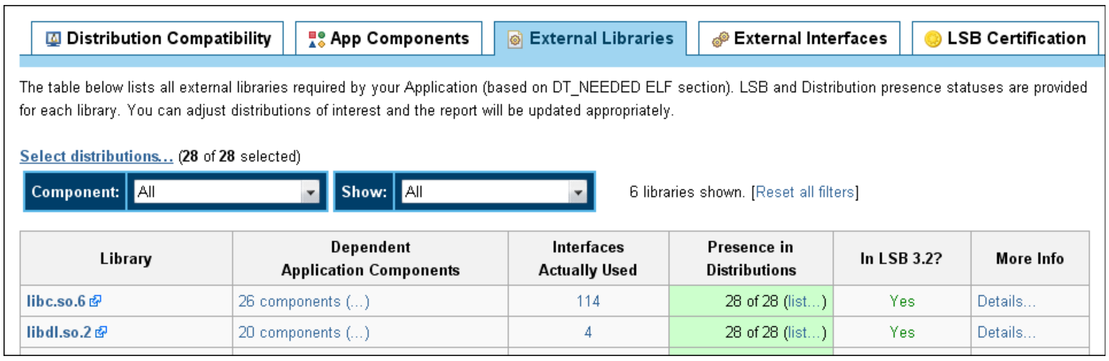
3.1.2 Application Components Report

The next tab presents information about particular components of composite applications (see Figure 2) like executable files, shared .so libraries and Perl, Python and shell scripts. For each component, compatibility status is shown – the number of distributions this component is compatible with (this helps easily identifying problematic components) with the next two columns providing detailed information about the number of problematic libraries and interfaces from the components external dependencies (those ones not provided by some distributions). The num-



Component	Distribution Compatibility	Missing Libraries	Missing Interfaces	Unneeded Libraries	LSB 3.2 Compatibility	Comments
PKG:/usr/bin/aconnect	26 of 28 (list...)	none	3 of 14 (...)		3 interfaces are since LSB 4.0	
PKG:/usr/bin/alsamixer	22 of 28 (list...)	1 of 5 (...)	25 of 45 (...)		1 non-LSB library, 6 non-LSB interfaces	

Figure 2: Application Components Report



The table below lists all external libraries required by your Application (based on DT_NEEDED ELF section). LSB and Distribution presence statuses are provided for each library. You can adjust distributions of interest and the report will be updated appropriately.

Select distributions... (28 of 28 selected)

Component: All   Show: All   6 libraries shown. [Reset all filters]

Library	Dependent Application Components	Interfaces Actually Used	Presence in Distributions	In LSB 3.2?	More Info
libc.so.6	26 components (...)	114	28 of 28 (list...)	Yes	Details...
libdl.so.2	20 components (...)	4	28 of 28 (list...)	Yes	Details...

Figure 3: External Libraries Report

ber of unused libraries is shown for each component (if any) and a special column summarizes LSB-compliance status of the components. Any cell in the table can be clicked to see the list of specific entries behind the summary numbers shown.

3.1.3 Reports on External Libraries and Interfaces

The third and the fourth tabs (see Figure 3) present information about the applications external libraries and external interfaces respectively (the union for all app components excluding internal dependencies).

For each library, the following information is shown:

- the list of app components that require this library;
- the actual number of interfaces that are used by the app from this library (unused libraries have red 0 in this column);
- the number of distributions that provide this library;
- LSB status of the library.

The fourth tab with external interfaces is quite similar. It just contains the list of interfaces instead of libraries and a couple of additional columns:

- interfaces version (like GLIBC_2.3);
- name of the library that provides this interface.

At both tabs, there are a number of filters that allow the view to be limited to only interesting records. For example it is possible to define specific distributions that are of interest for analysis (e.g. by excluding old distributions) – the compatibility column will be recalculated accordingly including the color coding. Also, it is possible to limit the view to only those entries that are relevant to a specific app component or library (in case of interfaces). A special filter allows the view to be focused on entries concerning particular problem types (entries with compatibility problems, unused libraries, non-LSB entries, deprecated entries, etc.).

3.1.4 LSB Certification Report

The fifth (the last) tab is devoted to the official certification report on compliance of the target application with the LSB standard (a specific LSB version is selected at the initial screen when specifying the application to check). The list of issues found is displayed in subsections that can be grouped either by application component or by issue type. If there are no critical issues then it is easily possible to apply for the formal certification right away by clicking a link in the report, which will smoothly bring the user to the online certification system at the Linux Foundation server and will ensure all the data necessary for initiating the certification process is transparently transferred to the server (first of all the technical certification report). As soon as the certification process is complete the application is automatically included into the official LSB-certified Product Registry.

3.2 Sharing Data about Applications with the Community

As a part of its Linux Ecosystem Knowledge Base, the Linux Foundation maintains information about real world applications. Thats why all ISVs (Independent Software Vendors) are invited to provide information about their applications to the Linux Foundation, which then can be shared with the community through the LSB Navigator web-portal [Nav].

Apart of the general information (name, home page, application type, etc.), this information basically includes the list of external dependencies of the application. The information about applications is useful to the Linux Foundation and the community for understanding which libraries and interfaces are most popular among ISVs. This in particular helps to make decisions on which libraries and interfaces to include/exclude to/from the LSB standard.

The Linux Application Checker is the easiest way to submit information about your application to the Linux Foundation. There is a special Upload Info link at the summary area of the App Checker reports. Clicking this link brings a form requesting a few additional informational items. After filling in these items it is possible to just press Upload button that will automatically put all the necessary information about the app right away to the Linux Foundation FTP-server. Alternatively, for those highly concerned with privacy, it is possible to save a file with all the necessary information, inspect it and then send to a special e-mail.

4 Conclusion

Differences between Linux distributions regarding the services, libraries and the library interfaces provided to applications create serious difficulties for ISVs, who have to create different builds of their applications for different Linux distributions. Fortunately, initiatives like Linux standardization and efforts of distribution vendors to support legacy applications help to form a common basis that can be found in most of distributions. Applications that use only this basis become portable across various distributions. The core of this basis is the LSB standard while there are actually many different surrounding bases dependent on the target segment of distributions. That is why it is very important for ISVs to have automated tools that can help them to understand this basis in the case of their specific segment of target distributions in the context of their particular application.

The leading IT-companies are committing significant resources (partly through the Linux Foundation) to making it possible to easily build portable applications for Linux. The Linux Foundation Ecosystem Knowledge Base and the Linux Application Checker tool presented in this paper that help ISVs analyze (and consequently fix appropriately) their applications for cross-distribution portability are the flagship achievements in this field at the moment. Additionally, Linux Application Checker is an official tool for certifying applications for compliance with the various versions of the LSB standard. The latest version of this open source tool can be downloaded from the pages of the LSB Infrastructure Program [Inf].

Bibliography

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

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

[LF] Linux Foundation. http://linuxfoundation.org/

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

[LVC] Russian Linux Verification Center. http://linuxtesting.org/

[Nav] LSB Navigator Portal. http://linuxfoundation.org/navigator/

[Rub07] V. Rubanov. Linux Standard Base (LSB): Single Linux Specification and Support Infrastructure. *Proceedings of SECR 2007*, 2007.